Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

Cooperative Behavior Acquisition for Multi-agent Systems by Q-learning

M. C. Xie, and A. Tachibana

Abstract— **In this paper, we focused on the problem of "trash collection", in which multiple agents collect all trash as quickly as possible. The goal of the present research is for multiple agents to learn to accomplish a task by interacting with the environment and acquiring cooperative behavior rules. We construct the learning agent using Q-learning, which is a representative technique of reinforcement learning. Q-learning is designed to find a policy that maximizes the learning for all states. The decision policy is represented by a function. The goal is for multiple agents to learn to accomplish a task by interacting with the environment and other agents. The action value function is shared among agents. The effectiveness of the learning is verified experimentally.**

Index Terms—**Action value function, Cooperative behavior, Q-learning, Multi-agent systems.**

I. Introduction

Multi-agent systems are systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks [1] In multi-agents systems, each agent must behave independently according to its state and environment, and, if necessary, must cooperate with other agents in order to perform a given task. Multi-agent systems have greater robustness and flexibility than conventional centralized management systems. However, it is difficult to predict in advance the actions of the agents and to assign action rules to multi-agent systems, because the interaction between agents is complicated.

Numerous studies regarding autonomous agents in the multi-agent systems have been conducted. Nolfi and Foreano [2] simulated a pursuit system with two agents (predator and prey) in real environments. They evolved both agents reciprocally using a genetic algorithm. Jim and Lee [3] evolved the autonomous agents with a genetic algorithm. Zhou [4] used both a fuzzy controller and a genetic algorithm. The fuzzy function displayed the position of the agent, and the genetic algorithm was used for learning. Fujita and Matsuo [5] learned the autonomous agent using reinforcement learning. The reinforcement learning method involves developing an agent's behavior by means of the interrelationship with the environment and resulting reinforcement signals. The reinforcement learning method can guarantee learning and adaptability without precise pre-knowledge about the environment.

We used a genetic algorithm to acquire the rules for an agent. Individual coding methods are performed (i.e., rule definition), and the learning efficiency is evaluated [6].

In the present paper, we focused on the problem of "trash collection", in which multiple agents collect all trash as quickly as possible. The goal of the present research is for multiple agents to learn to accomplish a task by interacting with the environment and acquiring cooperative behavior rules. We construct the learning agent using Q-learning, which is a representative technique of reinforcement learning. Q-learning is a method of allowing an agent to learn from a delayed system of rewards and punishments. Q-learning is designed to find a policy that maximizes the learning for all states. The decision policy is represented by a function. The goal is for multiple agents to learn to accomplish a task by interacting with the environment and other agents. The action value function is shared among agents. The effectiveness of the learning is verified experimentally.

II. Cooperative Action of Multi-agent

A. Multi-agent Systems

Multi-agent systems are the systems in which several interacting, autonomous agents pursue some set of goals or perform some set of tasks. A key pattern of interaction in multi-agent systems is goal- and task- oriented coordination, both in cooperative an in competitive situations. In the case of cooperation several agents try to combine their efforts to accomplish as a group what the individuals cannot, and in the case of competition several agents try to get what only some of them can have.

In [7] the following major characteristics of multi-agent systems are identified:

--each agent has just incomplete information and is restricted in its capabilities;

--system control is distributed;

--data is decentralized; and

--system control is distributed;

M. C. Xie is with the Department of Electrical and Computer Engineering, Wakayama National College of Technology, Nada-Cho, Gobo City, Wakayama-Ken, 644-0023, Japan (phone: 738-29-8377, fax: 738-29-8399, e-mail: xie@wakayama-nct.ac.jp)

A. Tachibana was with Mechatronics Engineering Course, Wakayama National College of Technology. He is now with Hitachi Advanced Digital, Inc. Yoshida-Cho, Totsuka-Ku, Yokohama City, Kanagawa-Ken, 244-0817, Japan (e-mail: a-tachibana@hitachi-ad.co.jp).

B. Cooperative Action of Agents

When multiple autonomous agents exist, the environment changes from static to dynamic, compared to the case of an individual agent. An agent engaged in cooperative action decides its actions by referring to not only its own information and purpose, but to those of other agents as well.

