Design Environment of Intelligent Multiagent Systems

Takahiro Uchiya[1] Syo Itazuro[1] Ichi Takumi[1] Tetsuo Kinoshita[2]

1 Nagoya Institute of Technology 2 Tohoku University
[1] Gokiso-chou, Syowa-ku, Nagoya, 466-8555 JAPAN,
[2] 2-1-1 Katahira, Aoba-ku, Sendai, 980-8577 JAPAN,
t-uchiya@nitech.ac.jp itazuro@uchiya.nitech.ac.jp
takumi@nitech.ac.jp kino@riec.tohoku.ac.jp

Abstract

The agent-oriented computing is a technique for generating the agent who operates autonomously according to the behavior knowledge. Moreover, agent can have the characteristic called "Learning" skill. More efficient operation of agents can be expected by realizing "Learning" skill. In this research, our aim is to support agent designer who designs and develops the intelligent agent system equipped with "Learning" skill. We propose interactive design environment for agent system with learning mechanism using repository-based agent framework called DASH framework. Proposed framework enables agent designer to design and implement the learning agents without highly expertise, therefore we can reduce the designer's burden. In this paper, we explain the DASH framework, Q-learning, Profit Sharing and proposed design environment. Moreover we show the effectiveness of the proposal method through the some experiments.

I. INTRODUCTION

In recent years, the network service develops greatly along with the rapid spread of the internet, and the importance of the network service in the society where we are surrounded has risen. User's needs are diversified and change frequently based on the appearance of new service and the update of existing service. Therefore, the necessity of the flexible system that performs automatically has risen. Nowadays, some researchers are focusing on the agent-oriented software computing as a means to achieve the flexible system.

The agent-oriented computing is a technique for generating the agents who operate autonomously by adding the function to the object to change own parameter and procedure responding to the environment. Moreover, agents can study the best action from the result of a past action. These characteristic is called "Learning" skill. More efficient operation of agents can be expected by realizing "Learning" skill.

In this research, we focus on the development of intelligent agent system. Our aim is to support agent designer who designs and develops the agent system equipped with "Learning" skill. We propose interactive design environment for agent system with learning mechanism using repository-based agent framework called DASH framework. Proposed framework enables agent designer to design and implement the learning agents without highly expertise, therefore we can dramatically reduce the designer's burden.

In this paper, we firstly explain DASH framework [1], Q-learning [2][3] and Profit Sharing (PS) [4] as the assumption knowledge. Next, we explain design environment which provides some functions to develop the learning agent. Finally, we verify the effectiveness of the proposal method through the some experiments.

II. DASH FRAMEWORK

In this research, DASH (Distributed Agent System based on Hybrid architecture) framework that is one of the agent frameworks is used. The DASH agent is a rule-based agent because it has some behavior rules of the if-then type defined by the agent designer. Each agent holds the knowledge in the form of "fact" and "rule". A rule is represented in the following form.

(rule Rule-name Condition-part (If-part)
--> Action-part (Then-part))

It behaves based on the domain knowledge-base that determines appropriate agent behavior (Fig. 1).

The inference mechanism consists of the inference engine, the working memory (WM), and the rule set.

The DASH agent operates as follows.

(P1) The inference engine that controls the operation knowledge searches for the rule that matches to the content referring to WM.
(P2) The matched rule is executed.

(P3) It returns to one when the content of WM is updated by executing the selected rule.

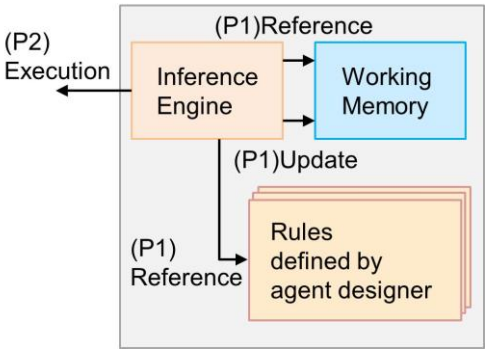

Fig.1: Inference mechanism of DASH agent

III. Q-LEARNING

A. Outline of Q-Learning

We use Q-Learning method for realizing intelligent single agent. Q-Learning [2][3] is a typical technique used in the reinforced learning field. Reinforcement learning is a technique to give agents the evaluation of the result of the action that the agent took. The agent gradually learns the best action to maximize this evaluation.

Q-Learning has shown that a sufficient number of trials in Markov decision processes engender an optimal solution. In this research, because of the simplification of the settings, we assume that the DASH agent is a single agent with a Markov property. Therefore, it is possible to realize highly reliable learning of agent behavior.

