Model-based Reinforcement Learning for Partially Observable Games with Sampling-based State Estimation

Hajime Fujita and Shin Ishii
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{hajime-f,ishii}@is.naist.jp

Abstract

We present a model-based reinforcement learning (RL) scheme for large-scale multi-agent problems with partial observability, and apply it to a card game, "Hearts". This game is a well-defined example of an imperfect information game. To reduce the computational cost, we use a sampling technique based on Markov chain Monte Carlo (MCMC) in which the heavy integration required for the estimation and prediction can be approximated by a plausible number of samples. Computer simulation results show that our RL agent can perform learning of an appropriate strategy and exhibit a comparable performance to an expert-level human player in this partially-observable multi-agent problem.

Keywords: model-based reinforcement learning, partially observable Markov decision process (POMDP), multi-agent problem, sampling technique, card game

1 Introduction

Reinforcement learning (RL) (Sutton & Barto, 1998) has been devoted much attention as an effective framework for strategic decision processes in multi-agent systems (Shoham, Powers, & Grenager, 2004). An optimal control problem in multi-agent environments, however, has a high degree of difficulty due to interactions among agents. These may make the Markov property of the state space fail because the changing behaviors of other agents provide dynamic nature. Although several RL researches based on the game theory have attained some remarkable results in a reasonably sized state space (Crites & Barto, 1996; Littman, 1994), it is difficult to deal with real-world applications due to their serious complexity.

In addition, the environments often have partial observability; the agents cannot directly access internal states of the environment, but can get only observations which contain partial information about the state. Decision-making problems in such a situation can be formulated as partially observable Markov decision processes (POMDPs) (Kaelbling, Littman, & Cassandra, 1998). When introducing this framework to realistic problems, however, serious difficulties arise because not only the estimation process for a large number of unobservable states, but also computing the optimal policy depending on the estimation require too heavy computation. To deal with large-scale multi-agent problems with partial observability and to follow the environmental dynamics, an estimation method with an effective approximation and explicit learning of its model are necessary.

In this article, we present an automatic strategy-acquisition scheme for large-scale multi-agent problems with partial observability, and deal in particular with the card game "Hearts". To estimate unobservable states, we use a sampling technique (Thrun, 2000) based on Markov chain Monte Carlo (MCMC) (Gilks, Richardson, & Spiegelhalter, 1996) for estimating the unobservable variables; the heavy integration due to the large state space is approximated by a plausible number of samples, each of which represents a discrete state. To predict the unknown environmental behaviors, we use a model-based approach (Sutton, 1990); the learning agent based on our RL method has multiple action predictors, each of which represents a policy of the opponent agent, and makes them learn independently. These ideas provide us with an effective solution for large-scale partially observable problems and ability to apply for various multi-agent settings, including games with multiple players; this is shown by computer simulations using expert-level rule-based agents. These results suggest that our method is effective in solving realistic multi-agent problems with partial observability.

2 Partially observable Markov decision process (POMDP)

A POMDP (Kaelbling et al., 1998) is a framework to make an agent learn and act in a partially observable environment, and consists of (1) a set of real states $\mathcal{S} = \{s_1, s_2, \cdots, s_{|\mathcal{S}|}\}$, (2) a set of observation states $\mathcal{O} = \{o_1, o_2, \cdots, o_{|\mathcal{O}|}\}$, (3) a set of actions $\mathcal{A} = \{a_1, a_2, \cdots, a_{|\mathcal{A}|}\}$, and (4) a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$. The dynamics of the model is represented as transition probability $P(s_{t+1}|s_t, a_t)$ and observation probability $P(o_t|s_t, a_t)$. The objective of each agent is to acquire the policy which maximizes an expected future reward in the partially observable world, in which the state s_t is not observable for each agent; only the observation o_t, which contains partial information about the state, is available. One way to obtain an optimal solution is to calculate a belief state $b(s_t) \equiv P(s_t|H_t)$, which summarizes the whole history $H_t = \{(o_t, -), (o_{t-1}, a_{t-1}), \cdots, (o_1, a_1)\}$ as a probability distribution over $\mathcal{S}$, and learn a value function $V(b_t)$ over the belief space. Although this formulation, called as "belief-state MDP", has the capability of solving a POMDP, an exact solution is hard to achieve because of the requirement for computing a policy over the entire belief space, whose cost increases exponentially with the increase in state number of the underlying MDP. Algorithms for computing an optimal policy, therefore, were considered impractical for large-scale domains, and recent research has focused on approximate algorithms that scale up the application area effectively (Hauskrecht, 2000).

