We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

11

A Reinforcement Learning System Embedded Agent with Neural Network-Based Adaptive Hierarchical Memory Structure

Masanao Obayashi, Kenichiro Narita, Yohei Okamoto,
Takashi Kuremoto, Kunikazu Kobayashi and Liangbing Feng
Yamaguchi University
Japan

1. Introduction

In this chapter, a way to realize intellectualization of robots (called "agents" here) is considered. We intend to achieve this by mimicing an intellectual way of human. Human learns many kinds of things from incidents driven by own actions and reflects them on the subsequent action as own experiences. These experiences are memorized in his/her brain and recollected and reused if necessary. In other words, human is not good at doing anything at first, but he/she will memory the meaningful things among his/her own experiences, at the same time oblivious of other memories without understood by him/her. He/She will accumulate knowledge gotten by experiences, and will make use of them when encounters unexperienced things.

The subject in this chapter is to realize the things mentioned above on agents. To be specific, the agent will be equipped with three main functions: "learning" taking out of the meaningful things from through experiences with trial and error, "memorization" memorizing the above meaningful things, and "the ability of associative recollection and its appropriate use" suitable for the situation. Of course, when it doesn't have such appropriate memories, the agent will learn them additively and moreover memorize them as new experiences. Repeating these processes, the agent will be more intellectual. In this intellectualization, there are a few models related to subject mentioned above, e.g., K-series model and their discrete KA–series model (D. Harter *et al.*, 2005 [1]) and DARWIN X-series models (J. L. Krichmar *et al.*, 2005 [2]). K-series and KA-series models have been developped by R. Kozuma and his colleagues. Their models are equipped with chaotic neurodynamics, which is very important to realize the brain model, hippocampal model and supervised learnimg ability. DARWIN X-series models have been developped by Edelman and his colleagues since 1981. Their models are also equipped with hippocampal model of spatial, episodic, and associative meory model. These two series models intend to realize the brain faithfully.

We have studied about this theme since 2006 [3] ~ [8]. This time our proposed model is not necessarily to realize the human brain faithfully, and intends to realize intellectualization of the agent functionally. At first we will introduce "reinforcement learning (RL, Sutton *et al.*, 1998 [9])", as experienced learning through trial and error, which is a learning algorithm

based on calculation of reward and penalty given through mutual action between the agent and the environment, and which is commonly executed in living things.
In the reinforcement learning, memorizing the environment and the agent's action corresponding to it as short term memory (STM), the agent will take out the meaningful thing from them, then it will memorize its refined information as long term memory (LTM). As a medium of this LTM, we will introduce Chaotic Neural Networks (CNNs, Aihara *et al.*, 1997 [10][11]) which is generally acknowledged to be an associative memory model of the brain. The memory structure takes the form of the adaptive hierarchical memory structure so as to deal with the increase of information. The structure consists of CNNs in consideration of the adjustment to non-MDP (Markov Decision Process) environment. When the agent is placed in a certain environment, the agent will search the appropriate experienced information in LTM. In such case of searching, as the mechanism of memory search, we introduce self-organizing maps (SOM, T. Kohonen, 2001 [12]) to find the appropriate experience. In fact, during the agent's exploration of the information adapting to the environment, the time series environmental information is necessary, so, we use the feedback SOM that its output is feedback to input layer to deal with the time series information. The whole structure of the our proposed system is shown in Fig. 1. To show the example of realization of the agent composed of functions mentioned above and the effectiveness of these methods, we carried out the simulation applied to the goal-oriented maze problem shown in Figs. 11,14.
As a result, it was verified that the agent constructed by our proposed idea would work well by making use of the experienced information, refer to Fig. 14, in unexperienced large scale goal-oriented maze problems and it got the goal in just about shortest steps. See Fig. 14.

2. Proposed system structure

The proposed system consists of three parts: memory, learning and discrimination. The memory consists of short-term memory (STM) and long-term memory (LTM). Figure 1 shows these overall structure.

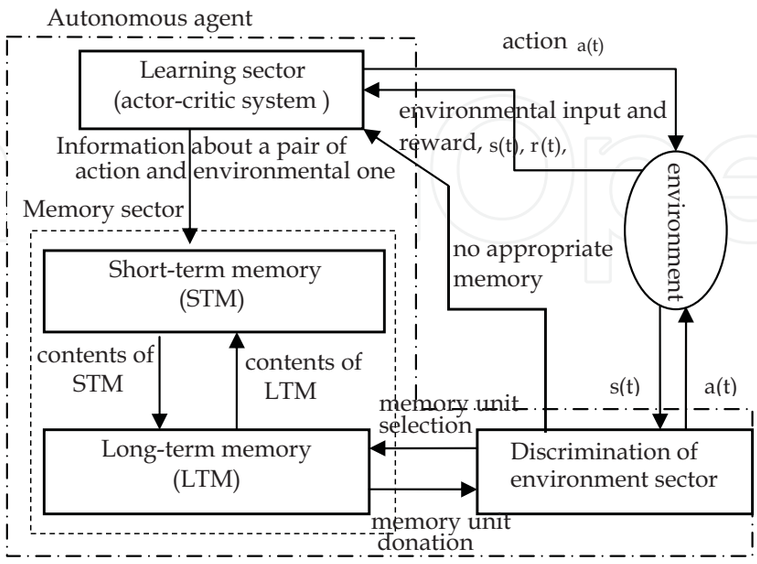


Fig. 1. Structure of the proposed RL embedded agent with adaptive hierarchical memory

Learning sector: actor-critic system is adopted. It learns the choice of appropriate actions to maximize the total predictive rewards obtained over the future considering the environmental information *s(t)* and reward *r(t)* as a result of executing action *a(t)*. Memory sector: memory setor consists of short-term-momory (STM) and long-term momory (LTM). Here, STM: it memorizes the learned path of the information (environmental information and its corresponding action) obtained in Learning sector. Unnecessary information is forgotten and only useful information is stored. LTM: it memorizes only the enough sophisticated and useful experience in STM. Environment discrimination sector: environment discrimination sector consists of initial operation part and environment discrimination part. This sector plays the role that the agent examines the environment through the agent's own behaviors and selects the memorized infromation in LTM corresponding to the current environment.

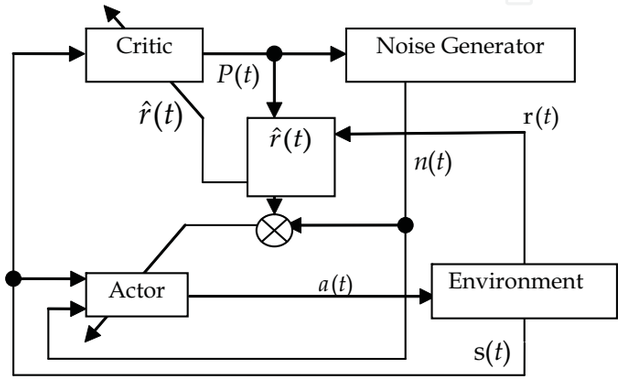


Fig. 2. The construction of the actor-critic system

3. Actor-critic reinforcement learning

Reinforcement learning (RL, Sutton *et al.*, 1998 [9]), as experienced learning through trial and error, which is a learning algorithm based on calculation of reward and penalty given through mutual action between the agent and the environment, and which is commonly executed in living things. The actor-critic method is one of representative reinforcement learning methods. We adopt it because of its flexibility to deal with both continuous and discrete state-action space environment. The structure of the actor-critic reinforcement learning system is shown in Fig. 2. The actor plays a role of a controller and the critic plays role of an evaluator in control field. Noise plays a part of roles to search the optimal action.

