Multiagent Reinforcement Learning Algorithm Research Based on Non Markov Environment

Xiangping Meng[1], Robert Babuška[2], Yu Chen[3], and Lucian Busoniu[2]

[1] Department of Electrical Engineering, Changchun Institute of Technology, P. R. China
xp_meng@ccit.edu.cn, mxp_1961@hotmail.com
[2] Delft Center for Systems and Control, Delft University of Technology, The Netherlands
{r.babushka, i.l.busoniu}@dcsc.tudelft.nl
[3] Department of Computer Engineering, Northeast China Institute of Electric Power, P.R.China
cycnen@sohu.com

Abstract

In this paper several multiagent reinforcement learning algorithms are investigated, compared and analyzed. An effective reinforcement learning algorithm based on non Markov environment is proposed. This algorithm uses linear programming to find the best-response policy, and avoids solving multiple Nash equilibria problem. The algorithm involves simple procedures and easy computations, and can guarantee good learning convergence in some situations. Experiment results show that this algorithm is effective.
Keyword: multiagent; reinforcement learning; markov environment; nash equilibria

I. Introduction

Especially, reinforcement learning techniques (Sutton and Barto, 1998) have attracted many researchers in investigating learning in multiagent systems. The progress made on reinforcement learning has opened the way for designing autonomous agents capable of acting in unknown environments by exploring different possible actions and their consequences. In single-agent systems, Q-learning (Watkins and Dayan, 1992) is an available reinforcement learning technique which possesses a firm foundation in the theory of Markov decision processes. However, to apply single-agent Q-learning to a multiagent system in a straightforward fashion is very difficult.

In researching multiagent Q-learning, most researches adopt the framework of general-sum stochastic games. Hu and Wellman define optimal Q-values as Nash Q-values [Hu and Wellman, 1998]. This algorithm needs Nash equilibria to be globally optimal or saddle points, and it might fail to converge if multiple equilibria exist. Greenwald and Hall proposed a correlated equilibrium (CE) learning algorithm [Greenwald and Hall, 2003]. This algorithm is more general than a NE, because it resolves the equilibrium selection problem by introducing four variants of CE-Q, based on four equilibrium selection functions, i.e. utilitarian, egalitarian, republican, and libertarian CE-Q learning. A CE can be computed easily via linear programming. Bowling and Veloso considered the rational

and convergent learning problem, and proposed a "win or learn fast"(WoLF) policy hill-climbing (PHC) learning algorithm [Bowling and Veloso, 2001]. The algorithm has good convergence properties; however its definition of Q-values is single-agent style not joint action style. So far multiagent reinforcement learning is less mature in many areas [Nikos Vlassis, 2003].

In this paper, we compare and analyse several multiagent learning algorithms, and attempt to combine Nash-Q, CE-Q and PHC algorithms' thoughts, and propose an improved reinforcement learning algorithm. The algorithm makes use of the linear programming ideas of CE-Q learning and the expression types of NE-Q learning in updating Q-values; and of the policy hill-climbing technique of PHC in updating π – values. The experiment results show this algorithm is effective in some situations.

II. Reinforcement Learning Algorithms

A. *Definitions of stochastic games*

A stochastic game is a tuple $(n, S, A^1, \cdots A^n, r^1, \cdots r^n, p)$, where *n* is the number of agents, S is the set of states, A^i is the set of actions available to agent i, and $r^i : S \times A^1 \times \cdots \times A^n \to R$ is the reward function for agent i, $p : S \times A^1 \times \cdots \times A^n \to \Delta(S)$ is the state transition probability map, where $\Delta(S)$ is the set of probability distributions over the state space S.

The goal of Stochastic games (SGs) is to maximize the discounted future reward of each agent. Each state in a stochastic game can be viewed as a matrix game with the reward $r^i(s, \vec{a})$ to player *i* determined by joint action $\vec{a}$ in state s, where $\vec{a} = (a^1, a^2, \cdots, a^n)$ is the joint actions of all agents. After playing the matrix game and receiving the payoffs, the players are transitioned to another state (or matrix game) determined by their joint actions. Therefore SGs contain both MDPs $(n = 1)$ and matrix games $(|s| = 1)$ as subsets of the framework [Bowling and Veloso, 2002].

