IDEAL: Interactive Design Environment for Agent system with Learning mechanism

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.

1. 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] for single agent's learning and Profit Sharing (PS) [4] for multiagent learning as the assumption knowledge. Next, we explain the design environment which provides some functions to develop the learning agents. Finally, we verify the effectiveness of the proposal method through the some experiments.

2. 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.

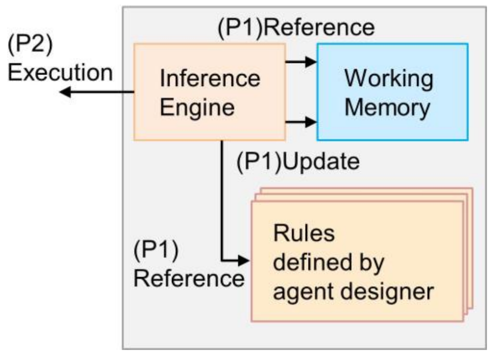


Fig.1: Inference mechanism of DASH agent

3. Q-LEARNING

3.1 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.

3.2 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.

3.3 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.

3.4 Update formula

3.4.1 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 $\max_{a' \in A(s')} Q(s',a')$.

3.4.2 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.

4. Profit Sharing

4.1 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 [4].

4.2 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.

4.3 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.

4.4 Update formula

4.4.1 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)*.

4.4.2 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.

5. PROPOSAL OF INTERACTIVE DESIGN ENVIRONMENT OF AGENT SYSTEM WITH LEARNING MECHANISM

5.1 IDEA

The Interactive Design Environment of Agent system (IDEA)[5] provides an interactive design environment for agent system designers. The following four mechanisms are introduced into IDEA to support agent design.

(M1) Mechanism of agent search support

This mechanism’s has the search condition input area for seeking agents from the repository, the search result display area, and the preview window of the agent knowledge.

(M2) Mechanism of agent programming support

This mechanism has an agent-programming editor based on a rule-based knowledge representation of the DASH framework. Using this editor, the designer can describe and test the agent programs.

(M3) Mechanism of agent simulation support

This mechanism has some interactive simulation functions to analyze agent’s behavior, such as ‘virtual distributed environment’, ‘exchange messages between designer and agents’, ‘dynamic knowledge modification’, and ‘message log analyzer’. Using these functions, the designer can monitor and control the behavior of agents in an interactive manner during the testing and debugging of agents.

