Hybrid Model for Multiagent Reinforcement Learning

Ville Könönen
Neural Networks Research Centre
Helsinki University of Technology
P.O. Box 5400, FI-02015 HUT, FINLAND
ville.kononen@hut.fi

***Abstract*— In this paper we propose a new method for reducing space and computational requirements of multiagent reinforcement learning based on Markov games. The proposed method estimates value functions by using two Q-value tables or function approximators. We formulate the method for symmetric and asymmetric multiagent reinforcement learning and discuss also some numerical approximation techniques. Additionally, we present a brief literature survey of multiagent reinforcement learning and test the proposed method with a simple example application.**

I. Introduction

Reinforcement learning methods have attained lots of attention in recent years. Although these methods and procedures were earlier considered to be too ambitious and to lack a firm foundation, they have now been established as practical methods for solving Markov decision processes. However, the requirement for reinforcement learning methods to work is that the problem domain where the methods are applied obeys the Markov property. In many real-world problems this property is not fully satisfied but many reinforcement learning methods can still handle these situations relatively well. Especially, in the case of two or more decision makers in the same system the Markov property does not always hold and more advanced methods should be used instead. One possible solution is to use competitive Markov decision processes since there exists a suitable theoretical framework for these processes and some learning methods have also been proposed.

Earlier work with multiagent reinforcement learning based on Markov games deals with methods where the whole state-action space is modeled with one complex function approximator. If opponents are modeled directly in the reinforcement learning method, as in the case of Markov games, the space and computational requirements of the learning model grow very quickly with the size of the problem instance. However, in many problems it is not necessary to model opponents' behavior in each state, i.e. it is possible to use a single-agent learning model in some states and a multiagent model in other states and agents can still perform almost optimally.

In this paper, we approach the problem by introducing a hybrid model and proposing appropriate learning methods for symmetric and asymmetric learning. The main idea of the proposed method is to divide the state space into two subspaces and to model the Q-function by using two separate Q-value tables. In addition, we discuss some numerical approximation techniques for this model and propose a suitable metric for evaluating the usefulness of a multiagent learning model in a particular state.

Multiagent reinforcement learning methods have been discussed earlier by many authors. Littman introduced a Q-learning method for Markov games with two players and a zero-sum payoff structure in [6]. This method is guaranteed to converge from arbitrary initial values to the optimal value functions. However, the zero-sum payoff structure can be a very restrictive requirement in some systems and thus Hu and Wellman [2] extended this algorithm to general-sum Markov games. Unfortunately, their method is guaranteed to converge only under very restrictive conditions. These limitations are relaxed in [7] and in [10] by adding some additional information about roles and payoff structures of the agents in the system.

Our previous contributions in the field of multiagent reinforcement learning include an asymmetric multiagent reinforcement learning model [3]. Additionally, we have proposed numerical methods for multiagent reinforcement learning in [4] and [5].

We begin the paper by introducing the background and basic solution concepts of game theory. Then we briefly go through the theory behind Markov decision processes and introduce some learning methods used with multiagent reinforcement learning problem. Finally, we propose our hybrid model and test the proposed method with a simple example problem.

II. Game Theory

This section is mainly concerned with the basic problem settings and definitions of game theory. We start with some preliminary information about mathematical games and then proceed to their solution concepts which are essential for the rest of the paper.

A. Basic Concepts

Mathematical games can be represented in different forms. The most important forms are the *extensive* form and the *strategic* form. Although the extensive form is the most richly structured way to describe game situations, the strategic form is conceptually simpler and it can be derived from the

extensive form. In this paper, we use games in strategic form for making decisions at each time step.

Games in strategic form are usually referred to as *matrix games* and particularly in the case of two players, if the payoff matrices for both players are separated, as *bimatrix games*. In general, an N-person matrix game is defined as follows:

Definition 1: A *matrix game* is a tuple $\Gamma = (A^1, \ldots, A^N, r^1, \ldots, r^N)$, where N is the number of players, A^i is the strategy space for player i and $r^i : A^1 \times A^2 \times \ldots \times A^N \rightarrow \mathbb{R}$ is the payoff function for player i.

In a matrix game, each player i simultaneously implements a strategy $a^i \in A^i$. In addition to pure strategies A^i, we allow the possibility that the player uses a random (mixed) strategy. If we denote the space of probability distributions over a set A by $\Delta(A)$, a randomization by a player over his pure strategies is denoted by $\sigma^i \in \Sigma^i \equiv \Delta(A^i)$.