B. *Single agent* **Q** *–learning*

In single-agent *Q*-learning [Watkins and Dayan, 1992], the goal of the agent is to learn the optimal *Q*-values, defined as

$$Q_*(s,a) = r(s,a) + \beta \cdot \sum_{s'} p(s' \mid s,a) \cdot v(s', \pi_*) \tag{1}$$

Where $Q_*(s,a)$ is the sum of the immediate reward obtained at state s for taking action a and the total discounted expected future rewards obtained by following the optimal policy thereafter. β is a discount factor, with $0 \le \beta < 1$. $v(s', \pi_*)$ is a value function that maximizes $Q_*(s,a)$ over all actions a. π_* is the optimal policy that maximize $Q_*(s,a)$ at each state s. Then

$$v(s', \pi_*) = \max_a Q_*(s,a) \tag{2}$$

The agent starts with arbitrary *Q*-values, and updates them as follows:

$$Q_{t+1}(s_t, a) = (1 - \alpha_t) \cdot Q_t(s_t, a) + \alpha_t \cdot [r_t + \beta \cdot \max_b Q_t(s_{t+1,} b)] \tag{3}$$

Under standard RL assumptions, the sequence Q_t converges to Q_* with probability 1, and the optimal policy is simply taking the action to maximize $Q_t(s,a)$ at any state s.

C. *Multiagent Q – learning*

In multiagent Q-learning, the Q-function of agent i is defined over states and joint- action vectors $\vec{a} = (a^1, a^2, \cdots, a^n)$, rather than state-action pairs. Then, the updating of Q value proceed as following:

$$Q^i_{t+1}(s, \vec{a}) = (1-\alpha) Q^i_t(s, \vec{a}) + \alpha \left[r^i_t + \beta \cdot V^i(s_{t+1}) \right] \tag{4}$$

Where $V^i(s_{t+1})$ is state value functions, and

$$V^i(s_{t+1}) = \max_{a^i \in A^i} f^i(Q^i_t(s_{t+1}, \vec{a})) \tag{5}$$

In this generic formulation, the keys elements are the learning policy, i.e. the selection method of the action $\vec{a}$, and the determining of the value function $V^i(s_{t+1})$. The different action selection / value function computation methods generate different multiagent learning algorithms.

D. *Several multiagent Q-learning algorithms*

We have pointed out in section *C* that different computation methods will generate different multiagent Q-learning algorithms.

If using a simple extension from the single-agent MDP to the multiagent SGs environment, the following formulation can resolve general multiagent Q-learning problems.

$$Q^i_{t+1}(s_t, a^i) = (1-\alpha) Q^i_t(s_t, a^i) + \alpha [r^i_t(s_t, a^i) + \beta V^i(s_{t+1})] \tag{6}$$

$$V^i(s_{t+1}) = \max_{a^i \in A^i} Q^i_t(s_{t+1}, a^i) \tag{7}$$

This method has very good convergence and occupies less memory space than the Q- values defined by joint actions. However, it can only be used in the limited context where every agent just considers its own maximum benefit but does not consider the other agents at all.

Another definition of multiagent Q-values is a function of all agents' joint actions:

$$Q^i_{t+1}(s_t, \vec{a}) = (1-\alpha) Q^i_t(s_t, \vec{a}) + \alpha [r^i_t(s_t, \vec{a}) + \beta V^i(s_{t+1})] \tag{8}$$

The key of the method is how to calculate $V^i(s_{t+1})$ and update the Q-values.

a. *Zero-sum SGs framework*

For strictly competitive game, or the zero-sum SGs framework, Littman suggested the minimax-Q learning algorithm, in which the state value is updated with the minimax of the Q values [Littman, 1994]. Then

$$V^1(s_{t+1}) = \max_{P^1 \in \prod(A^1)} \min_{a^2 \in A^2} \sum_{a^1 \in A^1} P^1(a^1) Q^1_t(s_{t+1}, a^1, a^2) = -V^2(s_{t+1}) \tag{9}$$

where P^1 is the mixed strategy of agent 1 and $P^1(a^1)$ is the probability of agent 1 playing action a^1.

