Opponent Modelling with Eligibility Trace for Multi-agent Reinforcement Learning

Hao Chen, Jian Huang, and Jianxing Gong

***Abstract*—Markov games and reinforcement learning algorithms are applied successfully in multi-agent learning systems such as Minimax-Q. Because of the interdependence between agents, it's time consuming to find the optimal policy when agents learning concurrently. Some algorithms accelerate convergences through spatial or action generalization, which requires domain-dependent prior knowledge. In order to improve learning efficiency directly, the opponent modelling Q(λ) algorithm is proposed which combines fictitious play in game theory and eligibility trace in reinforcement learning. A series of empirical evaluations were conducted in the classical soccer domain. Compared with several other algorithms, it is proved that the algorithm contributed in this paper significantly enhances the learning performance of multi-agent systems.**

***Index Terms*—Opponent modelling, markov Games, multi-agent, reinforcement learning.**

I. Introduction

In classical reinforcement learning (RL), single-agent learns through trial-and-error interactions with its environment directly, without relying on exemplary supervision or complete models of the environment [1]. Markov decision process (MDP) is the underlying of single-agent RL. It is an environment that the current state contains all the historical information. A finite MDP is a tuple $\mathsf{M} = \langle S, A, P, R, \gamma \rangle$, where S is a finite set of states, A is a finite set of actions, P is the transition probability matrix, $p(s'|s,a) \doteq \Pr\{S_{t+1}=s'|S_t=s,A_t=a\}$, R is a reward function $r(s,a) \doteq \mathbb{E}[R_{t+1}|S_t=s,A_t=a]$, γ is a discount factor. The target of solving MDPs is to find the optimal policy $\pi(a|s)$ to maximize the total expected discounted future rewards $\mathbb{E}[G_t]$ from time-step t, where $\pi(a|s) \doteq \Pr\{A_t=a|S_t=s\}$ is the probability take action a at state s, $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ is the discounted accumulated rewards.

In multi-agent systems, agents are located in a non-stationary scenario, where an agent's rewards from the environment depend on other agents' actions. Multi-agent RL tasks are not MDP, and classical RL algorithms may not applicable. Littman combined matrix games and Q-learning algorithm [2] and proposed Minimax-Q learning [3] to solve zero-sum game RL tasks. Trying to enhance the convergence rate, several improved algorithms were proposed based on Minimax-Q, such as Minimax-QS [4], Minimax-Q(λ) [5], Minimax-SARSA [5], HAMMQ [6], [7], etc. The Core ideas of these algorithms are illustrated in Section II.

Minimax-Q learning is a conservative algorithm, assuming that the opponent always chooses the best action. However, the opponent doesn't always follow an optimal policy. Therefore, it's a good idea to choose actions by evaluating the opponent's policy. Fictitious play [8] in game theory is consistent with this idea and was applied to opponent modelling (OM) [9]. In this paper, we present the opponent modelling Q(λ) (OMQ(λ)) algorithm, which combines fictitious play and eligibility trace [1] together to further improve the performance of the agent.

The remaining parts of this paper are organized as follows. Section II reviews the core ideas of several representative multi-agent RL algorithms and illustrates their contribution to accelerating convergence. Section III shows how to embed eligibility trace into opponent modelling and proposes the OMQ(λ) algorithm. Section IV conducts a series of empirical evaluations in Littman's classical soccer domain [3] and presents the superiority of our algorithm. Section V draws the conclusion of our research.