The targets of this study are partially observable and multi-agent problems; there are multiple agents in a common environment with partial observability. In this article, we then use the following notations. t indicates an action turn of the learning agent. The variables (state, observation and action) for agent i $(i = 0, \ldots, M)$ are denoted by s_t^i, o_t^i and a_t^i, where M is the number of opponent agents and $i = 0$ signifies the learning agent; s_t, o_t and a_t are the same as s_t^0, o_t^0 and a_t^0, respectively. A strategy of an agent i is denoted by ϕ^i. Note that we make an assumption that there is only one learning agent in the environment, and the other agents' strategies ϕ^i $(i = 1, \ldots, M)$ are fixed, for the time being. This assumption will be loosened later. An action sequence of the opponent agents is denoted by $u_t = \{a_t^1, \cdots, a_t^M\}$ and a history for the learning agent at its t-th action turn is given by $H_t \equiv \{(o_t, -, -), (o_{t-1}, a_{t-1}, u_{t-1}), \cdots, (o_1, a_1, u_1)\}$.

3 Model

In our RL method, an action is selected according to the greedy policy:

$$\pi(H_t) = \operatorname*{argmax}_{a_t} \; U(H_t, a_t), \tag{1}$$

where $U(H_t, a_t)$ is the utility function at a time step t. This function is defined as an expectation of a one-step-ahead utility value with respect to the belief state and transition probability:

$$U(H_t, a_t) = \sum_{s_t \in \mathcal{S}_t} P(s_t|H_t) U(s_t, a_t) \tag{2a}$$

$$U(s_t, a_t) = \sum_{s_{t+1} \in \mathcal{S}_{t+1}} P(s_{t+1}|s_t, a_t) \left[R(s_t, a_t) + V(s_{t+1}) \right], \tag{2b}$$

where $R(s_t, a_t)$ denotes an immediate reward at the time step $t+1$, and $V(s_{t+1})$ denotes the state value function of the next state s_{t+1}. In our application, the card game Hearts, the reward is defined as $R(s_t, a_t) = -n$ when the agent gets n penalty points (n may be 0) after the t-th play. The value function V is approximated by a normalized Gaussian network (NGnet) (Sato & Ishii, 2000) with a feature extraction technique to its 52-dimensional input; by considering the game property, a state s_t is converted to a 36-dimensional input p_t before the value function is updated so as to approximate the relationship between the input p_t and the scalar output $\sum_{i=t}^{13} R(s_i, a_i)$ according to the Monte Carlo RL method (Sutton & Barto, 1998).

In large-scale problems, it is difficult to learn the value function over the belief space (Hauskrecht, 2000). We then use a completely observable approximation (Littman, Cassandra, & Kaelbling, 1995); the agent maintains the state value function so that the self-consistency equation holds on the underlying MDP, and calculates the state-action utility value by a one-step-ahead prediction according to equation (2b). After that, according to equation (2a), it calculates the history-action utility value as an expectation of the state-action utility with respect to the belief state under the knowledge that the optimal value function for the belief space can be approximated well by a piecewise-linear and convex function (Smallwood & Sondik, 1973). The calculation of the utility function, however, includes three difficulties: (a) the computation for constructing the belief state $P(s_t|H_t)$ over possible current states is intractable due to the large state space and high dimensionality; (b) the prediction to possible next states is difficult because the environmental model $P(s_{t+1}|s_t, a_t)$ is unknown for the agent and may be changed in a multi-agent setting; and (c) the summation in equation (2) over possible current states and next states has computational intractability because there are so many candidates in a realistic problem. Some effective approximations, therefore, are required for avoiding the above difficulties.

