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

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



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



5

Cooperative Behavior Rule Acquisition for Multi-Agent Systems by Machine Learning

Mengchun Xie
Wakayama National College of Technology
Japan

1. Introduction

Multi-agent systems are the systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks (Wooldridge, 2000). In multi-agents systems, each agent must behave independently according to its states and environments, and, if necessary, must cooperate with other agents in order to perform a given task. Multi-agent systems have more robustness and flexibility than conventional centralized management one. However, it is difficult to beforehand predict the action of the agent and give the action rule for the multi-agent systems, because the interaction between agents is complicated.

The acquisition of an autonomous agent's intellectual action rule is a very interesting topic. Recently, extensive research has been done on models such as Robocap (Stone & eloso, 1996, Matsubara et al., 1998). Studying computation models of cooperative structure to accomplish a given task is known to be a difficult problem (Jeong & Lee, 1997).In the field of self-learning reactive systems, it is not even desirable to have a clear idea of a computational model. Thus, being adaptable, autonomous agents imply minimally pre-programmed systems. Numerous studies regarding autonomous agents in the multi-agent systems have been conducted. Nolfi and Foreano (Nolfi & Floreano, 1998) 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 (Jim & Lee, 2000) evolved the autonomous agents with a genetic algorithm. Zhou (Zhou, 2002) 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 (Fujita & Matsuo, 2005) 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.

In this chapter, we focused on the problem of "trash collection", in which multiple agents collect all trash as quickly as possible. The goal is for multiple agents to learn to accomplish a task by interacting with the environment and acquiring cooperative behavior rules. Therefore, for a multi-agent system, we discuss how to acquire the rules of cooperative action to solve problems effectively.

First we used a genetic algorithm as a method of acquiring the rules for an agent. Individual coding (definition of the rule) methods are performed, and the learning efficiency is evaluated.

Second, we construct the learning agent using the Q-learning which is a representative technique of reinforcement learning. Q-learning is a method to let an agent learn from delayed reward and punishment. It is designed to find a policy that maximizes for all states. The decision policy is represented by a function. The action vale function is shares among agents.
The third, we concentrate on an application of Multi-agent systems to disaster relief using Q-learning. We constructed a simplified disaster relief multi-agent system and acquired action rules by Q-learning. We then observe how the autonomous agents obtain their action rules and examined the influence of the learning situations on the system. Moreover, we discuss how the system was influenced by learning situation and the view information of the agent.

2. Cooperative action of multi-agent

2.1 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 (Jennings et al., 1998).
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. Occasionally, in order to proceed with a task in a changing environment, multi-agents must judge precise situations in order to make adaptive moves. In order to realize cooperative action, it is necessary to perform actions based on the rule that working as a group takes priority over the actions of the individual. Generally speaking, individual action is natural, whereas group action is acquired by learning to accomplish goals through cooperative actions.
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.

2.2 Establishment of the problems environment

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

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

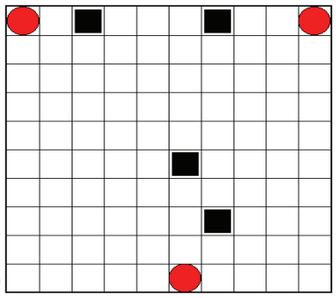
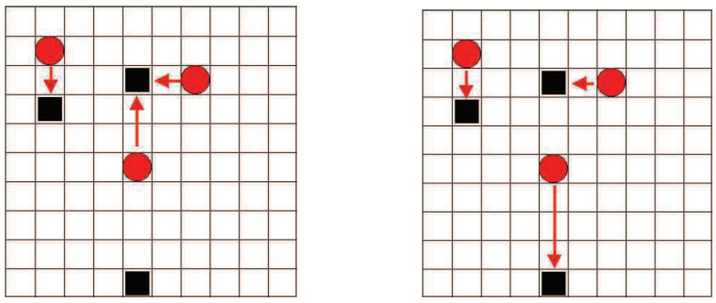


Fig. 1. An example of the problem environment

For the trash collection problem, an example of agent actions is shown in Figure 2 for cooperative behavior and non-cooperative behavior. If the priority of each agent is to act to increase individual profit, by collecting as much trash as possible, then the agents are expected to act as shown in Figure 2(a). However, if each agent has a complete understanding of the entire situation and has the priority of collecting all of the trash as fast as possible, then agents are efficient and act as shown in Figure 2(b). In this case, the priority action of an agent is called cooperative action. The goal of the present study is to have agents acquire, through learning, the rules that are necessary to take cooperative action.



(a) Non-cooperative action (b) Cooperative action

Fig. 2. Example of agent action

3. Acquisition of cooperative rule using GA

In this section, we use a genetic algorithm (GA) to acquire the cooperative action rule (Goldberg, 1989, Holland, 1975). The number of steps taken to collect all of the trash is used to evaluate the rule. We used a GA, which agents use to decide actions, to evolve and acquire the cooperative rules needed.

3.1 Structure of the system

The system to acquire the cooperative rules is described by the action rules of agents. An agent realizes the state from the field's information, compares the state with the internal rules, and then decides on its next action. The renewal of the rules is carried by GA. The structure of the system is as follows:

1. Development of the field - arrange the decided number of pieces of trash and agents on the established field.
2. Each agent determines its next move - each agent decides the direction of its next move based on the situation of the field, the internal rules and knowledge.