II. Markov Games And Multi-agent RL Algorithms

A. Markov Games Framework

Markov game is a combination of MDP and matrix games. It's presented as a tuple $\mathsf{MG} = \langle \mathsf{n}, \mathsf{S}, \mathsf{A}_{1\ldots n}, \mathsf{P}, \mathsf{R}_{1\ldots n}, \gamma \rangle$. In zero-sum Markov games (ZSMG), it is defined as $\mathsf{ZSMG} = \langle \mathsf{S}, \mathsf{A}, \mathsf{O}, \mathsf{P}, \mathsf{R}, \gamma \rangle$, where S is a finite set of states, A is a finite set of actions for the agent, O is a finite set of actions for the opponent, P is the transition probability matrix, $p(s'|s,a,o) \doteq \Pr\{S_{t+1}=s'|S_t=s,A_t=a, O_t=o\}$, R is a reward function $r(s,a,o) \doteq \mathbb{E}[R_{t+1}|S_t=s,A_t=a,O_t=o]$, γ is a discount factor.

Similar to MDP, the goal of ZSMG is to maximize the discounted future rewards. In each iteration, the agent and the opponent choose their own action based on the current state S_t. The next state and the reward of players are determined by the joint action (a, o).

B. Multi-agent RL Algorithms

In this section, we review the core ideas of several representative multi-agent RL algorithms. Also, the defects of these algorithms are analyzed.

On the basis of ZSMG, Minimax-Q learning algorithm was presented combined with Q-learning and minimax algorithm [10]. It's similar to classical Q-learning except that the *max* operator is replaced by the *minimax*. For deterministic action policies, the updating rule of Minimax-Q is:

Manuscript received November 10, 2018; revised April 1, 2019.

The authors are with the College of Artificial Intelligence, National University of Defense Technology, Hunan, Changsha 410073 China. (e-mail: nudtchenhao15a@163.com, nudtjHuang@hotmail.com, fj_gjx@qq.com).

DOI: 10.7763/IJMO.2019.V9.699

$$Q(s,a,o)=Q(s,a,o)+\alpha\left[r(s,a,o)+\gamma V(s')-Q(s,a,o)\right] \quad (1)$$

where α is the learning rate, $V(s)$ is the value of state in ZSMGs. For alternating Markov games, $V(s)$ is presented as:

$$V(s)=\max_{a\in A}\min_{o\in O}Q(s,a,o). \quad (2)$$

The agent selects action through *ε-greedy* policy, and the optimal policy is:

$$\pi^*(s)=\arg\max_{a\in A}\min_{o\in O}Q^*(s,a,o). \quad (3)$$

Similar to Q-learning, Minimax-Q converges to the equilibrium when the state-joint action pairs are visited infinitely [11]. But it's not easy for Minimax-Q, because agent's next state depends on the opponent's action in the current state. If the opponent never performs certain actions in certain states, it may be difficult for the agent to get the optimal policy. On the other hand, in the early stage of learning, Minimax-Q has a low learning efficiency and basically randomly selects actions. Also, similar to Q-learning, Minimax-Q is not efficient in learning because only one Q value is updated per iteration.

Minimax-Q(λ) is the combination of Minimax-Q and TD(λ) algorithms. Q values of several (s,a,o) tuples are updated each iteration, depending on the current TD error combined with eligibility traces of past events. Minimax-SARSA is an on-policy multi-agent RL algorithm, which combines the minimax algorithm and SARSA. At the beginning of the learning process, Minimax-SARSA converges slower than other algorithms because it depends on the actual policy followed. However, Minimax-SARSA presents a better performance than Minimax-Q when rewards obtained by explorations are terrible.

Minimax-QS embeds QS-algorithm into Minimax-Q. It defines the similarity among state-joint action pairs by spreading function $\sigma_t(s,a,o,s_i,a_i,o_i)\in[0,1]$. In [4], $\sigma_t(s,a,o,s_i,a_i,o_i)=g_t(s,s_i)\delta(a,a_i)\delta(o,o_i)$, where g_t is the state similarity function, δ is the Kronecker delta function. For each update, the Q value of tuple (s_i,a_i,o_i) is updated simultaneously according to the similarity degree to the tuple (s,a,o). The updating rule of Minimax-QS is:

$$\begin{aligned}&Q(s_i,a_i,o_i)=Q(s_i,a_i,o_i)+\\&\alpha\sigma(s,a,o,s_i,a_i,o_i)\left[r(s,a,o)+\gamma V(s')-Q(s_i,a_i,o_i)\right]\end{aligned} \quad (4)$$

where $V(s)$ is the same as (2). The performance of Minimax-QS depends not only on the type of environment but also on how the spreading function is defined. If the similar relationship between state-joint action pairs is not properly described, the performance will be degraded, i.e., Minimax-QS needs relevant prior knowledge in the field.