3.1 Structure and learning of critic

3.1.1 Structure of critic

Figure 3 shows the structure of the actor. The function of the critic is calculation of *P(t)*: the prediction value of sum of the discounted rewards that will be gotten over the future. Of course, if the value of *P(t)* becomes bigger, the performance of the system becomes better. These are shortly explained as follows:

The sum of the discounted rewards that will be gotten over the future is defined as *V(t)*.

$$V(t) \equiv \sum_{l=0}^{\infty} \gamma^n \cdot r(t+l), \tag{1}$$

where γ (0≤γ<1) is a constant parameter called discount rate.

Equation (1) is rewritten as

$$V(t) = r(t) + \gamma V(t+1). \tag{2}$$

Here the prediction value of *V(t)* is defined as *P(t)*.
The prediction error $\hat{r}(t)$ is expressed as follows:

$$\hat{r}(t) = \hat{r}_t = r(t) + \gamma P(t+1) - P(t). \tag{3}$$

The parameters of the critic are adjusted to reduce this prediction error $\hat{r}(t)$. In our case the prediction value *P(t)* is calculated as an output of a radial basis function neural network (RBFN) such as,

$$P(t) = \sum_{j=0}^{J} \omega_j^c y_j^c(t), \tag{4}$$

$$y_j^c(t) = \exp\left[-\sum_{i=1}^{n} (s_i(t) - m_{ij})^2 / \sigma_{ij}^2 \right]. \tag{5}$$

Here, $y_j^c(t)$: jth node's output of the middle layer of the critic at time t, ω_j^c : the weight of jth output of the middle layer of the critic, $s_i(t)$: i th state of the environment at time t, m_{ij} and σ_{ij} : center and dispersion in the i th input of j th node basis function, respectively, J : the number of nodes in the middle layer of the critic, n : the number of the states of the system (see Fig. 3).

3.1.2 Learning of parameters of critic

Learning of parameters of the critic is done by using commonly used back propagation method which makes prediction error $\hat{r}(t)$ go to zero. Updating rule of parameters are as follows:

$$\Delta\omega_i^c = -\eta_c \cdot \frac{\partial \hat{r}_t^2}{\partial \omega_i^c}, \quad (i = 1, \cdots, J). \tag{6}$$

Here η_c is a small positive value of learning coefficient.

3.2. Structure and learning of actor

3.2.1 Structure of actor

Figure 4 shows the structure of the actor. The actor plays the role of controller and outputs the control signal, action $a(t)$, to the environment. The actor basically also consists of radial basis function networks. The jth basis function of the middle layer node of the actor is as follows:

$$y_j^a(t) = \exp\left[-\sum_{i=1}^{n} (s_i(t) - m_{ij})^2 / \sigma_{ij}^2 \right], \tag{7}$$

$$a(t) = u_k(t) = \sum_{j=1}^{J} \omega_{kj} y_j^a(t) + n(t), \quad (k = 1, \cdots, K). \tag{8}$$

Here y_j^a : jth node's output of the middle layer of the actor, m_{ij} and σ_{ij} : center and dispersion in ith input of jth node basis function of the actor, respectively, K: the number of the actions, $n(t)$: additive noise, u_k : representative value of kth action, ω_{kj} : connection weight from jth node of the middle layer to kth output node. The action selection method to choose the representative u_k among all the candidates of actions is described at section 3.3.

3.2.2 Noise generator

Noise generator let selection of the output of the actor have diversity by making use of the noise. It comes to realize the learning of the trial and error according to the results of performance of the system by executing the selected action. Generation of the noise *n(t)* is as follows:

$$n(t) = n_t = noise_t \cdot \min\left(1, \exp(-P(t))\right), \tag{9}$$

where $noise_t$ is uniform random number of $[-1, 1]$, min $(\cdot)$: minimum of $\cdot$. As the $P(t)$ will be bigger (this means that the selected action goes close to the optimal action), the noise will be smaller. This leads to the stable learning of the actor.

3.2.3 Learning of parameters of actor

Parameters of the actor, ω_{kj}^a $(k = 1,...K, j = 1,\cdots, J)$, are adjusted by using the results of executing the output of the actor, i.e., the prediction error $\hat{r}_t$ and noise. k is the number of the selected and executed actions at the previous time.

$$\Delta\omega_{kj}^a = \eta_a \cdot n_t \cdot \hat{r}_t \cdot \frac{\partial u_k(t)}{\partial \omega_{kj}^a}. \tag{10}$$

$\eta_a (> 0)$ is the learning coefficient. Equation (10) means that $(-n_t \cdot \hat{r}_t)$ is considered as an error, ω_{kj}^a is adjusted as opposite to sign of $(-n_t \cdot \hat{r}_t)$. In other words, as a result of executing $u_k(t)$, e.g., if the sign of the additive noise is positive and the sign of the prediction error is positive, positive additive noise is sucess, so the value of ω_{kj}^a should be increased (see Eq. (8)), and vice versa.

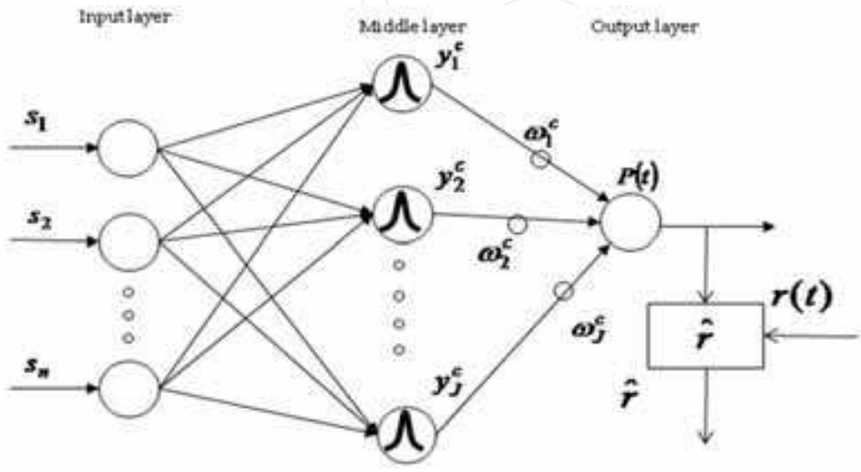


Fig. 3. Structure of the critic

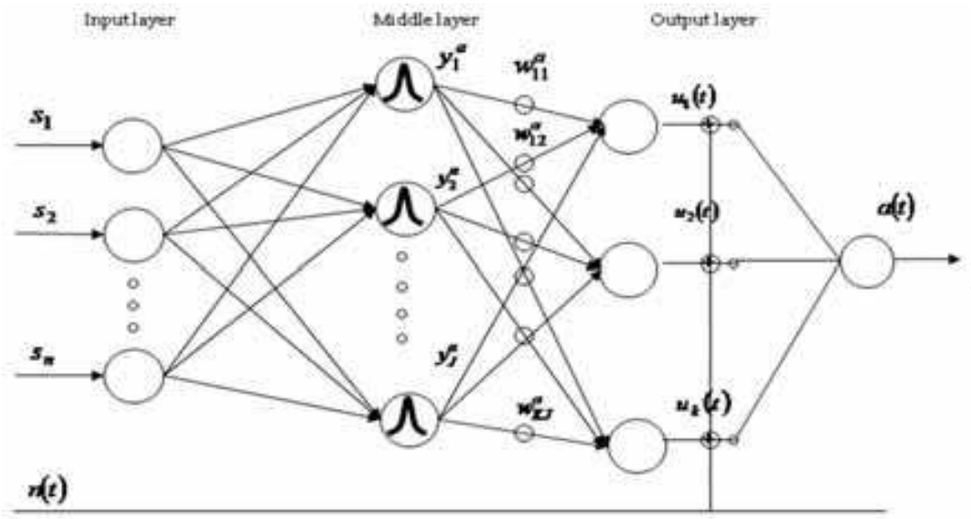


Fig. 4. Structure of the actor

3.3 Action selection method

The action u_b at time t is selected stochastically using Gibbs distribution Eq. (11).

$$P\left(u_b \mid \mathbf{s}(t)\right) = \frac{\exp\left(u_b(t)/T\right)}{\sum_{k=1}^{K} \exp\left(u_k(t)/T\right)} . \tag{11}$$

Here, $P\left(a_b \mid \mathbf{s}(t)\right)$: selection probability of b th action u_b, T : a positive constant called temperature constant.