B. Equilibrium Concepts

In decision problems with only one decision maker, it is adequate to maximize the expected utility of the decision maker. However, in games there are many players and we need to define more elaborated solution concepts. Next we will shortly present two relevant solution concepts of matrix games.

Definition 2: If N is the number of players, the strategies $\sigma^i_*, \ldots, \sigma^N_*$ constitute a *Nash equilibrium* solution of the game if the following inequality holds for all $\sigma^i \in \Sigma^i$ and for all i:

$$r^i(\sigma^1_*, \ldots, \sigma^{i-1}_*, \sigma^i, \sigma^{i+1}_*, \ldots, \sigma^N_*) \leq r^i(\sigma^1_*, \ldots, \sigma^N_*)$$

The idea of the Nash equilibrium solution is that the strategy choice of each player is a best response to his opponents' play and therefore there is no need for deviation from this equilibrium point for any player alone. Thus, the concept of Nash equilibrium solution provides a reasonable solution concept for a matrix game when the roles of the players are symmetric. However, there are decision problems in which one of the players has the ability to enforce his strategy to other players. For solving these kind of optimization problems we have to use a hierarchical equilibrium solution concept, i.e. the *Stackelberg equilibrium* concept. In the two-player case, where one player is acting as the leader (player 1) and the another as the follower (player 2), the leader enforces his strategy to the opponent and the follower reacts rationally to this enforcement.

The basic idea is that the leader enforces his strategy so that he enforces the opponent to select the response that leads to the optimal response for the leader. Algorithmically, in the case of finite bimatrix games where player 1 is the leader and player 2 is the follower, obtaining a Stackelberg solution $(a^1_s, a^2_s(a^1))$ can be seen as the following two-step algorithm:

1) $a^2_s(a^1) = \arg\max_{a^2 \in A^2} r^2(a^1, a^2)$
2) $a^1_s = \arg\max_{a^1 \in A^1} r^1(a^1, a^2_s(a^1))$

In the step 1, the follower's strategy is expressed as a function of the leader's strategy. In the step 2, the leader maximizes his own utility by selecting the optimal strategy pair. The only requirement is that the follower's response is unique; if this is not the case, some additional restrictions must be set.

III. Multiagent Reinforcement Learning

With two or more agents in the environment, the fundamental problem with single-agent Markov Decision Processes (MDPs) is that the approach treats the other agents as a part of the environment and thus ignores the fact that the decisions of the other agents may influence the state of the environment.

One possible solution is to use competitive multiagent MDPs, i.e. *Markov games*. In a Markov game, the process changes its state according to the action choices of all the agents and can thus be seen as a multicontroller MDP. Formally, we define a Markov game as follows:

Definition 3: A *Markov game* (*stochastic game*) is defined as a tuple $(S, A^1, \ldots, A^N, p, r^1, \ldots, r^N)$, where N is the number of agents, S is the set of all states, A^i is the set of all actions for each agent $i \in \{1, N\}$, $p : S \times A^1 \times \ldots \times A^N \rightarrow \Delta(S)$ is the state transition function, $r^i : S \times A^1 \times \ldots \times A^N \rightarrow \mathbb{R}$ is the reward function for the agent i. $\Delta(S)$ is the set of probability distributions over the set S.

Again, as in the case of single-agent MDP, we need a policy π^i for each agent i (the policy is assumed to be stationary):

$$\pi^i : S \rightarrow A^i, \forall i \in \{1, N\}. \tag{1}$$

In multiagent systems this policy function is not necessarily deterministic. However, here we assume that the randomization is performed inside the policy function and therefore π^i returns actions directly. The expected value of the discounted utility R^i for the agent i is the following:

$$\begin{aligned} V^i_{\pi^1,\ldots,\pi^N}(s) &= E_{\pi^1,\ldots,\pi^N}[R^i | s_0 = s] \\ &= E_{\pi^1,\ldots,\pi^N}\left[\sum_{t=0}^{\infty} \gamma^t r^i_{t+1} | s_0 = s\right], \end{aligned} \tag{2}$$

where r^i_{t+1} is the immediate reward for agent i after the state transition and γ is a discount factor. Moreover, the value for each state-action pair is

$$\begin{aligned} &Q^i_{\pi^1,\ldots,\pi^N}(s, a^1, \ldots, a^N) \\ &= E_{\pi^1,\ldots,\pi^N}[R^i | s_0 = s, a^1_0 = a^1, \ldots, a^N_0 = a^N] \\ &= r^i(s, a^1, \ldots, a^N) \\ &+ \gamma \sum_{s'} p(s' | s, a^1, \ldots, a^N) V^i_{\pi^1,\ldots,\pi^N}(s'). \end{aligned} \tag{3}$$