HAMMQ uses a heuristic function to induces action choice, and it requires a more precise domain-dependent prior knowledge of the field than Minimax-QS. Compared to Minimax-Q, HAMMQ only changes the policy of selecting actions without changing the way of updating Q values. For alternating ZSMG, the action choice rule of HAMMQ is:

$$\pi(s)=\begin{cases}\arg\max_{a\in A}\min_{o\in O}\left[Q(s,a,o)+\xi H_t(s,a,o)^{\beta}\right] & p\geq\varepsilon\\ a_{random} & otherwise\end{cases} \quad (5)$$

where ξ, β are real values, H is the heuristic function. For a given state s and opponent action o, H is expressed as:

$$H(s,a,o)=\begin{cases}\max_{a\in A}Q(s,a,o)-Q(s,a,o)+\eta & a=\pi_H(s)\\ 0 & otherwise\end{cases} \quad (6)$$

where η is a small real value, $\pi_H(s)$ is the prescribed action by the heuristic function. However, in practical applications, it is not easy to obtain an effective heuristic. Moreover, the performance of the algorithm mainly depends on the choice of heuristics.

III. Opponent Modelling Q(Λ) Algorithm

As depicted in Section II, Minimax-Q(λ), Minimax-QS and HAMMQ enhance the learning process by using temporal generalization, spatial generalization, and action generalization respectively. Minimax-QS and HAMMQ accelerate convergence by domain-dependent prior knowledge, the performance of the algorithm mainly depends on the choice of spreading functions or heuristics.

The OMQ(λ) presented in this section is the combination of fictitious play in game theory and eligibility trace in RL. Fictitious play technique helps the agent select the current optimal action by evaluating the past history of the action selected by the opponent, i.e., it assesses the opponent's policy at each iteration. It is proven that, in zero-sum games, the empirical distribution obtained by fictitious play converges to the Nash equilibrium [12]. Let $K(s,o)$ indicate the number of times opponent takes action o at state s, and $n(s)$ denotes the number of times opponent in state s. Per iteration, the agent chooses action by the following formula:

$$\pi(s)=\arg\max_{a\in A}\sum_{o\in O}\frac{K(s,o)}{n(s)}Q(s,a,o). \quad (7)$$

Eligibility trace is a mechanism to make learning more efficient. OMQ(λ) embeds eligibility traces into opponent modelling Q algorithm, which accelerates the learning efficiency. For each iteration, after the agent and the opponent taking action a and o respectively, the Q values of the past (s_i,a_i,o_i) tuples are updated on the basis of the traces. Let $e(s_i,a_i,o_i)$ denotes the trace of the tuple (s_i,a_i,o_i), and the updating rule of $Q(s_i,a_i,o_i)$ is expressed as:

$$\begin{aligned}&Q(s_i,a_i,o_i)=Q(s_i,a_i,o_i)+\\&\alpha e(s_i,a_i,o_i)\left[r(s,a,o)+\gamma V(s')-V(s)\right]\end{aligned}. \quad (8)$$

Based on the core idea of fictitious play, $V(s)$ here is defined as follows:

$$V(s)=\max_{a\in A}\sum_{o\in O}\frac{K(s,o)}{n(s)}Q(s,a,o) \quad (9)$$

$e(s_i,a_i,o_i)$ is decayed per iteration:

$$e(s_i,a_i,o_i)=\lambda\cdot\gamma\cdot e(s_i,a_i,o_i) \quad (10)$$

where λ is the trace-decay parameter, $\lambda\in[0,1]$. λ determines the rate at which the trace falls.