To avoid difficulty (a), we do not deal with the whole history H_t but do a one-step history $h_t = \{(o_t, -, -), (o_{t-1}, a_{t-1}, u_{t-1})\}$, which leads us to make an assumption (A): the belief state represents a simple one-step prior knowledge about states, but does not carry the complete likelihood information. The history H_t contains two kinds of information. The first is about impossible states at the t-th turn; for example, in the game Hearts, if an agent played ♡9 after a leading card ♣3 in a past trick, the agent no longer has any club cards at the t-th turn and any state in which this agent holds club cards is impossible. The second is about likelihood, considering the characteristics of the opponent agents; for example, in the same situation as above, it is unlikely for the agent to have any heart card higher than ♡9. Although the belief state $P(s_t|H_t)$, which is the sufficient statistic for the history H_t, should involve these two kinds of information, we partly ignore the latter kind by replacing the whole history H_t with a one-step history h_t; namely, the belief state $P(s_t|H_t)$ is approximated by the partial belief state $P(s_t|h_t)$ in this study. No impossible state, on the other hand, is considered in light of the former type of information, but each possible state has a one-step likelihood between the $(t-1)$-th and t-th time steps. Although the maintenance of likelihood over possible states requires heavy computation and a large amount of memory in realistic problems, this assumption enables us to estimate internal states easily at each time step.

To solve problem (b), the agent uses action predictors. Since the state transition of usual multi-agent games depends on the other players' actions, the transition probability $P(s_{t+1}|s_t, a_t)$ in equation (2b) is calculated by the product of action selection probabilities for M opponent agents, that is,

$$P(s_{t+1}|s_t, a_t) \approx P(s_{t+1}|s_t, a_t, \hat{\Phi}) = \prod_{i=1}^{M} P(a_t^i|o_t^i, \hat{\phi}^i), \tag{3}$$

where $\hat{\Phi} = \{\hat{\phi}^1, \dots, \hat{\phi}^M\}$. Note that $\hat{\phi}^i$ is not a real policy ϕ^i but a policy approximated by the agent. $P(a_t^i|o_t^i, \hat{\phi}^i)$ in equation (3), which represents the probability that the i-th agent's action is a_t^i for a given observation o_t^i, is calculated by the i-th action predictor ($i = 1, \dots, M$). The learning agent maintains M action predictors corresponding to the M opponent agents. The agent predicts that the i-th opponent agent selects an action a_t^i according to the soft-max policy:

$$P(a_t^i|o_t^i, \hat{\phi}^i) = \frac{\exp(F^i(o_t^i, a_t^i)/T^i)}{\sum_{\mathcal{A}^i} \exp(F^i(o_t^i, a_t^i)/T^i)}. \tag{4}$$

Note that the opponent agent's observation o_t^i is not observable to the agent, but can be determined from the estimated current state $\hat{s}_t$ without any ambiguity in usual games whose observation process is deterministic. $F^i(o_t^i, a_t^i)$ denotes the utility of action a_t^i for a given observation o_t^i of the i-th agent, which is an output of the action predictor. $\mathcal{A}^i$ denotes the set of possible actions for agent i, and T^i is a constant which denotes the assumed randomness of the agent i's policy. Equation (3) represents the behavior model of the environment. The action predictors thus approximate the environmental dynamics for the learning agent. Each predictor is implemented as Multi-Layered Perceptron (MLP), and its input and output are reduced to reasonably sized vectors by a feature extraction technique which is the same as our previous study (Ishii, Fujita, Mitsutake, Yamazaki, Matsuda, & Matsuno, 2005).

To avoid the computational intractability problem (c), we use sampling-based approximation; the agent obtains independent and identically distributed (i.i.d.) random samples, $\hat{s}_t$ and $\hat{s}_{t+1}$, whose probabilities are proportional to the partial belief state $P(s_t|h_t)$ and acquired environmental model $P(s_{t+1}|s_t, a_t, \hat{\Phi})$, respectively. By considering the two approximations described above, the utility function in equation (2) can be calculated as

$$\begin{aligned} U(H_t, a_t) &\approx \sum_{s_t \in \mathcal{S}_t} P(s_t|h_t) \sum_{s_{t+1} \in \mathcal{S}_{t+1}} P(s_{t+1}|s_t, a_t, \hat{\Phi}) \left[R(s_t, a_t) + V(s_{t+1})\right] \\ &\approx \frac{1}{K} \sum_{i=1}^{N} P(\hat{s}_t^{(i)}|h_t) \sum_{j=1}^{K} \left[R(\hat{s}_t^{(j)}, a_t) + V(\hat{s}_{t+1}^{(j)})\right]. \end{aligned} \quad (5)$$