Contrast to single-agent MDPs, finding the optimal policy π^i_* for each agent i can be seen as a game theoretical problem where the strategies the players can choose are the policies defined in Eq. (1).

A. Solving Markov Games

In the case of multiagent reinforcement learning, it is not enough to maximize the expected utility of individual agents. Instead, our goal is to find an equilibrium policy of the Markov game, e.g. a Nash equilibrium policy. The Nash equilibrium policy is defined as follows:

Definition 4: If N is the number of agents and Π^i is the policy space for agent i, the policies $\pi^1_*, \ldots, \pi^N_*$ constitute a Nash equilibrium solution of the game if the following inequality holds for all $\pi^i \in \Pi^i$ and for all i in each state $s \in S$:

$$V^i_{\pi^1_*,\ldots,\pi^i,\ldots,\pi^N_*}(s) \leq V^i_{\pi^1_*,\ldots,\pi^N_*}(s)$$

It is noteworthy that Definition 4 coincides with Definition 2 when individual strategies are replaced with policies. The Stackelberg equilibrium concept can be extended for policies in similar fashion. We refer to methods built on Markov games with the Nash equilibrium concept as symmetric methods and to methods that utilize the Stackelberg equilibrium concept as asymmetric methods.

For brevity, learning algorithms are presented next only in the case of two agents and in the asymmetric model the agent one is acting as the leader and the agent two as the follower.

B. Symmetric Learning in Markov Games

As in the case of single agent reinforcement learning, Q-values defined in Eq. (3) can be learned from observations on-line using some iterative algorithm. For example, in the two-agent case, if we use Q-learning, the update rule for the agent 1 is [2]:

$$\begin{aligned} Q^1_{t+1}(s_t, a^1_t, a^2_t) = (1-\alpha_t) Q^1_t(s_t, a^1_t, a^2_t) \\ + \alpha_t [r^1_{t+1} + \gamma \text{Nash}\{Q^1_t(s_{t+1})\}], \end{aligned} \tag{4}$$

where $\text{Nash}\{Q^1_t(s_{t+1})\}$ is a Nash equilibrium outcome of the bimatrix game defined by the payoff function $Q^1_t(s_{t+1})$. The corresponding update rule for the agent 2 is symmetric.

Note that it is guaranteed that every finite matrix game possesses at least one Nash equilibrium in mixed strategies. However, there exists not necessarily Nash equilibrium point in pure strategies and therefore $\text{Nash}\{Q^1_t(s_{t+1})\}$ in Eq. (4) returns the value of a mixed strategy equilibrium.

C. Asymmetric Learning in Markov Games

In the asymmetric case with Q-learning, we get the following update rules for the agents:

$$\begin{aligned} Q^1_{t+1}(s_t, a^1_t, a^2_t) = (1-\alpha_t) Q^1_t(s_t, a^1_t, a^2_t) + \alpha_t [r^1_{t+1} \\ + \gamma \max_{b \in A^1} Q^1_t(s_{t+1}, b, Tb)] \end{aligned} \tag{5}$$

$$\begin{aligned} Q^2_{t+1}(s_t, a^1_t, a^2_t) = (1-\alpha_t) Q^2_t(s_t, a^1_t, a^2_t) + \alpha_t [r^2_{t+1} \\ + \gamma \max_{b \in A^2} Q^2_t(s_{t+1}, g(s_{t+1}, a^c_{t+1}), b)], \end{aligned} \tag{6}$$

where $g(s_t, a^c_t)$ is the leader's enforcement and T is a mapping $T : A^1 \rightarrow A^2$ that conducts the follower's best response to the leader's enforcement.