B. Process of learning

In Q-Learning, an agent learns by updating the priority of an agent's respective rules based on the result of the action. The flow of concrete learning is shown below.

1. The agent observes the existing state (or environment), and searches for the rule that matches it. The definition of the state indicates the state of WM by the agent in the DASH framework.
2. When two or more matched rules exist, one is selected from them based on the agent's action selection technique.
3. The rule that is selected is executed.
4. The rule priority is updated from the execution result according to the update formula.
5. Check whether the agent's operation is done or not.
   When the agent's operation continues, it returns to 1.

C. Action selection technique

Reinforcement learning has no necessity for devising an action selection technique when aiming only at learning. Learning will be complete if all rules are executed many times so that learning can advance through trial-and-error. Therefore, it need only have selected the rule at random. However, the cases in which learning and system operation are requested simultaneously are actually the most common. For that case, it is necessary to devise the action selection technique to execute the high priority rule. The ε-greedy method and the soft max method, etc. are known as a typical action selection technique.

The ε-greedy method is a technique for the selection of the rule that the priority is the maximum by the probability 1-ε, and selects the rule at random by probability ε.

The soft max method is a technique for calculation of the selection probability of each rule according to the ratio of the priority of the rule, and selects the action based on the probability.

D. Update formula

- **Outlines of update**

In Q-Learning, the rule priority has been updated using the following update formula.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a' \in A(s')} Q(s',a')\right]$$

In the update formula, $Q(s,a)$ shows the priority of the rule that takes action a in the state s, and $Q'(s',a')$ shows the priority of the rule that takes action a' in the transition state s'. Moreover, r shows the reward obtained using the transition from s to s', and $A(s')$ shows the available action set in the state s'.

α is the learning rate for which the parameter range is $0<\alpha<1$.

γ is the discount rate for which the parameter range is $0<\gamma<1$.

Priority is updated every time in the state transition, and the agent has been updating the priority of the rule based on reward r and the value of high priority rule in the state s' expressed as $\max_{a' \in A(s')} Q(s',a')$.

- **Learning rate and discount rate**

The learning rate is a parameter representing the balance of priority whether we regard the original priority value as important or regard the newly obtained result such as available rule set as important when the priority of the rule is updated. The update formula shows that when α approaches 0, we regard the original priority value as important. Oppositely, when α approaches 1, we regard the newly obtained result as important. It is generally true that the learning rate is set to 0.1.

The discount rate is a parameter showing how importantly we regard the reward obtained in the future. Although reward r was obtained using the transition from state s to s', no complete guarantee exists of obtaining the best result from future action because it has not executed state s' yet. Therefore, it is necessary to discount the priority of executable rules in state s' to some degree. The update formula shows that when γ approaches 0, we disregard the future reward. When γ approaches 1, we

regard the future award as important. It is generally true that the discount rate is set to 0.9–0.99.

IV. Profit Sharing

A. Outline of Profit Sharing

We use Profit Sharing (PS) method for realizing intelligent multi-agent system. PS is known as one of the typical technique in the reinforcement learning field. The reinforcement learning is a technique to give agent the evaluation of the result of the action that the agent took. The agent gradually learns the best action to maximize this evaluation.

PS does not guarantee the optimal solution, however PS is suitable for multiagent reinforcement learning [3][4].

B. Process of learning

In the PS, agent learns by updating priority of rule in the environment at that time based on the result of the action. The flow of concrete learning is shown below.

1. The agent observes the existing state (or environment), and searches for the rule that matches to it. The definition of the state indicates the state of WM by the agent in the DASH framework.
2. When two or more matched rules exist, one is selected from them based on agent's action selection technique.
3. The selected rule is executed, and memorized as "episode-rule" by the agent.
4. Check whether the agent has transferred to the goal state or not. If the agent has not, returns to one.
5. Priorities of all episode-rule are updated from the execution result according to the update formula.
6. Check whether the agent's operation is done or not. When the agent's operation continues, returns to one.

C. Action selection technique

Reinforcement learning has no necessity for devising an action selection technique when aiming only at learning. Learning will be complete if all rules are executed many times so that learning can advance through trial-and-error. Therefore, it need only have selected the rule at random. However, the cases in which learning and system operation are requested simultaneously are actually the most common. For that case, it is necessary to devise the action selection technique to execute the high priority rule. The ε-greedy method and the soft max method, etc. are known as a typical action selection technique.

The ε-greedy method is a technique for the selection of the rule that priority is the maximum by the probability 1-ε, and selects the rule at random by the probability ε.

