Self-evaluated Learning Agent in Multiple State Games

Koichi Moriyama[1] and Masayuki Numao[2]

[1] Department of Computer Science, Tokyo Institute of Technology.
2–12–1, Ookayama, Meguro, Tokyo, 152–8552, Japan
moriyama@mori.cs.titech.ac.jp
[2] The Institute of Scientific and Industrial Research, Osaka University.
8–1 Mihogaoka, Ibaraki, Osaka, 567–0047, Japan
numao@ai.sanken.osaka-u.ac.jp

Abstract. Most of multi-agent reinforcement learning algorithms aim to converge to a Nash equilibrium, but a Nash equilibrium does not necessarily mean a desirable result. On the other hand, there are several methods aiming to depart from unfavorable Nash equilibria, but they are effective only in limited games. Based on them, the authors proposed an agent learning appropriate actions in PD-like and non-PD-like games through self-evaluations in a previous paper [11]. However, the experiments we had conducted were static ones in which there was only one state. The versatility for PD-like and non-PD-like games is indispensable in dynamic environments in which there exist several states transferring one after another in a trial. Therefore, we have conducted new experiments in each of which the agents played a game having multiple states. The experiments include two kinds of game; the one notifies the agents of the current state and the other does not. We report the results in this paper.

1 Introduction

We investigate the use of reinforcement learning in a multi-agent environment. Many multi-agent reinforcement learning algorithms have been proposed to date [6, 4, 2, 13, 1]. Almost all of them aim to converge to a combination of actions called Nash equilibrium. In game theory, a Nash equilibrium is a combination of actions of rational players. However, this combination is not optimal in some games such as the prisoner's dilemma (PD) [14]. There are, on the other hand, several methods aiming to depart from undesirable Nash equilibria and proceed to a better combination by handling rewards in PD-like games [7, 8, 23]. However, since they use fixed handling methods, they are inferior to normal reinforcement learning in non-PD-like games.

In our previous paper [11], we have constructed an agent learning appropriate actions in both PD-like and non-PD-like games through self-evaluations. The agent has two conditions for judging whether the game is like PD or not and two self-evaluation generators — one generates self-evaluations effective in PD-like games and the other generates them effective in non-PD-like games. The agent selects one of the two generators according to the judgement and generates a self-evaluation for learning. We showed results of experiments in several iterated games and concluded that the proposed method was effective in both PD-like and non-PD-like games.

N. Lavrač et al. (Eds.): ECML 2003, LNAI 2837, pp. 289–300, 2003.

However, the experiments were *static* ones in which there was only one state. The versatility for PD-like and non-PD-like games is indispensable in *dynamic* environments in which there exist several states transferring one after another in a trial. Therefore, we have conducted new experiments in each of which the agents played a game having multiple states. The experiments include two kinds of game; the one notifies the agents of the current state and the other does not. We report the results in this paper.

This paper consists of six sections. In Section 2, we introduce the generators and the conditions proposed in the previous paper. We show in Section 3 the new experiments that we have conducted in two kinds of game played by the agents. In Section 4, we discuss the result of the experiments and of this study itself. Related works are shown in Section 5. Finally, we conclude the paper and point out future works in Section 6.

2 Generating Self-evaluation

This section introduces our method proposed in the previous paper [11].

2.1 Background and Objectives

In game theory, an actor is called a *player*. A player maximizes his own payoff and assumes that all other players do similarly. A player i has a set of actions Σ_i and his strategy σ_i is defined by a probability distribution over Σ_i. When σ_i assigns probability 1 to an action, σ_i is called a *pure* strategy and we refer to it as the action. $\boldsymbol{\sigma} \equiv (\sigma_i)$ is a vector of strategies of players in a game, and $\boldsymbol{\sigma}_{-i}$ is a vector of strategies of players excluding i. A payoff of player i is defined as $f_i(\boldsymbol{\sigma}) \equiv f_i(\sigma_i, \boldsymbol{\sigma}_{-i})$. Then the *best response* of player i for $\boldsymbol{\sigma}_{-i}$ is a strategy σ_i satisfying

$$f_i(\sigma_i, \boldsymbol{\sigma}_{-i}) = \max_{\tau_i} f_i(\tau_i, \boldsymbol{\sigma}_{-i}).$$