Samples of current states $\hat{s}_t$ are obtained by the Metropolis-Hastings (MH) algorithm, the most popular Markov chain Monte Carlo (MCMC) technique (Gilks et al., 1996), in the following three steps: the first step is to sample a previous state $\hat{s}_{t-1}^*$ so as not to violate the whole history H_t, no impossible state being sampled (according to the former type of information described above); the second step is to calculate a one-step likelihood $P(\hat{s}_t^*|\hat{s}_{t-1}^*, a_t)$ by using the action predictors according to equation (3); and the last step is to replace $\hat{s}_t^*$ with $\hat{s}_t^{(i+1)}$ according to the probability $p = \min\{1, P(\hat{s}_t^*|\hat{s}_{t-1}^*, a_t)/P(\hat{s}_t^{(i)}|\hat{s}_{t-1}^{(i)}, a_t)\}$ and otherwise $\hat{s}_t^{(i)}$ remains. Note that a Markov chain is uniform according to assumption (A). These three steps are iterated N times, and the agent obtains estimated current states $\{\hat{s}_t^{(i)}|i = 1, \cdots, N\}$. Samples of next states $\hat{s}_{t+1}$ are obtained by a simple sampling technique, according to equations (3) and (4) given an estimated current state $\hat{s}_t^{(i)}$, an action a_t and an action sequence $\hat{u}_t = \{\hat{a}_t^1, \ldots, \hat{a}_t^M\}$ by the available fact that $\hat{s}_{t+1}^{(j)}$ can be determined without any ambiguity in usual games due to the deterministic nature of $P(s_{t+1}|s_t, a_t, u_t)$. This is iterated K times for each of N current states, and the agent obtains predicted next states $\{\hat{s}_{t+1}^{(j)}|j = 1, \cdots, K\}$ with the learned model. Two summations in equation (2) are simultaneously approximated by using KN samples in equation (5). The three approximations described above (the partial belief state, action predictor and sampling) enable us to solve large-scale and partially observable problems. In particular, they provide an effective solution to multi-agent problems whose underlying state space is discrete, including various multi-agent games.

4 Computer simulations

We applied our RL method to the card game Hearts, which is a well-defined example of large-scale and multi-agent problems with partial observability; it has about $52!/(13!)^4 \simeq 10^{28}$ states if every combination of 52 cards is considered, and many cards may be unobservable. To evaluate our method, we carried out computer simulations where an agent trained by our RL method played against rule-based agents which have more than 65 general rules for playing cards from their hands. The performance of an agent can be evaluated by the acquired penalty ratio, which is the ratio of the penalty points acquired by the agent to the total penalty points of the four agents. If the four agents have equal strength, their penalty ratio averages 0.25. The rule-based agent used in this study is much stronger than the previous one (Ishii et al., 2005), due to the improvement in the rules. Although the previous rule-based agent was an "experienced"-level player, the current rule-based agent has almost the same strength as an expert-level human Hearts player: when this rule-based agent challenged a human expert player, the acquired penalty ratio was 0.256. Since the outcome of this game tends to depend on the initial card distribution (for example, an expert player with a bad initial hand may be defeated by an unskilled player), we prepared a fixed data set for the evaluation; the data set is a collection of initial card distributions for 100 games, each of which was generated randomly in advance. In the evaluation games, the initial cards were distributed according to this data set. Since performance is influenced by seat position (that is, an agent may have an advantage/disadvantage based on its seat position if the agents have different strengths), we rotated the agents' positions for each initial hand to eliminate this bias; each of the 100 evaluation games was repeated four times with the four types of seating position. The performance of each agent, therefore, is evaluated by the 400 fixed and unbiased games. Note that learning of the agent was suspended during the evaluation games. Each learning run comprised several sets of 500 games, in which initial cards were distributed to the four agents at random and seat positions of the agents were determined randomly. In an experiment, accordingly, 400 evaluation games and 500 learning games were alternated.

