How an Adaptive Learning Rate Benefits Neuro-Fuzzy Reinforcement Learning Systems

Takashi Kuremoto[*,1], Masanao Obayashi[*], Kunikazu Kobayashi[**], Shingo Mabu[*]

[*] Graduate School of Science and Engineering, Yamaguchi University
Tokiwadai 2-16-1, Ube, Yamaguchi, 755-8611, Japan
{wu, m.obayas, mabu}@yamaguchi-u.ac.jp
[**] Schoo of Information Science and Technology, Aichi Prefectural University
Ibaragabasama 1522-3, Nagakute-Shi, Aichi, 480-1198, Japan
{wu, m.obayas, mabu}@yamaguchi-u.ac.jp

Abstract. To acquire adaptive behaviors of multiple agents in the unknown environment, several neuro-fuzzy reinforcement learning systems (NFRLSs) have been proposed Kuremoto et al. Meanwhile, to manage the balance between exploration and exploitation in fuzzy reinforcement learning (FRL), an adaptive learning rate (ALR), which adjusting learning rate by considering "fuzzy visit value" of the current state, was proposed by Derhami et al. recently. In this paper, we intend to show how the ALR accelerates some NFRLSs which are reinforcement learning systems with a self-organizing fuzzy neural network (SOFNN) and different learning methods including actor-critic learning (ACL), and Sarsa learning (SL). Simulation results of goal-exploration problems showed the powerful effect of the ALR comparing with the conventional empirical fixed learning rates.

Keywords: neuro-fuzzy system, swarm behavior, reinforcement learning (RL), multi-agent system (MAS), adaptive learning rate (ALR), goal-exploration problem.

1 Introduction

As an active unsupervised machine learning method, reinforcement learning (RL) has been developed and applied to many fields such as intelligent control and robotics since 1980s [1] – [3]. The learning process of RL is given by the trials of exploration and exploitation of a learner (agent) in unknown or non-deterministic environment. Valuable or adaptive actions which are optimized output of RL systems are obtained according to the modification of action selection policies using the rewards (or punishments) from the environment. Generally, there are four fundamental components in RL: state (observed information from the environment, input); policy (usually using probability selection to keep exploitation to an unknown environment); action (the output of the learner changing the current state) and reward (perceived/obtained by the learner during the transition of the state).

[1] A part of this work was supported by Grant-in-Aid for Scientific Research (JSPS No. 23500181, No. 25330287, No. 26330254).

In the history of RL research, there are some severe problems need to be solved theoretically:

(1) The explosion of state-action space in high dimension problems (the curse of dimensionality);

(2) The balance between exploration and exploitation;

(3) Learning convergence in partially observable Markov decision process (POMDP).

To tackle the first issue, linear approximation method [2], normalized Gaussian radial basis function classification method [3], fuzzy inference systems [4] – [8], etc. are proposed. Specially, neuro-fuzzy systems with a data-driven type self-organizing fuzzy neural network (SOFNN) proposed in our previous works showed their adaptive state identification ability for different RL algorithms such as actor-critic learning [5] [6], Q-learning, and Sarsa learning [7].

Meanwhile, Derhami et al. proposed to use adaptive learning rate (ALR) and adaptive parameter of state transition function to obtain a suitable balance of exploration and exploitation [8]. Effectiveness of the ALR has been confirmed by its application to a fuzzy controller with Q learning algorithm with simulation results of some benchmark problems such as boat problem and mountain-car problem in [8].

Furthermore, the third problem of RL mentioned above is more serious to a multi-agent system (MAS). Even the exploration environment is stable, the existence of other agents nearby the learner agent is uncertain. In [4] – [8], Kuremoto et al. proposed to calculate the reward of suitable distance between agents to modify the action policy and showed its higher learning convergence comparing with RLs by individuals independently.

In this paper, we adopt Derhami et al.'s ALR to Kuremoto et al.'s neuro-fuzzy reinforcement learning systems to improve the learning performance of agents in MASs. Goal-exploration problems were used in simulation experiments and the comparison between results with conventional empirical fixed learning rate and ALR is reported.