3. Movement of agents - all agents move simultaneously. A piece of trash is collected if an agent occupies the same position.
4. Repeat Items (2) ~ (3) until all of the trash is collected. Repetition of Items (2) ~ (3) is considered as one step.
5. When collection is completed, evaluate the actions of the agents based on the number of steps required to complete the action. Agents also learn on the basis of this evaluation.
6. The experiment is continued by returning to Item (1), and agents continue to learn.

3.2 Acquisition of cooperative rule by GA

The behavior of each agent is governed by a rule. A rule is a condition-action rule of the form. The condition part of the rule represents the situation of the field (the arrangement of agents and trash). The action part of the rule indicates the piece of trash toward which the agent moves at the next time step.

An agent compares the present state of the field with the action part of the rules and decides a rule that is closest to the present state of the field. An agent executes the decided action.

Because multiple agents are present in a dynamic environment, it is necessary for each agent to have the ability to learn from the environment in order to develop cooperative action. In this section, the acquisition of cooperative rules is performed using a GA. We propose two methods of individual representation are as follows:

Representation Method 1: one rule for one individual

This representation method takes one rule as a single individual, as shown in Table 1.

Distance from Agent A					Next Movement:
Agent1	Agent2		Trash1	Trash2	
Distance from Agent B					The No. Trash watch etc.
Agent1	Agent2		Trash1	Trash2	
Distance from Agent C					
Agent1	Agent2		Trash1	Trash2	

Table 1. Structure of the individual by Representation Method 1

The condition part of the rule represents the assumed field state at a distance. That is to say, each agent identifies the positions of agents to a central agent. The action part is the hamming distance from the central agent to each trash and to others agents rearranged in order of ascending proximity. The action part represents the piece of trash toward which the agent will move next.

Representation Method 2: Multiple rules for one individual

The form of a rule is as described in Method 1; however, one individual consists of a group of rules (Table 2).

Evaluation of the individual

Representation Method 1 takes one rule as a single individual for GA. An agent chooses a rule from the population consisting of such individuals and decides its next move. For the population of individuals, genetic operations, such as crossover, mutation and selection, are performed. The evaluation of each individual agent is based on that next move and the time required to finish collecting one piece of trash. The evaluation value of the individual agent is given a constant value when an agent applies the rule and collects one piece of trash. Once

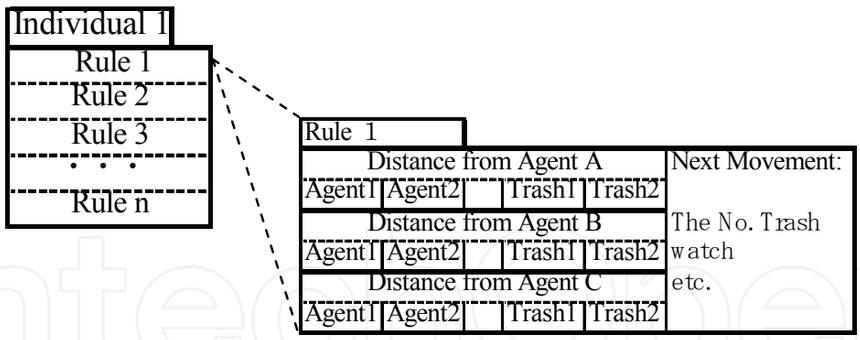


Table 2. Structure of the individual by Representation Method 2

collection is completed, the result is compared to the previous collection and the required number of steps is evaluated. The fewer the steps, the better the result.

For Representation Method 2, an agent has some individual (group of rules). Evaluation of individuals is performed based in the decrease in the number of steps required for collection compared to the previous collection by giving in proportion to the degree. Trash collection is performed once by one individual, and each individual is evaluated by changing the individual used for the trash collection. After finishing the evaluation of all individuals, a GA is selected based on the above evaluation.

3.3 Results and discussion

Using the representation method of the proposed individual, the acquisition of the cooperative action of the multi-agent system was attempted using a GA. In the experiment, the size of the field was 15 × 15. There were 15 pieces of trash and three agents. The average results for 20 repetitions until all of the trash was collected by the agents for the same initial distribution are shown below.

An example experimental result

An experimental result is shown in Figure 3. In this result, the agents are denoted by ● and pieces of trash are denoted by ■, and the lines indicate the tracks of agents.

When the cooperative action is learned using the Representation Method 1, each agent had 50 rules. In other words, the population size of the GA is 50. The number of steps required in order to collect all trash differs in every generation, but fluctuates in the range from 20 to 60 steps. That is because there are different rules taken for every generation by the GA. The shortest step of 20 was obtained by trial-and-error. As a result of the learning using the GA, this elite individual could be obtained in approximately 30 generations (Xie, 2006).

In Representation Methods 2, each agent has 10 rule groups. In short, the population size of the GA is 10. One rule group consists of 50 rules. That is, the results for the number of steps required for 10 individuals fluctuate in the range from 20 to 50 steps, which is fewer than that for Representation Method 1.

4. Cooperative behavior acquisition using Q-learning

In this section, we discuss the cooperative behavior acquisition of agents using the Q-learning which is a representative technique of reinforcement learning. Q-learning is a method to let an agent learn from delayed reward and punishment. It is designed to find a policy that maximizes for all states.