4. Hierarchical memory system

4.1 Associative Chaotic Neural Network (ACNN)

Chaotic Neural Network (CNN) has been developped by Aihara *et al.*, 1997 [10][11]), which is generally acknowledged to be an associative memory model of the brain. CNN is constructed with chaotic neuron models that have refractory and continuous output value. Its useful usage is as an associative memory network named ACNN. The followings are the dynamics of ACNN.

$$x_i(t+1) = f(v_i(t+1) + z_i(t+1)) , \tag{12}$$

$$v_i(t+1) = k_r \cdot v_i(t) - \alpha \cdot x_i(t) + a_i , \tag{13}$$

$$z_i(t+1) = k_f \cdot z_i(t) + \sum_{j=1}^{L} \omega_{ij} x_j(t) , \tag{14}$$

$$\omega_{ij} = \frac{1}{U} \sum_{p=1}^{U} (x_i^p \cdot x_j^p) . \tag{15}$$

$x_i(t)$: output of the ith neuron at step t, $v_i(t)$: internal state with respect to refractory of the ith neuron at step t, $z_i(t)$: internal state of the ith neuron with respect to mutual operation at step t, $f(\cdot)$: sigmoid function, ω_{ij} : connection weight from jth neuron to ith neuron, x_i^p : ith element of pth stored pattern, k_r : damping coefficient on refractory, k_f : damping coefficient on feedback, α : constant parameter, a_i : compound parameter with threshold and external input of ith neuron, $i, j = 1, \cdots, L$, L : the number of neurons in the CNN, U : the number of the stored patterns.

4.2 Network control

The dynamics of ACNN behaves chaotically or no-chaotically according to the value of the damping coefficient on refractory k_r. We would like the network to behave chaotically at first and to converge to one of the stored patterns when the state of the network becomes close to one of the stored patterns. Here, to realize this, we define network control as the control which makes transition of network from chaotic state to non-chaotic one by changing of the specified parameter k_r and vice versa. The network control algorithm of ACNN is shown in Fig. 5. The change of states of ACNN is defined by $\Delta x(t)$, total change of internal state $x(t)$ temporally, and when $\Delta x(t)$ is less than a predefined threshold value θ, the chaotic retrieval of ACNN is stopped by changing values of the parameter k_r into small one. As a result, the network converges to a stored pattern near the current network state.

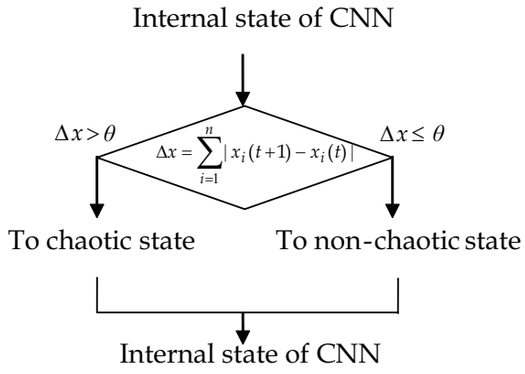


Fig. 5. Flow of the network control algorithm

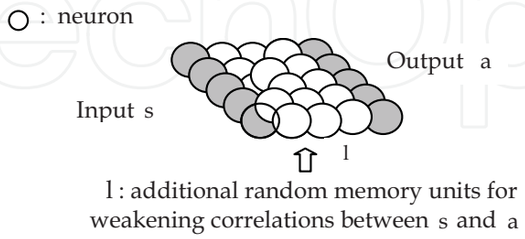


Fig. 6. Memory configuration of ACNN

4.3 Mutual associative type ACNN (MACNN)

4.3.1 Short-Term Memory (STM)

We make use of ACNN as a mutual associative memory system, called MACNN, namely, auto-associative memory matrix W_s is constructed with environmental inputs $s(t)$ and their

corresponding actions *a(t)* (refer to Fig. 6) . When *s(t)* is set as a part of the initial states of the ACNN, the ACNN retrieves *a(t)* with *s(t)* and *l(t)*, using the way of the described operation at 4.2. l is a random vector to weaken the correlation between *s(t)* and *a(t)*. The update equation of the memory matrix W_s is described as Eq. (16), here, λ_s is a forgetting coefficient, and η_s is a learning coefficient. λ_s is set to small, because that at the initial and middle learning stage W_s is not important. In case that these **s**, **l**, a are applied to MACNN, i.e., Eqs. (12) to (15), **s**, **l**, a are corresponding to $x_i(t)(i=1,..., L)$ through Eq. (15), its matrix type, Eq. (16).

$$W_S^{new} = \lambda_s \cdot W_S^{old} + \eta_S \left[\mathbf{s}^T \quad \mathbf{l}^T \quad a \right]^T \left[\mathbf{s}^T \quad \mathbf{l}^T \quad a \right] . \tag{16}$$

STM as one unit consists of plural MACNNs, and one MACNN memorizes information for one environmental state and action patterns (see Fig. 7). For example, STM has path information from start to goal on only one maze searching problem.

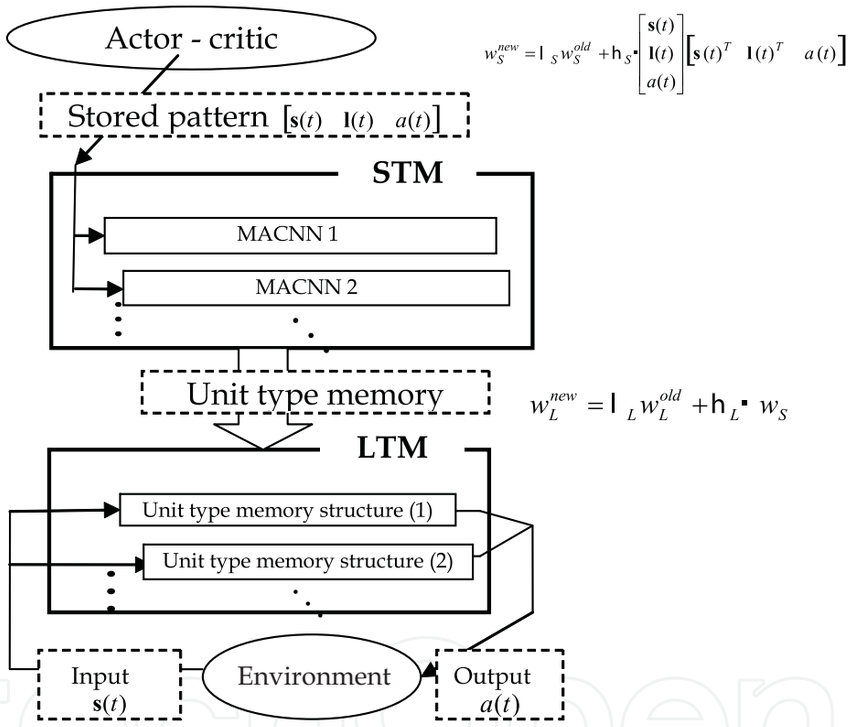


Fig. 7. Adaptive hierarchical memory structure

4.3.2 Long-Term Memory (LTM)

LTM consists of plural units. LTM memorizes enough refined information in STM as one unit (refer to Fig. 7). For example, when actor-critic learning has accomplished for a certain maze problem, information in LTM is updated as follows: In case that the current maze problem has not been experienced, the stored matrix W_L is set by Eq. (17) :

$$W_L = W_S . \tag{17}$$

In case that the current maze has been experienced and present learning is additive learning, the stored matrix is updated by Eq. (18);

$$W_L^{new} = \lambda_L \cdot W_L^{old} + \eta_L W_S \,. \tag{18}$$

λ_L is a forgetting coefficient, and η_L is a learning coefficient. λ_L is set to large value as same as one of η_L so as not to forget previous stored patterns.