Multi-agent systems enable problems to be solved more efficiently. In addition, multi-agent systems can solve problems that may be impossible for individual agents to solve, because multi-agents have one common aim and can adjust to their environment and perform cooperative actions. However, multi-agents are not always advantageous. For example, if multiple agents in the same environment act independently, then the situation that each agent has to deal with becomes more complex because each agent has to consider the other agents movements before performing an action. If an agent tries to perform an autonomously decided action, the agent may not be able to perform the action as planned because of disturbances caused by the other agents. Changes to the environment caused by the actions of other agents may make understanding the environment difficult for the agents.

C. Establishment of the Problems Environment

The present study considers the "trash collection" problem, in which trash is placed on a field of fixed size and agents must collect the trash as fast as possible.

As in Figure 1, the field is divided into $N \times N$ lattice. Agents are denoted by ● symbols, and trash is denoted by the ■ symbols.

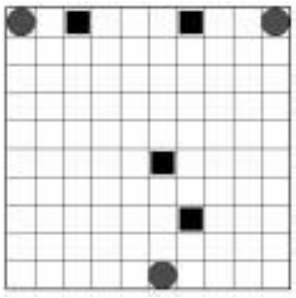

Fig. 1. An example of the problem environment. The field is denoted by lattice. Agents are denoted by ● symbols, and trash is denoted by the ■ symbols.

In the trash collection problem, the actions of agents are defined as follows:

1) The action of an agent is determined once per unit time.
2) An agent can move to an adjoining lattice, where up-and-down and right-and-left are connected per unit time.
3) An agent collects the trash when the agent has the same position as the trash.

III. Cooperative Behavior Acquisition Using Q-learning

A. Agent and Environment

The agent decides the action based on the condition of the perceived environment, and some changes in the environment are caused by the action. Therefore, the agent and the environment are related through their interactions.

An accessible environment is one in which the agent can obtain complete, accurate, and up-to-date information about the state of the environment. Most moderately complex environments are inaccessible. The more accessible the environment, the simpler it is to build an agent to operate in that environment [1]. When an environment is inaccessible, the information that can be perceived from the environment is limited and inaccurate, which involves a delay.

When there is only one agent, the environment is static. However, when another agent exists, the environment is not always static because the condition of the environment may be changed by other agent.

B. Composition of Agents

In this section, the composition and function of each agent are described. The structure of an agent is shown in Figure 2. The figure shows the individual components of an agent, and the transmission of information is expressed by the arrows. First, the agent perceives the condition of the environment in the detector. Next, the rule is compared with the condition that was perceived by the acting decider, and the action is decided. Finally, the action is carried out in the effector.

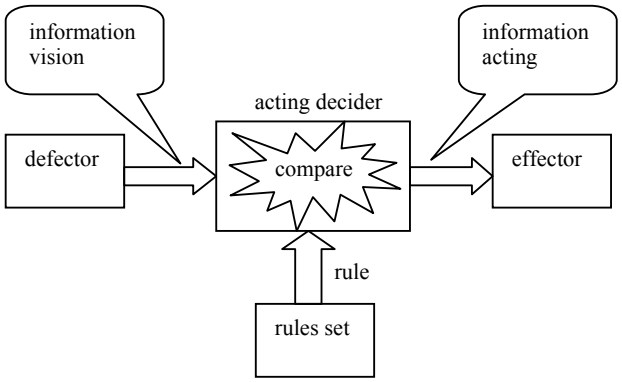

Fig. 2. Structure of an agent. It shows the individual components of an agent, and the transmission of information is expressed by the arrows.

The detector of the agent cannot always perceive the condition of the environment completely, which is similar to limits on human visual acuity. Therefore, an agent can perceive only a part of the condition, or state, of the environment.

The entire environment and the environment that an agent can perceive are shown in Figure 3 and Figure 4, respectively. Figure 3 shows the agents and their respective perceived environments within the entire environment. In Figure 3, the agents are indicated by yellow circles labeled by numbers. The frame that encloses each agent is the range that the agents can perceive, and this range is limited. Figure 4 shows the environments perceived by each agent. The X symbol represents the agents, and the numbers represent the other agents.