Above presented algorithms do not define how one selects the current state-action tuple (s_t, a^1_t, a^2_t). For example, it is possible to select states and actions pure randomly. However, it is often more efficient to explore state-action space by calculating an equilibrium solution of the matrix game associated with the current state and then deviate from this solution with some small probability, e.g. by using softmax action selection method.

IV. Hybrid Model

In the previously discussed learning methods there is a matrix game associated with each state. In many applications, it is not necessary to learn the exact model of the opponent in each state. By including problem specific information, the overall structure of the Markov game can be significantly simplified and agents can still perform almost optimally. This also reduces the computational requirements needed to solve the problem. We start the section by introducing the notation used with the partitioned state space and then proceed to the actual learning methods used with this hybrid model. For brevity, all mathematics in this section is presented in the case of two agents. However, the extensions to arbitrary number of agents are straightforward.

A. Partitioning State Space

The main idea of our method is to divide the state space into two subspaces: one that contains all states with utility values for single agent only (simple states, S_S) and the other containing states with matrix games (complex states, S_C). In the simple states it is possible to reduce the matrix games to vectors containing utility estimates for the agent and hence the learning process reduces to normal single-agent reinforcement learning. Note that this reduced learning model is consistent with the multiagent reinforcement learning model based on the Markov game since in the single-agent case, the Nash operator simply reduces to the maximum of the expected payoff values.

The overall learning process divides into four different cases depending on the types of the current and the next state of the system. This division is illustrated in Fig. 1. Next we will present the exact learning rules for the hybrid model in all cases.

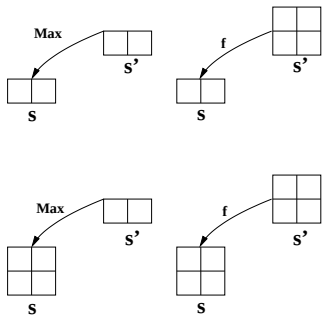

Fig. 1. Four different cases in the hybrid multiagent reinforcement learning. The f operator is used here to evaluate values of matrix games.

B. Learning Rules in the Hybrid Model

Actual learning rules used with the hybrid model depend on the current state and the state transition of the system. Thus, we get four different learning rules that are symmetric for both agents $i = 1, 2$.

1) Case 1: $s_t \in S_S$ and $s_{t+1} \in S_S$:

$$Q^i_{t+1}(s_t, a^i_t) = (1 - \alpha_t)Q^i_t(s_t, a^i_t) + \alpha_t[r^i_{t+1} + \gamma \max_{b \in A^i} Q^i_t(s_{t+1}, b)]$$

2) Case 2: $s_t \in S_C$ and $s_{t+1} \in S_S$:

$$Q^i_{t+1}(s_t, a^1_t, a^2_t) = (1 - \alpha_t)Q^i_t(s_t, a^1_t, a^2_t) + \alpha_t[r^i_{t+1} + \gamma \max_{b \in A^i} Q^i_t(s_{t+1}, b)]$$

3) Case 3: $s_t \in S_S$ and $s_{t+1} \in S_C$:

$$Q^i_{t+1}(s_t, a^i_t) = (1 - \alpha_t)Q^i_t(s_t, a^i_t) + \alpha_t[r^i_{t+1} + \gamma \mathrm{f}\{Q^i_t(s_{t+1})\}]$$

4) Case 4: $s_t \in S_C$ and $s_{t+1} \in S_C$:

$$Q^i_{t+1}(s_t, a^1_t, a^2_t) = (1 - \alpha_t)Q^i_t(s_t, a^1_t, a^2_t) + \alpha_t[r^i_{t+1} + \gamma \mathrm{f}\{Q^i_t(s_{t+1})\}]$$

In the above equations, the operator $f\{Q^i(s_t)\}$ evaluates the games associated with the states s_t. Possible choices for the operator f are for example the Nash equilibrium (leading to symmetric learning) operator or the Stackelberg equilibrium operator (leading to asymmetric learning). Note that albeit the learning rules are similar for both agents, the implementation of the operator f does not have to be symmetric and hence the actual learning rules could be different.

C. Numeric Approximations

Solving problems with large state-action spaces requires the use of function approximators such as neural networks. The traditional approach is to use function approximators to estimate value functions of the learning agents directly. A natural extension to the hybrid model is to estimate the Q-function with two function approximators: one for the simple states and one for the complex states, see Fig. 2.