b. *Full cooperative SGs framework*

Littman's Friend-Q learning algorithm can be used in the strictly cooperative, or team games, SGs framework. In this method all the players' reward functions are equivalent – with uniquely valued equilibria [Littman, 2001].

$$V^i(s_{t+1}) = \max_{\vec{a} \in A} Q^i_t(s_{t+1}, \vec{a}) \tag{10}$$

c. *General-sum SGs framework*

In general-sum SGs framework, there are several different Q-learning algorithms, such as Nash Q-learning, CE Q-learning, and WoLF's policy hill-climbing etc.

1. *Figures and Tables*

In Nash Q-learning, the agent attempts to learn its equilibrium Q-values, starting from an arbitrary guess. To this end, the Nash Q-learning agent maintains a model of other agents' Q-

values and uses this information to update its own Q-values. The updating rule is based on the expectation that agents would take their equilibrium actions in each state. Based on the learned Q-values, agents can then derive the Nash equilibrium and choose their actions accordingly [Hu and Wellman, 1998; Hu and Wellman, 2003].

$$V^i(s_{t+1}) \in NASH_i(Q^1(s_{t+1}),\cdots,Q^n(s_{t+1})) \tag{11}$$

A Nash equilibrium is a collection of strategies for each of the players such that each player's strategy is a best-response to the other players' strategies. So, no player can get a higher payoff by changing strategies given that the other players don't change strategies either [Bowling and Veloso, 2001]. Despite its limitations, such as non-uniqueness, the Nash equilibrium serves as the fundamental solution concept for general-sum games.

2. CE Q-learning

Greenwald and Hall proposed a multiagent Q-learning algorithm, i.e. Correlated-Q (CE-Q) learning, based on the correlated equilibrium (CE) solution concept. CE-Q generalizes both Nash-Q and Friend-and-Foe-Q in general-sum games, and the set of correlated equilibria contains the set of Nash equilibria; in constant-sum games, the set of correlated equilibria contains the set of minimax equilibria [Greenwald and Hall, 2003].

$$V^i(s_{t+1}) \in CE_i(Q^1(s_{t+1}),\cdots,Q^n(s_{t+1})) \tag{12}$$

Where $CE_i(Q^1(s_{t+1}),\cdots,Q^n(s_{t+1}))$ denotes the *i*th player's reward according to some correlated equilibrium determined by the rewards $Q^1(s_{t+1}),\cdots,Q^n(s_{t+1})$ in the general-sum game. CE Q-learning generalizes Nash Q-learning, since a Nash equilibrium is a correlated equilibrium that can be factored into independent distributions over each individual agent's action space.

3. WoLF policy hill-climbing algorithm

The win-or-learn-fast (WoLF) policy hill-climbing algorithm (PHC), also called the variable learning rate algorithm, was proposed by Bowling and Veloso [Bowling and Veloso, 2001; Bowling and Veloso, 2003]. The basic idea is to vary the learning rate used by the algorithm in such a way as to guarantee convergence, without sacrificing rationality. The principle has a simple intuition, i.e., learn quickly while losing and slowly while winning. The method of determining when the agent is winning is to compare the current policy's expected payoff with that of the average policy over time. This principle aids in convergence by giving more time for the other players to adapt to changes in the player's strategy that at first appear beneficial, while allowing the player to adapt more quickly to other players' strategy changes when they are harmful.This algorithm is the same as the simple extending from single-agent to multiagent Q-learning algorithm in update Q-values – formulations (6) and (7). But the update of the π-values is as follows:

$$\pi^i(s,a) \leftarrow \pi^i(s,a) + \begin{cases} \delta & \text{if } a = \arg\max_{a'} Q_t^i(s,a') \\ \dfrac{-\delta}{|A^i|-1} & \text{otherwise} \end{cases} \tag{13}$$

$$\delta = \begin{cases} \delta_w & \text{if } \sum_a \pi^i(s,a)Q^i(s,a) > \sum_a \bar{\pi}^i(s,a)Q^i(s,a) \\ \delta_l & \text{otherwise} \end{cases} \tag{14}$$

Where $\bar{\pi}^i(s,a)$ is estimate of average policy.