The conditions (position) of all of the agents are included in the entire environment, as shown in Figure 3. Agent 1 and

Agent 2 perceive different environments. However, even though each agent is uniquely located in the environment, Agent 3 and Agent 4 perceive the environment to be the same. This problem is called the imperfect perception problem and becomes one of the important problems in agent design. Increasing the visual field range of the agent is considered to be one solution to this problem.

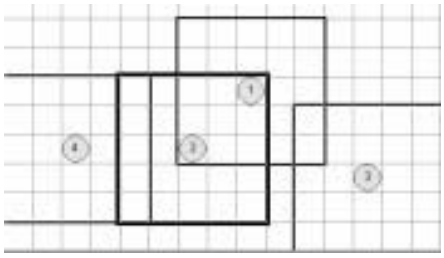

Fig. 3. Entire environment. The agents are indicated by yellow circles labeled by numbers. The frame that encloses each agent is the range that the agents can perceive.

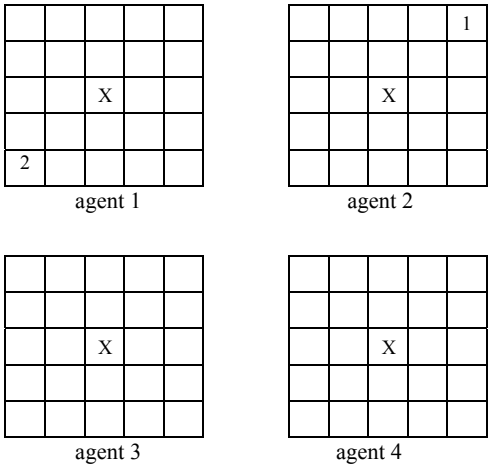

Fig. 4. Environment perceived by each agent. The X symbol represents the agents, and the numbers represent the other agents.

The behavior of each agent is governed by rules. A rule is a condition-action rule of form. The condition part of the rule represents the condition of the environment that the defector perceives. The action part of the rule indicates the action of the agent. When the number of patterns of the condition of an environment that the defector perceives increases, the number of rules increases. An example of condition-action rules is shown in Table 1.

TABLE I
EXAMPLE OF RULES

Condition of environment	action
2000000000000X000000010020	up
0020000010000X000000020000	down
0000000000000X000000002000	left
0020000000000X000000001000	right
0000000000000X000002000000	down

The effector decides the action by the rule. For example, if the condition of the environment was perceived as "000000000000X000002000000", the action "down" is chosen by the agent from the rules in Table 1.

C. Q-learning

A prominent algorithm in reinforcement learning is the Q-learning algorithm. In a finite Markov decision process (MDP) environment, the goal is to find the optimal policy in each state visited in order to maximize the value of a performance metric, e.g., long-run discounted reward using Q-learning.

In Q-learning, which can be implemented in simulators or in real time, the algorithm iterations are the *Q*-factors. Let $Q(s, a)$ denote the *Q*-factor for state s and action a. Let r_{t+1} denote the immediate reward earned in going from state s_t to state s_{t+1}, and let γ denote the discounting factor. When the system transitions from state s_t to state s_{t+1} and a_t denotes the action chosen in state s_t, then the *Q*-factor for state s_t and action a_t is updated as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \quad (1)$$

where $0 < \alpha \leq 1$, which denotes a step-size parameter in the iteration, must satisfy some standard stochastic-approximation conditions, and all other *Q*-factors are kept unchanged. Theoretically, the algorithm is allowed to run for infinitely many iterations until the *Q*-factors converge.

D. Architectural Design of Agent Systems

The agent simulation system is made using the Java language. This system can simulate the actions of "trash collection" agents on a *15* × *15* field. The system consists of the main window on the left and the log window on the right. In the main window, actions and the number of steps of the agent, among other items, are displayed. The check box is set to be ON, when confirmation of the action of the agent is desired. In the log window, information on the action of the agent is displayed. It is possible to preserve displayed information in a text file. In addition, the graph can be opened in Excel.

E. Action Acquisition by Q-learning

The action-value function in which the agent acts is renewed. When an agent picks up a piece of trash, the reward that affects the renewal of the action value is assigned to the agent. In addition, the field is initialized when all of the trash on the field has been picked up. However, the action-value function and the agent learning number of steps are not initialized. It is repeated for the specified number of learning steps.