(M4) Mechanism of agent registration support

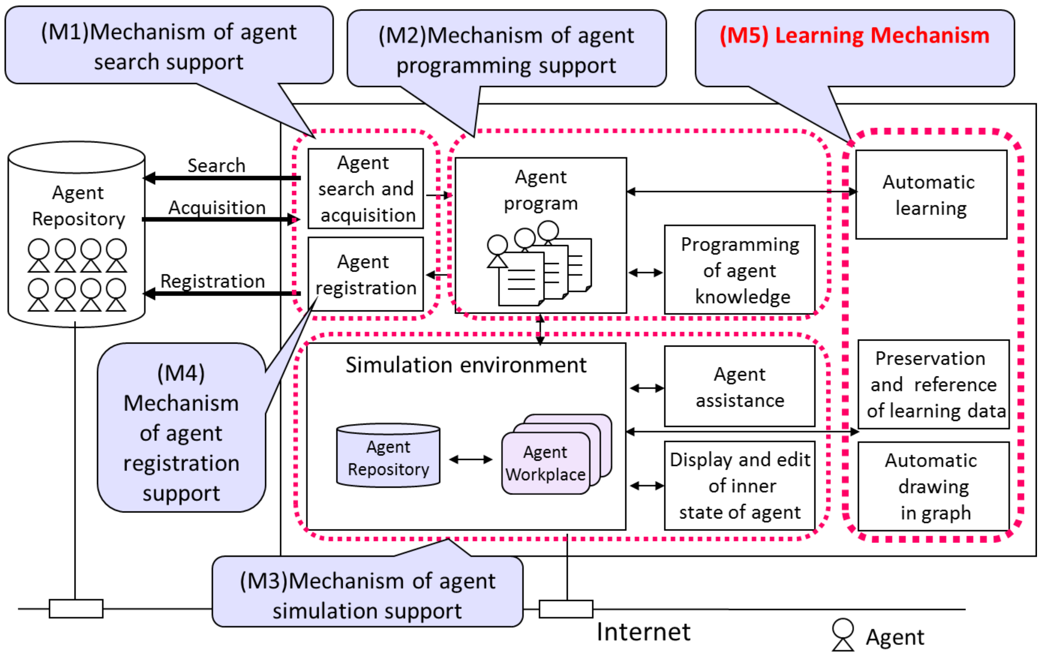

(M1)Mechanism of agent search support
(M2)Mechanism of agent programming support
(M5) Learning Mechanism
Agent Repository
Search
Acquisition
Registration
Agent search and acquisition
Agent registration
Agent program
Programming of agent knowledge
Automatic learning
(M4) Mechanism of agent registration support
Simulation environment
Agent Repository
Agent Workplace
Agent assistance
Display and edit of inner state of agent
Preservation and reference of learning data
Automatic drawing in graph
(M3)Mechanism of agent simulation support
Internet
Agent

Fig.2: Overview of Interactive Design Environment of Agent system with Learning mechanism (IDEAL).

This mechanism provides an interface to store the completed agent system to the repository.

5.2 Introduction of learning mechanism

In this research we newly introduce the learning mechanism of intelligent agents and integrate it into IDEA.

The new design environment is called IDEAL (Interactive Design Environment for Agent system with Learning mechanism) which helps agent designer to design and implement the Q-Learning or PS based learning agents (Fig. 2). Then we focus on the (M5) learning mechanism. This mechanism has following functions.

5.3 Functions of learning mechanism

5.3.1 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.

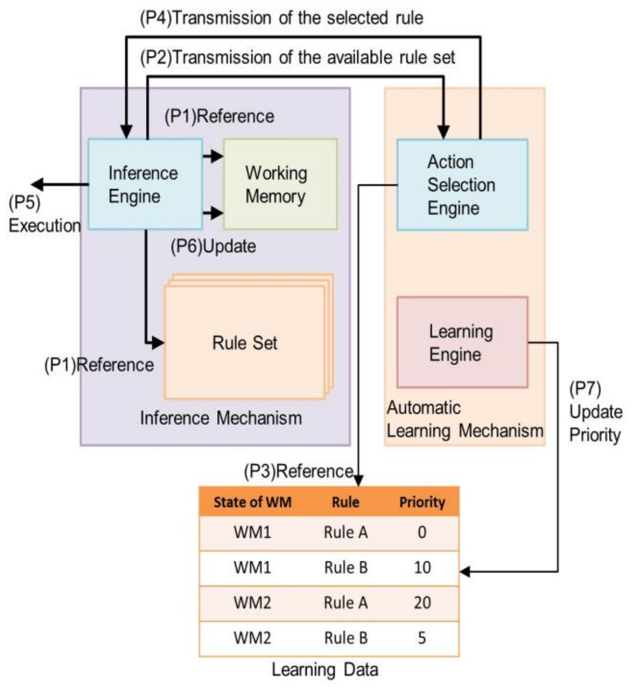
5.3.2 Preservation and reference function of learning data

After the 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.

5.3.3 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.



State of WM	Rule	Priority
WM1	Rule A	0
WM1	Rule B	10
WM2	Rule A	20
WM2	Rule B	5

Fig.3: Flow of operation

5.4 Flow of operation

This mechanism operates as follows (Fig. 3).

(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

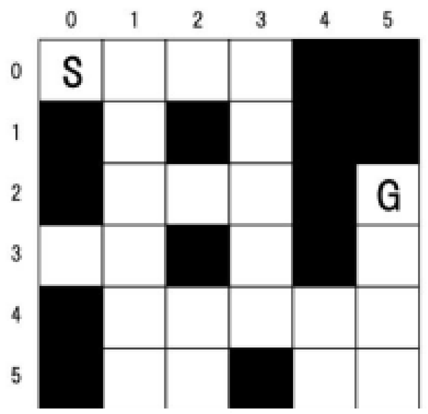


Fig. 4: Maze problem.

6. EXPERIMENT

6.1 Experiment of Q-learning

6.1.1 Experiment 1: 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.

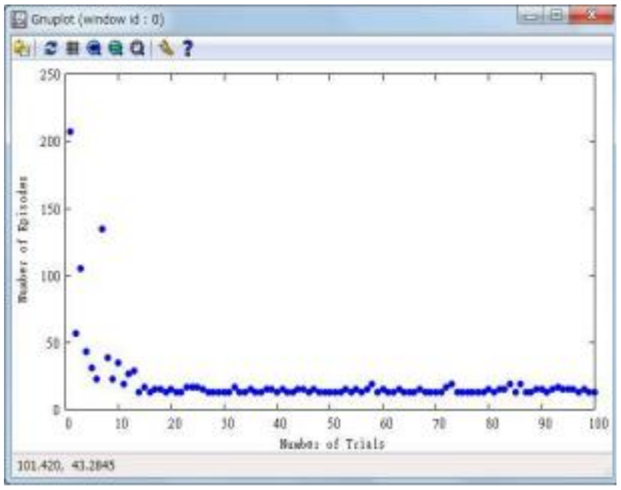


Fig. 5: Result obtained from the ε-greedy method.

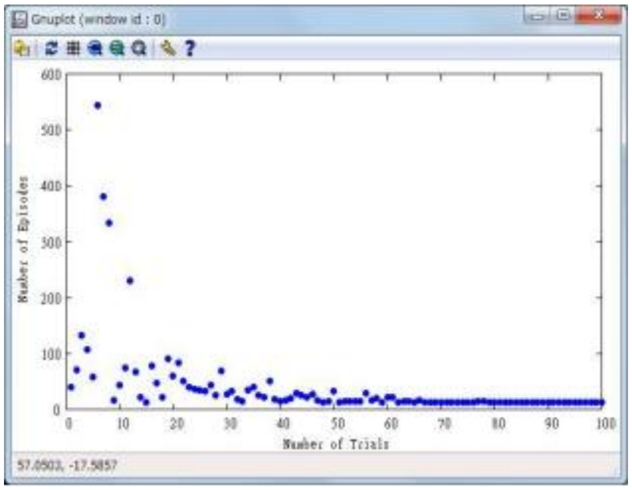


Fig. 6: Result obtained from the soft max method.

6.1.2 Experiment 2: Investigation of the reduction rate of the description amount

[Outlines]

To show that the agent designer's burden is reduced by the proposed mechanism, we investigated the reduction rate of the program description amount.

・Description amount A: The amount of all descriptions necessary for the learning function when we implement the learning agent on DASH without using the proposed method.

・ Description amount B: The description amount necessary for the learning function when we implement the learning agent on DASH with the proposed method.

We calculated the reduction rate of description amount related to learning by measuring A and B above. The reduction rate of description amount used by this experiment is the following.

Reduction rate = $(1-\frac{B}{A})\times 100$

[Results and consideration]

The experiment 2 result is shown below. Table I shows that the reduction rate of description amount related to learning is about 97.6%. By this result, we confirmed that our proposed method can reduce the burden of implementation of the agent's learning skill. Therefore, we achieved the support of the agent designer who has no expert knowledge related to agent programming. Therefore, we were able to verify the effectiveness of the proposed method.

Table I: Reduction rate of description amount related to learning

Description amount A (lines)	Description amount B (lines)	Reduction rate (%)
187	4	97.6

Table II: Result of the questionnaire (convenience)

Score	1	2	3	4	5	Average
Automatic learning	0	0	1	4	4	4.33
Automatic drawing in graph	0	0	0	5	4	4.44
Preserving the graph	0	1	5	3	0	3.22
Preservation and reference of learning data	0	0	6	3	0	3.33

6.1.3 Experiment 3: Questionnaire survey

[Outlines]

To verify the effectiveness of the proposed method, we asked nine test users with experience in agent design using DASH framework, to use our prototype, and to answer a questionnaire about each function's convenience, operability, and improvement. The effectiveness and operability are given rankings according to the five-grade evaluation system.

[Results and consideration]

Results of the questionnaire are as follows. Table II and Table III show evaluations related to convenience and operability. Table IV shows a comprehensive evaluation.

Table III: Result of the questionnaire (operability)

Score	1	2	3	4	5	Average
Automatic learning	0	0	3	6	0	3.67
Automatic drawing in graph	0	0	3	4	2	3.89
Preserving the graph	0	1	6	1	1	3.22
Preservation and reference of learning data	0	2	6	1	0	2.89

Table IV: Result of the questionnaire (comprehensive evaluation)

Score	1	2	3	4	5	Average
Comprehensive evaluation	0	0	2	7	0	3.78

From the questionnaire results, we were able to verify that the convenience of automatic learning function and automatic drawing in graph function are rated higher. However, operability is rated lower. Therefore, we must refer to improvement by free answer for improving each function's interface. Moreover, preserving the graph function and preservation and reference function of learning data are rated lower about both of convenience and operability. The reason for the low score is, as test users said, these are developing functions. For that reason, they have no idea how to use them. Therefore, we must expand functions and implement functions that are easy to understand and use.

6.2 Experiment of Profit Sharing

6.2.1 Experiment 4: 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.

6.2.2 Experiment 5: Investigation of the reduction rate of description amount

[Outlines]

In order to show that the agent designer's burden is reduced by the proposed mechanism, we investigated the reduction rate of program description amount.

・Description amount A: The amount of all descriptions necessary for the learning function when we implement the learning agent on DASH without proposed method

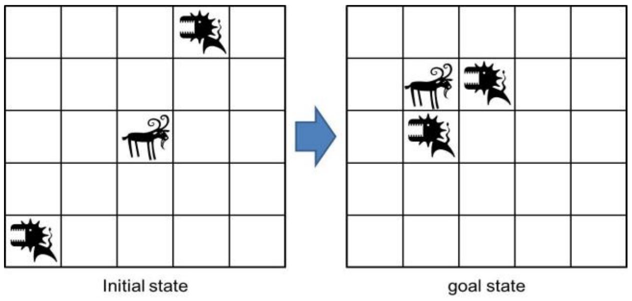


Fig.7: The tracking problem

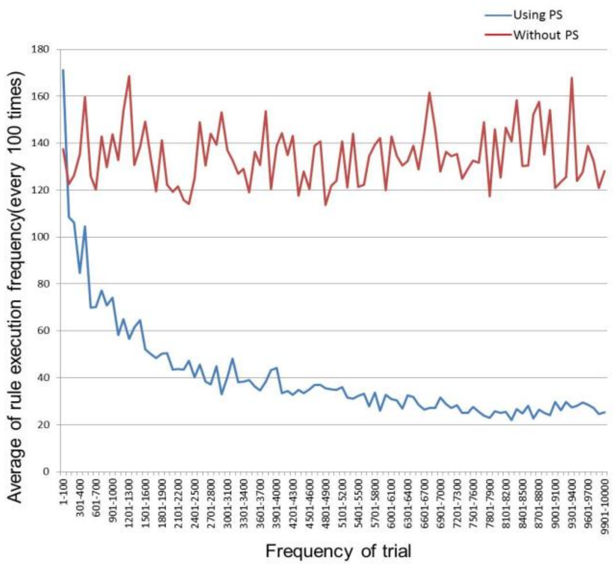


Fig.8: Result of tracking problem

・Description amount B: The description amount necessary for the learning function when we implement the learning agent on DASH with proposed method.

We calculated the reduction rate of description amount concerning learning by measuring above A and B. The reduction rate of description amount used by this experiment is as follows.

The reduction rate = $(1-\frac{A-B}{A})\times 100$

[Results and consideration]

The result of experiment 5 is shown below. Table.V shows that the reduction rate of description amount concerning learning is about 94.3%. By this result, we confirmed that our proposal method is able to reduce the burden of the implementation of agent's learning skill. Therefore, we achieved the support of the agent designer who doesn't have the expertise knowledge concerning agent programming. Hence, we could verify the effectiveness of proposal method.

Table. V: The reduction rate of description amount concerning learning

Description amount A (lines)	Description amount B (lines)	Reduction rate (%)
194	11	94.3

7. 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 of 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.

8. ACKNOWLEDGEMENT

This work was supported by Grant-in-Aid for Scientific Research (C) 25330258 and JST CREST.

9. 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] Takahiro Uchiya, Takahide Maemura, Hideki Hara, Kenji Sugawara and Tetsuo Kinoshita, Interactive Design Method of Agent System for Symbiotic Computing, International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), Vol.3, No.1, 2009, pp.57-74.