The soft max method is a technique for the calculation of the selection probability of each rule according to the ratio of the priority of the rule, and selects the action based on the probability.

D. Update formula

・**Outlines of update**

In the PS, the rule priority has been updated by using the following update formula.

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha * r(t),$$
$$t = 0, \cdots episode - 1$$

In the update formula, $Q(s_t,a_t)$ shows the priority of the rule in the environment at that time that takes action *a* in the state *s*. Moreover, *r(t)* shows the reward function obtained by the transition from *s* to *s'*.

α is the learning rate that parameter range is $0<\alpha<1$.

Priorities of all episode-rule are updated when the agent transfers to the goal state, and the agent has been updating the priority of the rule based on the reward function *r(t)*.

・**Learning rate**

The learning rate is a parameter representing the balance of priority whether we regard the original priority value as important or regard the newly obtained result such as available rule set as important when the priority of the rule is updated. The update formula shows that when α approaches 0, we regard the original priority value as important. Oppositely, when α approaches 1, we regard the newly obtained result as important. It is generally true that the learning rate is set to 0.1.

V. PROPOSAL OF DESIGN ENVIRONMENT OF INTELLIGENT AGENT SYSTEM

A. Outline of learning mechanism

The design environment proposed by this research helps agent designer to design and implement the Q-Learning or PS based learning agents. This environment has following functions.

B. Functions

・**Automatic learning function**

The rule priority is automatically updated by using the Q-Learning scheme or PS scheme. The automatic learning mechanism is newly introduced as a mechanism to update priority, and it operates in cooperation with the inference mechanism built into existing DASH framework. The automatic learning mechanism is composed by the action selection engine and the learning engine.

[Action selection engine]

This engine selects one action. The ε-greedy method and the soft max method are implemented as an action selection technique, and agent designer chooses one method when they start the learning agent.

[Learning engine]

By using the update formula, this engine updates the priority of the executed rule that the action selection engine selected.

• **Preservation and reference function of learning data**

After learning process proceeds to some degree, the rule name and the priority of each rule are preserved with the file of Comma Separated Value as learning data. This file is called a learning data file. When the same agent works again, agent's operation begins after reading the learning data file and setting the priority of each rule.

Therefore, it is possible to interrupt or restart the learning act. Moreover, we have a future plan to enable new agent designer to use the learning result of other users.

• **Automatic drawing in graph and preservation function**

To visually confirm the appearance that the agent's operation advances efficiently, an automatic drawing is performed as the graph of learning process. We assume that one trial is from the initial state to the target state, and the number of the execution of the rule of every one trial is displayed automatically by the graph form. Moreover, to confirm the past learning process, the function to preserve the graph data as the DAT file is provided.

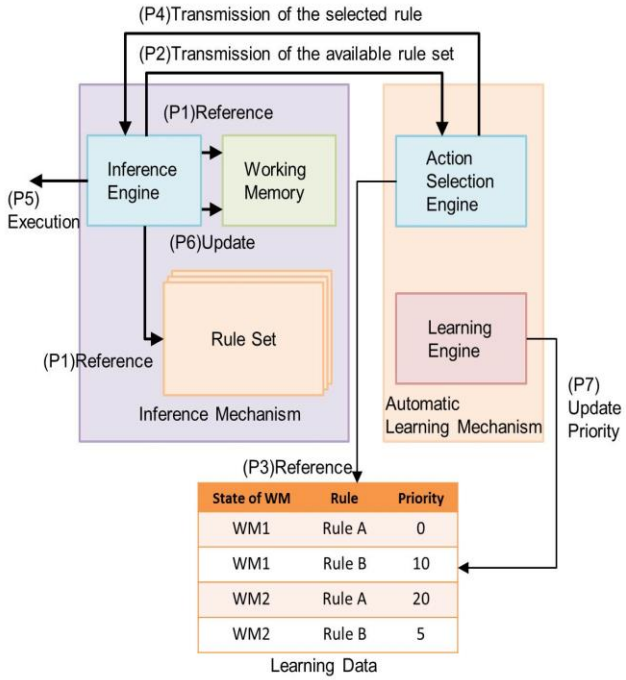

Fig.2: Flow of operation

C. Flow of operation

This mechanism operates as follows (Fig. 2).