For alternating Markov games, the OMQ(λ) implemented in this paper is described in Algorithm 1.

Algorithm 1. OMQ(λ)
Input: S: a finite set of states, A: a finite set of agent's actions, O: a finite set of opponent's actions, $r(s,a,o)$: reward from the environment, $V(s)$: value function, $Q(s,a,o)$: state-joint action value function, $e(s,a,o)$: trace function of tuples, $K(s,o)$: numbers of times opponent takes action o at state s, $n(s)$: numbers of times opponent in state s, T: list of (s,a,o) tuples, the eligibility traces

1. **Initialize:** $\forall s\in S, V(s)\leftarrow 0, n(s)\leftarrow 0$;
   $\forall s\in S\wedge\forall a\in A\wedge\forall o\in O, Q(s,a,o)\leftarrow 0, e(s,a,o)\leftarrow 0$;
   $\forall s\in S\wedge\forall o\in O, K(s,o)\leftarrow 0$;
2. Observe current state s
3. **loop**
4. Agent select action a by (7); opponent select action o through its policy
5. Receive the reward $r(s,a,o)$ from the environment and observe next state s'
6. Update $Q(s,a,o)=Q(s,a,o)+\alpha[r(s,a,o)+\gamma V(s')-Q(s,a,o)]$
7. **for all** $(s_i,a_i,o_i)\in T$ **do**
8. Calculate trace decay with (10)
9. Update $Q(s_i,a_i,o_i)$ with (8)
10. **if** $e(s_i,a_i,o_i)<\delta$ **then**
11. **delete** (s_i,a_i,o_i) from T
12. **end if**
13. **end for**
14. Update $V(s)$ with (9)
15. $\forall a_i\neq a$, $e(s,a_i,o)\leftarrow 0$; $e(s,a,o)\leftarrow 1$
16. **if** $(s,a,o)\notin T$ **then**
17. $T\leftarrow T\cup(s,a,o)$
18. **end if**
19. $s\leftarrow s'$
20. **end loop**

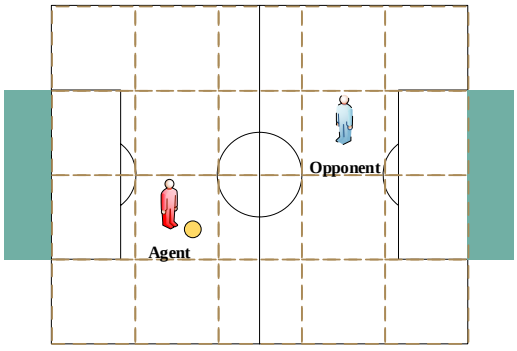

Fig. 1. The initial state of the soccer domain.

It is worth noting that the list T and a small real number δ is used to control the space complexity of OMQ(λ) algorithm.

Compared to classical Minimax-Q, OMQ(λ) is more rational. If the opponent never takes action o in state s, the Q value of tuple $(s,a,o,)$ is ignored by (7). As the number of iterations increases, the model of the opponent is more precise, and the action selected by the agent is more reasonable. Taking the advantages of eligibility trace, OMQ(λ) converges faster. Compared to Minimax-QS and HAMMQ, OMQ(λ) doesn't require prior knowledge and heuristic information of the domain, making it more convenient in practical applications.

IV. Experiments in The Classical Soccer Domain

A. The Classical Soccer Domain

The soccer domain proposed by Littman is a classical lab environment for adversarial multi-agent RL. The Agent and the opponent are placed in a 4×5 grid world, and the initial state of players are shown in Fig. 1. Each cell can only be occupied by one player, and the ball occupies the same cell as one of the players. Each iteration, plays can choose an action from up, right, down, left and stand. The possession of the ball is given randomly to the agent or the opponent at the beginning of the game.