This algorithm requires two learning rate parameters, δ_l and δ_w, with $\delta_l > \delta_w$. This can be determined by comparing the expected Q-value estimates of following the current policy π in the current state with that of following the average policy $\bar{\pi}$. If the expectation of the current policy is smaller (i.e. the agent is "losing") then the larger learning rate, δ_l is used

This algorithm's convergence properties are good, but the definition of Q-values is not based on the joint action, so it is suitable for self-interested multiagent situations.

III. An improved multiagent reinforcement learning algorithm

To overcome the disadvantage of the above algorithms, we combine the thoughts of WoLF, Nash Q-learning and CE Q-learning algorithms, and propose an improved multiagent reinforcement learning algorithm.

The algorithm makes use of the linear programming ideas of CE-Q learning and the expression types of NE-Q learning in updating Q-values; and makes use of the policy hill-climbing technique of WoLF in updating $\pi-$ values.The definition of Q values use the type of joint actions, and the updating of Q values is following

$$Q_{t+1}^{i}(s_t, a^1, \cdots, a^n) = (1-\alpha_t)Q_t^i(s_t, \vec{a}) + \alpha_t[r_t^i(s_t, \vec{a}) + \beta V^i(s_{t+1})] \tag{15}$$

Where $V^i(s_{t+1})$ is the state value function, and is determined by the following:

$$V^i(s_{t+1}) = \sum_{\vec{a}} \prod_{j=1}^{n} \pi_*^j(s_{t+1}, a^j) \cdot Q_t^i(s_{t+1}, \vec{a}) \tag{16}$$

$\pi_*^i(s_{t+1}, a^i)(i = 1, 2, \cdots n)$ are obtained by maximizing the following:

$$\pi_*^i(s_{t+1}, a^i) = \arg\max_{\pi^i} \sum_{\vec{a}} \pi^i(s_{t+1}, a^i) \cdot \prod_{j=1, j\neq i}^{n} \pi_t^j(s_{t+1}, a^j) \cdot Q_t^i(s_{t+1}, \vec{a}) \tag{17}$$

π values update as follows:

$$\pi_{t+1}^i(s, a) = \pi_t^i(s, a) + \Delta\delta_{sa} \tag{18}$$

$$\Delta\delta_{sa} = \begin{cases} \min\left(\pi_t^i(s, a),\ \delta/\left(\left|A^i\right|-1\right)\right) & \text{if } a = \arg\max_{a^{i'}} Q_t^i(s_t, a^1, \cdots, a^{i'}, \cdots, a^n) \\ -\min\left(\pi_t^i(s, a),\ \delta/\left(\left|A^i\right|-1\right)\right) & \text{otherwise} \end{cases} \tag{19}$$

$$\delta = \begin{cases} \delta_w & \text{if } \sum_{a^i} \pi^i(s, a^i) Q^i(s, a^1, \cdots, a^i, \cdots, a^n) > \sum_{a^i} \overline{\pi}^i(s, a^i) Q^i(s, a^1, \cdots, a^i, \cdots, a^n) \\ \delta_l & \text{otherwise} \end{cases} \tag{20}$$

This improved algorithm is following as Table 1. It extends the WoLF-PHC algorithm to multiagent joint action Q- values. In the updating of Q-values it only uses linear programming to calculate π_*, so the computation is simple. The convergence properties of this method are almost the same as that of the Nash Q-learning and WoLF-PHC algorithms, furthermore, this algorithm avoids resolving complex multiple Nash equilibria problem.