The vector $\boldsymbol{\sigma}$ is a *Nash equilibrium* if, for all i, σ_i is the best response for $\boldsymbol{\sigma}_{-i}$. On the other hand, the vector $\boldsymbol{\sigma}$ is *Pareto optimal* if there is no vector $\boldsymbol{\rho}$ satisfying

$$\forall i \quad f_i(\boldsymbol{\rho}) \geq f_i(\boldsymbol{\sigma}) \quad \text{and} \quad \exists j \quad f_j(\boldsymbol{\rho}) > f_j(\boldsymbol{\sigma}).$$

It means that there is no combination of strategies in which someone gets more payoff than $\boldsymbol{\sigma}$ without those who get less.

Generally, a Nash equilibrium is not Pareto optimal. Table 1 shows a game having only a single Nash equilibrium, which is not Pareto optimal. This game is an example of the *prisoner's dilemma* (PD) [14]. In this game, since the best response of a player is D regardless of the other player's action, there is only one Nash equilibrium $\boldsymbol{\sigma} = (D, D)$; but it is not Pareto optimal because (C, C) is the role of $\boldsymbol{\rho}$ in the definition.

In reinforcement learning, an actor is called an *agent* and it learns from cycles of action selection and reward acquisition [18]. Although reinforcement learning is for a single agent environment, there are many proposals to extend it for a multi-agent environment [6, 4, 2, 13, 1]. However, almost all of them aim to converge to a Nash equilibrium without considering Pareto optimality. Hence, in a PD game of Table 1, they will converge to an unfavorable result $\boldsymbol{\sigma} = (D, D)$ purposely.

Table 1. Prisoner's Dilemma [14]. Player A selects a row and B selects a column. Each takes cooperation C or defection D. (x,y) refers to the payoff of A and B, respectively

$A \setminus B$	C	D
C	$(2, 2)$	$(0, 3)$
D	$(3, 0)$	$(1, 1)$

On the other hand, there are several proposals to have the combination of actions depart from undesirable Nash equilibria and proceed to a better combination in PD-like games through reinforcement learning with reward handling methods [7, 8, 23]. However, since the methods of these proposals are fixed and applied unconditionally, the methods are inferior to normal reinforcement learning in non-PD-like games. Hence, when we equip agents with these methods, we have to check the game in advance. These cannot also be used in an environment changing from game to game.

Hence, in the previous paper [11], we have constructed an agent learning appropriate actions in both PD-like and non-PD-like games. The agent has two reward handling methods and two conditions for judging the game. We call the handled rewards *self-evaluations* and the reward handling methods *self-evaluation generators*. Each generator generates self-evaluations appropriate in either PD-like or non-PD-like games, and the conditions judge whether the game is like PD or not. In each learning cycle, the agent judges the game through the conditions, selects one of the two generators according to the judgement, generates a self-evaluation, and learns through the evaluation.

Before introducing the generators and the conditions in detail, we classify games in the next subsection.

2.2 Classification of Symmetric Games

We classify *symmetric* games into four classes in terms of game theory. A symmetric game is that in which all players have a common set of actions Σ and a common payoff function f. A Nash equilibrium which consists only of pure strategies is called a *pure strategy Nash equilibrium* (PSNE), and a PSNE in which all players' actions are identical is called a *symmetric PSNE* (SPSNE). Here, we classify a symmetric game into one of the following class by a set of SPSNEs N and a set of Pareto optimal combinations of actions P of the game.

Independent: $N \neq \emptyset,\ N \subset P$
All SPSNEs are Pareto optimal.

Boggy: $N \neq \emptyset,\ N \cap P = \emptyset$
None of SPSNEs are Pareto optimal.

Selective: $N \neq \emptyset,\ N \cap P \neq \emptyset,\ N \not\subset P$
There are Pareto optimal SPSNEs and non-Pareto optimal SPSNEs.

Rival: $N = \emptyset$
There is no SPSNE.