4.4 Adaptive hierarchical memory structure

Fig. 7 shows the whole configuration of the adaptive hierarchical memory structure. When an environmental state is given to the agent, at first it is sent to LTM for confirming whether it already exists in the memory or not. If it is the same as the stored information, the recalled action corresponding to it is executed, otherwise, it is used to learn at the actor-critic system. After learning the pair of the enough refined and trained environmental state s and action a in STM is sent to LTM to be stored. If it comes to be different from the stored pattern on the way to use, information about it in LTM is used to relearn at the actor-critic system in STM.

5. Discrimination of the environment

The information that the agent got through its own experienced and momorized is used to discriminate whether it is applicable to the current environment, or not. In this section, the structure of the environment discrimination and how to discriminate it are explained. The discrimination of environment is composed of the initial operation part and the memory selection part.

5.1 Initial operation

To decide whether the agent has the momory corrresponding to the current environment, the agent behaves with next features,

i. The agent behaves predefined n_{init} steps randomly without use of its own memory.
ii. The agent behaves according to two rules: One is that the agent does not return back to the paths which the agent passed during this initial operation, the other is that the agent does not strike the wall. These rules make the speedy search and collection of the information of the agent possible.

5.2 Discrimination of the environment using feedback SOM

The agent discriminates the environment by the feedback SOM. The feedback SOM consists of three layers: input layer, competition layer and output feedback layer. The sturucture of the feedback SOM is shown in Fig. 8. At first the agent investigates the evironment by executing the initial operation. In the initial opereation, during the n_{init} steps, the winner occurs every each steps, i.e., the number of n_{init} comes into winners. Using these data, the agent discriminates the environment. Concretely the agent gets these data for all the environment the agent faced and memorizes the time series of winners. When the agent is placed at the undiscriminated situation, the agent begins the initial operation and gets the above data and compares them with memorized data that is refined about specified environment through the actor-critic learning. If two data agree, after then the agent behaves using the memorized data, especially, action. In the opposite case, the agent begins learning about the current environment using the actor-critic system. The algorithm of the feedback SOM to get the time series data of the winners for each step of the initial operation about the environment is as follows:

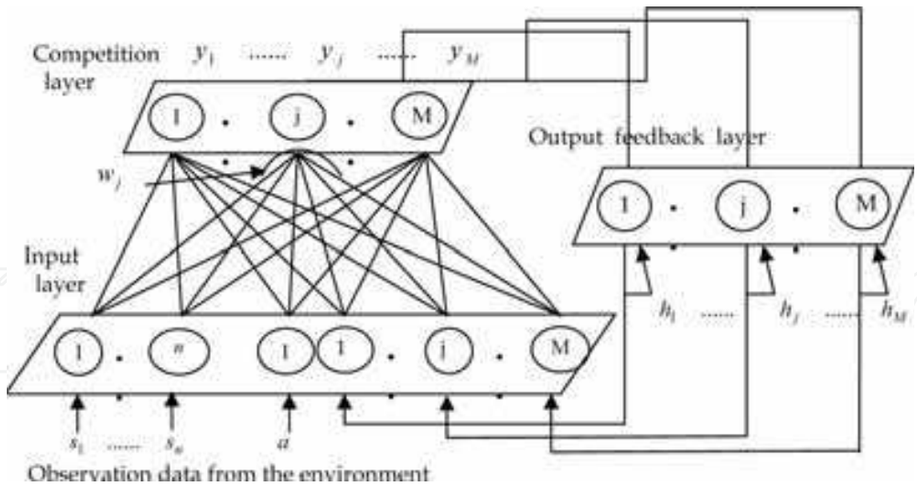


Fig. 8. The used feedback SOM

Algorithm of the feedback SOM

Step 1. Set random small values to the connenction weights

$$w_{ji}(j=1,\cdots,M, i=1,\cdots,n+M)\,.$$

Step 2. Give the input signal to the input layer as follows:

$$\begin{aligned} I(t) &= \{\mathbf{s}(t), a(t); \beta\mathbf{h}(t-1)\} \\ &= \{s_1(t),\cdots,s_n(t), a(t); \beta h_1(t-1),\cdots,\beta h_M(t-1)\}, \end{aligned} \tag{19}$$

where $I(t)$: input vector at time t, $s_i(t)$: ith observation data from environment, $h_j(t)$: feedaback data from competition at time, β is a positive constant representing the rate of considering the history information, n : the number of environmental states, M : the number of outputs of the competition layer.

Step 3. Calculate the distance d_j between the input vector and all the neurons in the competitive layer at time t.

$$d_j = \sqrt{\sum_{i=1}^{n+1+M} (I_i(t) - w_{ji})^2}\,, \quad (j=1,\cdots,M). \tag{20}$$

Step 4. Find the neuron j^* (called the winner neuron) which has the smallest distance d_{j^*} and calculate y_i as follows :

$$\begin{aligned} j^* &= \arg\min_{1\le j\le M} d_j \\ y_j(t) &= \begin{cases} 1, & j = j^*. \\ 0, & j \ne j^* \end{cases} \end{aligned} \tag{21}$$

Step 5. Calculate the output of neurons in the output feedback layer as follows :

$$h_j(t) = (1-\gamma)y_j(t) + \gamma h_j(t-1)\,, \tag{22}$$

where γ is a positive constant retaining the past information.

Step 6. Update the values of connection weights of the winner neuron and around it as follows :

$$w_j(k) = w_j(k-1) + \eta\Lambda(j, j^*)\{I(t) - w_j(k-1)\}$$
$$\Lambda(j, j^*) = \exp(-\frac{||j - j^*||}{\sigma^2}) \quad , \tag{23}$$

Step 7. where $w_j(k)$: jth connection weight vector in the competition layer, η is a positive learning coefficient, k is repetition number of renewal of the weights, and σ is deviation from the center and then σ become smaller according to progress of the learning.

Step 8. Repeat Step 2 to Step 6 until the predefined times is over.

5.3 Selection of the memorized information corresponding to the current environment

Figure 9 shows the flow of the memorized environment selection in the case of n_{init} = 5. In the figure, during n_{init} = 5 steps, the number 1 and number 3 of the memorized environments were selected three times at time t, t-3, t-4 and two times at time t-1, t-2, respectively.

Threshold set algorithm of the memorized environment selection

Step 1. After learning of the feedback SOM, give the environment to the feedback SOM again to decide the value of threshold.

Step 2. Repeat the initial operation ($=n_{init}$ steps) n_{repeat} times and get the data of $n_{init} \times n_{repeat}$ neurons which won.

Step 3. Record the number of firing (winning) times of each neuron in the neurons of competition layer in $n_{init} \times n_{repeat}$ data.