The player's objective is to bring the ball into the opponent's goal (right for the agent and left for the opponent), and get a score of +1. When a player with the ball wants to move to a cell occupied by another player, the player loses the ball and the move fails. Any action that causes the player to go out of bounds is ignored. The end condition of the game is that any player brings the ball into the opponent's goal or the player's steps reach the limit. After the game, the players' position and the possession of the ball are reset.

B. Experimental Setup

In this section, we compared the performance differences of OMQ(λ) with Minimax-Q, Minimax-Q(λ), Minimax-QS, HAMMQ, and OM. A sequence of 100 sessions was run for each algorithm, and each session consists of 700 matches of 10 games.

The parameters used in every algorithm are the same. The player who scores the goal receives a reward +1 from the environment. The learning rate α was initialized to 1.0 and decayed at a rate of 0.9999954 for each iteration. The exploration rate ε was 0.2 and the discount factor γ was 0.9. The values of the above parameters are identical to those used in [3]-[6]. The Q values of all (s,a,o) tuples and the value functions $V(s)$ of all states are initialized to 0. The maximum steps for every player were set to 15. For Minmax-QS, the linearly decreasing spreading function was identical to [4], where $\sigma_t(s,a,o,s_i,a_i,o_i)=g_t(s,s_i)=\tau^d$, τ=0.7, and d=1. For algorithms Minimax-Q(λ) and OMQ(λ), λ=0.05. For HAMMQ, $\xi=\beta=\eta=1$, the heuristics suggested action was defined as the direction in which the player attacks (i.e., left or right), ignoring the state of another player.

Evaluation 1: The purpose of the first set of experiments was to assess the performance of the 6 algorithms versus a random opponent, i.e., the agent was a player using an RL algorithm mentioned above, and the opponent chose action randomly. We calculated the average accumulated number of goals scored by the agent from 700 matches, over 100 sessions.

Evaluation 2: The purpose of the second set of experiments was to assess the performance of the OMQ(λ) agent against the opponent using multi-agent RL algorithm. In this paper, we used Minimax-QS, which performed excellently in evaluation 1, as the opponent of OMQ(λ). For comparison, we also employed the Minimax-Q, Minimax-Q(λ), Minimax-QS and HAMMQ agents to

compete with the Minimax-QS opponent respectively. We calculated the average cumulative number of goal difference (i.e., the difference between the number of goals and the number of goals lost) got by the agent from 700 matches, over 100 sessions.

C. Results Analysis

Experimental results of evaluation 1 are shown in Fig. 2, Fig. 3 and Table I. Overall, OMQ(λ) converges fastest and has the highest cumulative goals in the experiments. Fig. 2 presents the 4 best among the 6 algorithms, which are Minimax-QS, HAMMQ, OM, and OMQ(λ). The agent using OM closed to the performance of the HAMMQ agent around 400 matches. However, the Minimax-QS agent only approached the performance of the HAMMQ agent after 600 matches. In the experiments, no algorithm exceeds the average goals (over 100 sessions) of the OMQ(λ) agent in 700 matches. The OMQ(λ) agent scored an average of around 5.7 goals at the end of matches, and the other 3 agents only got an average of around 4.7 goals at the end of experiments. As illustrated in Section III, the performance of HAMMQ is strictly related to the heuristic chosen, so in practice, it requires accurate prior knowledge in the field. It should be noted that the heuristic chosen in this paper is not optimal. A better heuristic function should consider how to get the ball from the opponent.

Fig. 3 presents the learning curves of the first 50 matches for the agent using the 6 algorithms respectively against a random opponent. The HAMMQ agent performed best at the beginning of the experiment benefiting from the heuristic function. In conjunction with Fig. 2, after 8 matches, the OMQ(λ) agent approached the performance of HAMMQ and showed an overwhelming advantage after 100 matches. Profiting from the eligibility trace, the OMQ(λ) agent scored more goals than the OM agent in each match (averaged over 100 sessions) throughout the learning process. And it's worth noting that the symmetrical environment makes it easy to define the spreading function, i.e., in practical applications, finding the proper spreading function may be time consuming.