Table 1. Improved multiagent reinforcement learning algorithm

1. Initialize:
   (a) Select initial learning rate α, δ_w, δ_l and discount factor β and ε
   (b) Let $t = 0$
   (c) For all $s \in S$, $a^1 \in A^1$, $a^2 \in A^2, \cdots, a^n \in A^n$,
   Let $Q_0^i(s, a^1, a^2, \cdots, a^n) = 0$
   $$\pi_0^i(s, a^i) = \frac{1}{|A^i|}, \quad \bar{\pi}_0^i(s, a^i) = \frac{1}{|A^i|}, \quad C^i(s) = 0$$
   (d) Initialize s_0
   (e) Choose action $a^1, a^2, \cdots, a^n$ based on probability $\pi(s, a)$ with some exploration in state s_0
2. Loop
   (a) Execute action $a^1, a^2, \cdots, a^n$ in state s_t
   (b) Observe reward $(r_t^1, \cdots, r_t^n)$ and new state s_{t+1}
   (c) Update Q values using the following rule for $i = 1, 2, \cdots, n$
   $$Q_{t+1}^i(s, a^1, \cdots, a^n) = (1-\alpha)Q_t^i(s, \vec{a}) + \alpha[r_t^i + \beta \cdot \prod_{i=1}^n \pi_*^i(s_{t+1}, a^i) \cdot Q_t^i(s_{t+1}, \vec{a})]$$
   Where $\pi_*^i(s_{t+1}, a^i)$ is calculated by (17)
   (d) Update estimate of average policy $\bar{\pi}^i$ for $i = 1, 2, \cdots, n$
   $$C^i(s) = C^i(s) + 1$$
   $$\forall a' \in A^i, \quad \bar{\pi}^i(s, a') = \bar{\pi}^i(s, a') + \frac{\pi^i(s, a') - \bar{\pi}^i(s, a')}{C^i(s)}$$
   (e) Update $\pi^i(s, a)$ by (18) and δ selected by (20)
   (f) Choose the next action for new states s_{t+1} according to $\pi^i(s, a)$
3. Loop end

IV. Experiments and Results

A. Grid world examples

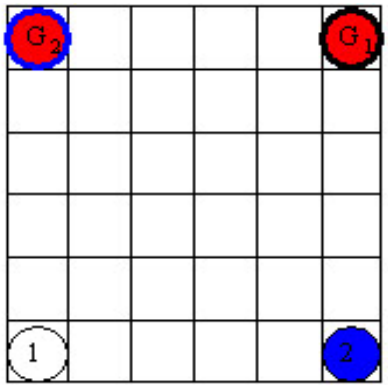

Fig. 1. Grid World 6×6 without obstacles

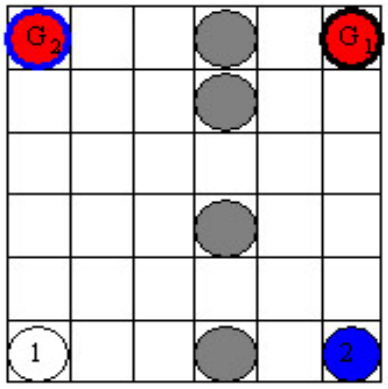

Fig. 2. Grid World 6×6 with 4 obstacles

We examined our improved multiagent reinforcement learning algorithm on a 6×6 gridworld without obstacles and with 4 obstacles as shown in Figure 1 and Figure 2, respectively. Two agents play

these games – denoted as agent 1 and agent 2. Each agent has 6×6 = 36 states in Figure 1, but in Figure 2, two agents can not enter the cells occupied by obstacles, so each agent only has 32 states. Two agents start from the two lower corners, trying to reach their goal cells G_1 and G_2 opposite to them in the top row. An agent can move only one cell a time, and in four possible compass directions: Left, Right, Down, Up. Actions are executed simultaneously. If the agents stay in the previous cell (excluding a goal cell), they receive a negative -1 payoff. The game ends when any one agent reaches its goal. Reaching the goal earns a positive 10 points reward.

In case both agents reach their goal cells at the same time, they both obtain a reward of 10 points. In other cases agents do not earn any payoff. The objective of each agent in this game is to reach its goal with a minimum number of learning steps. We Assum that the agents do not know the locations of their goals at the beginning of the learning period, and that they do not know their own and the other agent's payoff functions. Agents choose their actions simultaneously, and can observe the previous actions of both agents and the current state. They also observe the immediate rewards after both agents choose their actions.

B. Experiment results

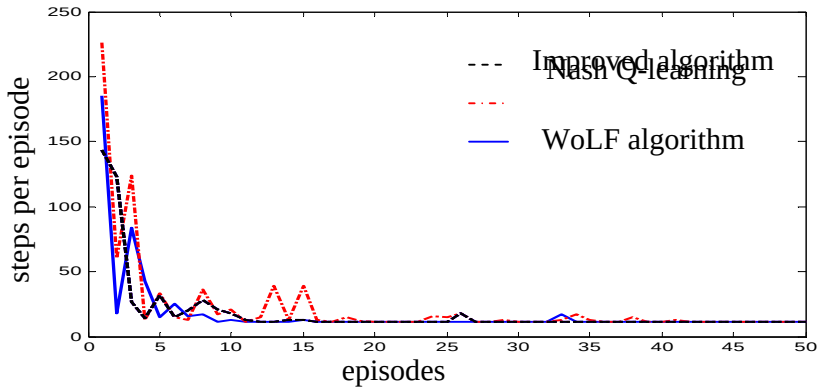

Fig. 3. The number of learning steps vs the number of learning iterations (without obstacles)

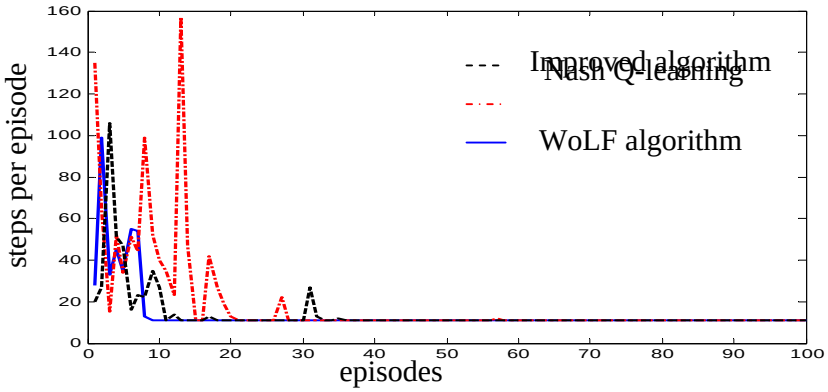

Fig. 4. The number of learning steps vs the number of learning iterations (with 4 obstacles)

In this example problem experiment, we use the following initial parameters:

$\alpha = 0.3;\ \beta = 0.95;\ \delta_l = 0.004;\ \delta_w = 0.001;\ \varepsilon = 0.3;$

Where α, δ_l, δ_w and ε are all decaying gradually in iterations. The decaying formula of δ_l and δ_w is the same as PHC algorithm.

$$\alpha_k = \frac{1}{1+k/500};\ \ \varepsilon = \varepsilon / episodes;\ \ \delta_w = \frac{1}{1000+k/10};\ \ \delta_l = 4\delta_w$$

We measure learning time in episodes (trials). We train the agents over 50 and 100 episodes on the gridworlds in figures 1 and 2, respectively.
Both agents choose their actions by using ε-greedy action selection with random exploration.
In the experiment we compared our improved multiagent learning algorithm with the WoLF-PHC and the Nash Q-learning algorithms. The comparison is in terms of the number of steps taken to reach the goal as a function of the count of learning episodes. In each game both agents start up from their initial positions and the game runs until the both agents all reach their goal positions. Every such game constitutes one learning trial, also called as an episode, and the learned Q-values are passed from iteration to iteration. The experiment results are shown as Figure 3 and Figure 4, respectively.