Step 4. Repeat from Step 1 to Step 3 until finishing of records of above firing times for all environments.

Step 5. Fix the threshold (*threshold_win*) to decide whether the winner neuron corresponding memorized environment is adopted or not as a winner neuron.

Selection algorithm of the environment in the memorized environments in LTM

Step 1. Put the agent on the start point in the environment.

Step 2. Start the initial operation (n_{init} steps) and get the information of the observation and action at the each 1 step operation.

Step 3. Decide the winner neuron by the above information for each step.

Step 4. Calculate each total number of winner neurons which are corresponding to each memorized environment as follows:

Comparison the winner neuron in the current environment with the winner neuron in the memorized environment.

if ($win_{i,win\_neuron} > threshold\_win$)

$count_i = count_i + 1$;

win_{ij} : firing number of jth neuron of the ith memory

threshold_ win: threshold to decide whether the neuron is adopted or not as a winner neuron

$count_i$: total number of winner neurons corresponding to the ith memorized environment

Step 5. Repeat from Step 3 to Step 4 until the agent finishes the initial operation of n_{init} steps.
Step 6. Select the maximum count for each memorized environment
Step 7. Distinguish whether the ith memorized environment with the selected count is able to correspond to the current environment by next process:

```
if (count_i ≥ threshold_count)
    the agent may have the memory corresponding to the current
    environment, go to Step 8,
else
    the agent may not have the memory corresponding to the current
    environment, go to learning process.
    threshold_count: threshold to distinguish whether the selected
    memorized environment is adopted or not
```

Step 8. Request the tender of MACNN unit corresponding to the current environment.
Step 9. Start the recollection by the MACNN unit.
Step 10. When the behavior by recollection using MACNN failed, that is, the agent goes into the wall or goes over the predefined steps before arrival of the goal, Go Step 2.

Note: In the simulation of the next section, n_{init}, n_{repeat}, $threshold\_win$ and $threshold\_count$ are set to 5, 30, 10, 4, respectively

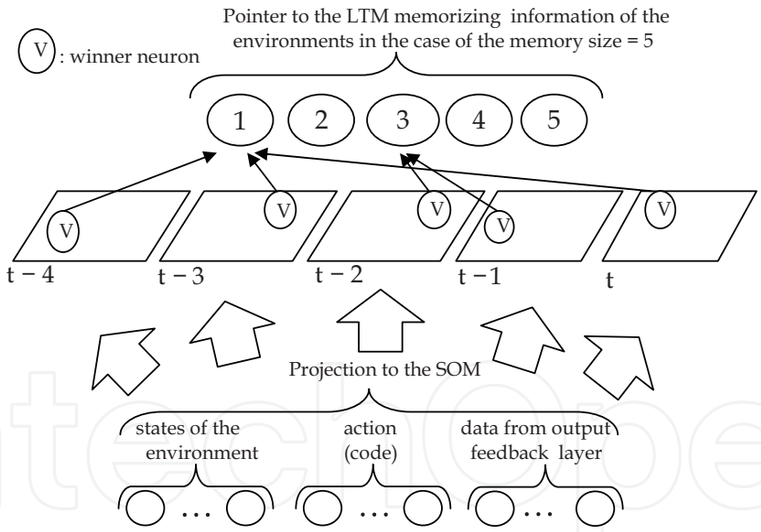


Fig. 9. Flow of the memorized environment selection in the case of n_{init}=5

6. Simulation

In this study, our proposed method is applied for the agent to find, memorize, recollect and reuse the optimal paths of the plural small and large scale mazes.

6.1 Simulation condition

An agent can perceive whether there is aisle or not at the forward, right-forward, left-forward, right, and left as the state ***s*** of the environment (refer to Fig. 10). An agent can

move 1 lattice to forward, back, left, and right as action a (see Table 1). Therefore in actor-critic, a state $\boldsymbol{s}$ of the environment consists of 20 inputs (n = 5 directions ×4 lattice in Fig. 10) in Fig. 8. The content of an input is defined as the distance from the agent to wall, and has value of 1, 2, 3, and 4. In the case that there is a wall next to the agent, the content of input is value 1, and so on. The number of kinds of action a is 4 (= K in Fig. 4). The number of hidden nodes of RBFN is equal to 32 (=J) in Fig. 3 and 4. And the number of units l is equal to 21 in Fig. 6. When the agent gets the goal, it is given the reward, 1.0. For the case of a collision with wall, reward is -1.0, and for each action except for collision is - 0.1. Flow of the whole algorithm of the simulation is shown in Table 2.

	up	down	left	right
code	1000	0100	0010	0001

Table 1. Action and its code taken by the agent

<table>
<tr><td colspan="3">Flow of the whole algorithm of the simulation</td></tr>
<tr><td colspan="3">Step 1 : Set the maze to the agent.</td></tr>
<tr><td colspan="3">Step 2 : Begin the initial operation (n_{init}. steps).</td></tr>
<tr><td colspan="3">Step 3 : Distinguish whether the memorized environment selected by the result of the initial opereation is adopted or not.</td></tr>
<tr><td colspan="2">In the case of existence of the appropriate memorized environment</td><td>In the case of absence of the appropriate memorized environment</td></tr>
<tr><td colspan="2">Step4 : Start the behavior using the selected unit memory in LTM.</td><td>Step 4 : Switch to learn the maze by actor-critic method.</td></tr>
<tr><td>Arrival to the goal</td><td>Failure of recollecction</td><td rowspan="2">Step 5 : Switch to learn the MACNN and the feedback SOM by use of the data of learning sector (Step 4).</td></tr>
<tr><td>Step 5 : End of the process</td><td>Step 5 : Go back to Step 2</td></tr>
<tr><td colspan="2">Step 6 : Go back to Step 1</td><td>Step 7 : Go back to Step 1.</td></tr>
</table>

Table 2. Flow of the whole algorithm of the simulation

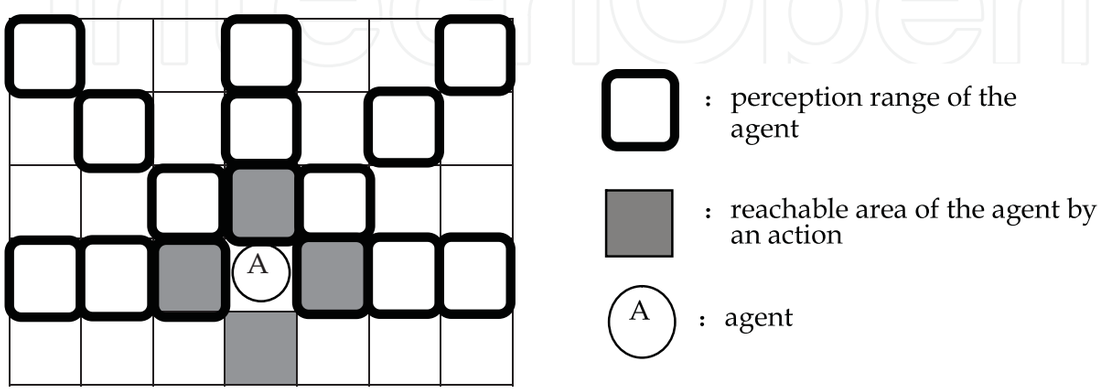


Fig. 10. Ability of perception and action of the agent

6.2 Parameters used in the simulation

Parameters used in this simulation are shown in Table 3-4. These parameters are decided by trial and error. The number of mutual retrieval Systems(MACNNs) is 11 in STM and call this layers 1
unit memory structure (see Fig. 7). LTM has 4 units type memory structure to memorize 4 mazes (see Figs. 7 and 11). Table 1 shows the number of kinds of actions and their codes taken by the agent.