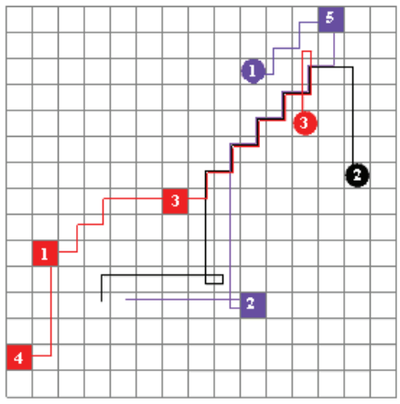


Fig. 3. An example experimental result

4.1 Agent and environment

The agent decides the action based on the condition of perceived environment, and some changes are caused environment by the action. Therefore, agent and environment are relationships of the interaction.

An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state. Most moderately complex environments are inaccessible. The more accessible can environment is, the simpler it is to build agent to operate in it (Alex et al., 2005). In the meantime, when the environments are inaccessible, the information which can be perceived from the environment is limited, and it is inaccurate, and it entails delay.

When there is only one the agent, the environment is static. However, it is not always static for the environment, when other agent exists. Because, the condition of the environment may be due to be changed by other agent.

4.2 The composition of agents

In this section, the composition and each function of the agent are described. The structure of an agent is shown in Figure 4. The figure has shown each component of the agent, and the transmission of information has been expressed in the arrow. First, the agent perceives the condition of the environment in the detector. Next, the rule is compared with the condition that it was perceived by the acting decider, and the action is decided. Finally, the action is carried out in effector.

The detector of the agent can't always perceive the condition of the environment completely. It is similar to there being a limit at the human visual acuity. Therefore, an agent can perceive only a part of condition of the environment or the state just in the neighborhood.

The whole environment and the environment which an agent perceived are respectively shown in Figure 5 and Figure 6. Figure 5 shows whole of the arranged field of agents. The round sign is an agent, and the number is the label of the agent in Figure 6. The frame which each agent is enclosed with is the range that an agent can perceive, and it is the range that it has been limited. Figure 6 is the environment which each agent perceived. The symbol X is an agent itself, and the number is other agents.

The conditions (position) of all the agents are included in the whole environment as like Figure 5. An agent 1 and an agent 2 perceive the respectively different environment.

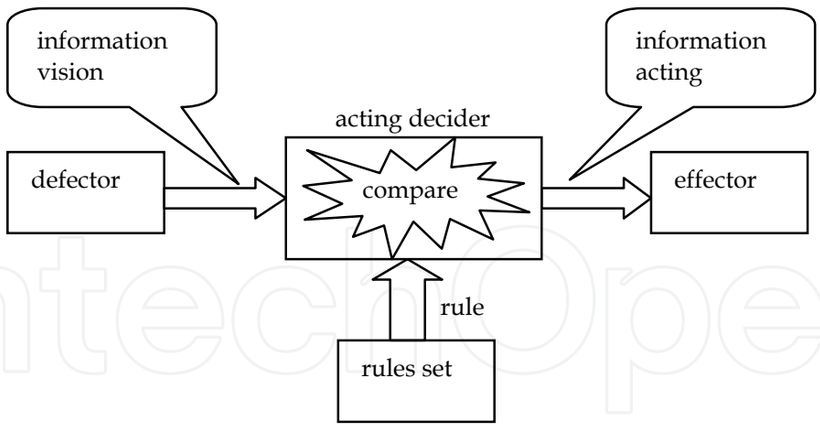


Fig. 4. Composition of an agent

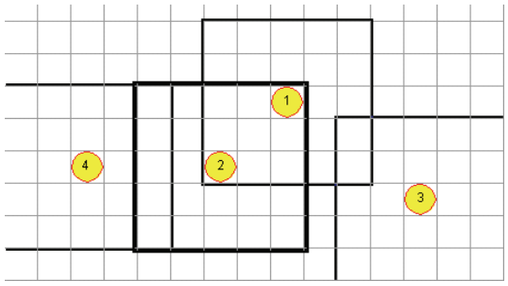


Fig. 5. The whole environment

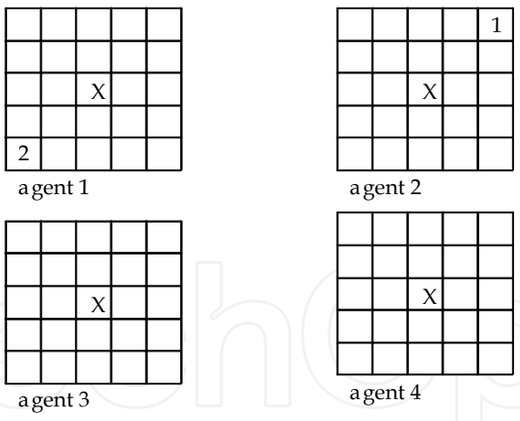


Fig. 6. The environment which each agent perceives

However, though each agent is located in the different environment, an agent 3 and an agent 4 perceive it as a condition of the same environment. This problem is called the imperfect perception problem, and it becomes one of the important problems in the agent design. It is considered that the visual field range of the agent is increased in order to avoid this problem.

The behavior of each agent is governed. A rule is a condition-action rule of form. The condition part of the rule represents the situation of the environment which 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 which the defector perceives increases, the number of rules increases. An example of the condition-action rules is shown in Table 3.

Situation of environment	action
200000000000X000000010020	up
002000010000X000000020000	down
000000000000X000000002000	left
002000000000X000000001000	right
000000000000X000002000000	down

Table 3. An example of rules

The effector decides the action by the rule. For example, if the condition of the environment was perceived with "000000000000X000002000000", the action to the "down" is chosen on the action of the agent in Table 3 rules.