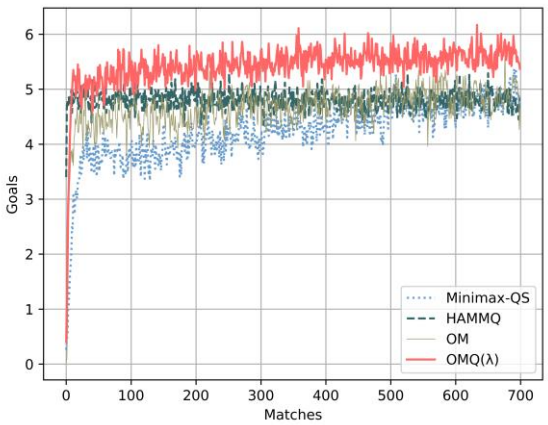

Fig. 2. Average goals of the agent using Minimax-QS, HAMMQ, OM, and OMQ(λ) respectively versus a random opponent.

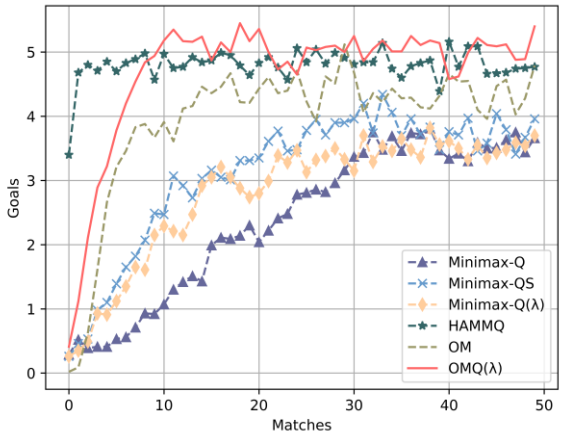

Fig. 3. Average goals (the first 50 matches) of the agent using Minimax-Q, Minimax-QS, Minimax-Q(λ), HAMMQ, OM, and OMQ(λ) respectively versus a random opponent.

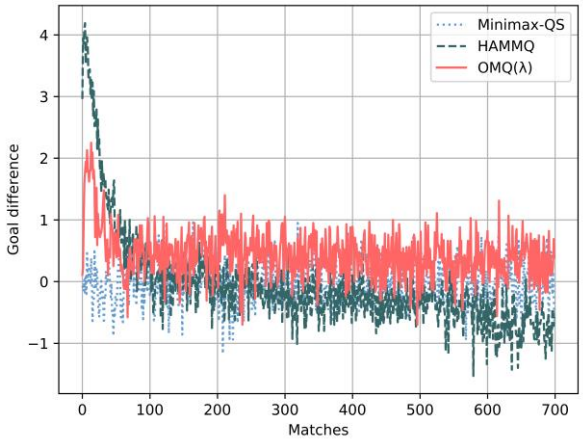

Fig. 4. Average goal difference of the agent using Minimax-QS, HAMMQ, and OMQ(λ) respectively versus a Minimax-QS opponent.

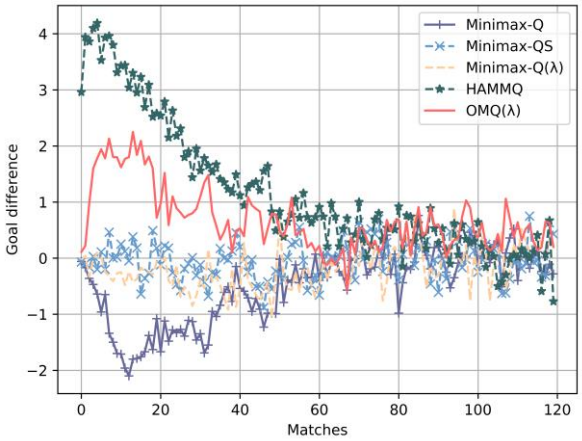

Fig. 5. Average goal difference (the first 120 matches) of the agent using Minimax-Q, Minimax-QS, Minimax-Q(λ), HAMMQ, and OMQ(λ) respectively versus a Minimax-QS opponent.