For a game in the *independent* class, the combination of actions is desirable when all players select rational actions independently. Conversely, the combination is unfa-

vorable for a game in the *boggy* class. In the game, the players actually get less if all of them select the more profitable action. This is the origin of the name "boggy". PD is in this class. In a game in the *selective* class, it is a problem which SPSNE is desirable. The *rival* class consists of games having some gainers and some losers (e.g. a game in which a player has to yield a way to another). In the two-person two-action case, the independent and the boggy classes are both in Categories I and IV[1] of Weibull [21], and the selective and the rival classes are in Categories II and III, respectively.

2.3 Generating Self-evaluation

In this paper, we use *Q-learning* [20] that is a representative reinforcement learning method. Q-learning updates the *Q-function* representing estimates of future rewards from each cycle of action selection and reward acquisition. At time t, an agent recognizes the current state s_t, takes an action a_t, obtains a reward r_{t+1}, and recognizes the next state s_{t+1}. Afterwards, Q-learning updates the Q-function Q_t as follows.

$$\begin{aligned} &Q_t(s,a) = Q_{t-1}(s,a) \quad \text{if } s \neq s_t \text{ or } a \neq a_t, \\ &Q_t(s_t,a_t) = Q_{t-1}(s_t,a_t) + \alpha\,\delta_t. \end{aligned} \tag{1}$$

In the update rule, δ_t is a temporal difference (TD) error:

$$\delta_t \triangleq r_{t+1} + \gamma \max_a Q_{t-1}(s_{t+1},a) - Q_{t-1}(s_t,a_t). \tag{2}$$

The agent selects an action using the Q-function with a randomizing method, e.g. the softmax method [18], that adds some randomness to the selection.

We now introduce two self-evaluation generators. In an agent A_i at time t, a self-evaluation $r'_{i,t+1}$ is generated by adding a term $\lambda_{i,t+1}$ to a reward $r_{i,t+1}$, which is then used to update the Q-function. We omit the subscript i showing "the agent itself (A_i)" in the following.

$$r'_{t+1} \triangleq r_{t+1} + \lambda_{t+1}. \tag{3}$$

We propose two λ's, which we call the *neighbor's rewards* (NR) and the *difference of rewards* (DR). NR is effective for a game in the boggy class and DR is effective for a game in the independent class.

$$\lambda^{NR}_{t+1} \triangleq \sum_{A_k \in N_i \backslash A_i} r_{k,t+1} \tag{4}$$

$$\lambda^{DR}_{t+1} \triangleq r_{t+1} - r_t \tag{5}$$

In Formula 4, $N_i \backslash A_i$ is a set of agents that excludes A_i from a set of A_i's neighbors, N_i. NR is effective in a game in which the neighbors' rewards decrease as the agent selects profitable actions. Since DR emphasizes the difference between the present reward and the last reward, it makes the agent sensitive to change of rewards and the agent has a tendency to take self-interested actions.

[1] Categories I and IV are the same except for action names.

Table 2. Auto-Select (AS-) AA, AR, and RR: Q-functions for the judgement

AS-	Formula 6	Formula 7
AA	Q^{act}	Q^{act}
AR	Q^{act}	Q^{recog}
RR	Q^{recog}	Q^{recog}

Although NR and DR are effective in games in the boggy class and in the independent class, respectively, they are harmful when their use is reversed because the two classes are the opposite. Therefore, we have to devise how the agent appropriately selects the two λ's according to the game. In the previous paper, we have proposed two conditions by the following formulae for judging the class of game and selecting λ.

$$Q_{t-1}(s_t, a) < 0 \quad \text{for all } a, \tag{6}$$

$$r_{t+1} < Q_{t-1}(s_t, a_t) - \gamma \max_a Q_{t-1}(s_{t+1}, a). \tag{7}$$

Formula 6 means that there is no hope whatever action the agent currently selects. Formula 7 is derived from the TD error (Formula 2). This formula shows that the current situation is worse than what the agent learned, because it shows that the present reward is less than the estimate calculated from the learned Q-function on the assumption that the TD error is 0, i.e. the learning is completed. We can think that the agents should refrain from taking self-interested actions that bring such worse situation. Thus, we make a rule that NR is used as a self-evaluation generator if at least one of these formulae is satisfied; DR is used otherwise. We call this rule *auto-select* (AS).