4.3 Q-Learning

A prominent algorithm in reinforcement learning is the Q-Learning. In an environment of finite Markov decision process (MDP), the goal is to find the optimal policy in each state visited to maximize the value of a performance metric, e.g , long-run discounted reward using Q-Learning (Schleiffer, 2005).

A policy defines the action in every state visited. Let A denote the set of actions allowed and let S denote the set of states. We will assume that both A and S are finite. Let r_{t+1} denote the immediate reward earned in going from state s_t to state s_{t+1}, when action $a_t \in A$ is selected in state s_t, and let γ denote the discounting factor. Then with s_t as the starting state, for the policy π, the total discounted reward - generally referred to as discounted reward - is defined as (Bradtkes & Duff, 1994, Parr & Russell, 1998):

$$R_t^{\pi} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T = \sum_{k=1}^{T-t} r^{k-1} r_{t+k} \tag{1}$$

where T denotes the finished time of the system. If the Markov chain associated with the policy π is irreducible and a periodic, the discounted reward, R_t^{π} , is independent of the starting state s_t.

In Q-Learning, which can be implemented in simulators or in real time, the algorithm iterates are the so-called Q-factors. $Q(s,a)$ will denote the Q-factor for state $s \in S$ and $a \in A$. The updating in Q-Learning is asynchronous. One iteration of the algorithm is associated with each state transition in the simulator. In one transition, only one Q-factor is updated. Let $Q(s,a)$ denote the Q-factor for state s and action a. 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:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{2}$$

where $0 \le \alpha \le 1$, 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.

4.4 Action acquisition by Q-learning

The action-value function is renewed in which the agent acts. When an agent pick up a trash, the reward which affects the renewal of the action value is given for the agent. And, the field is initializes, when all trashes on the field are picked up. But, action-value function and agent learning number of step would not be initialized. It is repeated to learning number of step decided.