Table I. shows the cumulative number of wins of 700 matches (averaged over 100 sessions) for the agent using the 6 algorithms respectively versus a random opponent. It can be seen intuitively that OMQ(λ) won the most in the whole learning process, up to 3789 wins.

The second set of experimental results are shown in Fig. 4, Fig. 5 and Table II. Fig. 4 present average goal difference of

the agent using Minimax-QS, HAMMQ, and OMQ(λ) respectively against a Minimax-QS opponent. It is worth mentioning that these algorithms perform admirably in evaluation 1 when competing against random opponents. Average goal difference of the Minimax-QS agent fluctuated at the equilibrium from the start, because its opponent also used the same algorithm. The HAMMQ agent occupied the upper hand at the beginning, but eventually, it was crushed by the opponent. OMQ(λ) is the best performing learning algorithm in this experiment. It took about 630 matches for the Minimax-QS opponent to learn how to compete with the OMQ(λ) agent.

TABLE I: THE CUMULATIVE NUMBER OF WINS OF RL AGENTS VERSUS A RANDOM OPPONENT OF 700 MATCHES

Agent vs. Opponent	Wins
Minimax-Q vs. random	2936±87 vs. 201±13
Minimax-QS vs. random	2952±91 vs. 200±14
Minimax-Q(λ) vs. random	2951±77 vs. 197±14
HAMMQ vs. random	3355±41 vs. 202±13
OM vs. random	3442±93 vs. 202±13
OMQ(λ) vs. random	3789±66 vs. 201±14

TABLE II: THE CUMULATIVE NUMBER OF WINS OF RL AGENTS VERSUS A MINIMAX-QS OPPONENT OF 700 MATCHES

Agent vs. Opponent	Wins
Minimax-Q vs. Minimax-QS	2624±102 vs. 2722±101
Minimax-QS vs. Minimax-QS	2776±95 vs. 2695±105
Minimax-Q(λ) vs. Minimax-QS	2674±94 vs. 2684±99
HAMMQ vs. Minimax-QS	2351±80 vs. 2564±220
OMQ(λ) vs. Minimax-QS	3295±88 vs. 2983±127

Fig. 5 shows the first 120 matches average goal difference of the agent using Minimax-Q, Minimax-QS, Minimax-Q(λ), HAMMQ, and OMQ(λ) respectively versus a Minimax-QS opponent. Throughout the learning process, the curves of the agent using Minimax-QS and Minimax-(λ) are similar, but the Minimax-QS agent is slightly better at the beginning. The Minimax-Q agent performed worst at the first 50 matches and caught up the Minimax-QS agent after about 450 matches. In conjunction with Fig. 4, agents using HAMMQ performed best in the first 100 matches and got the largest goal difference (averaged 4.2 over 100 sessions), but after 100 matches, the advantage was recovered by the Minimax-QS opponent and then overtaken after 200 matches. It can be seen from the curves that the OMQ(λ) agent also got advantages at the initial stage, and maintained the longest time.

Table II. shows the cumulative number of wins of 700 matches (averaged over 100 sessions) for the agent using the 5 algorithms respectively versus a Minimax-QS opponent. Benefiting from opponent modelling and TD(λ), the OMQ(λ) agent performed best among 5 algorithms.

It must be noted that the results presented in this paper are far from convergence, and the values of parameters here are designed for comparative experiments.

V. CONCLUSION

This paper contributed the OMQ(λ) algorithm for multi-agent RL, which combines fictitious play in game theory and eligibility trace in RL. Core ideas and the algorithm of OMQ(λ) were elaborated. A series of comparative experiments were conducted in the classical soccer domain, and the results showed that our algorithm achieved the best performance in against with a random opponent and a Minimax-QS opponent without any domain-dependent prior knowledge. While ensuring learning efficiency, OMQ(λ) is more convenient and versatile, compared to algorithms that require prior knowledge.

REFERENCES

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction* MIT Press, 1998, ch. 1, p. 17.

[2] W. J. C. H. Christopher and P. Dayan. "Q-learning," *Machine Learning*, 1992, pp. 279-292.

[3] L. L. Michael, "Markov games as a framework for multi-agent reinforcement learning," *Machine Learning*, 1994, pp. 157-163.

[4] H. C. R. Carlos and A. H. R. Costa, "Experience generalization for concurrent reinforcement learners: The minimax-QS algorithm." in *Proc. International Joint Conference on Autonomous Agents and Multiagent Systems*, 2002, pp. 1239-1245.

[5] B. Bikramjit, S. Sen, and J. Peng, "Fast concurrent reinforcement learners." in *Proc. International Joint Conference on Artificial Intelligence* Morgan Kaufmann Publishers Inc, 2001, pp. 825-830.

[6] A. C. B. Reinaldo, C. H. C. Ribeiro, and A. H. R. Costa, "Heuristic selection of actions in multiagent reinforcement learning." in *Proc. International Joint Conference on Artificial Intelligence, Hyderabad, India*, 2007, pp. 690-695.

[7] A. C. B. Reinaldo *et al*., "Heuristically-accelerated multiagent reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 44, no. 2, 2014, p. 252.

[8] F. Drew and D. K. Levine, *The Theory of Learning in Games, The Theory of Learning in Games*, MIT Press, 1998, pp. 177–198.

[9] W. Uther and M. Veloso "Adversarial reinforcement learning," *Studia Mathematica*, 1997, vol. 12, pp. 143-144.

[10] J. R. Stuart and P. Norvig, "Artificial intelligence: A modern approach," *Applied Mechanics and Materials*, 2010, pp. 2829-2833.

[11] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, 1992, pp. 279-292.

[12] K. Miyasawa, "On the convergence of learning process in 2x2 non zero person game," *Research Memo*, Princeton University, 1961.

Hao Chen was born in Shandong province, China, 1993. He received the bachelor's degree in electrical engineering and automation from the College of Electrical Engineering, Hebei University of Technology (HEBUT), Tianjin, China, in 2015, and the master's degree in control science and engineering from the College of Mechatronics and Automation, National University of Defense Technology (NUDT), Hunan, China, in 2017.

He is currently a Ph.D. student at the College of Artificial Intelligence, NUDT, China. His current research interests include mission planning, reinforcement learning, and intelligent confrontation simulation.

Jian Huang was born in Zhejiang province, China, 1971. She received the B.S. degree in control science and engineering from the Department of Automatic Control, NUDT, Hunan, in 1994, where she received the master's degree and the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation, NUDT, in 1997 and 2000, respectively.

She now is a professor at the College of Artificial Intelligence, NUDT. She has been a visiting professor in Technische Universiteit Delft. She is the co-author of the book: HLA simulation system integrated design (Changsha, Hunan, National University of Defense Technology Press, 2008). Her current research interests include mission planning, system simulation, and artificial intelligence.

Prof. Huang was selected as an innovation talent in the military in 2012. She won the first prize and the second prize of the military science and technology advancement for four times. Prof. Huang was awarded the National March Eighth Red Flag Bearer and the second-class post allowance for outstanding professional.

Jianxing Gong was born in Fujian province, China, 1976. He received the B.E. degree in control science and engineering from Harbin Institute of Technology, Heilongjiang, China, in 2001, and got the master's degree and Ph.D. degree in control science and engineering from the College of Mechatronics and Automation, NUDT, in 2003 and 2007, respectively.

He now is an associate professor at the College of Artificial Intelligence, NUDT. His current research interests include system modelling, system simulation, and mission planning.