(P1) Reference of the content of WM
(P2) Transmission of the available rule set to the Action Selection Engine
(P3) Selection of the rule in consideration of learning data
(P4) Transmission of the selected rule to the Inference Engine
(P5) Execution of the rule
(P6) Update of the content of WM
(P7) Update of the priority in leaning data

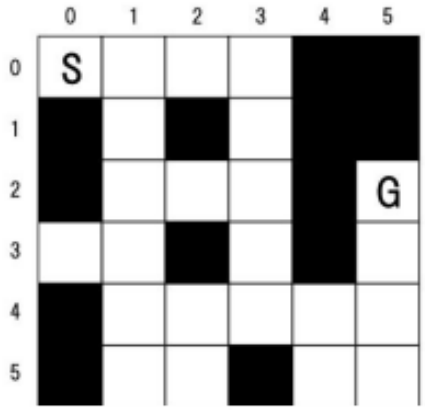

Fig. 4: Maze problem.

VI. EXPERIMENT

A. Experiment of Q-learning: Confirmation of the operation

[Outlines]

We created the following sample agent called "Meiro.dash". Meiro.dash is an agent that solves the maze problem portrayed in Fig. 4. The agent only has information about the room that can be moved from each room. It searches for the shortest route from start (S) to goal (G).

We confirmed that Q-Learning was performed appropriately by setting the following parameters.

・Learning rate: 0.1
・Discount rate: 0.9

Moreover, we confirmed the operation of the automatic graph description function. We observed the change of the rule execution frequency to reach the target goal on 100 times with two cases. One used the ε-greedy method. The other used the soft max method.

[Results and consideration]

As results of experiments, we obtained two graphs using the automatic graph description function. Fig. 5 shows the result obtained using the ε-greedy method; Fig. 6 shows the result obtained using the soft max method. A horizontal axis shows the trial frequency (Number of Trials) and the vertical axis shows the movement frequency (Number of Episodes). In each case, we confirmed that the movement frequency settled gradually to the optimal value. Therefore, we were convinced that appropriate learning was performed correctly.

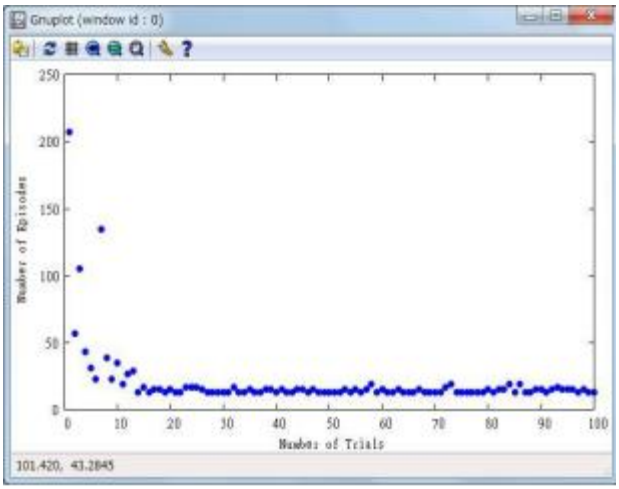

Fig. 5: Result obtained from the ε-greedy method.

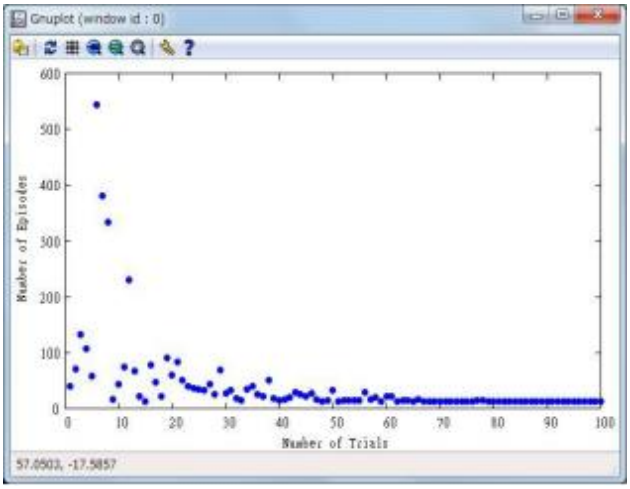

Fig. 6: Result obtained from the soft max method.

B. Experiment of PS: Confirmation of the operation

[Outlines]

We created the following sample agents who solve the tracking problem shown in Fig. 7. There are two Lion-Agents and one Goat-Agent, and Lion-Agents search for the shortest route to Goat-Agent using PS. Goat-Agent moves randomly. These Lion-Agents have only information about where he is, and other two agents are. We define that goal state is the state that both of two Lion-Agents adjoin the Goat-Agent.

We confirmed that PS was performed appropriately by setting following parameters.

- Learning rate: α=0.1
- Reward function: *r(t) = 100*

We observed the change of the rule execution frequency to reach the goal state on 10000 times, and compared the case using PS and the case without PS (movement randomly).

[Results and consideration]

As the result of experiment, we obtained a graph shown in Fig.8. A horizontal axis shows the frequency of trial, and a vertical axis shows the average of movement frequency every 100 trials.

The graph shows that the movement frequency settled gradually to the optimal value. Therefore, we were convinced that appropriate learning was correctly performed.

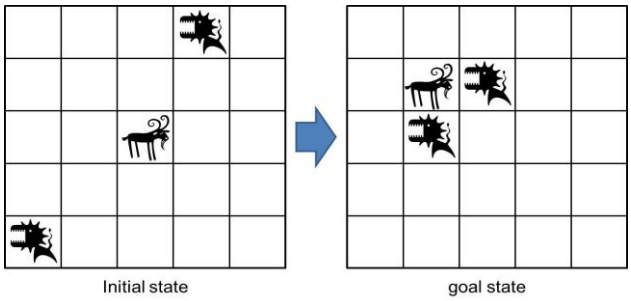

Fig.7: The tracking problem

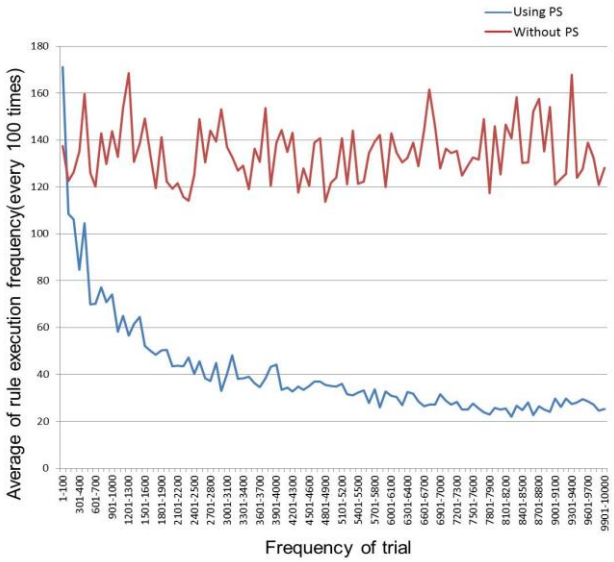

Fig.8: Result of tracking problem

VII. CONCLUSION

In this paper, we proposed the design support mechanism which provides some functions to develop the intelligent agent which equipped with reinforcement learning skill. Moreover, we verify the effectiveness of the proposal method through the some experiments. As a result, we confirmed the agent designer's burden is reduced by the proposed mechanism.

Our future issues are as listed below.

- **Improvement the usability using GUI**

Now, agent's designer must determine some parameter's values, such as learning rate, using both GUI

and property section of DASH file. Because of the lack of uniformity, designer may not know in where he fills in the parameter's values, and it may cause the confusion. Therefore, we should develop the gui-based unified input scheme, and improve the usability.

- **Improvement and expansion of functions**

We should improve and expand each prototype function to improve its usability.

ACKNOWLEDGEMENT

This work was supported by Grant-in-Aid for JST CREST.

REFERENCES

[1] Kenji Sugawara, Hideki Hara, Tetsuo Kinoshita, Takahiro Uchiya: Flexible Distributed Agent System programmed by a Rule-based Language, Proceedings of the Sixth IASTED International Conference of Artificial and Soft Computing, 2002, pp.7-12.

[2] Peng, J. and Williams, R.J.: Efficient Learning and Planning Within the Dyna Framework, Adaptive Behaviour, Vol.1, No.4, pp.437-454(1993)

[3] Rummery, G. A. and Niranjan, M.: On-line Q-learning Using Connectionist Systems, Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University (1994)

[4] Grefenstette, J.J.: Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms, Machine Learning, Vol.3, 1988, pp.225-245.

[5] Arai, S., Miyazaki, K. and Kobayashi, S: Generating Cooperative Behavior by Multi-Agent Reinforcement Learning, in Proceedings of the 6th European Workshop on Learning Robots, 1997, pp.143-157.

[6] Arai, S., Miyazaki, K. and Kobayashi, S.: Cranes Control Using Multi-agent Reinforcement Learning, in Proceedings of International Conference on Intelligent Autonomous System 5, 1998, pp.335-342.

[7] Bellifemine, F., Poggi, A. and Rimassa, G. JADE – a FIPA-compliant agent framework. In Proc. of Practical Application of Intelligent Agents and Multi Agents, 1999, pp.97-108.

[8] Renquist, N. R., Andrew, H. and Andrew, L.: Jack – Summary of An Agent Infrastructure. In Proc. of 5th International Conference on Autonomous Agents, 2001.