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 trashes on the field are picked up）
        Initialize state of field
}
```

The trash collection carries out the action based on action-value function. Action rule of trash collection agent is shown as Table 4. Action part is replaced with the action-value function.

The action of the agent is decided by the value of action-value function moved to the each direction. For example, the action is decided by the each value of *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 (Watkins, 1989, Bridle, 1990):

1. ε-greedy policies

It is a method that most of time they choose an action that has maximal estimated action value, but with probability ε they instead select at random. That is, all nongreedy action 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 when it explores it chooses equally among all action. This means that it is as likely to choose the worst-appearing action as it is to choose the next-to-best action.

2. Softmax policies

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

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

Table 4. Environment 's state and action-value

4.5 Experiment results and discussion

Using the system as has been mentioned in 4.2, the 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 in 1 step. And, each agent has the ability of perceiving the condition of the environment of the circumference of 2 squares. It is called a task that the trash is picked up. The reward is given, when each agent picked up the trash. And, action-value function Q is shared between each agents.

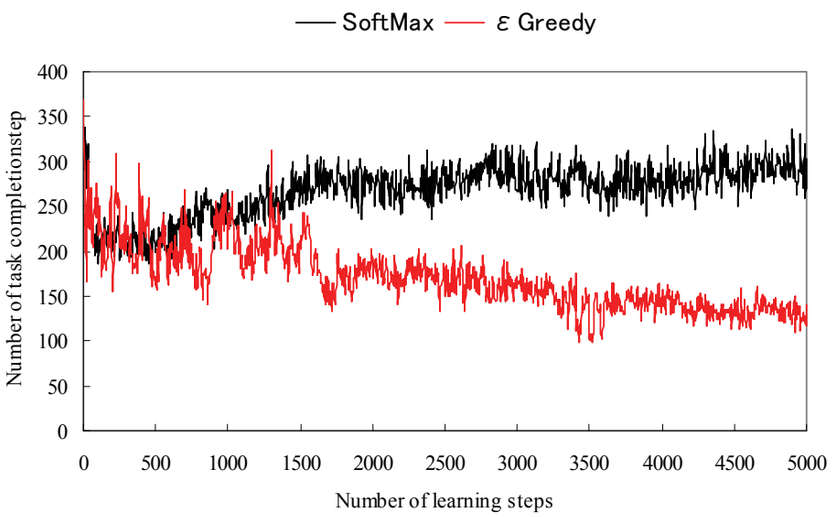


Fig. 7. Comparison policies

The effect of ε-greedy policies and softmax policies on the learning of the agent is shown in the Figure 7. The task completion step decreases with the increase of the learning step, when ε-greedy policies were used. However, it reversely increases on the task completion step, when the softmax policies were used. This is because the action value to each direction is equalized with the increase of the learning step, and the difference of the probability moved to the each direction decreased using the softmax policies.

The effects of step-size parameter, reward size and discount-rate parameter in update Q-factor were examined.

The step-size parameter shows which degree is renewed when the learning-value function is updated. It is a small positive fraction which influences the rate of learning. The range of the step-size is from 0.1 to 1. Learning results of different step-size α = 0.1, 0.4 and 0.7 were shown in Figure 8. In Figure 8, when step-size α is 0.4, there are smaller tasks completion

steps than α is 0.1. However, in Figure 8, the learning has not been very much stabilized in the time of step-size 0.7. From this fact, it was proven that it can not skillfully learn when the step-size too is greatly set.

The reward size is a value given when the agent picked up a trash. It is shown in Figure 9 as a result of the learning as the reward size is set to 10, 100 and 1000. In this Figure, it was proven that the reward size does not affect the learning. Then in this chapter, we use the same reward size value for all experiments. It seems to be important to give the reward than its value.

The discount rate shows the degree of reference of Q-factor in next state. The learning results are shown in Figure 10 as the discount rate was set from 0.1 to 0.9. It was proven that it could not learn skillfully, when the discount rate is too small.

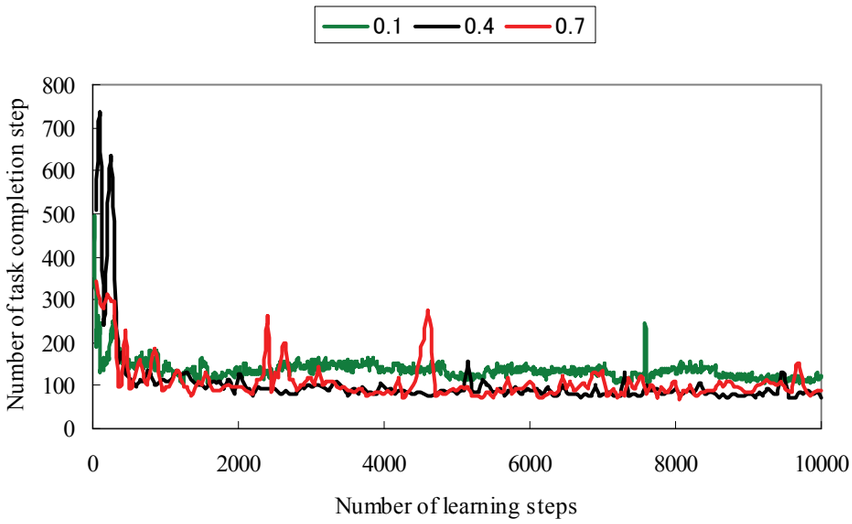


Fig. 8. Step-size comparison

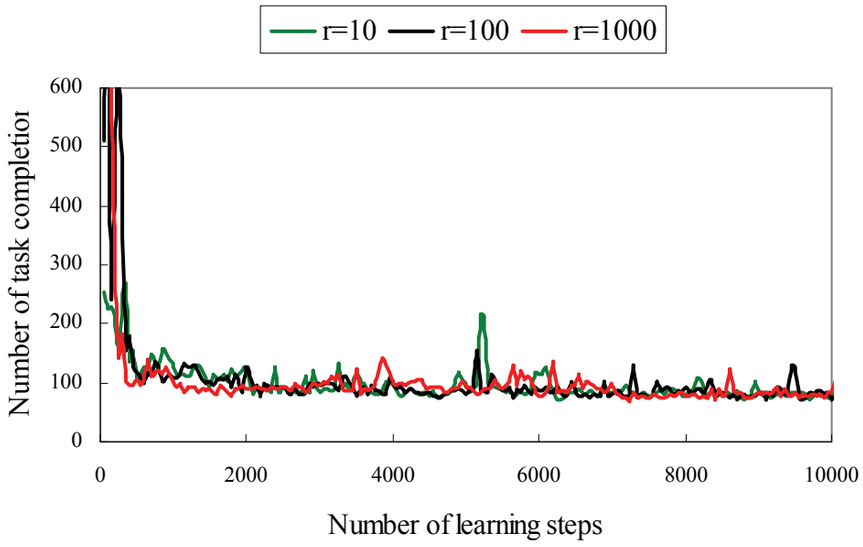


Fig. 9. Comparison of reward size

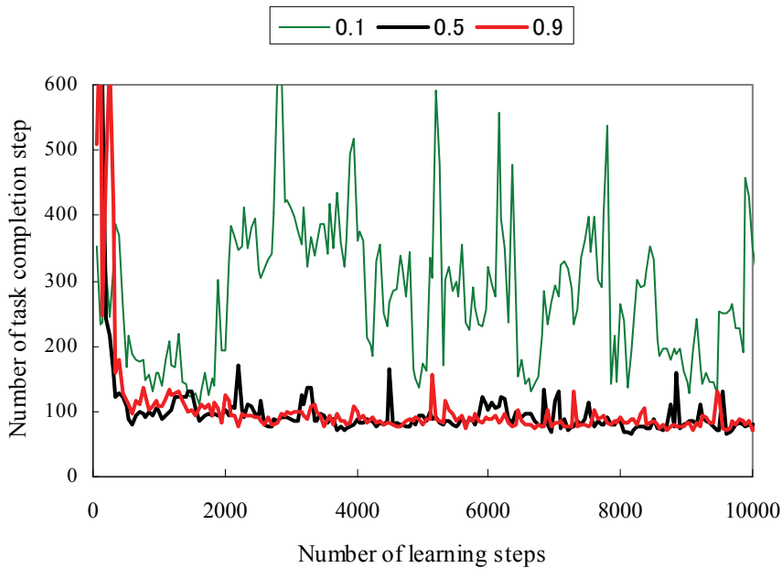


Fig. 10. Comparison of discount rate

5. Disaster relief multi-agent systems using Q-learning

In this section, we concentrate on an application of Multi-agent systems to disaster relief using Q-learning. We constructed a simplified disaster relief multi-agent system and acquired action rules by Q-learning. We then observe how the autonomous agents obtain their action rules and examined the influence of the learning situations on the system. Moreover, we discuss how the system was influenced by learning situation and the view information of the agent.

5.1 Disaster relief multi-agent systems