Figure 1 shows the result when the agent trained by our method challenged the three rule-based agents. The abscissa of the lower panel denotes the number of training games and the ordinate denotes the penalty ratio acquired by each agent. Each point and error bar represent an average and standard deviation of the penalty ratio, respectively, for the 400 evaluation games over 17 learning runs, each consisting of 5,500 training games. The penalty ratio of the RL agent decreased with the learning process, and after about 5,000 training games, the agent became significantly stronger than the rule-based agents. Since the agent showed a better performance than the expert-level rule-based agents after only several thousand training games, the new RL method based on a sampling method is a salient improvement over the previous one, both in learning speed and in strength. Although the three rule-based agents have the same rules, there is a distinct difference in their performances. This comes from the fact that the relative seat position was not changed even with the rotation during the evaluation games.

When the RL agent challenged the three rule-based agents used in our previous work, it showed a better performance from the beginning of learning and finally became much better than the rule-based agents; this is why we have developed the new rule-based agent which is much stronger than the previous one. The drastic improvement by our new RL method is attributed to the following two facts. First, the ability to approximate the utility function in

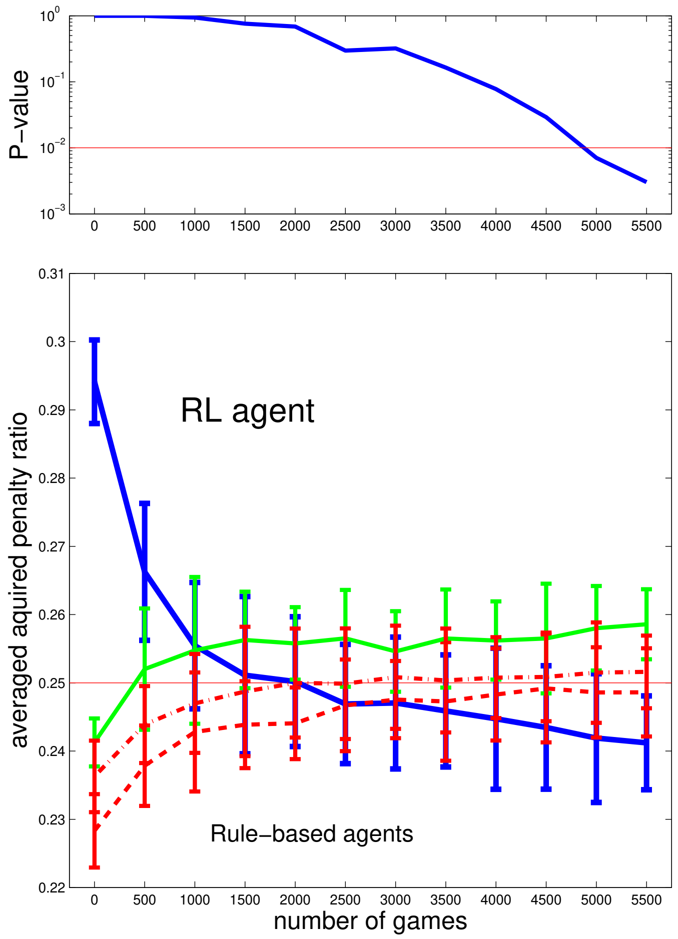

Figure 1: Computer simulation result in an environment where there are one learning agent trained by our RL method and three rule-based agents.

upper panel: P-values of the t-test where the null hypothesis is "the RL agent has the same strength as the rule-based agents" and the alternative hypothesis is "the RL agent is stronger than the rule-based agents". The test was done independently at each point on the abscissa. The horizontal line denotes the significance level of 1%. After about 5,000 training games, the RL agent was significantly stronger than the rule-based agents.

lower panel: The abscissa denotes the number of training games and the ordinate denotes the penalty ratio acquired by each agent. We executed 17 learning runs, each consisting of 5,500 training games. Each point and error bar represents the average and standard deviation, respectively, for the 400 evaluation games over the 17 runs. The constant T^i in equation (4) was 1.0, and the numbers of samples in equation (5) were $N = 80$ and $K = 20$.