The flow of the trash collection agent learning is as follows.

```
Initialize Q(s, a) arbitrarily
 Initialize state of field
   (Arrange initial position of agents and trash)

Loop (until decided step)
{
     Decide randomly action a, then act

     IF (pick up a trash)
   r=M (M is any positive number)
     ELSE
   r=0 (don't pick up)

     Acquire present state s
     Acquire next state s'
     Acquire max Q(s', a') in the next state

     Renew Q-factor
          Q(s,a) ← (1−α)Q(s,a) + α(r + γ max Q(s',a'))

   IF (All trash on the field is picked up)
    Initialize state of field
}
```

F. Action Decide

The trash collection carries out the action based on the action-value function. The action rule of a trash collection agent is as shown in Table 2. The action part is replaced by the action-value function.

TABLE II
STATE AND ACTION-VALUE OF THE ENVIRONMENT

state	action-value function			
	up	down	left	right
A	Q(A, up)	Q(A, down)	Q(A, left)	Q(A, right)
B	Q(B, up)	Q(B, down)	Q(B, left)	Q(B, right)
		……		
Z	Q(Z, up)	Q(Z, down)	Q(Z, left)	Q(Z, right)

The action of the agent is decided based on the value of the action-value function for movement in each direction. For example, the action is decided by the values Q(B, up), Q(B, down), Q(B, left) and Q(B, right) when the state of the environment is B. As a method for deciding the action from these four values, there are two techniques, are as follows:

(1) ε-greedy policies

In ε-greedy policies, most of time the agents choose an action that has a maximal estimated action value, but when the probability is ε the agents instead make their selection randomly. That is, all non-greedy actions are given the minimal probability of selection, and the remaining bulk of the probability is given to the greedy action.

Although ε-greedy action selection is an effective and popular means of balancing exploration and exploitation in reinforcement learning, one drawback is that during exploration the agents choose equally among all action. This means that the likelihood of choosing the worst action is as high as that of choosing the next-to-best action.

(2) Softmax policies

Softmax policies vary the action probabilities as a graded function of the estimated value. The greedy action is still given the highest selection probability, but all of the other actions are ranked and weighted according to their value estimates.

IV. EXPERIMENT RESULTS AND DISCUSSION

Using the proposed system, an experiment on the learning of the agent was carried out. The size of the field was 15×15. There were 10 pieces of trash and five agents. An agent moves to the one square of relative position - up, down, right, and left - in one step. In addition, each agent has the ability to perceive the condition of 2 squares of the surrounding environment. The task requires the trash to be picked up. A reward is given when an agent picks up a piece of trash. Also, the action-value function Q is shared between agents.

A. Action Selection Policies Comparison

The effect of ε-greedy policies and softmax policies on the learning of the agent is shown in Figure 5. The task completion step decreases with the increase of the learning step when ε-greedy policies are used. Conversely, the task completion step increases, when the softmax policies are used. This is because the action value for each direction is equalized with the increase of the learning step, and the difference in the probability of movement in each direction decreased when the softmax policies were used.

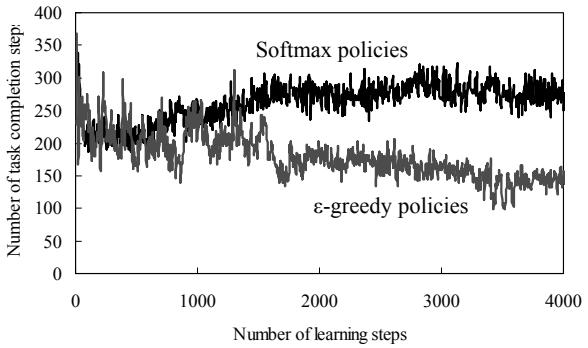

Fig. 5. Comparison of policies. The task completion step decreases with the increase of the learning step when ε-greedy policies are used. Conversely, the task completion step increases, when the softmax policies are used.

B. Effects of Step-size Parameter

The effects of the step-size parameter on Q-factor update were examined. The step-size parameter indicates the degree that is renewed when the learning-value function is updated. The step-size parameter is a small positive fraction that influences the rate of learning. The step-size range is from 0.1 to 1. The learning results for step sizes of ε= 0.1, 0.4 and 0.7 are shown in Figures 6 and 7.