The actual learning of the approximators can be done e.g. using a VAPS-like learning rule for multiagent domains, see [4]. Note that the use of two distinct approximators for multiagent reinforcement learning is one instance of the *Occam's razor* principle in the sense that the agent's Q-function is modeled with as simple model as possible with regard to the problem specific a priori information.

In addition, the hybrid model reduces computational requirements of evaluating the value of a state. For example, if we are using the Nash equilibrium concept, general computational complexity of the problem is still unknown. On the other hand, the complexity of finding the maximum element from a fixed set of elements is linear.

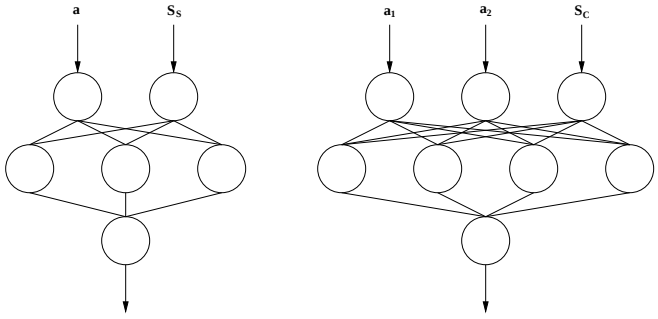

Fig. 2. An example of two approximators used to approximate utility functions in the hybrid model. Left: approximator for the simple states, right: approximator for the complex states.

D. Evaluating Dependencies between Agents

The division of the state space can be accomplished by using problem specific a priori information. However, it is also possible to try to evaluate dependencies between agents in a particular state, i.e. to calculate how much the agent's payoff values depend on the opponent's action choices. Let a_{ij} be the Q-value of the agent 1 (acting as a row player in the corresponding matrix games) when he selects the action i and the opponent selects the action j. Then it is sufficient to measure the total difference of the columns in the payoff matrix. Formally this can be expressed as:

$$d = \beta \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{k=1}^{M} |a_{ki} - a_{kj}|^p \right)^{\frac{1}{p}}, \qquad (7)$$

where M and N are the numbers of action choices for the agent 1 and the agent 2, respectively. β is a scaling coefficient, perhaps a function of M and N, and $p \geq 1$ is an arbitrary parameter. Clearly, d is zero if and only if all columns in the corresponding payoff matrix are equal. In the next section, we demonstrate the metric with a simple example problem.

The metric proposed in Eq. (7) can not be used for packed Q-tables or function approximators. Hence, if the metric is used online for deciding which states should be packed (transferred to S_s), the actual packing can be done only once. Moreover, it is hard to evaluate the cost of the packing online. The only reliable way to do the evaluation is to compare the performance of the original and the packed system side-by-side.

Note that the above proposed metric depends on the absolute difference of utility values. In many problems, particularly in control problems, it would be more desirable to take into account only the ordering of the different actions.

V. Example Application

In this section, we solve a simple example problem by using hybrid multiagent reinforcement learning methods (symmetric with the Nash operator and asymmetric) discussed earlier in this paper. This problem is a variation from the commonly used grid world problem, which is used for testing single-agent and multiagent reinforcement learning algorithms in many works,

e.g. in [9], [8], [1] and [7]. Our test case is the same as in [2], where the problem was solved using a tabular version of the symmetric multiagent reinforcement learning algorithm.

A. Grid World Example

In a grid world problem, we have a grid world containing 16 cells and two competing agents (Fig. 3). The agents start from the lower corners 1 and 2, respectively, and on each round they can move to adjacent cells (4-neighborhood). In addition, there are two distinct goal positions, one for each agent. The agents get large positive payoffs when they reach the goal positions. In the symmetric learning model both agents get small negative payoffs when they try to move to the same position and the agents are returned back to their original positions. In the asymmetric learning model, only the agent 1 (leader) gets the negative payoff and thus tries to avoid the collision by its enforcements. Hence, the ultimate goal of the agents is to reach the goal cell using as few moves as possible.

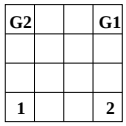

Fig. 3. The game board used in the example problem. Agents are initially located in the cells marked with numbers **1** and **2**. Goal cells are marked with **G1**- and **G2**-symbols.

B. Associated Markov Game

We characterize the problem with the following Markov game:

- A state in this problem is a pair $s = (p_1, p_2)$, i.e. the positions of agents. Hence, the state space of this example consists of 16*16=256 states.
- Both agents get a positive payoff of 0.9 when they find the right goal cell.
- The action set for both agents is $A^i = \{\text{Left}, \text{Right}, \text{Up}, \text{Down}\}, i = 1, 2$. The agents are restricted to stay on the game board.
- The discount factor γ is 0.99.
- Both agents select their actions by using the softmax action selection method.
- In the asymmetric model the agent 1 is acting as the leader.
- In the symmetric model, if the agents collide both agents get a negative reward of -0.1 and in the asymmetric model only the leader gets this negative reward.
- In the symmetric model, the Nash solution concept is used. The solution is calculated by using the *Lemke-Howson* algorithm that is fully deterministic. Both agents always select the first equilibrium.

Critical points in the grid world problem are states where the agents have an option to take moves that lead to the same cell, i.e. states where the distance between agents is exactly 2 cells. Therefore, it is possible to reduce the problem by using single-agent reinforcement learning in all states where the distance between agents is more than 2 cells. Note that the number of these critical points increases much slower than the total number of states when the size of the game board is enlarged.

The learning rate parameter α and the temperature parameter in the softmax action selection are modified linearly with time. Note that the learning rate decaying scheme used in the simulation runs does not obey the conditions defined in stochastic approximation theory. However, in real problems, the learning rate decaying scheme satisfying these theoretical conditions often induces very slow convergence and thus this kind of scheme is seldom used in real-world applications and in empirical research.

When an agent reaches the goal position, the agents are moved back to their initial positions (randomized cells on the game board excluding goal cells) and the learning process is restarted with a new episode. The maximum length of the episode is restricted to 10 moves. We repeated test runs (learning of the optimal payoff function) 50 times in every test case. Some equilibrium paths generated by the hybrid model (both asymmetric and symmetric reinforcement learning) are illustrated in Fig. 4.

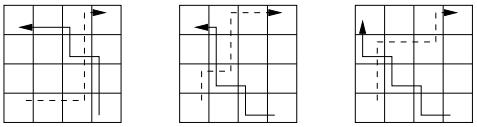

Fig. 4. Three optimal paths generated by both symmetric and asymmetric multiagent reinforcement learning models.

In Table I, averaged learning times are presented. All simulations were ran by using one 1 GHz Alpha processor. In both cases, with symmetric and asymmetric models, there were a significant improvement on the learning time when the hybrid model was used. Additionally, the asymmetric model was much faster than the symmetric model.

TABLE I

AVERAGED LEARNING TIMES (CPU TIME IN SECONDS) FROM 50 TEST RUNS. ALL TEST RUNS WERE RAN ON ONE 1 GHZ ALPHA PROCESSOR.

	normal	hybrid
symmetric	237	180
asymmetric	78	48

In Fig. 6, we test two different states in our example problem by using the metric proposed in Eq. (7) with $p = 2$. The coefficient β was the inverse of the total number of the summations in Eq. (7). The compared states are shown in Fig. 5. The model used in the comparison was the asymmetric model and the metric is calculated only for the agent 1 (leader). The state corresponding to positions marked with 1 gets lower values than the state marked with 2 since both agents are capable to reach the goal position and to end the game in the state 2. This induces strong dependence of the opponent's action choices for both agents.

The convergence curves of the hybrid model in the asymmetric case are shown in Figs. 7 and 8. Convergence properties

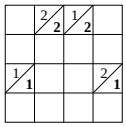

Fig. 5. Example positions used for dependency measurements. States discussed in the text are marked with boldfaced numbers. Agents are marked with normal font.

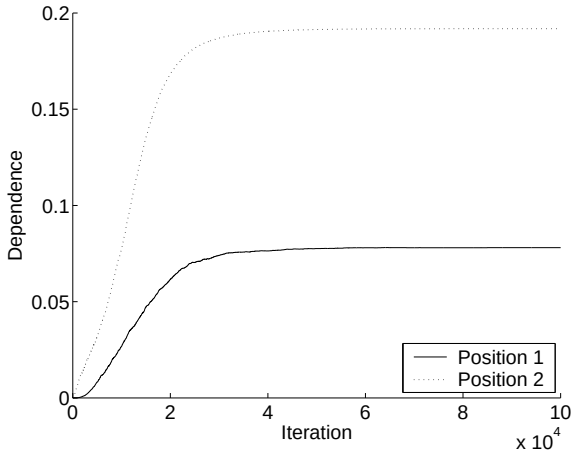

Fig. 6. Leader's dependence of the opponent's actions.

in the symmetric case are analogous. Both agents converged with an equal pace. Moreover, the convergence of the simple states is slower than the convergence of the complex states since learning process converges quickly to the paths containing mostly complex states. With larger boards, the relative portion of the simple states is larger and these states are visited more frequently.