We consider the "disaster relief" problem, in which the injured are placed on a field of fixed size and agents must rescue the injured persons as fast as possible. It aims to rescue the injured person efficiently as the whole system when the agent is achieving own target. This can be considered as a multi-agent system (Xie & Okazaki, 2008).

5.2 Representation of the perceived environment

An agent has constant view in the disaster relief multi-agent system. The agent can perceive the surrounding environment and can recognize other agents and injured individuals within its range of vision. When an agent has a visual field N, it is assumed that the agent can move N steps.

An example within the range of vision is shown in Figure 11 Agent 1's visual field is 2, and Agent 1 can recognize two injured individuals within its visual field.

In the disaster relief multi-agent system, in order to handle visual field information that the agent receives from the environment by reinforcement learning, it is necessary that pattern is provided of visual field information by replacing with the numerical value. The agent expresses the information of other agents and injured individuals within its visual field range numerically and arranges this information in a fixed order. This string is called the visual field pattern.

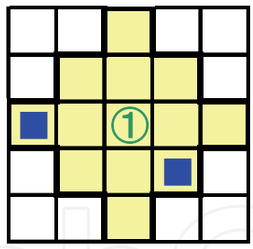
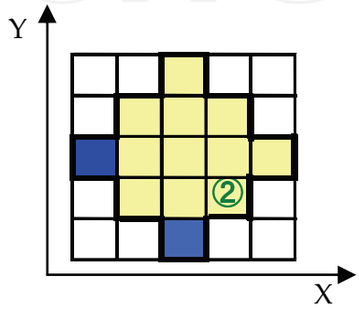


Fig. 11. Range of vision of an agent



position of agent 1 is centered at (0,0)

coordinate (-2, 0) P (injured)
coordinate (0, -2) P (injured)
coordinate (1, -1) A (agent 2)

visual filed pattern: -20P0-2P1-1A

Fig. 12. Creation of visual field information pattern

The technique for creating the pattern of the visual field information is shown in Figure 12. First, Agent 1 perceives the relative position of the other agents and injured individuals in a fixed order, and the coordinates of Agent 1 are assumed to be (0, 0). In this example, the relative positions and patterns are as follows:

at (-2,0) an injured person (P) >> -20P

at (0,-2) an injured person (P) >>0-2P

at (+1,-1) other agent 2 (A) >>1-1A

The string -20P0-2P1-1A is the visual field pattern in proportion to recognized visual field information that there are some injured individuals and other agents. Here, the visual field range is limited to 9. Compared to the case in which the information of all masses in the visual field range is acquired, there is the merit in which the string of the visual field pattern shortens on this technique.

In reinforcement learning, the wider the range of the vision pattern, the more information can be recognized. However, reinforcement learning has some disadvantages in that the number of states of the visual field pattern increase and the learning speed is decreased.

5.3 Experiment and discussion

In the disaster relief multi-agent system constructed using Q-learning, experiment and consideration are carried out by changing the learning conditions. One movement of the agent is set to correspond to one step, and all agents are assumed to move simultaneously. The action value functions are shared between agents. In all of the experiments, the field size is 10×10, the number of rescue agents is three, the step size α is 0.1, the discount rate γ is 0.9, the probability ε is 0.1, and the number of iterations is 100.

5.3.1 Effect of number of learning iterations

The relationship between the number of steps, which depended on the rescue, and the number of injured individuals that are rescued varied with the number of learning iterations of the agent, as shown in Figure 13. Figure 13 shows the learning results for 10, 100, 1,000, and 10,000 learning iterations. There were 10 injured individuals and three agents. In one step, each agent moves one square to an adjacent position, namely, up, down, right, or left. The visual field of each agent is 2.

The horizontal axis shows the number of individuals rescued, and the vertical axis shows the number of steps, which depends on the specific conditions of the rescue. As the number of learning iterations increases, the number of steps required in order to effect a rescue decreases. This is an effect by the learning of agents. Moreover, the number of steps required to effect rescue increased rapidly, when the number of injured individuals exceeded eight. Since there is no information that agent is obtained and agents choose the random action, when the injured were in visual field outside of all agents.