In Figure 6, when the step size, α, is 0.4, there are smaller task completion steps than when α is 0.1. However, in Figure 7, the learning has not been stabilized significantly by the time the step size reaches 0.7. Therefore, skillful learning is not possible when the step size too is set to be too large.

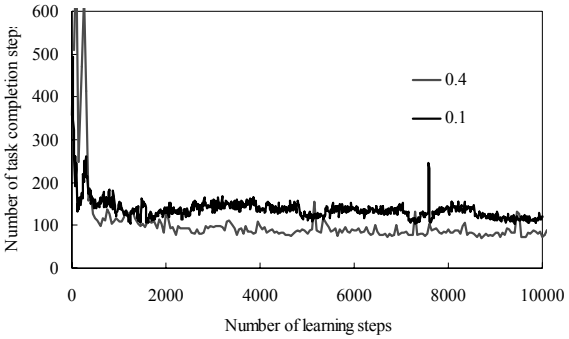

Fig. 6. Step-size parameter indicates the degree. The step-size parameter is a small positive fraction that influences the rate of learning. The learning results for step sizes of ε = 0.1 and 0.4.

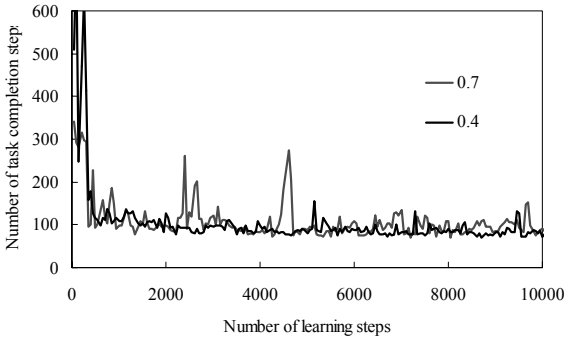

Fig. 7. Step-size parameter indicates the degree. The step-size parameter is a small positive fraction that influences the rate of learning. The learning results for step sizes of ε = 0.4 and 0.7.

V. Conclusion

In the present study, we attempted to acquire a cooperative action rule for agents through the use of Q-learning. We used the Q-learning based on the fact that the action-value is high when agents act cooperatively, and each agent acts cooperatively by learning to finish trash collection as fast as possible without any indication.

The task completion step decreased with the increase of the learning step when ε-greedy policies were used. Conversely, the task completion step increased with the increase of the learning step when softmax policies were used. This is because the action value for each direction is equalized with the increase of the learning step, and the difference in the probability of movement in each direction decreased when the softmax policies were used.

In present study, the number of states is very large because all of the environments in which the agents appear during learning are used. In future research, we intend to improve the representation methods of the action rule in order to solve this problem.

References

[1] Michael Wooldridge: Intelligent Agent, *Multiagent Systems,* edited by Gernard Weiss, 2000

[2] S. Nolfi and D. Floreano, Co-evolving predator and prey robots: do 'arm races' arise in artificial evolution? *Artificial Life 4* (*4*), 1998, pp.311-335

[3] K. C. Jim and C. Lee, Talkin helps: evolving communicating robots for the predator- prey pursuit problem, *Artificial Life*, No. 6, 2000, pp.237-254.

[4] Changjiu Zhou, Robot learning with GA-based fuzzy reinforcement learning agents, *Information Science*, Vol. 145, 2002, pp.45-68.

[5] K. Fujita and H. Matsuo, Multi-agent Reinforcement Learning with the Partly High-dimensional State Space, *The transactions of the institute of electronics, information and communication engineers*, Vol. J88-D-I No. 4, 2005, pp.864-872.

[6] M. C. Xie, "Cooperative Behavior Rule Acquisition for Multi-agent Systems Using a Genetic Algorithm", Proceedings of the IASTED International Conference on Advances in Computer Science and Technology, 2006, pp.124-128.

[7] N. R. Jennings, K. Sycara, and M. Wooldridge, A roadmap of agent research and development, *Autonomous Agent and Multi-Agent Systems*, 1998, pp.7-38.