2 Neuro-Fuzzy Reinforcement Learning Systems

2.1 Reinforcement Learning

Markov decision process (MDP) is used in reinforcement learning (RL) algorithms. Let a state space $\mathbf{X}(x_1, x_2, \ldots, x_n) \in R^n$, an action space $\mathbf{A}(a_1, a_2, \ldots, a_m) \in R^m$, a state transition (from the current stat s_t to the next state s_{t+1}) policy π with probability $P^{a_t}_{s_t s_{t+1}, \pi} : \mathbf{X} \times \mathbf{A} \times \mathbf{X} \in [0, 1]$, and a reward function $R = \sum_{t=0}^{\infty} r_t \in R^1$, where is the reward $r_t \in R^1$ obtained during the state transition. The RL algorithm is to find an optimal policy π^* that can yield the maximum discounted expected reward R^*, i.e. the maximum value function of state $V(\mathbf{X}) \equiv \underset{P(s_{t+1}|s_t,a_t)}{E}\{R\} = \underset{P(s_{t+1}|s_t,a_t)}{E}\{\sum_{t=0}^{\infty}\gamma^t r_t\}$, or the maximum value function of state-action $Q(\mathbf{X}, \mathbf{A}) \equiv E\{V(\mathbf{X}) \mid \pi\}$.

To avoid a mass calculation to whole MDP state transition, and deal with unknown transition probabilities, there is an efficient RL algorithm named temporal difference learning (TD-learning) [2] to yield the maximum value functions. A TD error $\varepsilon \in R^1$ is defined by Eq. (1), and it is used to update state value function in the learning process, for example, Eq. (2).

$$\varepsilon \equiv r_{t+1} + \gamma V^{\pi}(\mathbf{x}_{t+1}) - V^{\pi}(\mathbf{x}_t) \qquad (1)$$

$$V^{\pi}(\mathbf{x}_t) \leftarrow V^{\pi}(\mathbf{x}_t) + \alpha\varepsilon \qquad (2)$$

where $0 \le \gamma \le 1$ and $0 \le \alpha \le 1$ are a damping rate and a learning rate, respectively.

When action value is also considered during the state transition, the modification of the state-action value function $Q^{\pi}(s_t, a_t)$ can be used to instead of state value function $V^{\pi}(\mathbf{x}_t)$.

$$Q^{\pi}(s_t, a_t) \leftarrow Q\pi(s_t, a_t) + \alpha(r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t) \qquad (3)$$

$(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ appears in Eq. (3), so this learning rule is named as "sarsa learning" [2].

2.2 An Actor-Critic type Neuro-Fuzzy Reinforcement Learning System

To deal with continuous state space, an actor-critic type neuro-fuzzy reinforcement learning system is proposed by Kuremoto et al. [5] [6]. The system is able to be used as an internal model of an autonomous agent which output a series of adaptive actions in the exploration of the unknown environment. The information processing flow is as follows:

Step 1 Agent observes states x_t from the environment;

Step 2 Agent classifies the inputs into *k* classes $\phi_k(x_t)$ by Fuzzy net;

Step 3 Agent outputs an action according to the value function $A_m(x_t)$ (i.e. Actor);

Step 4 Agent receives rewards from the environment after the action is executed;

Step 5 TD-error, concerning with the state value function $V(x_t)$ in Critic, is calculated;

Step 6 Modify the action value function using TD-error (Eq. (1)).

Step 7 Return to Step 1if the state is not the terminal state, else end the current trail.

For an *n*-dimension input state space $\mathbf{x}(x_1(t), x_2(t),...,x_n(t))$, a fuzzy inference net is designed with a hidden layer composed by units of fuzzy membership functions $B_i^k(x_i(t))$, i.e., Eq. (4), to classify input states.

$$B_i^k(x_i(t)) = \exp\left\{-\frac{(x_i(t) - c_i^k)^2}{2\sigma_i^{k^2}}\right\} \tag{4}$$

Here c_i^k, σ_i^k denotes the mean and the deviation of *i*th membership function which corresponding to *i*th dimension of input $x_i(t)$, respectively.

Let $K(t)$ be the largest number of fuzzy rules, we have Eq. (5) :

$$\text{if } (\, x_1(t) \text{ is } B_1^k(x_1(t)), ,..., \; x_n(t) \text{ is } B_n^k(x_n(t))\,) \text{ then}$$

$$\phi_k(\mathrm{x}(t)) \;=\; \prod_{i=1}^{n} B_i^k(x_i(t)) \tag{5}$$

where $\phi_k(\mathrm{x}(t))$ means the fitness of the rule R^k for an input set $\mathrm{x}(t)$.

To determine the number of membership functions and rules of fuzzy net, a self-organized fuzzy neural network (SOFNN) which is constructed adaptive membership functions and rules driven by training data and thresholds automatically [5] [6]. The self-organizing process of SOFNN is given as follows.

Only one membership function is generated by the first input data (for example, the position of agent) , the value of its membership's center equals to the value of input data, and the value of width of all Gaussian function units is fixed to an empirical value. The number of rule for membership functions is one at first, and the output of the rule R^1 equals to $\phi_1(\mathbf{x}(1)) = \prod_{i=1}^{n} B_i^1(x_i(1))$ according to Eq. (5).

For the next input state $\mathbf{x}(x_1(t), x_2(t),..., x_n(t))$, a new membership function is generated if Eq. (6) is satisfied.

$$\max_s B_{i,s}(x_i(t)) < F \tag{6}$$

Here $B_{i,s}(x_i(t))$ denotes the value of existed membership functions calculated by Eq. (4) and $s = 1,2,..., L_i(t)$ indicates the *s*th membership function with the maximum number $L_i(t)$. *F* denotes a threshold value of whether an input state is evaluated enough by existing membership functions.

A new rule is generated automatically when a new membership function is added according to Eq. (5). Iteratively, the fuzzy net is completed to adapt to input states.

The weighted outputs of fuzzy net are used to calculate the value of states (Critic) and actions (Actor) according to Eq. (7) and Eq. (8), respectively.

$$V(\mathbf{x(t)}) = \frac{\sum_k v_k(t)\phi_k(\mathbf{x}(t))}{\sum_k \phi_k(\mathbf{x}(t))} \tag{7}$$

$$A_j(\mathbf{x(t)}) = \frac{\sum_k w_{kj}(t)\phi_k(\mathbf{x}(t))}{\sum_k w_{kj}(\mathbf{x}(t))} \tag{8}$$

Here v_k, w_{kj} are the weighted connections between fuzzy net rules $\phi_k(\mathbf{x}(t))$ and critic function $V(\mathbf{x}(t)) \in R^1$, actor function $A_j(\mathbf{x}(t)) \in R^m$. A_j denotes the jth action selected by agent according to a stochastic policy Eq. (9), where j = 1, 2, …, m.

$$P\left(a_t = a_j \mid \mathbf{x}(t)\right) = \frac{\exp\left(A_j\left(\mathbf{x}(t)\right)/T\right)}{\sum_m \exp\left(A_m\left(\mathbf{x}(t)\right)/T\right)} \tag{9}$$

Here $T > 0$ is a constant named temperature of Boltzmann distribution.

Actor function $A_j(\mathbf{x}(t)) \in R^m$ belongs to an m-dimension real number space in a goal –exploration problem. When the input space is a continuous Euclidean space, for example, a 2-D plane, the number of states and the number of actions may be infinity. To overcome this problem, the speed of action can be limited to 1 grid (mass) per step, and the direction may be given by the linear combination of discrete action values in 4 directions: $A_{up}(\mathbf{x}(t)), A_{down}(\mathbf{x}(t)), A_{left}(\mathbf{x}(t)), A_{right}(\mathbf{x}(t))$. In fact, we used two bigger Action values to design candidate actions in x axis and y axis. For example, $A_{right}(\mathbf{x}(t))$ in x axis is chosen by Eq. (9), and $A_{up}(\mathbf{x}(t))$ in y axis, then the velocity angle is given by Eq. (10)

$$\theta = \arctan \frac{A_{up}(\mathbf{x}(t))}{A_{right}(\mathbf{x}(t))} \tag{10}$$

The movement of the agent is calculated by adding Eq. (11) from the current coordinates.

$$\left(\frac{A_{right}(\mathbf{x}(t))}{\sqrt{A_{right}(\mathbf{x}(t))^2 + A_{up}(\mathbf{x}(t))^2}}, \frac{A_{up}(\mathbf{x}(t))}{\sqrt{A_{right}(\mathbf{x}(t))^2 + A_{up}(\mathbf{x}(t))^2}}\right) \tag{11}$$

TD learning rule given by Eq. (1) become to Eq. (12) and Eq. (13), where β_v, β_w denote learning rates for connections of Critic and Actor.

$$v_k(t+1) = v_k(t) + \beta_v \varepsilon_{TD}(\mathbf{x(t)})\phi_k(\mathbf{x(t)}) \tag{12}$$

$$w_{kj}(t+1) = w_{kj}(t) + \begin{cases} \beta_w \varepsilon_{TD}(\mathbf{x}(t))\phi_k(\mathbf{x(t)}) & a_t = a_j \\ 0 & otherwise \end{cases} \tag{13}$$

When multiple agents explore an unknown environment at the same time, cooperative exploration, i.e., swarm learning may provide more efficient performance comparing with individual learning. Let $D(\mathbf{x}^p(t), \mathbf{x}^q(t))$ express the Euclidean distance between agent *p* and agent *q*, i.e. Eq. (14).

$$D(\mathbf{x}^p(t), \mathbf{x}^q(t)) = \sqrt{\sum_{i=1}^{n}(x_i^p(t) - x_i^q(t))^2} \tag{14}$$

where $i = 1,2,..n$ is the element number of input vectors.

Then we give a positive reward r_{swarm} to the agent when Eq. (15) is satisfied, or a negative reward r_{swarm} in the opposite.

$$\min\_dis \le D(\mathbf{x}^p(t), \mathbf{x}^q(t)) \le \max\_dis \tag{15}$$

$$r_{\mathbf{t+1}} = r_t \pm r_{swarm} \tag{16}$$

Here $\min\_dis, \max\_dis$, denotes near limit distance, far limit distance, respectively.

2.3 Adoption of Adaptive Learning Rate

Learning rates in Eq. (12) and Eq. (13) are decided empirically as fixed values conventionally. However, they need to be reduced to improve the learning convergence during the iterations of exploration process. It is indicated by Derhami et al. that the visited states should use smaller learning rates, meanwhile, less visited states with larger learning rates to modify the exploration policy [8]. So they proposed an adoptive learning rate (ALR) to balance the exploitation/exploration as follows.

Let Φ_t^k be the accumulation firing strength of rule $\phi^k(\mathbf{x}(t))$, the visit value to the state $\mathbf{x}(t)$ is normalized by Eq. (17).

$$FV\left(\phi^k(\mathbf{x}(t))\right) = \frac{\sum_{k=1}^{K_t} \phi^k\left(\mathbf{x}(t)\right)\Phi_t^k}{\sum_{k=1}^{K_t} \Phi_t^k} \tag{17}$$

where $\Phi_t^k = \Phi_{t-1}^k + \phi^k(\mathbf{x}(t))$, $\Phi_{t=0}^k = 0$, $0 \le FV\left(\mathbf{x}(t)\right) \le 1$.

The ALR then is given by Eq. (18).

$$\alpha_t^{ALR}(\phi^k(\mathbf{x}(t))) = \min\left(\frac{\alpha_{\min} K_t}{FV(\mathbf{x}(t))}, \alpha_{\max}\right) \tag{18}$$

where K_t is the number of fuzzy rules, $\alpha_{\min}, \alpha_{\max}$ are the boundaries of ALR.

2.4 A Sarsa Learning type Neuro-Fuzzy Reinforcement Learning System

When a RL agent is in a partially observable Markov decision Process (POMDP), there may be different adaptive actions need to be output though the observed states are the same. Actor-critic type neuro-fuzzy reinforcement learning system was hard to find the optimal solution in POMDP [7]. So a sarsa learning (SL) type neuro-fuzzy reinforcement learning system is proposed recently [7].

The mathematical description of Fuzzy net is as same as which in actor-critic learning type system. Two steps states are observed in sarsa learning algorithm (See sarsa learning rule Eq. (3)), and these state-action value functions are given by following:

$$Q(\phi(\mathbf{x}(t)), a_j, \mathbf{w}_j^{Q_t}) = \frac{\sum_k w_{kj}^{Q_t} \phi^k(\mathbf{x}(t))}{\sum_k \phi^k(\mathbf{x}(t))} \tag{19}$$

$$Q(\phi(\mathbf{x}(t+1)), a_j, \mathbf{w}_j^{Q_{t+1}}) = \frac{\sum_k w_{kj}^{Q_{t+1}} \phi^k(\mathbf{x}(t+1))}{\sum_k \phi^k(\mathbf{x}(t+1))} \tag{20}$$

where $k = 1,2,..K^t$ is the number of fuzzy rules, $j = 1,2,..m$ is action number.
The learning rule of QL becomes to:

$$w_{kj}^{Q_t} \leftarrow w_{kj}^{Q_t} + \begin{cases} \alpha_t^{ALR} \varepsilon^{sarsa} \phi_k(\mathbf{x}(t)) / \sum_{k=1}^{K} \phi_k(\mathbf{x}(t)) & a_t = a_j \\ 0 & otherw \end{cases} \tag{21}$$

$$w_{kj}^{Q_{t+1}} \leftarrow w_{kj}^{Q_t+1} + \begin{cases} \alpha_{t+1}^{ALR} \varepsilon^{sarsa} \phi_k(\mathbf{x}(t+1)) / \sum_{k=1}^{K} \phi_k(\mathbf{x}(t+1)) & a_{t+1} = a_j \\ 0 & otherw \end{cases} \tag{22}$$

where $\alpha_t^{ALR}, \alpha_{t+1}^{ALR}$ are adaptive learning rates described by Eq. (17) and Eq. (18), TD error is given by Eq. (23).

$$\varepsilon_{TD}^{Q_{t+1}} = r_t + \gamma\, Q(\phi(\mathbf{x}(t+1)), a_{t+1}, \mathbf{w}_{t+1}^{Q_{t+1}}) - Q(\phi(\mathbf{x}(t)), a_t, \mathbf{w}_t^{Q_t}) \tag{23}$$

3 Simulation Experiments

To confirm how ARL can benefit the learning performance of neuro-fuzzy type reinforcement learning systems, a goal-exploration problem was used to in the simulation experiment (Fig. 1). The problem is assumed that agents explore a goal area in an unknown 2-D square surrounded by walls. All candidate actions may be chosen before the learning processes, and RL algorithms aforementioned find adaptive actions of agent to explore the goal or exploit its experience to move to the goal. Adaptive actions mean that agents choose to avoid obstacles, other agents and wall, and find the shortest path from their start positions to the goal area.

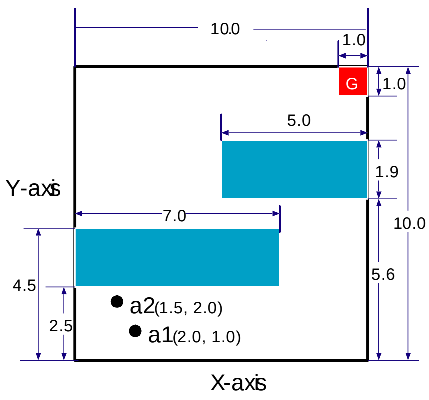
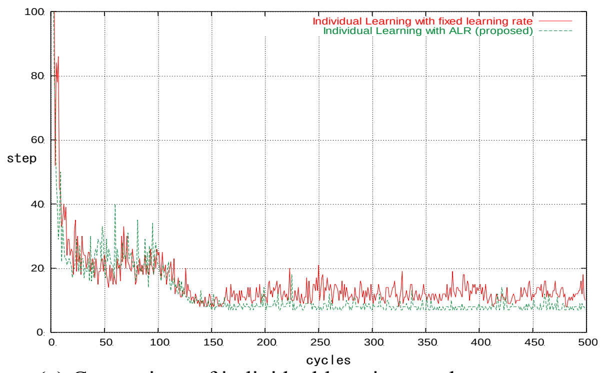
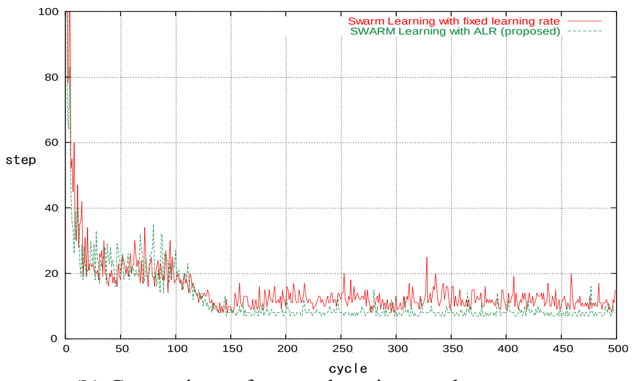


Fig. 1. Simulation environments for goal-directed exploration experiment.



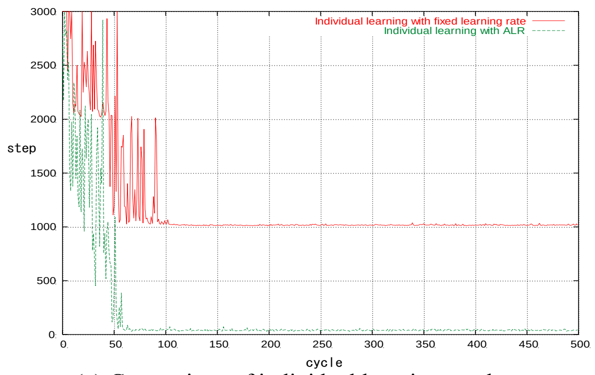
(a) Comparison of individual learning results



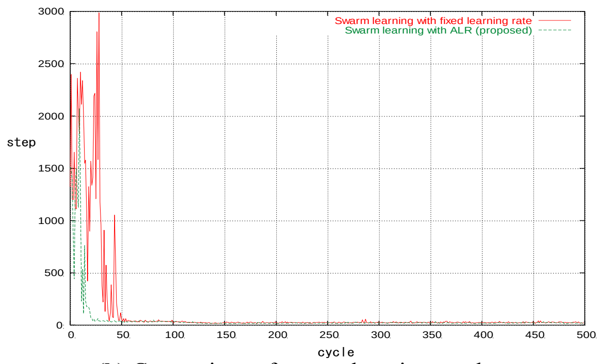
(b) Comparison of swarm learning results

Fig. 2. Learning curves of different learning methods by the actor-critic type system.

Agents observed the state of environment by the information of its position (x, y) in Simulation I, and their vicinities in 4 directions: up, down, left, right in Simulation II. In the last case, if the one step ahead is a path that an agent can arrive it by one time motion, then the value of the direction is 0, oppositely, if there is a wall, or other agent (s), or an obstacle, then the input is 1. So the number of states in the environments of Simulation I is $x \times y$ in a grid scale, and more in the continuous space. In the case of Simulation III, the number of states is 4 bits, 16 states (i.e., 0010, 0011, …).



(a) Comparison of individual learning results



(b) Comparison of swarm learning results

Fig. 3. Learning curves of different learning methods by the sarsa learning system

3.1 Simulation I: A Maze-like Environment

An exploration environment is shown in Fig. 1. There were walls on the four sides, obstacles in the exploration area, and a goal area was fixed. Two agents a1 and a2 start from (2.0, 1.0) and (1.5, 2.0) to find the goal area between (9.0, 9.0) to (10.0, 10.0). Each agent observed its position and position of others in the square, and moved 1.0 length per step toward an arbitrary direction. The method to decide the direction of one time movement was to choose 2 orthogonal directions whose value of action function were higher than others. According to Eq. (10) and Eq. (11) the action direction and its value were calculated. Agents did not have any information of the goal position before they arrived at it.

When the distance rewards defined by Eq. (14) – Eq. (16) were not considered, 2 agents learned to search the goal as fast as they can individually (*individual learning*);

meanwhile, when agents used the swarm rewards in the same environment to modify their policies, the learning was called *swarm learning*.

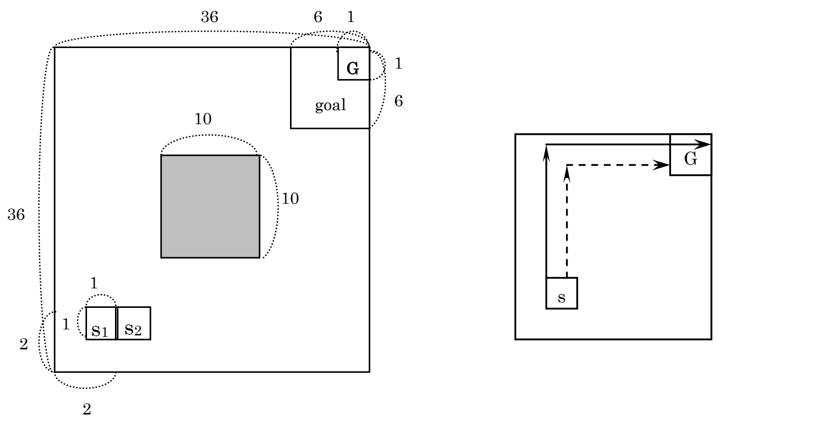
Actor-critic type neuro-fuzzy reinforcement learning system described in Section 2.2 was used as a conventional system [5] [6]. When adaptive learning rate (ALR) shown in Section 2.3 was adopted in, the change of the learning performance was investigated.

The values of parameters used in these simulations were shown in Table 1. The number of learning iterations (cycles) for one simulation was set to be 500. In one trial (cycle), i.e., an exploration from the start to the goal, the limitation steps of exploration was 5,000. Because the stochastic property of RL, the simulation results were given by the averages of 5 simulations.

Table 1. Parameters used in Simulation I and Simulation II

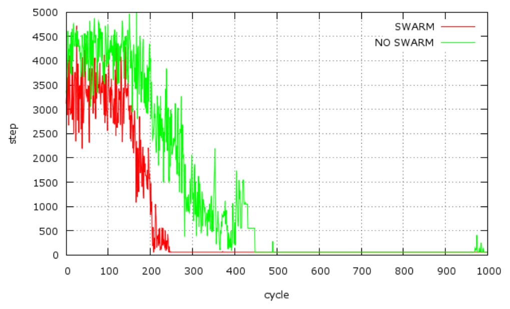
Description	Symbol	Quantity
Dimension of input vector	n	2
The number of actions	m	2
Standard deviation of membership	σ_i^k	0.1
Threshold of fitness	F	0.4
Initial weight between rules and critic	v_k	1.0
Initial weight between rules and actor	w_{kj}	0.25
TD learning coefficient for critic	β_v, β_v^{ALR}	0.3, (0.001-0.6)
TD learning coefficient for actor	β_w, β_w^{ALR}	0.3, (0.001-0.6)
Discount of TD error	γ	0.9
Temperature of Bolzmann distribution	T	2.0 (0.2 after 100 cycles)
Reward for goal arrived	r_{goal}	1000.0
Reward for wall or agent crashed	r_o	-10.0
Reward for corner crashed	r_c	-20.0
Reward for swarm formed	r_{swarm}	0.5
Reward for swarm unformed	$r_{a\text{-}swarm}$	$-\lvert dis \rvert$
Minimum distance between agents	$\min\_dis$	1.0
Maximum distance between agents	$\max\_dis$	3.0

When we confirmed the learning curves of these different learning methods, not only swarm learning with ALR but also individual learning with ALR showed their prior learning performance comparing with the conventional fixed learning rate. The average results are shown in Fig. 3. In detail, the average lengths of the path explored by agents were 1182.01 using fixed learning rate and individual learning, 41.62 using ALR and individual learning, 26.61 using fixed learning rate and swarm learning, 22.64 using ALR and swarm learning calculated by the last 100 cycles data (from cycle 400 to cycle 500).

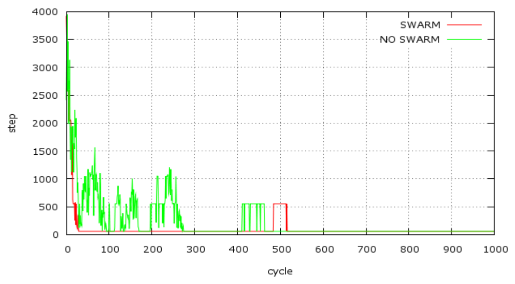


(a) Exploration environment **(b)** Minimum path (broken arrow lines) and quasi-minimum path (solid arrow lines)

Fig. 4. A goal-exploration problem for sarsa learning type neuro-fuzzy reinforcement learning system.



(a) Using a fixed learning rate $\alpha = 0.03$



(b) Using ALR $\alpha_t^{ALR} \in [0.001, 0.3]$

Fig. 5. Learning curves of different learning methods by the sarsa learning system.

3. 3 Simulation II: A POMDP Environment

An exploration environment with an obstacle as shown in Fig. 4 (a) was used in Simulation II. The sarsa learning type neuro-fuzzy reinforcement learning system was adopted as the action learning method of autonomous agents. Two agents were input in 4 dimensions, i.e., up, down, left, and right directions with values 0 and 1. The actions were also in 4 directions 1 grid/step ($a_j, j = 1,2,3,4$). The optimal path of the agents should be a transition of a same state (0, 0, 0, 0) even from the start position (2, 2) and (3, 2) to the goal area from (30, 30) to (36, 36) avoiding to crash the 10x10 obstacle in the center of the exploration square. So it was an environment under partially observable Markov decision process (POMDP) and the optimal solution was difficult to be found, alternatively, quasi-optimal solution was available as shown in Fig. 4 (b).

The dramatic improvement for the learning performance by ALR was confirmed with the learning curves of sarsa learning type neuro-fuzzy reinforcement learning system dealing with the POMDP problem as shown in Fig. 5. Swarm learning effects stand out clearly and ALR accelerated learning convergence efficiently.

4 Conclusions

Adaptive learning rate (ALR) was adopted into neuro-fuzzy reinforcement learning systems in this paper. The concept was founded on the balance management of exploration and exploitation of reinforcement learning process. Higher learning rate serves larger modification of value functions dealing with unexplored states. Simulations of goal-navigated exploration problem showed the effectiveness of the proposed method. And adequate distance between agents also accelerated the exploration process. It is expected to apply these effective RL learning systems to autonomous agent in web intelligence or robots in real environment in the future.

References

1. L. P. Kaelbling, M. L.Littman: Reinforcement Learning: A Survey, J. Artificial Intelligence Research, 4 237—285 (1996)
2. R. S. Sutton, and A. G. Barto: Reinforcement learning: an introduction, The MIT Press, Cambridge (1998)

3 K. Samejima and T. Omori: Adaptive internal state space construction method for reinforcement learning of a real-world agent, Neural Networks, 12, 1143--1155 (1999)

4. X. S. Wang, Y. H. Cheng and J. Q. Yi: A fuzzy Actor–Critic reinforcement learning network, Information Sciences, 177, 3764--3781 (2007)
5. T. Kuremot, M. Obayashi, and K. Kobayashi: Adaptive swarm behavior acquisition by a neuro-fuzzy system and reinforcement learning algorithm, International Journal of Intelligent Computing and Cybernetics, 2(4), 724-744 (2009)
6. T. Kuremoto, Y. Yamano, M. Obayashi, and K. Kobayashi: An improved internal model for swarm formation and adaptive swarm behavior acquisition, Journal of Circuits, Systems, and Computers, 18(8), 1517-1531 (2009)
7. Kuremoto T., Yamano Y. Feng L.-B, Kobayashi K., and Obayashi M.: "A fuzzy neural network with reinforcement learning algorithm for swarm learning", Lecture Notes in Electronic Engineering (LNEE), Vol.144, pp.101-108, 2011.
8. V. Derhami, V. J. Majd, and M. N. Ahmadabadi: Exploration and exploitation balance management in fuzzy reinforcement learning, Fuzzy Sets and Systems, 161(4), 578－595 (2010)