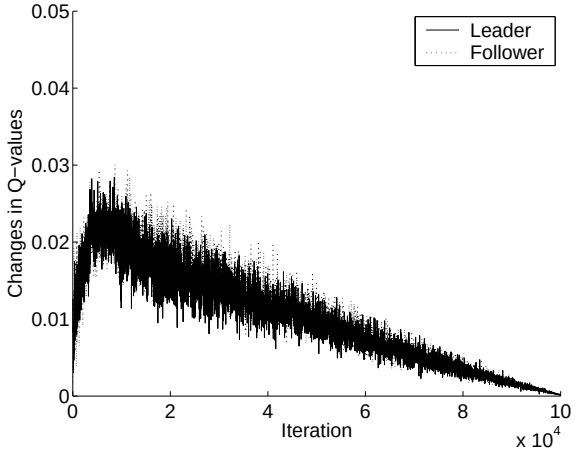

Fig. 7. Convergence of the asymmetric model. Simple states.

VI. Conclusions and Future Research

A novel method for multiagent reinforcement learning was presented in this paper. The proposed method divides the state space into two subspaces: one containing states where the opponent is not modeled and the another where the opponent is modeled. Additionally, the proposed method was tested with a simple example problem. As most of the computational burden of the multiagent reinforcement learning methods come from equilibrium point calculation, the hybrid model reduces the time needed to learn the payoff function significantly.

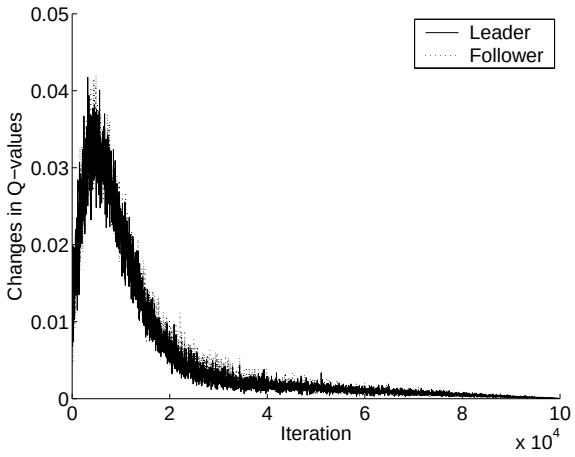

Fig. 8. Convergence of the asymmetric model. Complex states.

Numerous papers have been written on model-free learning in Markov games. However, there are no many applications built on multiagent reinforcement learning methods based on Markov games. Thus, in future research, we will test these learning methods with applications, e.g. from automated pricing domain.

References

[1] Amy Greenwald and Keith Hall. Correlated-Q learning. In *Proceedings of the AAAI-2002 Spring Symposium Workshop on Collaborative Learning Agents*, Stanford, CA, 2002. AAAI Press.

[2] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, Madison, WI, 1998. Morgan Kaufmann Publishers.

[3] Ville J. Könönen. Asymmetric multiagent reinforcement learning. In *Proceedings of the 2003 WIC International Conference on Intelligent Agent Technology (IAT-2003)*, Halifax, Canada, 2003. IEEE Press.

[4] Ville J. Könönen. Gradient based method for symmetric and asymmetric multiagent reinforcement learning. In *Proceedings of the Fourth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2003)*, Hong Kong, China, 2003. Springer-Verlag.

[5] Ville J. Könönen. Policy gradient method for multiagent reinforcement learning. In *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, 2003.

[6] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ, 1994. Morgan Kaufmann Publishers.

[7] Michael L. Littman. Friend-or-Foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, Williamstown, MA, 2001. Morgan Kaufmann Publishers.

[8] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.

[9] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[10] Xiaofeng Wang and Tuomas Sandholm. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *Advances in Neural Information Processing Systems*, volume 15, Cambridge, MA, 2002. MIT Press.