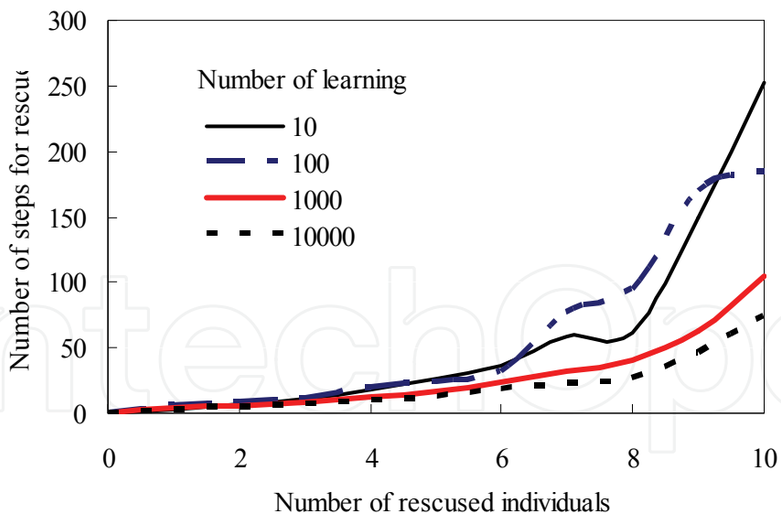


Fig. 13. Effect of number of learning iterations

5.3.2 Density of the injured

The relationship between number of steps required for rescue and the rescue rate of injured individuals is shown in Figure 14, where the injured of the different density were arranged. The number of learning iterations is 10,000, the visual field is 3, and number of injured individuals is 5, 10, and 20.

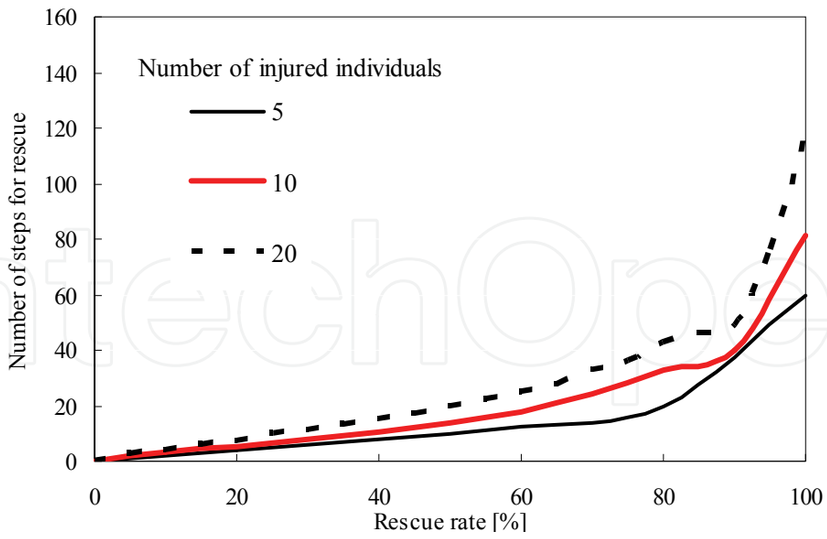


Fig. 14. Effect of density of injured

In Figure 14, the horizontal axis shows the rescue rate, i.e., the ratio of the number of individuals rescued to the total number of injured individuals in the entire field. The experimental results revealed that the number of steps necessary to effect a rescue increased as the number of injured individuals increased. The frequency with which the rescue agent moves on the field is considered to have increased as the number of injured individuals increased.

5.3.3 Visual field range

When the visual field range of the agent is changed, the relationship between the number of individuals rescued and the number of steps required for rescue is shown in Figure 15. The experimental conditions assume that the number of iterations if 10,000, the number of injured is 10, and the visual field range is varied as 2, 3, and 4.

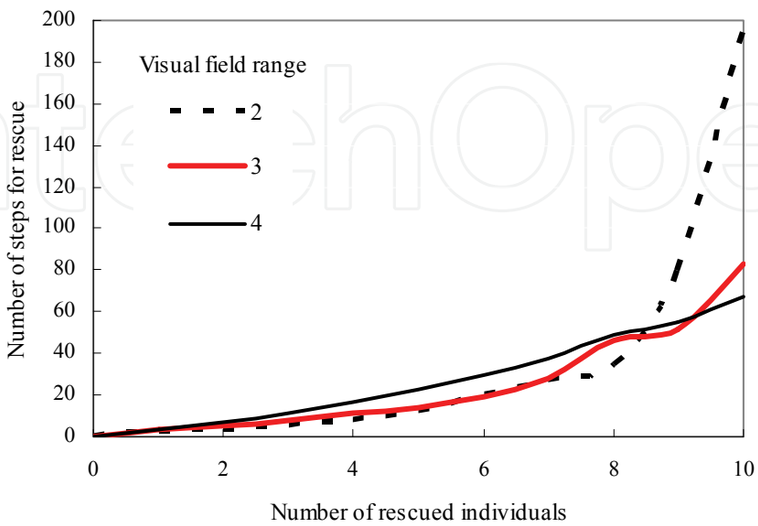


Fig. 15. Effect of visual field range

This figure shows that the fewest steps in which all injured individuals could be rescued occurred when the visual field range was 4. This is because the visual field range of the rescue agent is wide, and the situation in which injured individuals are located outside the visual fields of all rescue agents rarely occurs. However, in some cases, a visual field range of 2 allowed the quickest rescue when the number of injured individuals was small. The reason for this is that the visual field pattern number decreased, because the visual field range is narrow, and a more accurate action value function could be obtained. Moreover, it became a result of combining both characteristics of visual field range 2, 4 in visual field range of 3.

5.3.4 Acquisition of the cooperation action rule

It is possible that the agent efficiently rescued after it carried out Q-learning because the cooperation action was accurately carried out between agents. To confirm the cooperative behavior of agents, a number of action rules acquired from the action value function were examined after 10,000 learning iterations (Figure 16).

In this Figure, agents are denoted by ● symbols, and the injured individuals are denoted by ■ symbols. In rule 1, for Agent 1, the probability of moving to the left, where nearest injured individual 1 is located, increased most. However, in rule 2, the probabilities of moving toward the bottom and the right, where an injured individual 2 is located, were highest, because another agent, Agent 2, is positioned near the other injured individual 1. As a result, Agent 1 performed a cooperation action.

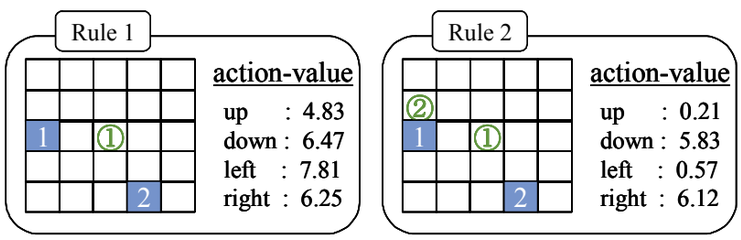


Fig. 16. Example of an action rule

6. Conclusions

In this chapter, we attempted to acquire a cooperative action rule for agents through the use of a GA and Q-Learning, and carried out an application of Multi-agent system to disaster relief by Q-Learning.

We used two representation methods to describe individual agents and to learn cooperative action through a GA. We used the GA based on the fact that each agent acts cooperatively by learning to finish trash collection as fast as possible without any indication. However, further research on the individual evaluation of GA and crossover methods is required. The rule and individual of the present study are intended to learn the most efficient action in response the arrangement of the field by repeating tasks using the same arrangement. However, this makes the learned rules less efficient if the arrangement of the field is different than that used in training. At this stage, we were not able to determine clearly whether agents engaged in cooperative action.

We use the Q-Learning based on the fact that the action-value is high when agents will act cooperatively, and each agent acts cooperatively by learning to finish trash collection as fast