V. Conclusion

In this paper we overview several multiagent reinforcement learning algorithms, compare the WoLF-PHC algorithm with Nash Q-learning and proposed an improved multiagent reinforcement learning algorithm. This improved algorithm only uses linear programming to calculate the optimal policies π_* in each state, so the computation is simple. The convergence properties of this method are almost the same as those of Nash Q-learning and WoLF-PHC , but this algorithm avoids resolving complex multiple Nash equilibria problem. Two 6×6 grid games,one without obstacles and one with 4 obstacles were used as example problems for validating the algorithm. The results show the algorithm is effective and converges well.
We will continue researching the convergence condition in theory and real applications in future work.

References

[1] Bowling, M., and Veloso, M., Rational and convergent learning in stochastic games. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Seattle,Washington. 2001, pp. 1021–1026.
[2] Bowling, M., and Veloso, M., Multiagent learning using a variable learning rate. Artificial Intelligence, 2002, 136, pp. 215-250.
[3] Greenwald, A., and Hall, K., Correlated Q-learning. In Proceedings of the Twentieth International Conference on Machine Learning, 2003, pp. 242–249 K.
[4] Hu, J., and Wellman, M. P., Nash Q-learning for general-sum stochastic games. Journal of Machine Learning Research, 2003, 4, pp. 1039–1069.
[5] Hu, J., and Wellman, M. P., Multi-agent reinforcement learning: Theoretical framework and an algorithm. In Proceedings of the Fifteenth International Conference on Machine Learning, 1998, pp. 242–250.
[6] Kaya, M., and Alhajj, R., Modular Fuzzy Reinforcement Learning Approach With Internal Model Capabilities for Multi agent Systems, IEEE Transactions on systems, man, and cybernetics-Part B: Cybernetics, 2004, 34(2), pp. 1210-1223
[7] Littman, M. L., Markov games as a framework for multiagent reinforcement learning. In Proceedings of the 11th International Conference on Machine Learning, New Brunswick, NJ, 1994, pp. 157–163,.
[8] Littman, M. L., Friend-or-foe: Q-learning in general-sum games. In Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 322–328.
[9] Murakoshi, K., and Mizuno, J., A parameter control method in reinforcement learning to rapidly follow unexpected environmental changes, BioSystems, 2004.

[10] Shoham, Y., Powers, R., and Grenager, T., Multi-agent reinforcement learning: a critical survey. Technical report, Computer Science Department, Stanford University, Stanford, 2003.
[11] Singh, S., Jaakkola, T., Littman, M. L., and Szepesvari, C., Convergence results for single-step on-policy reinforcement-learning algorithms, Machine Learning, 2000, 39, pp. 287–308.
[12] Sutton, R.S., Sutton, R.S., Reinforcement Learning: An Introduction. The MIT Press / Bradford Books, 1998.
[13] Sakaguchi, Y., and Takano, M., Reliability of internal prediction/estimation and its application. I. Adaptive action selection reflecting reliability of value function, Neural Networks, 2004, 17, pp. 935–952.
[14] Vlassis, N., A Concise Introduction to Multiagent Systems and Distributed AI, Technical report, University of Amsterdam, 2003.
[15] Watkins, C. J. C. H., Dayan, P., Q-learning. Machine Learning, 1992, 8(3/4), pp. 279-292.

Xiangping Meng. received her B.S. and Ph.D. from Northeastern University, China, in 1983 and 2000 respectively, and a M.S. from Northeastern Institute of Electric Power Engineering, China, in 1986. She has been working at a Postdoctor at Jilin University from 2000 to 2002. She is working at Changchun Institute of Technology since 1986. Her research interests include intelligent control, data mining, intelligent computing and reinforcement learning and so on.

Robert babushka. received a M.Sc. degree in control engineering from the Czech Technical University in Prague, in 1990, and a Ph.D. degree from the Delft University of Technology, the Netherlands, in 1997. Currently, he is a Professor at the Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology. His research interests include the use of fuzzy set techniques and neural networks in nonlinear system identification and control, with applications in process industry, biotechnology and biomedical systems.

Yu Chen. master student at Northeastern Institute of Electric Power Engineering, China. Her research interests include intelligent control, intelligent computing, reinforcement learning and data mining.