Parameters used in actor-critic			
σ . width coefficient	0.1	η_a . learning coefficient	0.7
η_c . learning coefficient	0.7	γ . discount rate	0.85
T . temperature coefficient	0.4 (within 3steps)	T . temperature coefficient	0.1 (more than 3 steps)
Forgetting and Learning coefficients used in memory sector			
λ_S . forgetting coefficient for STM	0.89	η_S . learning coefficient for STM	1.00
λ_L . forgetting coefficient for LTM	1.00	η_L . learning coefficient for LTM	1.00
Network control parameters of MACNN			
	Chaos /Non-chaos		Chaos /Non-chaos
α . constant parameter	10.0/1.00	k_r . damping coefficient of refractory	0.99/0.10
ε . a steepness parameter	5.0/5.0	k_f . damping coefficient of feedback	0.30/030
a . compound parameter	3.0/3.0	-	- -

Table 3. Parameters used in the simulations

Feedback SOM	
The number of nodes in the input layer	20+4+40
The number of nodes in the competitive layer	40
The number of nodes in the state layer	40
β . : the rate of considering the past information	3.0
γ . : forgetting rate	0.7
η . learning coefficient	0.5 → 0.0 (linear transformation)
σ : width coefficient	0.0 → 1.0 (linear transformation)

Table 4. Parameters of the feedback SOM used in the simulations

6.3 Simulations and results

6.3.1 Confirmation of learning and recollection in the case of a simple small scale maze

At first we confirm by a simple small scale maze whether learning and recollection in our proposed system work well or not. We gave the agent a maze as shown in Fig. 11(a), and let the agent learn and memorize it. In Fig. 11(a), S means the start position of the agent and G means the goal position. I means the position where the agent begins the initial operation. Its numbered place (cell) means the position where each environment was found in LTM. Where, ■ means wall recognized as value 1, □ means aisle recognized as value 0 by the agent. The result of the simulation using the maze 1 as a simple small scale maze, the agent reached the goal through the shortest path as the real line with arrow shown in Fig. 11(a). Let us explain how the maze was solved by the agent concretely as follows:

The whole algorithm until the agent reaches to the goal in case that maze 1 is given, as follows:

Step 1. Give the maze 1 to the agent which does not learn and memorize anything.

Step 2. Switch to the learning sector because of no learned and memorized mazes, and the agent learns the maze 1 by the actor-critic method. As a result, the memory corresponding to the maze 1 is generated as the number 1 of the memory in LTM.

Step 3. The agent learns the MACNN and the feedback SOM by use of the results of learning at Step 2.

Step 4. The agent executes the initial operation (n_{init}=5 steps) n_{repeat}(=30) times and records the winner neuron number for the maze 1.

Step 5. The agent begins on the initial operation again for maze 1.

Step 6. The agent inputs the data gotten from the initial operation to the discrimination of environment sector. As a result of the discrimination, the agent gets the memory 1 (maze 1).

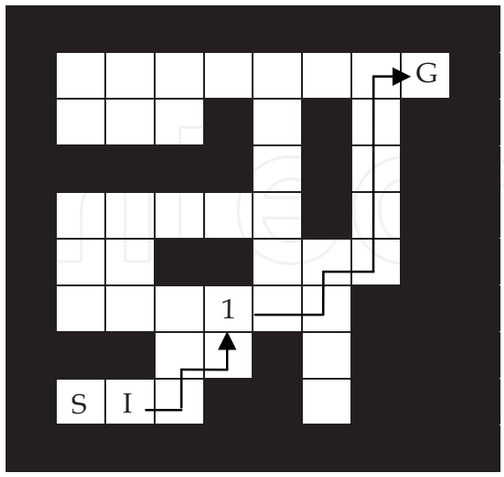
Step 7. The agent begins the recollection using the memory 1, i.e. MACNN1. It reaches the goal at shortest steps.

6.3.2 Generation and discrimination of plural simple small scale mazes

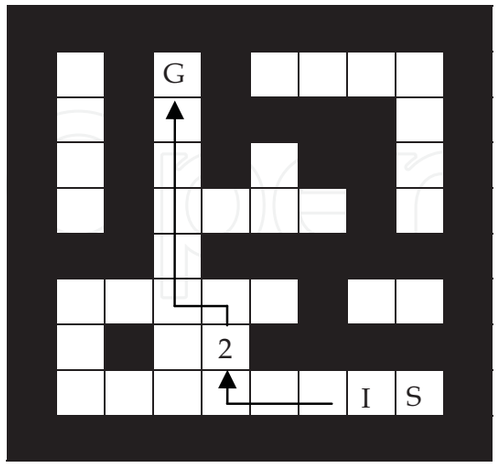
We consider four simple small scale mazes as shown in Fig. 11(a) to (d). At first the agent learns and memorizes the maze 1 by the way mentioned above 6.3.1, next we gave the agent the maze 2 as shown in Fig. 11(b). For the maze 2, after the agent executed the initial operation, the agent judged the memory 1 (maze 1) could not be used since the memory 1 is not corresponding to the current maze, it switched to the learning sector and memorized the maze 2 as memory 2 in LTM (refer to Table 2). Similarly, maze 3 and 4 are learned and memorized as memory 3 and 4 in LTM.

The winner neuron numbers at the each initial operation step when given the environment the same as the memory are shown in Fig. 12. In Fig. 12, it is found that though there are only four memories, the winner neuron numbers are overlapping in spite of the difference of the environments each other. Next, we check the differences of the Hamming distance between above 4 mazes each other. As mentioned at 6.3.1, ■ means wall recognized as value 1, □ means aisle recognized as value 0 by the agent. There is 15 bits (5 different directions times 3 different distances) in the perception range of the agent. The Hamming distance between the four mazes is shown in Table 5. From Table 5, it is found that there is no overlapping environments. However we encountered an example of the failure of the

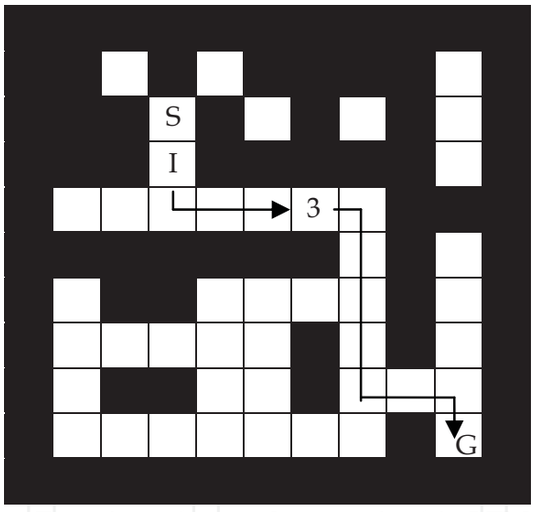
agent's taking the goal like following. After learning and memorizing above 4 mazes, we gave the agent maze 4 again. The situation of the failure case is shown in Fig. 13.



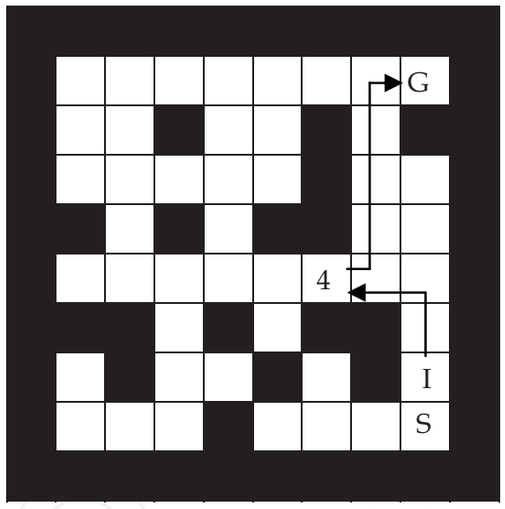
a) Experienced maze 1 memorized in LTM



b) Experienced maze 2 memorized in LTM



c) Experienced maze 3 memorized in LTM



d) Experienced maze 4 memorized in LTM

Fig. 11. Learned and memorized paths of the agent on the each maze.