In Formula 7, the left-hand side is the present reward (r). It becomes a subject of discussion because, in this paper, the Q-function is learned by *self-evaluations* (r'), not by rewards. Accordingly, we also introduce a normal Q-function for the judgement. We call this normal Q-function Q^{recog} because it is used for recognizing games, and refer to the Q-function learned by self-evaluations as Q^{act} because it is used for action selection. Since the discussion is not concerned with Formula 6, we are able to introduce two types of agent; one uses Q^{act} and the other uses Q^{recog} in Formula 6. We call them AS-AR and AS-RR, respectively. We also introduce AS-AA that uses Q^{act} in both formulae [10]; it can be thought to substitute rewards for self-evaluations in Formula 7. Table 2 shows the relation of ASs.

In summary, the proposed agents AS-AA, AS-AR, and AS-RR learn actions in the following cycle:

1. Sense the current state s_t.
2. Select and take an action a_t by Q^{act} with a randomizing method.
3. Obtain a reward r_{t+1} and sense the next state s_{t+1}.
4. Recognize the game by Formulae 6 and 7, then select λ_{t+1}.
5. Generate a self-evaluation r'_{t+1} from r_{t+1} and λ_{t+1} by Formula 3.
6. Update Q^{act} by Q-learning with s_t, a_t, r'_{t+1}, s_{t+1} and Q^{recog} by Q-learning with s_t, a_t, r_{t+1}, s_{t+1}.
7. Back to 1.

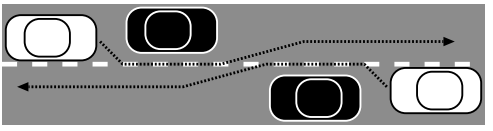


Fig. 1. Narrow Road Game: Black cars are parking on the road. Two white cars simultaneously appear at both side of the road

3 Experiments

We have conducted three experiments using two kinds of game having multiple states played by multiple homogeneous agents. One kind is called *Markov games* or *stochastic games* [6, 4] in which each agent is notified of the current state, and the other kind is *state-unknown games* in which each agent is not informed of the state. We used seven types of agents for comparison: *random* (RD), *normal* (NM), the *neighbors' rewards* (NR), the *difference of rewards* (DR), and the three types of *auto-select* (AS-AA, AS-AR, and AS-RR). RD selects actions randomly, and NM learns by normal Q-learning.

Each experiment was conducted twenty-five times. Q-functions were initialized to 0 in each trial. We used the softmax method [18] as a randomizing method for action selection and a parameter of the method called temperature was set to 1. Learning rate α and discount factor γ in Formula 1 were both set to 0.5.

3.1 Markov Game

A Markov game [6, 4] has several matrix games each of which has a label called *state*. In the game, the combination of all player's actions makes a state transfer to another and every player knows which state he is in now.

Here we use a game named Narrow Road Game. Suppose a road with two parking cars and two cars which simultaneously appear at both sides of the road (Figure 1). Each car selects one of actions: GO or WAIT. Both cars want to pass the narrow point as soon as possible. However, if both take GO, they cannot pass because of the point. If both take WAIT, on the other hand, nothing changes.

Each car (agent) receives a reward 1 when it takes GO and succeeds in passing and -0.5 when it fails to pass or it takes WAIT. A cycle is finished after both agents have passed or they have taken four actions. Every agent learns after every reward acquisition and knows whether the opposite remains and how many actions they have taken. After one agent succeeds in passing, it gets out of the cycle, and the remaining agent takes an action again. Then if the remaining agent takes GO, it passes unconditionally, and otherwise, it has to take an action again. These are summarized in Figure 2. Since payoff matrices (i.e. states) change with the number of agents and every agent knows which state it is in now, this is a Markov game. The set of neighbors, N_i, in Formula 4 includes all agents in a game.

Figure 3 shows the average of summed reward of two agents at the 100th cycle in 25 trials. In each cycle, the rewards each agent obtains are accumulated and two agents' accumulated rewards are summed. The maximum summed reward is 1.5 when

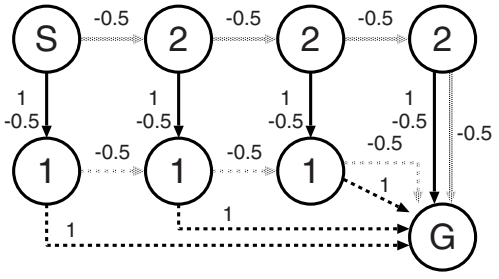


Fig. 2. State Transition in Narrow Road Game: A circle and an arrow show a state and a state transition, respectively. A number in a circle shows how many agents are remaining. Black arrows show transitions in the case in which one of the agents succeeds in passing, and gray ones show transitions in other cases. Dashed arrows show transitions by one remaining agent. Numbers with an arrow show payoff with each transition. With a solid black arrow, the upper number is payoff for the going agent and the lower is payoff for the waiting agent

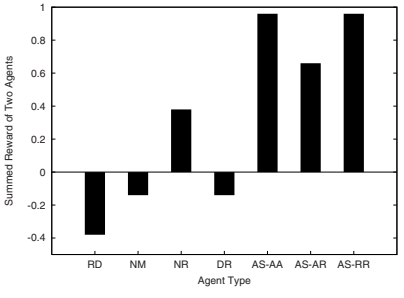


Fig. 3. Result of Narrow Road Game: It shows the average of summed reward of two agents at the 100th cycle in 25 trials

one agent passes at the first action phase (i.e. the other waits) and the other passes at the second, and the minimum is −4 when both fail to pass in four action phases.

From the figure, we can see the following:

- The three AS versions outperform the other methods. However, as a result of paired *t*-tests, there is no significant difference between the result of AS-AR and those of other methods. AS-AA and AS-RR are also not significantly different from NR.
- The summed reward about 1 in AS-AA and AS-RR means that the probability of taking more than two action phases is less than a half.

3.2 State-Unknown Games

The previous subsection shows the experimental result of a Markov game. In a Markov game, every agent is able to know which state it is in now. However, in a real situation, we cannot precisely know the current state in many cases and we have to guess it.

Therefore, in this subsection, we show the results of two experiments in state-unknown games.

We used the *tragedy of the commons* (TC) type games. TC is introduced by Hardin [3], in which there is an open pasture with several rational herdsmen each of whom grazes cattle in the pasture. Since each herdsman tries to maximize his profit, each one brings as many cattle as possible to the pasture. If all herdsmen bring their cattle unlimitedly, however, it is overgrazing and the pasture goes wild. This is the tragedy.

Mikami et al. [7] conducted an experiment consisting of ten agents. In a cycle, each agent takes one of three actions (*selfish*, *cooperative*, and *altruistic*) simultaneously and receives a reward. An action a_i of an agent A_i ($i = 0, 1, ..., 9$) follows a base reward $R(a_i)$ and a social cost $C(a_i)$. R is 3, 1, and -3 and C is $+1$, ± 0, and -1 when the agent takes a selfish, a cooperative, and an altruistic action, respectively. After all agents take actions, A_i receives a reward r_i defined as

$$r_i \triangleq R(a_i) - \sum_j C(a_j).$$

The set of A_i's neighbors, N_i, in Formula 4 is defined as

$$N_i \triangleq \left\{ A_k \,|\, k = (i + j) \bmod 10,\ j = 0, 1, 2, 3 \right\},$$

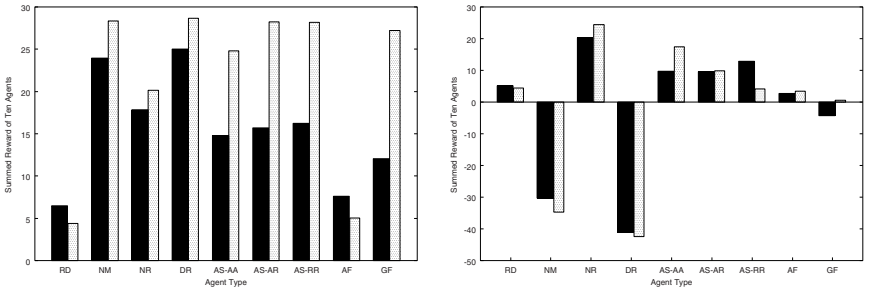
and A_i uses the combination of actions of N_i as a state in Q-learning. A_i is able to know only the combination, not all agents' actions.

In our previous paper [11], we modified the cost C as $+c$, ± 0, and $-c$ when the agent took a selfish, a cooperative, and an altruistic action, respectively. c was a constant common for all agents and the game was identical with Mikami's when $c = 1$. We showed the results of two experiments, $c = 1$ and $c = 0$, which were in the boggy class and in the independent class, respectively, and concluded that the proposed methods (ASs) were effective in both games.

In this paper, the cost parameter c is changed *without informing agents* in each trial. Since the results in the previous paper were at the 3000th cycle, we changed the parameter c in a trial after 3000 cycles and continued the trial further 3000 cycles. We have conducted two experiments; the parameter c is changed from 1 (boggy) to 0 (independent) and vice versa. We show the results at the 6000th cycle (i.e. the 3000th cycle after the change) with the previous results for comparison. We look at the results of the experiments with those of Mikami's *average filter* $(\text{AF})^2$ [7] and of our past method called the *general filter* (GF) [9].

Figure 4 shows the average of summed reward of ten agents at the 6000th cycle in 25 trials. From the figure we can see that, in the experiment from $c = 1$ (boggy) to 0 (independent), the results are considerably different from the previous static ones, especially those of ASs. On the other hand, the results are similar in the experiment from $c = 0$ to 1.

[2] They proposed two types of AF; but due to space constraint, only the result of type 1 is shown here.



(a) $c = 1 \rightarrow 0$ (b) $c = 0 \rightarrow 1$

Fig. 4. Result of the Tragedy of the Commons: Each thick, black bar shows the average of summed reward of ten agents at the 6000th cycle (i.e. the 3000th cycle after change) in 25 trials. Each thin, gray bar shows the result at the 3000th cycle in 25 trials of a previous static experiment with changed c, i.e. 0 in (a) and 1 in (b) [11]

4 Discussion

In Narrow Road Game, ASs are the best of all. It seems slightly strange because ASs only chose NR or DR that were worse than ASs, but we can be convinced if we interpret as the game requiring agents to use NR and DR properly. In fact, when two agents are remaining, which is in the rival class, one agent has to give a way to the other and NR may be effective in this class. On the other hand, when only one is remaining, which is in the independent class, it has to go decisively and DR is effective in this class.

In state-unknown games, although the result of '0 to 1' is not affected by the change, the result of '1 to 0' is worse than that of $c = 0$. It is because an agent's policy learned before the change had deleterious effect on its actions after the change. Since $c = 1$ is in the boggy class, the agent mainly learned through NR before the change. Hence, the agent's Q^{act} function for altruistic actions must have been more than that for selfish actions. The function for cooperative actions may also have been. On the other hand, not only selfish actions, but also cooperative ones bring positive rewards after the change in this game. Therefore, even if altruistic actions were withdrawn appropriately by learning after the change, the action the agent would choose may have been not the selfish one, which is desirable in $c = 0$, but the cooperative one. Consequently, if the agent learns through same Q-functions before and after the change, we cannot avoid this problem even if we humans choose NR and DR properly. Thus, in order to learn appropriate actions in these games, what we have to improve is, probably, not the conditions for selecting self-evaluation generators, but the selected generators, i.e. NR and DR.

We have constructed an agent which is able to learn suitable actions in *several* games by means of Q-learning with self-evaluations instead of rewards. As far as we know, no such work yet exists in a reinforcement learning context. There are indeed some works which handles reward functions for learning suitable actions [7, 8, 23], but they differ from this work because each of them performs for only *one* game. If we use

their works for constructing agents, we have to decide in advance whether we are to use them according to the game. If we use this work, on the other hand, it will surely reduce problem of judging the game.

This work aims to discover suitable conditions or *meta-rules* for learning. In this paper, we introduce two conditions for judging the boggy game. Although we can introduce other meta-rules, it will be difficult. Thus, we should devise methods for constructing meta-rules themselves. It becomes a meta-learning problem [19] in a learning context. We can also search function space of self-evaluations by genetic algorithm.

5 Related Works

Although there are only a few works which handle rewards for learning suitable actions in a reinforcement learning context, there are several methods in a genetic algorithm (GA) context. Mundhe et al. [12] proposed several fitness functions for the tragedy of the commons. Their proposed fitness functions are used if a received reward is less than the best of the past; otherwise, normal functions are used. Thus, unlike Mikami's [7] and Wolpert's [23], they are conditionally used but only applied to games in the boggy class.

There are a few works aiming to obtain meta-rules. Schmidhuber et al. [17] proposed a reinforcement learning algorithm called the *success story algorithm* (SSA). It learns when it evaluates actions in addition to normal actions. Zhao et al. [24] used the algorithm in a game in which agents have to cooperate to gain food, as they escape from a pacman. In a GA context, Ishida et al. [5] proposed agents each of which had its own fitness function called *norm*. If the agent's accumulated reward becomes under a threshold, the agent dies and a new one having a new norm takes the agent's place. They conducted only an experiment in the tragedy of the commons.

Uninformed changes in state-unknown games are similar to *concept drifts* in a symbolic learning context, which are changes of concepts driven by some hidden contexts. A learner in the environment having concept drifts usually uses a kind of windowing method to cast off expired inputs. Widmer et al. [22] presented a flexible windowing method which modified the window size according to the accuracy of learned concepts to the input sequence in order to follow the concept drifts more flexibly. They also proposed a learner that reuses some old concepts learned before. Sakaguchi et al. [16] proposed a reinforcement learning method having the forgetting and reusing property. In their method, if the TD error is over a threshold, a learner shifts the current Q-function to an adequate one it has or creates a new Q-function if it does not have. They conducted experiments only in single agent environments.

In game theory, each player takes defection in the prisoner's dilemma (PD). However, when humans play the game, the result is different from that of theories. Rilling et al. [15] requested thirty-six women to play PD and watched their brains by functional Magnetic Resonance Imaging (fMRI). They reported that several parts of reward processing in their brains were activated when both players cooperated, and explained that rewards for cooperation were then *generated in their brains*. This shows that there are common features in a human's brain processes and in the proposed method of this work.

6 Conclusion

Most of multi-agent reinforcement learning methods aim to converge to a Nash equilibrium, but a Nash equilibrium does not necessarily mean a desirable result. On the other hand, there are several methods aiming to depart from unfavorable Nash equilibria, but they are effective only in limited games. Hence, we have constructed an agent learning appropriate actions in many games through reinforcement learning with self-evaluations.

First we defined a symmetric pure strategy Nash equilibrium (SPSNE) and categorized symmetric games into four classes — independent, boggy, selective, and rival — by SPSNE and Pareto optimality. Then we introduced two self-evaluation generators and two conditions for judging games in the boggy class. We proposed three types of methods, which were auto-select (AS-) AA, AR, and RR.

We have conducted experiments in Narrow Road Game and state-unknown games. The result of Narrow Road Game shows that the versatility for PD-like and non-PD-like games is indispensable in games having multiple states. In '0 to 1' state-unknown game, there is no effect from the change, thus the proposed methods (ASs) seem robust to multiple states. On the other hand, the result of '1 to 0' state-unknown game shows that there exist games in which ASs are fragile. However, discussion about this ineffectiveness in Section 4 points out that the fragility is derived from not the conditions for selecting self-evaluation generators, but the selected generators.

We are able to point out several future works. First, since we have conducted only empirical evaluation in this paper, we have to evaluate the proposed methods theoretically. Next, since we have classified only symmetric games, we need to extend the class definition so that it can deal with asymmetric games. Furthermore, we conducted experiments only in simple games with homogeneous agents, and thus, we have to conduct experiments in more complex problems which consist of heterogeneous agents and evaluate the proposed methods by the results.

References

1. M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.
2. C. Claus and C. Boutilier. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In *Proc. 15th National Conference on Artificial Intelligence, AAAI-98*, pages 746–752, Madison, Wisconsin, U.S.A., 1998.
3. G. Hardin. The Tragedy of the Commons. *Science*, 162:1243–1248, 1968.
4. J. Hu and M. P. Wellman. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proc. 15th International Conference on Machine Learning, ICML'98*, pages 242–250, Madison, Wisconsin, U.S.A., 1998.
5. T. Ishida, H. Yokoi, and Y. Kakazu. Self-Organized Norms of Behavior under Interactions of Selfish Agents. In *Proc. 1999 IEEE International Conference on Systems, Man, and Cybernetics*, Tokyo, Japan, 1999.
6. M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. 11th International Conference on Machine Learning, ML'94*, pages 157–163, New Brunswick, New Jersey, U.S.A., 1994.

7. S. Mikami and Y. Kakazu. Co-operation of Multiple Agents Through Filtering Payoff. In *Proc. 1st European Workshop on Reinforcement Learning, EWRL-1*, pages 97–107, Brussels, Belgium, 1994.
8. S. Mikami, Y. Kakazu, and T. C. Fogarty. Co-operative Reinforcement Learning By Payoff Filters. In *Proc. 8th European Conference on Machine Learning, ECML-95*, (Lecture Notes in Artificial Intelligence 912), pages 319–322, Heraclion, Crete, Greece, 1995.
9. K. Moriyama and M. Numao. Constructing an Autonomous Agent with an Interdependent Heuristics. In *Proc. 6th Pacific Rim International Conference on Artificial Intelligence, PRICAI-2000*, (Lecture Notes in Artificial Intelligence 1886), pages 329–339, Melbourne, Australia, 2000.
10. K. Moriyama and M. Numao. Construction of a Learning Agent Handling Its Rewards According to Environmental Situations. In *Proc. 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS-2002*, pages 1262–1263, Bologna, Italy, 2002.
11. K. Moriyama and M. Numao. Generating Self-Evaluations to Learn Appropriate Actions in Various Games. Technical Report TR03-0002, Department of Computer Science, Tokyo Institute of Technology, 2003.
12. M. Mundhe and S. Sen. Evolving agent societies that avoid social dilemmas. In *Proc. Genetic and Evolutionary Computation Conference, GECCO-2000*, pages 809–816, Las Vegas, Nevada, U.S.A., 2000.
13. Y. Nagayuki, S. Ishii, and K. Doya. Multi-Agent Reinforcement Learning: An Approach Based on the Other Agent's Internal Model. In *Proc. 4th International Conference on MultiAgent Systems, ICMAS-2000*, pages 215–221, Boston, Massachusetts, U.S.A., 2000.
14. W. Poundstone. *Prisoner's Dilemma*. Doubleday, New York, 1992.
15. J. K. Rilling, D. A. Gutman, T. R. Zeh, G. Pagnoni, G. S. Berns, and C. D. Kilts. A Neural Basis for Social Cooperation. *Neuron*, 35:395–405, 2002.
16. Y. Sakaguchi and M. Takano. Learning to Switch Behaviors for Different Environments: A Computational Model for Incremental Modular Learning. In *Proc. 2001 International Symposium on Nonlinear Theory and its Applications, NOLTA-2001*, pages 383–386, Zao, Miyagi, Japan, 2001.
17. J. Schmidhuber, J. Zhao, and N. N. Schraudolph. Reinforcement Learning with Self-Modifying Policies. In [19], pages 293–309, 1997.
18. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
19. S. Thrun and L. Pratt, editors. *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, 1997.
20. C. J. C. H. Watkins and P. Dayan. Technical Note: Q-learning. *Machine Learning*, 8:279–292, 1992.
21. J. W. Weibull. *Evolutionary Game Theory*. MIT Press, Cambridge, MA, 1995.
22. G. Widmer and M. Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, 23:69–101, 1996.
23. D. H. Wolpert and K. Tumer. Collective Intelligence, Data Routing and Braess' Paradox. *Journal of Artificial Intelligence Research*, 16:359–387, 2002.
24. J. Zhao and J. Schmidhuber. Solving a Complex Prisoner's Dilemma with Self-Modifying Policies. In *From Animals to Animats 5: Proc. 5th International Conference on Simulation of Adaptive Behavior*, pages 177–182, Zurich, Switzerland, 1998.