equation (2) is improved by replacing the analog approximation method with the discrete sampling-based one. In our previous work, to calculate the utility function, we applied the mean-field-like analog approximation to the problem whose state space is discrete by changing the order of summations; we calculated the summation over current states with the approximation before calculating the summation over the next states. In this study, on the contrary, the summations are calculated in a straightforward way with the sampling-based approximation. It enables us to calculate the expectation with a higher approximation accuracy. Second, the expected future reward could be evaluated with higher accuracy by the state value function. In our previous work, we made the value function learn over the observation space. Although this is an effective method for large-scale POMDP problems, it is difficult to obtain an accurate value due to the perceptual aliasing property in partially observable environments. In this study, in contrast, the learning agent makes a prediction to the next states and evaluates a value by the value function over the state space. This enables the agent to perform accurate value prediction. The ideas used in our previous work were adequate for an environment with moderate complexity, constituted by the previous rule-based agents, but the limitation of that method precluded a more remarkable result. In this study, we have improved the old model so that the method can work well within only several thousand training games, even in the harder environment constituted by the expert-level rule-based agents.

Figure 2 shows the result when one agent trained by our RL method, one agent trained by our previous RL method, and two rule-based agents played against each other. We executed 16 learning runs, each consisting of 4,000 training games. Although the penalty ratio of the RL agent became smaller than the rule-based agents after 3,500 training games, the ratio of the previous RL agent did not decrease and it remained much weaker than the other agents. This result shows that our new agent can acquire a better strategy than the previous one through a direct match. In the previous experiment (Fig.1), our method was based on the assumption that there is only one learning agent in the environment. In this experiment, our method was applied directly to the multi-agent environment, in which there are multiple learning agents, and the new RL method showed good performance even in this complex setting.

Figure 3 shows the result when two RL agents trained by our method and two rule-based agents played against each other. We executed 16 learning runs, each consisting of 4,000 training games. The penalty ratios of both RL agents came to acquire a smaller penalty ratio than the rule-based agents after about 5,000 training games. The setting of this experiment is more difficult than the previous one (Fig.2), because the learning speed of a learning agent trained by the new RL method is much faster than that of a learning agent trained by the previous method. In other words, the environment changes its dynamics more rapidly. Even with this difficult multi-agent setting, the RL agents could adapt to the change, and showed good performance. This ability is attributed to the fast learning owing to using three action predictors.

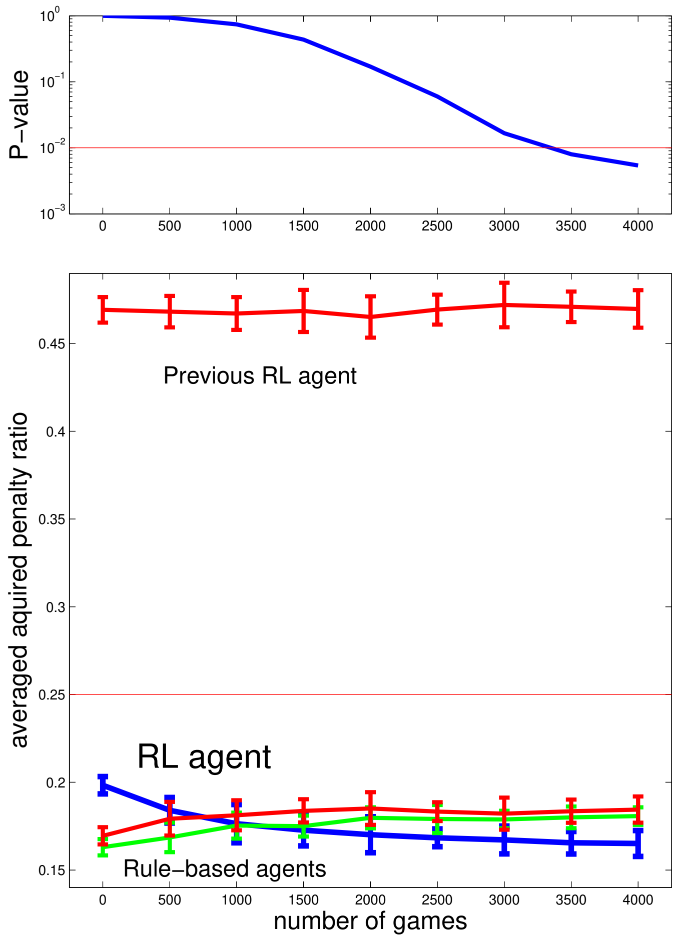

Figure 2: Computer simulation result in an environment where there are one learning agent trained by our RL method, one learning agent trained by our previous method, and two rule-based agents.

upper panel: P-values of the t-test where the null and alternative hypotheses are the same as in the previous experiment (Fig.1). After about 3,500 training games, the RL agent was significantly stronger than the rule-based agents, but the previous RL agent was not (P-values are not shown because they remained around 1).

lower panel: The abscissa and the ordinate denote the same as in Fig.1, but note that the scale of the ordinate is larger here. We executed 16 learning runs, each consisting of 4,000 training games. The parameter values and other experimental setups are also the same.

To validate the general strength of the learning agent, we carried out a direct match to a human expert player. Figure 4 shows the result when one RL agent trained by our method, two rule-based agents, and an expert-level human Hearts player played together.

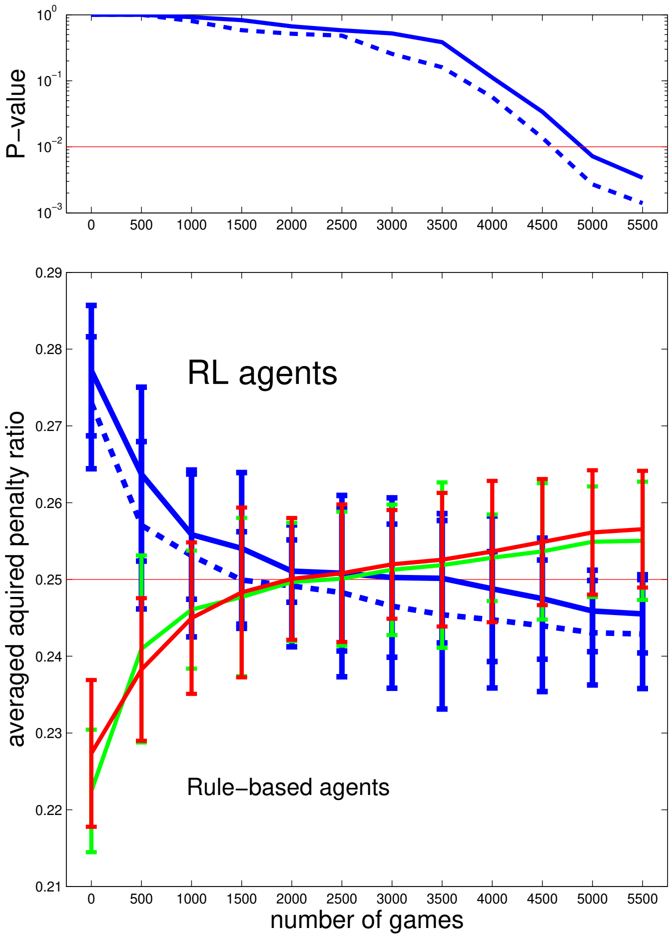

Figure 3: Computer simulation result in an environment where there are two learning agents trained by our RL method and two rule-based agents.

upper panel: P-values of the t-test where the null and alternative hypotheses are the same as in the previous experiment. After about 5,000 training games, the two RL agents were significantly stronger than the rule-based agents.

lower panel: The abscissa and the ordinate denote the same as in the previous experiment (Fig.1). We executed 18 learning runs, each consisting of 5,500 training games. The parameter values and other experimental setups are also same.

We used another evaluation data set of 25 games, with seat rotation; 100 evaluation games were done before learning and after 1,000, 2,000, 3,000, 4,000 and 5,000 training games. We repeated the training and evaluation runs twice. Each point denotes the average of 200 (2×100) evaluation games. The learning run was executed twice, and each point of the figure represents a mean over 200 games. The RL agent successfully acquired a general

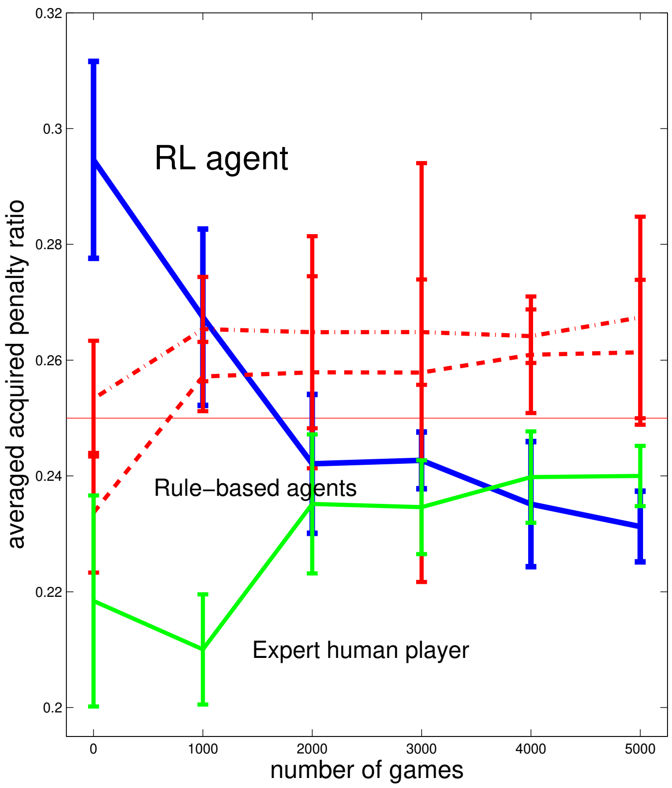

Figure 4: Computer simulation result when one learning agent trained by our RL method, one human expert player and two rule-based agents played together. 100 evaluation games were done before learning and after 1,000, 2,000, 3,000, 4,000 and 5,000 training games. We repeated the training and evaluation runs twice. The abscissa and the ordinate denote the same as in the previous experiment. The parameter values are also the same. Each point denotes the average of 200 (2×100) evaluation games.

strategy which is comparable to or slightly better than the strategy of the human expert player.

5 Conclusion

In this study, we developed a new model-based RL scheme for a large-scale multi-agent game with partial observability, and applied the method to a realistic card game, “Hearts”. Since this game is a partially observable game, it is necessary for decision-making, to estimate unobservable states and predict opponent agents’ actions. This realistic game, however, has a very large state space, so it is difficult to estimate all possible current states and make a prediction to all possible next states. In our method, therefore, we avoided this computational intractability by using a sampling technique based on MCMC. We then proposed a model-based RL method, in which the action is selected to maximize the one-step-ahead utility prediction. Although this value prediction involves intractable integrations over possible states, the above sampling method has reduced the computational cost sufficiently to allow the agent to select a preferable action. Computer simulations showed that our model-based RL method is effective for acquiring good strategy in this realistic partially observable problem. In the current study, we have partly discarded the likelihood information, because an exact maintenance of the belief in the discrete state space is not easy with a restricted computer resource. It will be our future work to cope with this difficulty.

References

Crites, R. H., & Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS '96)*, Vol. 8, pp. 1017–1023.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (Eds.). (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall.

Hauskrecht, M. (2000). Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, *13*, 33–94.

Ishii, S., Fujita, H., Mitsutake, M., Yamazaki, T., Matsuda, J., & Matsuno, Y. (2005). A reinforcement learning scheme for a partially-observable multi-agent game. *Machine Learning*, *59*, 31–54.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99–134.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML '94)*, pp. 157–163.

Littman, M. L., Cassandra, A. R., & Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML '95)*, pp. 362–370.

Sato, M., & Ishii, S. (2000). On-line EM algorithm for the normalized Gaussian network. *Neural Computation*, *12*, 407–432.

Shoham, Y., Powers, R., & Grenager, T. (2004). Multi-agent reinforcement learning: a critical survey. In *Proceedings of AAAI Fall Symposium on Artificial Multi-Agent Learning*, pp. 1–13.

Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable processes over a finite horizon. *Operations Research*, *21*, 1071–1088.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning (ICML '90)*, pp. 216–224.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Thrun, S. (2000). Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems (NIPS '00)*, Vol. 12, pp. 1064–1070.