This cause may be considered as follows: When the agent starts from start point S, it can select two directions, i.e., up and left, the agent can take move to. When the agent executes the initial operation, in other words, when the winner neuron numbers at the each initial operation are set first, if the selection rate of the upward step of the agent are biased, the upward direction are selected mainly, after memorizing of their data in LTM. However, when the agent begins the initial operation and the steps to the left are mainly selected, the winner neuron count had become less than the value of the *threshold_count* (refer to 5.3). Though the agent has the memory corresponding to the current maze, the agent judged that the agent does't have the experience of the current maze because of the small value of the *threshold_count*, and as a result, it switched to the learning sector. To solve this problem, the number of steps on the initial operation should be increased and the *threshold_count* is appropriately decided.

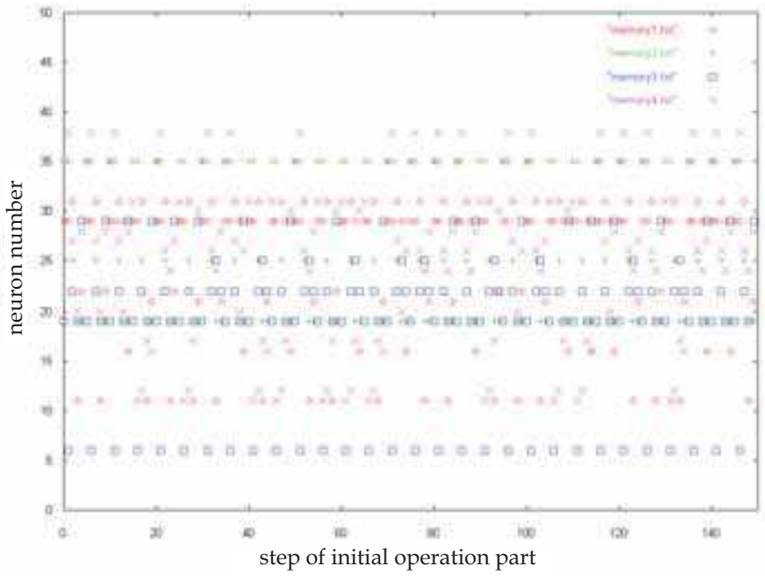


Fig. 12. The winner neuron numbers at the each initial operation step when given the environment the same as the memory

	maze 2	maze 3	maze 4
maze 1	6	6	6
maze 2		6	6
maze 3			2

a) At step 1

	maze 2	maze 3	maze 4
maze 1	8	4	8
maze 2		6	8
maze 3			8

b) At step 2

	maze 2	maze 3	maze 4
maze 1	8	10	10
maze 2		2	6
maze 3			4

c) At step 3

	maze 2	maze 3	maze 4
maze 1	10	10	10
maze 2		6	8
maze 3			6

d) At step 4

	maze 2	maze 3	maze 4
maze 1	8	4	2
maze 2		10	10
maze 3			4

e) At step 5

Table 5. Hamming distance of each other of the four mazes on the initial 5 steps

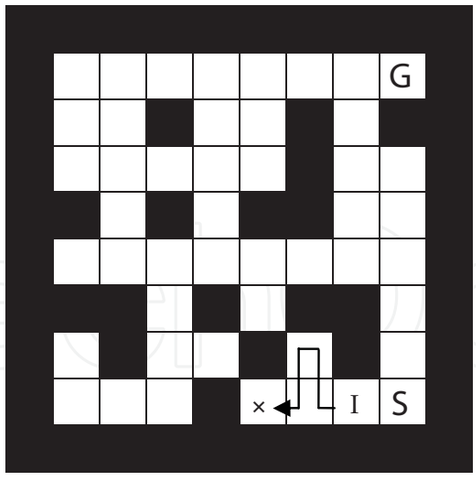


Fig. 13. The moving path in the case of failure

6.3.3 In the case of a large-scale maze

The large scale maze to be solved by the agent is shown in Fig. 14. This maze is constructed by using the four mazes shown in Fig. 11. In this maze, the shortest steps, i.e., optimal steps is 108. Before the agent tries to explore this maze, the agent learned and memorized above four mazes orderly. In the Figure, × means the position where the agent failed the choice of the memorized information in LTM, i.e., the action corresponding to the current environment under use of the learned and memorized environment. The number shows the memory number the agent selected using the initila operation. In a lot of the agent's trials to this maze, the steps until the agent got the goal is between 110 and 140. Because of the exploring steps at the initial operation process, they are more than the shortest steps. As a result, it is said that it may be possible the agent with our proposed system could reach the goal in any case of environments, by additive learning and memorizing for the unknown environment.

7. Conclusions

Living things learn many kinds of things from incidents driven by own actions and reflects them on the subsequent action as own experiences. These experiences are memorized in their brain and recollected and reused if necessary. They will accumulate knowledge gotten by experiences, and will make use of them when encounters unexperienced things.

The subject in this research was to realize the things mentioned above on an agent. In this research, we tried let the agent equip with three main functions: "learning", i.e., reinforcement learning commonly used by living things, "memorization", and "the ability of associative recollection and its appropriate use" suitable for the situation, i.e., chaotic neural network.

This time we realized a part of subjects of above functions on the agent. However, a lot of unsolved problem are still left. One of them is too difficult to decide the various kinds of parameters and thresholds approproately. Another one is to utilize the ability of feedback SOM well. SOM has the feature that input patterns similar to each other are placed in the SOM retaining the neighboring relationship. This is useful in the case of existing observation together with noise because that actually almost all observation data include noises. In such cases, making use of this characteristic of feedback SOM, the agent may realize things mentioned before.