as possible without any indication. The task completion step decreases with the increase of the learning step, when ε-greedy policies were used. However, it reversely increases on the task completion step, when the softmax policies were used. This is because the action value to each direction is equalized with the increase of the learning step, and the difference of the probability moved to the each direction decreased using the softmax policies. In present, the number of states is very large because all the environments where the agent appeared under learning are used. In future research, we intend to improve the representation methods of action rule to solve that problem.

In the disaster relief multi-agent system, the number of steps required to effect the rescue of an injured individual decreased with the increase in the number of learning iterations. This is considered to be the effect of the learning of agents. However, the number of steps, which depends on the rescue situation, increased rapidly when the number of injured individuals in the field decreased. When the injured individuals are outside the visual field of all rescue agents, there is no information available to the agents, and so the agents perform random actions. The effect of density and visual field range of the injured individuals on the system was also examined. The number of environment patterns becomes large because, at present, all of the environments that the agent considers during learning are considered. In the future, the number of environment patterns should be decreased.

7. References

Wooldridge, M. (2000). Intelligent Agent, in: *Multiagent Systems,* Gernard Weiss, 27-73, The MIT Press, 0262731312, Cambridge, Massachusetts

Stone, P. & eloso, M. (1996). Using Machine learning in the Soccer Server, *Proc. of IROS-96 Workshop on Robocup*

Matsubara, H.; Frank, I. & Tanaka, K. (1998). Automatic Soccer Commentary and RoboCup, *The 2nd Proceedings of RoboCap Workshop*

Jeong, K. & Lee, J. (1997). Evolving cooperative mobile robots using a modified genetic algorithm, *Robotics and Autonomous Systems*, Vol. 21, 197-205

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

Jim, K.C. & Lee, C. (2000). Talkin helps: evolving communicating robots for the predator-prey pursuit problem, *Artificial Life*, No.6, 237-254

Zhou, C. (2002). Robot learning with GA-based Fuzzy reinforcement learning agents, *Information Science*, Vol. 145, 45-68

Fujita, K. & Matsuo, H. (2005). 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, 864-872

Jennings, N.R.; Sycara, K. & Wooldridge, M. (1998). A roadmap of agent research and development, *Autonomous Agent and Multi-Agent Systems*, 1:7-38

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, 0201157675, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press

Xie, M.C. (2006). 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*, pp.124-128

Alex, V.; Conradie, E. & Aldrich, C. (2005). Development of neurocontrollers with evolutionary reinforcement learning, Computers & Chemical Engineering Vol.30, 1-17

Schleiffer, R. (2005). An intelligent agent model, *European Journal of Operational Research*, Vol.166, No.1, pp.666-693

Bradtkes, S. J. & Duff, M. O. (1994). Reinforcement Learning Method for Continuous- Time Markov Decision Problems, *Advances in Neural Information Processing Systems 7*, pp.393-400

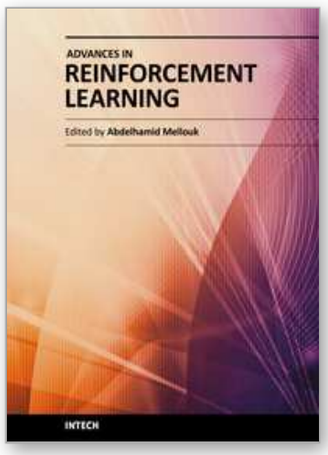
Parr, R. & Russell, S. (1998). Reinforcement Learning with Hierarchies of Machines, *Advances in Neural Information Processing Systems 10*, pp.1043-1049

Watkins, C. H. (1989). Learning from Delayed Rewards, *Ph.D. thesis*, Cambridge University

Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimates of parameters, Advances in Neural Information Processing Systems: *Proceedings of the 1989 Conference*, pp.211-217

Xie, M.C.& Okazaki, K. (2008). Application of Multi-Agent Systems to Disaster Relief Using Q-Learnin, *Proceedings of the IASTED International Conference on Software Engineering and Applications* ,pp.143–147

IntechOpen



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:

Mengchun Xie (2011). Cooperative Behavior Rule Acquisition for Multi-Agent Systems by Machine Learning, 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/cooperative-behavior-rule-acquisition-for-multi-agent-systems-by-machine-learning

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

© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen