Learning Desirable Actions in Two-Player Two-Action Games

Koichi Moriyama
Department of Computer Science, Tokyo Institute of Technology.
W8–61, 2–12–1, Ookayama, Meguro, Tokyo, 152-8552, JAPAN.
moriyama@mori.cs.titech.ac.jp

Abstract

Reinforcement learning is widely used to let an autonomous agent learn actions in an environment, and recently, it is used in a multi-agent context in which several agents share an environment. Most of multi-agent reinforcement learning algorithms aim to converge to a Nash equilibrium of game theory, but it does not necessarily mean a desirable result. On the other hand, there are several methods aiming to depart from unfavorable Nash equilibria, but they use other agents' information for learning and the condition whether or not they work has not yet been analyzed and discussed in detail. In this paper, we first see the sufficient conditions of symmetric two-player two-action games that show whether or not reinforcement learning agents learn to bring the desirable result. After that, we construct a new method that does not need any other agents' information for learning.

1. Introduction

In this paper, we consider the environment consisting of multiple autonomous actors called agents. In such a multi-agent system, each agent should coordinate each other, but the coordination is too hard to be implemented in advance if the number of agents is large. Therefore, it is desirable that each agent autonomously adapts to coordinate each other. One method for the adaptation is *machine learning* that gradually lets agents adapt to select proper actions. There are a great number of machine learning methods in the world, but especially *reinforcement learning* is used as a typical method of single agent learning because it does not need answers, but only scholar values called *rewards* for learning.

Here we investigate the use of reinforcement learning in a multi-agent environment. Of course, there are many multi-agent reinforcement learning algorithms proposed to date, e.g. [2, 1, 6]. Almost all of them aim to converge to a *Nash equilibrium* that is a combination of actions of rational players in game theory. However, this combination is not good in some games such as Prisoner's Dilemma (PD) [7]. On the other hand, several methods try to depart from bad Nash equilibria and proceed to a better combination by handling rewards in PD-like games [3, 10]. Since they use fixed handling methods, they are worse than normal reinforcement learning in non-PD-like games, but the condition whether or not they work has not yet been analyzed and discussed in detail. The author proposed an agent learning appropriate actions in both PD-like and non-PD-like games through self-evaluations [4, 5]. However, it needs other agents' reward information for learning, which is not desirable.

In this paper, we first see the sufficient conditions of symmetric two-player two-action games that show whether the learning agent works or not. After that, we construct a new self-evaluation generation method that does not need any other agents' information.

This paper consists of 5 sections. In Section 2, we discuss the backgrounds and objectives. Next, in Section 3, we derive the sufficient conditions whether the learning agent works or not. In Section 4 we propose a new self-evaluation generation method that does not need any other agents' information. Finally, we conclude the paper in Section 5.

2. Backgrounds 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. Let the set of all possible probability distributions over Σ_i be $PD(\Sigma_i)$. Then, his strategy σ_i is an element of $PD(\Sigma_i)$. 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

$$f_i(\sigma_i, \boldsymbol{\sigma}_{-i}) = \max_{\tau_i \in PD(\Sigma_i)} f_i(\tau_i, \boldsymbol{\sigma}_{-i}). \quad (1)$$

Table 1. Prisoner's Dilemma [7]. 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	$(3,0)$	$(1,1)$

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

$$\forall i \quad f_i(\boldsymbol{\rho}) \geq f_i(\boldsymbol{\sigma}) \quad \text{and} \quad \exists j \quad f_j(\boldsymbol{\rho}) > f_j(\boldsymbol{\sigma}). \tag{2}$$

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

Generally, a Nash equilibrium is not Pareto optimal. Table 1 shows a game having only a single Nash equilibrium that is not Pareto optimal. This game is an example of *Prisoner's Dilemma* (PD) [7]. In this game, since the best response of a player is D regardless of the opponent'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 Formula 2. Like PD, games having two players each of whom has two actions are called 2×2 games.

In reinforcement learning, an actor is called an *agent*. An agent senses a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}(s_t)$ in a discrete time t. $\mathcal{S}$ is a set of possible states in the environment and $\mathcal{A}(s_t)$ is a set of possible actions in the state s_t. After selecting an action, it receives a reward $r_{t+1} \in \mathbb{R}$ and senses a new state s_{t+1} [8] [1]. A representative reinforcement learning method is Q-learning [9]. Q-learning updates the *action value function* $Q_t(s, a)$ by the following rules to make it approach the true *value*, which means the expected sum of rewards discounted by $0 < \gamma < 1$, i.e. $E\left(\sum_{k=0}^{\infty} \gamma^k r_{t+1+k}\right)$, of the state s and action a.

$$Q_t(s,a) = \begin{cases} Q_{t-1}(s_t, a_t) + \alpha\, \delta_t & \text{if } (s,a) = (s_t, a_t), \\ Q_{t-1}(s,a) & \text{otherwise.} \end{cases} \tag{3}$$

$0 < \alpha \leq 1$ is a parameter called a learning rate and δ_t is called a TD error that becomes 0 when $Q_{t-1}(s_t, a_t)$ becomes the true value of (s_t, a_t):

$$\delta_t \triangleq r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} Q_{t-1}(s_{t+1}, a) - Q_{t-1}(s_t, a_t). \tag{4}$$

For all s and a, $Q_t(s, a)$ is proved to converge to true values when the agent visits to all states and takes all actions infinitely and decreases the learning rate α properly [9] [2].

If Q_{t-1} equals true values, we can select an action a_t in a state s_t from Q_{t-1} by

$$a_t = \arg \max_{a' \in \mathcal{A}(s_t)} Q_{t-1}(s_t, a'). \tag{5}$$

However, if the agent selects such action between learning, Q_t may converge to a local optimum because the agents may not visit all states. To avoid it, we usually use stochastic methods like the *softmax method* [8] for action selection. The softmax method calculates the probabilities $p_{s_t}(a)$ of each action $a \in \mathcal{A}(s_t)$ by

$$p_{s_t}(a) \triangleq \frac{\exp(Q_{t-1}(s_t, a)/T)}{\sum_{a' \in \mathcal{A}(s_t)} \exp(Q_{t-1}(s_t, a')/T)} \tag{6}$$

and uses them for action selection. $T > 0$ is called a temperature that controls the effect of randomness.

Although Q-learning is for a single agent environment, there are many proposals to extend it for a multi-agent environment, e.g. [2, 1, 6]. 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.

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* [3, 10]. These methods uses other agents' reward / influence information or its prediction for learning. Since the methods are fixed and applied unconditionally, they are inferior to normal reinforcement learning in non-PD-like games. However, the condition whether or not they work has not yet been analyzed and discussed in detail.

The author proposed an agent learning appropriate actions in both PD-like and non-PD-like games through reward handling [4, 5]. The agent has two reward handling methods and two conditions for judging the game. We call the handled reward *self-evaluation* and the reward handling methods *self-evaluation generators*. A self-evaluation is $r + r'$, in which r is a reward and r' is an added parameter generated by the agent, which also depends on (s, a) of all agents in the environment. 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

1 Here we assume that r_{t+1} and s_{t+1} depend on only the (old) state s_t and the action a_t. Later, in the multi-agent case, we assume that they depend on only (s_t, a_t) of *all agents sharing the environment.*

2 For simplicity, the explanation of Q-learning here is not necessarily precise. For example, the value should depend on the transition of states and actions, and Q_t converges to the values of optimal transition. See [8, 9] for details.

Table 2. Symmetric 2x2 game. r_{CC}, r_{CD}, r_{DC}, and r_{DD} are variables showing rewards

$A \setminus B$	C	D
C	(r_{CC}, r_{CC})	(r_{CD}, r_{DC})
D	(r_{DC}, r_{CD})	(r_{DD}, r_{DD})

game through the conditions, selects one of the two generators according to the judgement, generates a self-evaluation, and learns through the evaluation. However, since it is based on the reward handling methods described above, it also needs other agents' information for learning. It is not desirable because not only is it a problem how to communicate the information reliably, but, if each agent has real autonomy, it can tell a lie when the information is requested from other agents.

In this paper, we first see the sufficient conditions of symmetric 2×2 games that show whether the learning agent learns to bring the desirable result or not. After that, we construct a new self-evaluation generator that does not need any other agents' information.

3. Sufficient conditions of symmetric 2×2 games

In this section, we see the sufficient conditions of symmetric 2×2 games for identifying whether a learner takes desirable actions or not. We use a 2×2 game in Table 2 for consideration. It is symmetric because there are no difference between stances of players. The relations between the reward variables considering here is as follows [7] [3]:

Prisoner's Dilemma (PD) (C, C) is desirable
$r_{DC} > r_{CC} > r_{DD} > r_{CD}$,

Deadlock Game (DG) (D, D) is desirable
$r_{DC} > r_{DD} > r_{CC} > r_{CD}$.

The learner here is an agent using one state Q-learning with the softmax method. We refer to the two actions as C and D following Table 2 and to $Q(s, a)$ as $Q(a)$ because there is only one state.

Here we analyze the sufficient conditions statically, that is, we discuss the case in which the learners do not change their actions from beginning to end. On the other hand, the learners freely change their actions in the verifications shown later.

3 When we consider only the ordering of rewards, there are four kinds of symmetric one-shot 2×2 games, but we consider only the two here. The rest two are Chicken Game and Stag Hunt Game [7].

3.1. Normal Q-learning

First we consider an agent using normal Q-learning with the softmax method. Suppose the desirable action is C. Then the probability of selecting action D should be small. Here we set the upper limit of the probability $1/n$; of course, $n > 2$. Then, the probability $p(D)$ is

$$p(D) \equiv \frac{\exp(Q(D)/T)}{\exp(Q(C)/T) + \exp(Q(D)/T)} \leq \frac{1}{n}. \quad (7)$$

From the formula, we can derive ("log" is natural logarithm)

$$Q(C) \geq Q(D) + T \log(n-1). \quad (8)$$

Next, when the opponent takes actions C with probability p, $Q(C)$ and $Q(D)$ become as follows because they are the expected sum of discounted rewards.

$$\begin{aligned} Q(C) &= \frac{r_{CC}}{1-\gamma} p + \frac{r_{CD}}{1-\gamma}(1-p) \\ &= \frac{r_{CC} - r_{CD}}{1-\gamma} p + \frac{r_{CD}}{1-\gamma}, \end{aligned} \quad (9)$$

$$\begin{aligned} Q(D) &= \frac{r_{DC}}{1-\gamma} p + \frac{r_{DD}}{1-\gamma}(1-p) \\ &= \frac{r_{DC} - r_{DD}}{1-\gamma} p + \frac{r_{DD}}{1-\gamma}. \end{aligned} \quad (10)$$

From them,

$$\begin{aligned} &Q(C) - Q(D) \\ &= \frac{r_{CC} + r_{DD} - r_{CD} - r_{DC}}{1-\gamma} p + \frac{r_{CD} - r_{DD}}{1-\gamma}, \end{aligned} \quad (11)$$

$$\begin{aligned} &Q(D) - Q(C) \\ &= \frac{r_{CD} + r_{DC} - r_{CC} - r_{DD}}{1-\gamma} p + \frac{r_{DD} - r_{CD}}{1-\gamma}. \end{aligned} \quad (12)$$

From Formulae 8 and 12, we get

$$(1-\gamma)T \log(n-1) \leq (r_{CC} + r_{DD} - r_{CD} - r_{DC})p + r_{CD} - r_{DD}. \quad (13)$$

Since the right hand side of Formula 13 is a linear function of p and $0 \leq p \leq 1$, the sufficient condition for satisfying Formula 8 is

$$(1-\gamma)T \log(n-1) \leq \begin{cases} r_{CD} - r_{DD} \\ \quad \text{if } r_{CC} + r_{DD} \geq r_{CD} + r_{DC}, \\ r_{CC} - r_{DC} \quad \text{otherwise,} \end{cases} \quad (14)$$

and the sufficient condition for violating Formula 8 is

$$(1-\gamma)T \log(n-1) > \begin{cases} r_{CC} - r_{DC} \\ \quad \text{if } r_{CC} + r_{DD} \geq r_{CD} + r_{DC}, \\ r_{CD} - r_{DD} \quad \text{otherwise.} \end{cases} \quad (15)$$

Similarly, if the desirable action is D, then

$$Q(D) \geq Q(C) + T\log(n-1). \tag{16}$$

We get the sufficient condition for satisfying Formula 16

$$(1-\gamma)T\log(n-1) \leq \begin{cases} r_{DD} - r_{CD} \\ \quad \text{if } r_{CD} + r_{DC} \geq r_{CC} + r_{DD}, \\ r_{DC} - r_{CC} \quad \text{otherwise,} \end{cases} \tag{17}$$

and the sufficient condition for violating Formula 16

$$(1-\gamma)T\log(n-1) > \begin{cases} r_{DC} - r_{CC} \\ \quad \text{if } r_{CD} + r_{DC} \geq r_{CC} + r_{DD}, \\ r_{DD} - r_{CD} \quad \text{otherwise.} \end{cases} \tag{18}$$

Since the left hand side and the right hand side in Formula 15 are larger than 0 and smaller than 0 in the PD setting, respectively, agents using normal Q-learning do not take desirable action C in the setting. On the other hand, Formula 17 can be satisfied in the DG setting by agents using normal Q-learning.

3.2. Q-learning with the opponent's rewards (NR)

Here, we see the case in which agents use the opponent's rewards as the added parameter r' of self-evaluations. It is introduced in [4, 5] as NR (Neighbors' Rewards) based on [3] [4]. In this case, a self-evaluation becomes the sum of both players' rewards. Hence, Formulae 9 and 10 become as follows.

$$\begin{aligned} Q(C) &= \frac{2r_{CC}}{1-\gamma}p + \frac{r_{CD}+r_{DC}}{1-\gamma}(1-p) \\ &= \frac{2r_{CC} - r_{CD} - r_{DC}}{1-\gamma}p + \frac{r_{CD}+r_{DC}}{1-\gamma}, \end{aligned} \tag{19}$$

$$\begin{aligned} Q(D) &= \frac{r_{DC}+r_{CD}}{1-\gamma}p + \frac{2r_{DD}}{1-\gamma}(1-p) \\ &= \frac{r_{CD}+r_{DC}-2r_{DD}}{1-\gamma}p + \frac{2r_{DD}}{1-\gamma}. \end{aligned} \tag{20}$$

From them,

$$\begin{aligned} &Q(C) - Q(D) \\ &= \frac{2(r_{CC}+r_{DD}-r_{CD}-r_{DC})}{1-\gamma}p + \frac{r_{CD}+r_{DC}-2r_{DD}}{1-\gamma}, \end{aligned} \tag{21}$$

$$\begin{aligned} &Q(D) - Q(C) \\ &= \frac{2(r_{CD}+r_{DC}-r_{CC}-r_{DD})}{1-\gamma}p + \frac{2r_{DD}-r_{CD}-r_{DC}}{1-\gamma}. \end{aligned} \tag{22}$$

4 In [3], the average of summed reward is used. The authors of [10] consider the effect of actions of the agent that is also similar to this self-evaluation.

If the desirable action is C, from Formulae 8 and 21, we get

$$\begin{aligned} &(1-\gamma)T\log(n-1) \\ &\quad \leq 2(r_{CC}+r_{DD}-r_{CD}-r_{DC})p + r_{CD} + r_{DC} - 2r_{DD}. \end{aligned} \tag{23}$$

Since the right hand side is a linear function of p and $0 \leq p \leq 1$, the sufficient condition for satisfying Formula 8 is

$$(1-\gamma)T\log(n-1) \leq \begin{cases} r_{CD} + r_{DC} - 2r_{DD} \\ \quad \text{if } r_{CC} + r_{DD} \geq r_{CD} + r_{DC}, \\ 2r_{CC} - r_{CD} - r_{DC} \quad \text{otherwise,} \end{cases} \tag{24}$$

and the sufficient condition for violating Formula 8 is

$$(1-\gamma)T\log(n-1) > \begin{cases} 2r_{CC} - r_{CD} - r_{DC} \\ \quad \text{if } r_{CC} + r_{DD} \geq r_{CD} + r_{DC}, \\ r_{CD} + r_{DC} - 2r_{DD} \quad \text{otherwise.} \end{cases} \tag{25}$$

Similarly, if desirable action is D, we get the sufficient condition for satisfying Formula 16

$$(1-\gamma)T\log(n-1) \leq \begin{cases} 2r_{DD} - r_{CD} - r_{DC} \\ \quad \text{if } r_{CD} + r_{DC} \geq r_{CC} + r_{DD}, \\ r_{CD} + r_{DC} - 2r_{CC} \quad \text{otherwise,} \end{cases} \tag{26}$$

and the sufficient condition for violating Formula 16

$$(1-\gamma)T\log(n-1) > \begin{cases} r_{CD} + r_{DC} - 2r_{CC} \\ \quad \text{if } r_{CD} + r_{DC} \geq r_{CC} + r_{DD}, \\ 2r_{DD} - r_{CD} - r_{DC} \quad \text{otherwise.} \end{cases} \tag{27}$$

In this case, Formulae 24 and 26 can be satisfied in the PD setting and in the DG setting, respectively.

3.3. Verification

To verify the discussion, the author conducted two experiments, in which one is for the Prisoner's Dilemma setting and the other is for the Deadlock Game setting. The parameter of Q-learning is as follows: Learning Rate $\alpha = 0.005$, Discount Factor $\gamma = 0.5$, and Temperature $T = 1$. In Formula 7, set $n = 30$.

The rewards in each games are as follows[5]:

Prisoner's Dilemma (PD)

$r_{DC} = 5.684,\ r_{CC} = 4.684,\ r_{DD} = 1,\ r_{CD} = 0,$

Deadlock Game (DG)

$r_{DC} = 4.684,\ r_{DD} = 2.684,\ r_{CC} = 2,\ r_{CD} = 0.$

In this reward setting, the sufficient conditions in previous subsections say that NR works in PD and does not work in DG, while normal Q-learning works in DG.

5 $(1-\gamma)T\log(n-1) \equiv (1-0.5)\times 1 \times \log 29 \simeq 1.684.$

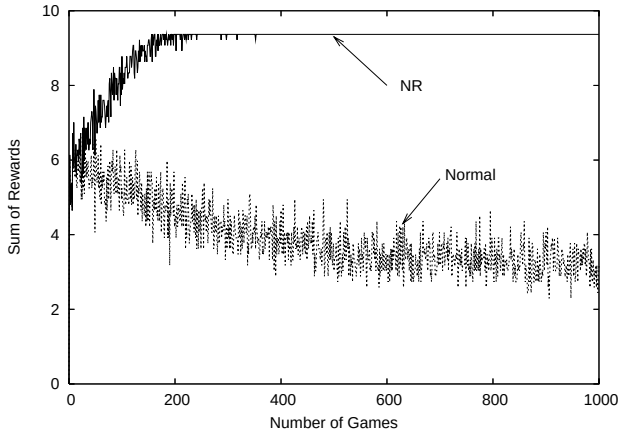

Figure 1. Result of PD: NR works and Normal does not

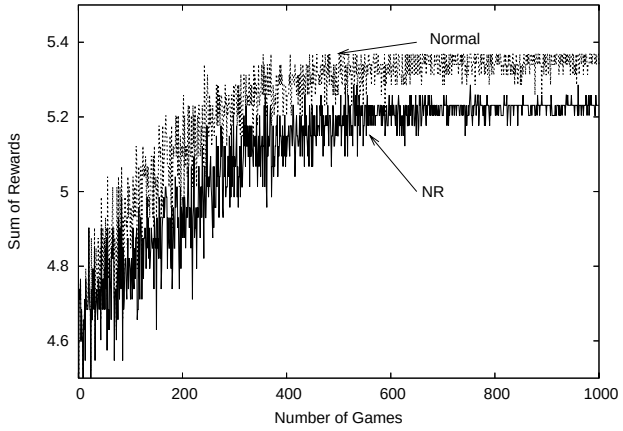

Figure 2. Result of DG: Normal works and NR does not

The results of the experiments are shown in Figures 1 and 2. The experiments are in self-play games, in which both players are agents having same algorithms. The x axes are the number of played games and the y axes show the sum of rewards of two agents averaged in 25 trials. In PD, the ideal, maximum value is $4.684 \times 2 = 9.368$, that is achieved by NR while not by normal Q-learning. It is consistent with the consequence of the conditions. In DG, the ideal, maximum value is $2.684 \times 2 = 5.368$, that is almost achieved by normal Q-learning while not by NR. It is also consistent with the consequence of the conditions. Indeed, about 10% of NR agents take C at the 1000th game, that is larger than $1/n \equiv 1/30$.

4. Self-evaluation generator without others' information

4.1. How to calculate self-evaluation

Formulae 8 and 16 say that the action value function $Q(a)$ of a desirable action a should be larger than the function of another action plus $T\log(n-1)$. In other words, agents can learn desirable actions by letting $Q(a)$ be larger than the function of another action plus $T\log(n-1)$. It means that we are able to let the added parameter r' be independent from others because $T\log(n-1)$ can be calculated locally. That is, we are able to construct a self-evaluation generator without other agents' information.

Let $q(a)$ be an additional action value function for the action value function $Q(a)$. Then, since the added parameter $r'(a)$ is independent from other agents and $q(a)$ is the expected sum of discounted $r'(a)$, the relation between $r'(a)$ and $q(a)$ is

$$q(a) = \frac{r'(a)}{1-\gamma}. \tag{28}$$

Let $a_{\max} \equiv \arg\max_{a' \in \{C,D\}} Q(a')$, then $q(a)$ has to satisfy

$$Q(a) + q(a) \geq Q(a_{\max}) + T\log(n-1) \tag{29}$$

to make the probability of selecting another action be less than $1/n$. Therefore, $r'(a)$ should be

$$r'(a) \geq (1-\gamma)\Big\{T\log(n-1) + Q(a_{\max}) - Q(a)\Big\}. \tag{30}$$

4.2. Verification

To verify Formula 30, we construct two types of self-evaluation generators, one is ToC and the other is ToD, each of which utilizes the parameter $r'(a)$ equal to the right hand side of the formula. The desirable action of ToC and ToD is C and D, respectively. The other settings are identical with those in Section 3.3.

The results of the experiments are shown in Figures 3 and 4. We can see that ToC almost converges to the ideal value in Prisoner's Dilemma and ToD also does in Deadlock Game. It shows that ToC generates self-evaluations by which the agent learns a desirable action in the Prisoner's Dilemma setting without other agents' information.

5. Conclusion

In this paper, we discuss the sufficient conditions of whether or not a Q-learner with or without a self-evaluation generator, which uses the softmax method for action selection, is able to learn an desirable action in symmetric 2×2 games. It shows that (1) agents having normal Q-learners cannot select the desirable action C in Prisoner's Dilemma, (2) there are some game settings in Deadlock Game in

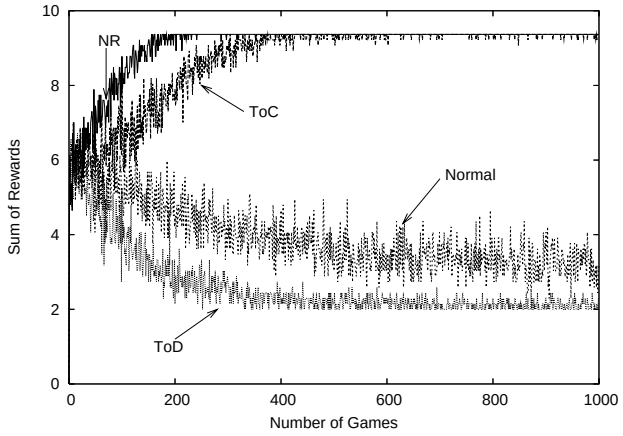

Figure 3. Result of PD: ToC works without opponent's information

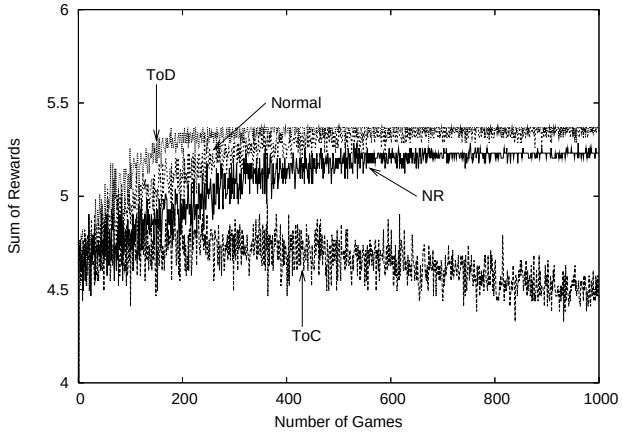

Figure 4. Result of DG: ToD works without opponent's information

which agents having *Q*-learners with NR cannot achieve the condition of upper limit of probability of undesirable action, and (3) we are able to construct a self-evaluation generator that lets the agent learn a desirable action without other agents' information.

Section 4.2 shows that To*C* fails to learn in Deadlock Game and To*D* fails in Prisoner's Dilemma, too. It is similar to that NR fails to achieve the condition of upper limit of probability of undesirable actions in Deadlock Game, while normal *Q*-learning fails in Prisoner's Dilemma. In [4, 5], the author has constructed an agent judging whether or not the game is like PD to learn appropriate actions in both PD-like and non-PD-like games through self-evaluations. We should integrate this work and the judging method to select the self-generators according to the game. Then, we can realize an autonomous agent that overcomes the trade-off property of the self-generators.

References

[1] J. Hu and M. P. Wellman, "Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm", *Proc. 15th International Conference on Machine Learning, ICML'98*, pp. 242–250, Madison, Wisconsin, U.S.A., 1998.

[2] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning", *Proc. 11th International Conference on Machine Learning, ML'94*, pp. 157–163, New Brunswick, New Jersey, U.S.A., 1994.

[3] S. Mikami and Y. Kakazu, "Co-operation of Multiple Agents Through Filtering Payoff", *Proc. 1st European Workshop on Reinforcement Learning, EWRL-1*, pp. 97–107, Brussels, Belgium, 1994.

[4] 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.

[5] K. Moriyama and M. Numao, "Self-Evaluated Learning Agent in Multiple State Games", *Proc. 14th European Conference on Machine Learning, ECML-2003*, (Lecture Notes in Artificial Intelligence 2837), pp. 289–300, Cavtat-Dubrovnik, Croatia, 2003.

[6] Y. Nagayuki, S. Ishii, and K. Doya, "Multi-Agent Reinforcement Learning: An Approach Based on the Other Agent's Internal Model", *Proc. 4th International Conference on MultiAgent Systems, ICMAS-2000*, pp. 215–221, Boston, Massachusetts, U.S.A., 2000.

[7] W. Poundstone, *Prisoner's Dilemma*, Doubleday, New York, 1992.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[9] C. J. C. H. Watkins and P. Dayan, "Technical Note: Q-learning", *Machine Learning*, 8:279–292, 1992.

[10] D. H. Wolpert and K. Tumer, "Collective Intelligence, Data Routing and Braess' Paradox", *Journal of Artificial Intelligence Research*, 16:359–387, 2002.