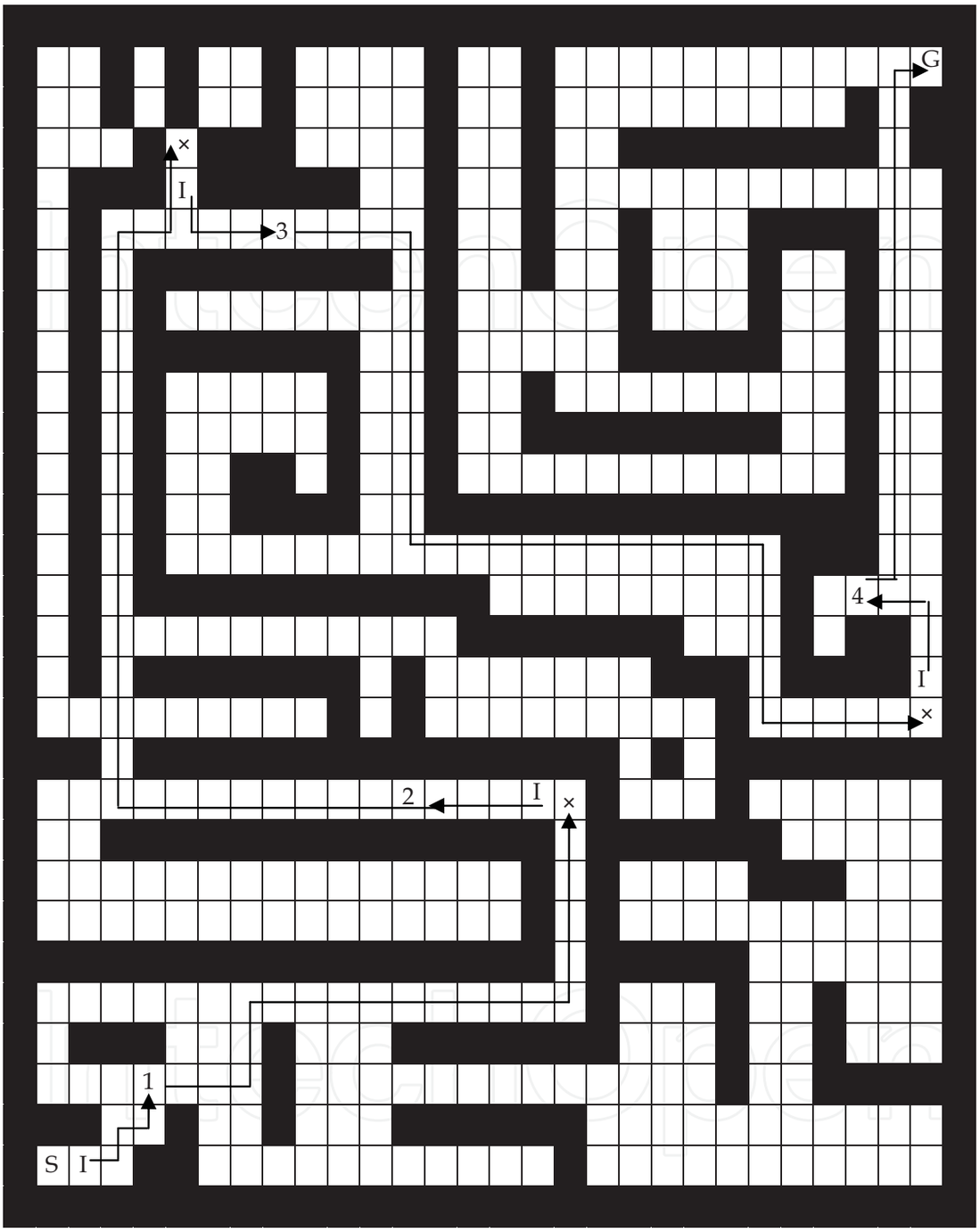


Fig. 14. The path the agent found and memorized in LTM on the large scale goal searching problem using the experienced mazes shown in Fig. 11.

8. Acknowledgements

We would like to thank that a part of this study was supported by JSPS-KAKENHI (No.20500207 and No.20500277).

9. References

[1] D. Harter, R. Kozma: "Chaotic Neurodynamics for Autonomous Agents", IEEE Trans. on *Neural Networks, Vol*. 16, pp.565-579, 2005

[2] J. L. Krichmar, K. Seth, D. G.Nitz, J. G. Fleischer, G. M. Edelman:"Spatial Navigation and Causal Analysis in a Brain-Based Device Modeling Cortical-Hippocampal Interactions", Neuroinformatics, Vol.3, No.3, pp.197-221, 2005

[3] Obayashi M., Omiya R., Kuremoto T., Kobayashi K.: "Shapes of Non-monotonous Activation Functions in Chaotic Neural Network." *IEEJ Transactions on EIS*, Vol.126, No11, pp.1401-1405, 2006 (*in Japanese*).

[4] M. Obayashi, K. Narita, K. Kobayashi, T. Kuremoto: "A Transient Chaotic Associative Memory Model with Temporary Stay Function", *IEEJ Transactions on EIS*, Vol.128, No.12, pp.1852-1858, 2008 (*in Japanese*)

[5] M. Obayashi, K. Narita, T. Kuremoto, K. Kobayashi: "A Reinforcement Learning System with Chaotic Neural Networks-Based Adaptive Hierarchical Memory Structure for Autonomous Robots", *Proceedings of International Conference on Control, Automation and Systems 2008* (*ICCAS 2008*), pp. 69-74, 2008 (Seoul, Korea)

[6] Obayashi, M., Yano, Y., Kobayashi, K., and Kuremoto, T.: "Chaotic Dynamical Associative Memory Model Using Supervised Learning". *Proceedings of the 13th International Symposium on Artificial Life and Robotics (AROB2008)*, pp.555-558, January 31-February 2, 2008 (Beppu, Japan).

[7] M. Obayashi, T. Kuremoto, K. Kobayashi: "A Self-Organized Fuzzy-Neuro Reinforcement Learning System for Continuous State Space for Autonomous Robots", *Proceedings of International Conference on Computational Intelligence for Modeling, Control and Automation 2008* (*CIMCA08*), pp.552-559, 10-12 Dec. 2008 (Vienna, Austria)

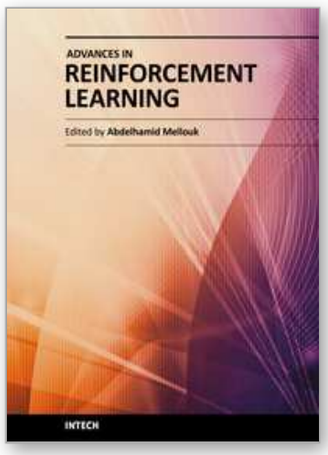
[8] Obayashi, M., Kuremoto, T., and Kobayashi, K., Feng L.: "Intelligent Agent Construction Using the Idea of the Attention Characteristic Pattern in a Chaotic Neural Network", *Proceedings of the 15th International Symposium on Artificial Life and Robotics (AROB2010)*, pp.597-600, February 4-6, 2010 (Beppu, Japan), and to be appeared as a paper of *Artificial Life and Robotics*, Vol. 15, 2010

[9] R.S. Sutton, A.G. Barto:"Reinforcement Learning", *The MIT Press*, 1998

[10] Aihara, K. and Takabe, T. and Toyoda, M.: "Chaotic Neural Networks", *Physics Letters A*, pp.333-340, 1990

[11] M. Adachi, K. Aihara:"Associative Dynamics in a Chaotic Neural Network", *Neural Networks*, Vol. 10, No. 1, pp.83-98, 1997

[12] T.kohonen: "Self-organizing Maps", *Springer*, 2001



Advances in Reinforcement Learning
Edited by Prof. Abdelhamid Mellouk

ISBN 978-953-307-369-9
Hard cover, 470 pages
Publisher InTech
Published online 14, January, 2011
Published in print edition January, 2011

Reinforcement Learning (RL) is a very dynamic area in terms of theory and application. This book brings together many different aspects of the current research on several fields associated to RL which has been growing rapidly, producing a wide variety of learning algorithms for different applications. Based on 24 Chapters, it covers a very broad variety of topics in RL and their application in autonomous systems. A set of chapters in this book provide a general overview of RL while other chapters focus mostly on the applications of RL paradigms: Game Theory, Multi-Agent Theory, Robotic, Networking Technologies, Vehicular Navigation, Medicine and Industrial Logistic.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Masanao Obayashi, Kenichiro Narita, Yohei Okamoto, Takashi Kuremoto, Kunikazu Kobayashi and Liangbing Feng (2011). A Reinforcement Learning System Embedded Agent with Neural Network-Based Adaptive Hierarchical Memory Structure, Advances in Reinforcement Learning, Prof. Abdelhamid Mellouk (Ed.), ISBN: 978-953-307-369-9, InTech, Available from: http://www.intechopen.com/books/advances-in-reinforcement-learning/a-reinforcement-learning-system-embedded-agent-with-neural-network-based-adaptive-hierarchical-memor

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen