Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 926267, 8 pages
http://dx.doi.org/10.1155/2013/926267

Research Article

Multiagent Reinforcement Learning with Regret Matching for Robot Soccer

Qiang Liu,[1,2] Jiachen Ma,[1,2] and Wei Xie[1,2]

[1] *School of Astronautics, Harbin Institute of Technology, Harbin 150001, China*
[2] *School of Information and Electrical Engineering, Harbin Institute of Technology (Weihai), Weihai 264209, China*

Correspondence should be addressed to Qiang Liu; hit_liuqiang@163.com

Received 4 April 2013; Revised 19 July 2013; Accepted 20 July 2013

Academic Editor: Yudong Zhang

This paper proposes a novel multiagent reinforcement learning (MARL) algorithm Nash-Q learning with regret matching, in which regret matching is used to speed up the well-known MARL algorithm Nash-Q learning. It is critical that choosing a suitable strategy for action selection to harmonize the relation between exploration and exploitation to enhance the ability of online learning for Nash-Q learning. In Markov Game the joint action of agents adopting regret matching algorithm can converge to a group of points of no-regret that can be viewed as coarse correlated equilibrium which includes Nash equilibrium in essence. It is can be inferred that regret matching can guide exploration of the state-action space so that the rate of convergence of Nash-Q learning algorithm can be increased. Simulation results on robot soccer validate that compared to original Nash-Q learning algorithm, the use of regret matching during the learning phase of Nash-Q learning has excellent ability of online learning and results in significant performance in terms of scores, average reward and policy convergence.

1. Introduction

Multi-robot system (MRS) has received more and more attention because of its broad application prospect, which has several research platforms including formation [1], foraging [2], prey-pursuing [3, 4], and robot soccer [5–7]. Robot soccer is associated with robot architecture, cooperation, decision making, planning, modeling, learning, vision tracking algorithm, sensing, and communication, which owns all the key features of MRS. And the robot soccer system is discussed as a test benchmark in this paper [8].

Though reinforcement learning (RL), for example, Q-learning [9–11] can be directly applied in MRS for decision-making, it violates the static environment assumption of Markov Decision Process (MDP) [12]. For MRS action selection of the learning robot is unavoidably affected by actions of other agents, so multiagent reinforcement learning (MARL) involving joint state and joint action is more suitable and promising method for MRS [13–16].

MARL based on Stochastic Game (SG) that can be also called Markov game (MG) has a solid theoretical foundation for MRS, which has developed several branches such as MiniMax-Q learning [17], Nash-Q learning [18], FF-Q learning [19], and CE-Q learning algorithms [20]. Agents adopting the above algorithms can also be called equilibrium learners [17, 20, 21], which is one method of handling the loss of stationarity of MDP. These algorithms learn joint action values which are stationary and in certain circumstances guarantee that these values can converge to Nash equilibrium (NE) values [22] or correlated equilibrium (CE) values. Using these values, the agents'policy corresponds to the agent's component of some nash or correlated equilibrium [23]. So based on the fundamental solution concept of NE for MG, Nash-Q learning algorithm that finds NE at each state in order to obtain NE policies for Q value updating is an effective and typical MARL method.

For single agent learning scenario, Q-learning is guaranteed to converge to the optimal action independent of the action selection strategy. However, in a multiagent setting, the action selection policy becomes crucial for convergence to any joint action. A big challenge in defining a suitable strategy for the selection of actions is to strike a balance between exploring the usefulness of actions that have been attempted only a few times and exploiting those in which the agents' confidence in obtaining a high reward is relatively strong. This is known as the exploration and exploitation problem [24].

Regret matching can better harmonize the relation between exploration and exploitation. Regret has been studied both in game theory [25] and computer science [26, 27]. Regret measures how much worse an algorithm performs compared with the best static strategy whose goal is to guarantee at least zero average regret [23]. Regret matching [25] belonging to no-regret algorithms guarantees that the joint action will asymptotically converge to a set of points of no-regret that can be referred to as coarse correlated equilibrium in MG [28]. Because Nash equilibrium is in fact coarse correlated equilibrium [28], it can be inferred that regret matching that leads joint action to points of coarse correlated equilibrium can effectively improve the convergence rate of original Nash-Q learning algorithm.

This paper is organized as follows. Section 2 reviews multiagent reinforcement learning and Nash-Q learning algorithm. Section 3 briefly describes regret matching algorithm and then shows how to incorporate regret matching technique into original Nash-Q learning algorithm. Section 4 describes the structure of reinforcement learning of soccer robot. Section 5 presents simulation demonstration of our algorithm in robot soccer. Section 6 draws a conclusion and summarizes some important points about this paper.

2. Multiagent Reinforcement Learning and Nash-Q Learning

2.1. Markov Game. Markov game (MG) can be viewed as an extension of MDP to multiagent environments [29, 30], where all agents select their actions simultaneously. The reward that each agent gets depends on their joint action of all agents and the current state as well as the state transitions according to the Markov property [31]. MG is the theory foundation of MARL and Figure 1 shows the architecture. A reinforcement framework of MG can be defined by the following.

An n-agent MG Γ is a tuple $\langle n, S, A_1 \ldots, A_n, T, r_1, \ldots, r_n \rangle$, where n represents the number of agents, S is the state space, A_i is the action space of agent i $(i = 1, \ldots, n)$, $T : S \times A_1 \times \cdots \times A_n \times S \rightarrow \Delta(S)$ is the transition function which depends on the actions of all agents and $\Delta(S)$ is the set of probability distributions over state space S, and $r_i : S \times A_1 \times \cdots \times A_n \times S \rightarrow R$ is the reward function for agent i which also depends on the actions of all agents. Given state s, each agent independently chooses corresponding action $a_1, \ldots, a_n$ and then receives rewards $r_i(S, a_1, \ldots, a_n)$, $i = 1, \ldots, n$. The next state s' arrives after joint action $(a_1, \ldots, a_n)$ is taken at state s based on fixed transition probabilities and the following equation is satisfied:

$$\sum_{s' \in S} P\left(s', S, a_1, \ldots, a_n\right) = 1. \tag{1}$$

In a discounted MG, the objective of each agent is to maximize the discounted sum of rewards with discount factor $\beta \in [0, 1)$. Denote π_i as the strategy of agent i. For a given initial state s, agent i tries to maximize

$$V_i\left(s, \pi_1, \pi_2, \ldots, \pi_n\right) = \sum_{t=0}^{\infty} \beta^t E\left(r_1^t \mid \pi_1, \pi_2, \ldots, \pi_n, s_0 = s\right). \tag{2}$$

2.2. Comparing among Existing Algorithms. The traditional Q-learning algorithm [9] for computing an optimal policy in an MDP with unknown reward and transition functions is as follows:

$$\begin{gathered} Q(s, a) \longleftarrow (1 - \alpha) Q(s, a) + \alpha \left[r(s, a) + \beta V\left(s'\right)\right], \\ V(s) \longleftarrow \max_{a \in A} Q(s, a). \end{gathered} \tag{3}$$

The simplest way to extend this to the multiagent MG setting is just to add a subscript to the formulation above and the definition of the Q values assumes that they depend on the joint action of all agents. Meanwhile V should be updated with computation outcome of the Q values corresponding to respective algorithm.

The Minimax-Q learning algorithm as the first MARL extends the traditional Q-learning to the domain of two-player zero-sum multiagent MG environment. In Minimax-Q learning, V is updated with the minimax of the Q values:

$$V_1(s) \longleftarrow \max_{p_1 \in \prod(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} p_1\left(a_1\right) Q_1\left(s, a_1, a_2\right). \tag{4}$$

The policy used in the Minimax-Q learning algorithm can guarantee that it receives the largest value possible in the absence of knowledge of the opponent's policy.

Hu and Wellman [21] extended the Minimax-Q algorithm to n-player general-sum MG. The extension requires that each agent maintains Q values for all of the agents. And the linear programming solution used to find the equilibrium of zero-sum games is replaced by the quadratic programming solution for finding an equilibrium in n-player general-sum games. Nash-Q updates the V values based on some NE in the game defined by the Q-values:

$$V_i(s) \longleftarrow \text{Nash}_i\left(Q_1(s, a), \ldots, Q_n(s, a)\right), \tag{5}$$

where $Q_i(s, a)$ denotes the payoff matrix to player i and Nash_i denotes the Nash payoff to that player.

Since Nash-Q is limited to zero-sum and common-payoff games in essence, Littman reinterpreted it as the Friend-or-Foe-Q (FF-Q) learning framework [19]. Although FF-Q can be applied in multiple players scenario, for simplicity we show how the V are updated in a two-player game:

$$\begin{gathered} \text{Friend: } V_1(s) \longleftarrow \max_{a_1 \in A_1, a_2 \in A_2} Q_1\left(s, a_1, a_2\right), \\ \text{Foe: } V_1(s) \longleftarrow \max_{p_1 \in \prod(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} p_1\left(a_1\right) Q_1\left(s, a_1, a_2\right). \end{gathered} \tag{6}$$

Thus Friend-Q updates V similarly to regular Q-learning, and Foe-Q updates as does minimax-Q.

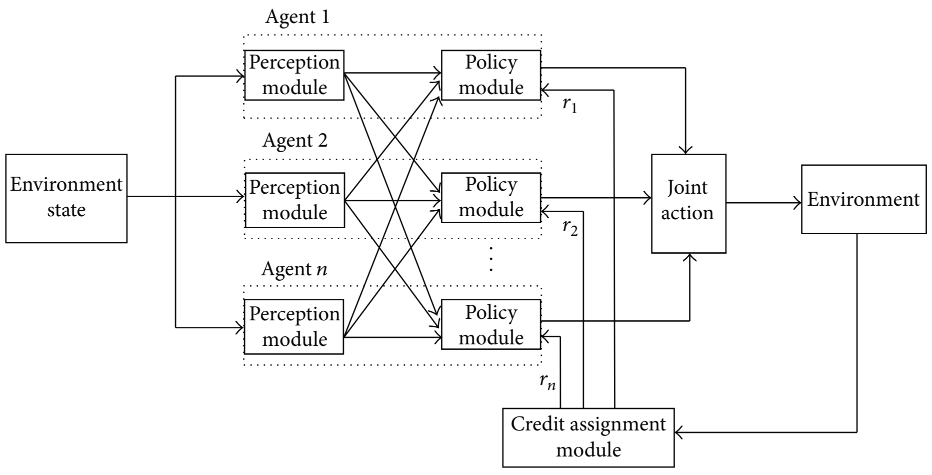

Figure 1: Architecture of MARL based on Markov game.

The above algorithms in this section all depend on some methods of computing the NE for the matrix game defined by Q values of all players in each state. The value for each player of a mutually agreed-on equilibrium is the value function used in the Q update process. Instead of computing Nash equilibria of Q stage games, the agent can compute other solution concepts. One option is computing the CE. This is the technique used by Greenwald and Hall in the unambiguously named Correlated-Q (CE-Q) algorithm [20]. A CE is more general than an NE, since it allows dependencies among the agents' probability distributions, while maintaining the property that agents are optimizing. Compared to NE, CE can be computed easily via linear programming. CE-Q learning is similar to Nash-Q but instead uses the value of a correlated equilibrium to update V:

$$V_i(s) \longleftarrow \mathrm{CE}_i(Q_1(s,a),\dots,Q_n(s,a)). \tag{7}$$

Like Nash-Q, it requires agents to select a unique equilibrium; an issue that the authors address explicitly by suggesting several possible selection mechanisms.

2.3. Nash-Q Learning. The following is based on [18]. Extending Q-learning to the multiagent learning domain with NE concept, Nash-Q equilibrium value is defined as the expected sum of discounted rewards when all agents follow specified Nash equilibrium strategies from the next period on. The literature usually uses the terms policy and strategy interchangeably. A Nash equilibrium is a joint strategy where each agent's strategy is a best response to the others' strategies.

In MG Γ, a Nash equilibrium point is a tuple of n strategies $(\pi_1^*,\dots,\pi_n^*)$ such that for all $s\in S$ and $i=1,\dots,n$,

$$\begin{aligned} &v_i(s,(\pi_1^*,\dots,\pi_n^*)) \\ &\quad \ge v_i(s,\pi_1^*,\dots,\pi_{i-1}^*,\pi_i,\pi_{i+1}^*\dots,\pi_n^*) \quad \forall \pi_i\in\Pi_i, \end{aligned} \tag{8}$$

where Π_i is the set of strategies available to agent i. Q_i^* is defined as a Nash-Q function for agent i and $Q_i^*(s,a_1,\dots,a_n)$ is called Nash-Q equilibrium value. Nash-Q function of agent i is defined over $(s,a_1,\dots,a_n)$, as the sum of agent i's current reward plus its future rewards when all agents follow the joint NE strategy. That is,

$$\begin{aligned} Q_i^*&(s,a_1,\dots,a_n) \\ &= r_*^i(s,a_1,\dots,a_n) \\ &\quad + \beta\sum_{s'\in S} P(s' \mid s,a_1,\dots,a_n)\, v^i(s',\pi_1^*,\dots,\pi_n^*), \end{aligned} \tag{9}$$

where $(\pi_1^*,\dots,\pi_n^*)$ is the joint Nash equilibrium strategy, $r_i(S,a_1,\dots,a_n)$ is agent i's one stage reward in state s and under joint action $(a_1,\dots,a_n)$, and $V_i(s',\pi_1^*,\dots,\pi_n^*)$ is agent i's total discounted reward over infinite periods starting from state s' given that agents follow the equilibrium strategies.

In the case of multiple equilibria, different NE strategy profiles may select different Nash-Q functions. In this paper, the learning agent picks the NE that yields the highest expected payoff to them as a whole. The learning agent indexed by i learns about its Q values by forming an arbitrary guess at time 0. One simple guess would be letting $Q_i^0(s,a_1,\dots,a_n)=0$ for all $s\in S$, $a_1\in A_1,\dots,a_n\in A_n$. At each time t, agent i observes the current state and then takes its action. After actions were taken, agent i observes its own reward, actions taken by all other agents, others' rewards, and the new state s'. It then calculates a Nash equilibrium $\pi_1(s')\cdots\pi_n(s')$ and updates its Q values according to

$$\begin{aligned} Q_i^{t+1}(s,a_1,\dots,a_n) &= (1-\alpha_t)\,Q_i^t(s,a_1,\dots,a_n) \\ &\quad + \alpha_t\left[r_i^t + \beta \mathrm{Nash}Q_i^t(s')\right], \end{aligned} \tag{10}$$

where $\mathrm{Nash}Q_i^t(s')=\pi_1(s')\cdots\pi_n(s')\cdot Q_i^t(s')$.

$\mathrm{Nash}Q_i^t(s')$ is agent i's payoff in state s' for the selected equilibrium. Note that $\pi_1(s')\cdots\pi_n(s')\cdot Q_i^t(s')$ is a scalar. The learning algorithm is as follows:

```
Initialize:
    Let $t = 0$, get the initial state $s_0$;
    Let the learning agent be indexed by $i$;
```

```
        For all $s \in S$ and $a_j \in A_j$, $j = 1, \ldots, n$, let
        $Q_j^t(s, a_1, \ldots, a_n) = 0$.
    Loop
        Choose action $a_i^t$;
        Observe $r_1^t, \ldots, r_n^t$; $a_1^t, \ldots, a_n^t$, and $s^{t+1} = s'$;
        Update $Q_j^t$ for $j = 1, \ldots, n$.
```

$$Q_j^{t+1}(s, a_1, \ldots, a_n) = (1 - \alpha_t) Q_j^t(s, a_1, \ldots, a_n) + \alpha_t \left[r_j^t + \beta \mathrm{Nash} Q_j^t(s') \right], \tag{11}$$

```
        where $\alpha_t \in (0, 1)$ is the learning rate, and
        $\mathrm{Nash}Q_j^t(s')$ is defined in (10).
        Let $t := t + 1$.
```

For obtaining the NE $\pi_1(s') \cdots \pi_n(s')$, agent i need to know $Q_t^1(s'), \ldots, Q_t^n(s')$. Agent i should have conjectures about those Q-functions at the beginning of play. As the game proceeds, agent i observes other agents' immediate rewards and previous actions. That information can then be used to update agent i's conjectures on other agents' Q-functions. Agent i updates its beliefs about agent j's Q-function, according to the same updating rule (10) it applies to its own:

$$Q_j^{t+1}(s, a_1, \ldots, a_n) = (1 - \alpha_t) Q_j^t(s, a_1, \ldots, a_n) + \alpha_t \left[r_j^t + \beta \mathrm{Nash} Q_j^t(s') \right]. \tag{12}$$

Note that $\alpha_t = 0$ for $(s, a_1, \ldots, a_n) \neq (s_t, a_1, \ldots, a_n)$. Therefore (12) does not update all the entries in the Q-functions. It updates only the entry corresponding to the current state and actions chosen by the agents. This type of updating is called asynchronous updating [18].

3. Regret Matching Algorithm for Action Selection

By observing human ways of handling problems, we can conclude that a human often reflects how regretful it is for the decision that he had made. Through reflecting on past action and feeling regretful, a human can learn more experience, find improved action under complicated environment, and enhance the learning efficiency. Regret enables him to obtain better policy and to make progress quickly. In case that people of community all adopt such idea, then the joint action will bring each one good reward.

Based on the above notion, no regret learning algorithms are proposed and have been widely studied and applied in multiagent learning. No regret learning algorithms consist of a lot of algorithms which guarantee that the joint action will converge asymptotically to a set of points of no-regret that can also be called coarse correlated equilibrium [32]. A no-regret point represents a case for which the average reward which an agent actually obtained is as much as the counterpart that the agent "would have" obtained had that agent used a different fixed strategy at all previous time steps [28]. Figure 2 shows that Nash equilibrium belongs to not only correlated equilibrium but also coarse CE. In other words, it is important to note that convergence to a NE point also implies convergence to a coarse correlated equilibrium point (no-regret point).

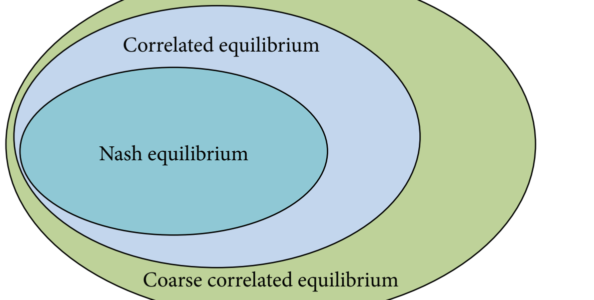

Figure 2: Relationship between Nash, correlated, and coarse correlated equilibrium.

The prominent feature of regret matching [25] as a branch of no regret learning algorithms is that compared to other learning algorithms, for example, fictitious play [33], it can be easily applied in large scale MRS [28]. The detailed description of regret matching can be found in [25]. And a new algorithm Nash-Q learning with regret matching is proposed to increase the rate of convergence in MG. In the proposed algorithm, regret matching is used to select the action in each state to increase the convergence rate toward Nash equilibrium policy.

According to the above notation, we define the average regret $R_i^{a_i}(s, t)$ of agent i at time t and in state s as

$$R_i^{a_i}(s, t) = \frac{1}{N} \sum_{m=0}^{N-1} \left(r_i(s, a_i, a_{-i}(m)) - r_i(s, a(m)) \right), \tag{13}$$

where a_{-i} denotes the collective $(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots\ a_n)$, of agents' action except agent i, a represents the joint action $(a_1, \ldots, a_n)$ of all agents, and N represents the number of state s visited.

Equation (13) shows that average regret for $a_i \in A_i$ of agent i would represent the average improvement in his reward if it had chosen $a_i \in A_i$ in all past steps and all other agents' actions had remained unchanged up to time t. Regret matching based each agent i computes $R_i^{a_i}(s, t)$ for every action $a_i \in A_i$ using the following iterative equation:

$$R_i^{a_i}(s, t) = \frac{t-1}{t} R_i^{a_i}(s, t-1) + \frac{1}{t} \left(r_i(s, a_i, a_{-i}(t)) - r_i(s, a(t)) \right). \tag{14}$$

Note that at each time step $t > 0$, agent i updates all entries included in his average regret assemble $R_i(s, t) = [R_i^{a_i}(s, t)]_{a_i \in A_i}$. In regret matching after agent i computed its average regret assemble $R_i(s, t)$, action $a_i(s, t)$ is selected according to the probability distribution $p_i(t)$, as shown in the following equation:

$$p_i^{a_i}(t) = \Pr\left[a_i(s,t) = a_i\right] = \frac{R_i^{a_i}(s,t)}{\sum_{a_i' \in A_i}\left[R_i^{a_i'}(s,t)\right]}, \tag{15}$$

where $p_i(t)$ is the uniform distribution over A_i. In other words, an agent using regret matching selects a particular action at any time step with probability proportional to the average regret for not selecting that particular action in the past time steps.

If all agents of one team choose regret matching algorithm for robot soccer, then the joint action will converge asymptotically to a set of points of coarse CE. So it can be inferred that regret matching can effectively improve the convergence rate of original Nash-Q learning, which is validated by the following simulation.

4. Action-Based Soccer Robot

4.1. Environment States and Joint Action of Robot. Robot soccer is an very challenging and interesting domain for the application of machine learning algorithms to real world problems. Research groups have applied a lot of different machine learning approaches to many facets of autonomously soccer playing MRS [34].

Behavior-(action-) based approaches are very suited for soccer because they have outstanding performance than deliberative control in uncertain and dynamic environments. Behavior design of the robot (agent) soccer team is based on the following two characteristics. Firstly, points are scored by kicking the ball across the opponent team's goal. Secondly, robots should avoid kicking the ball toward the wrong directions, lest they score against their own team [35].

In this paper, environment states represented in Figure 3 are used to activate the robot. For simplicity each team is composed of three agents (players) as shown in Figure 5. Based on [36], a motor schema-based reactive control system is used for action designing in which each agent is provided three preprogrammed actions (behavior assemblages) that correspond to steps in achieving the task as shown in Table 1. These actions are in turn composed of more primitive behaviors called motor-schemas. Several motor-schemas are described as follows.

Move_to_kickspot: high gain to draw the robot to a point one-half of a robot radius behind the ball. If the robots bumps the ball from that location, the ball is propelled in the direction of the opponent's goal. Avoid_teammates: gain sufficiently high to keep the robots on the team spread apart. Move_to_half_point: high gain to draw the robot to a point halfway between the ball and the defended goal. Swirl_ball: a ball dodging vector with gain sufficiently high to keep the robots from colliding with the ball. Move_to_defended_goal: high gain to draw the robot to the defend goal [35].

Shoot ball action is showed in Figure 4. Being analogous to Figure 4, chase ball action is composed of three primitive schemas: move_to_halfway_point, swirl_ball, and avoid_teammates. Goal keeping action is composed

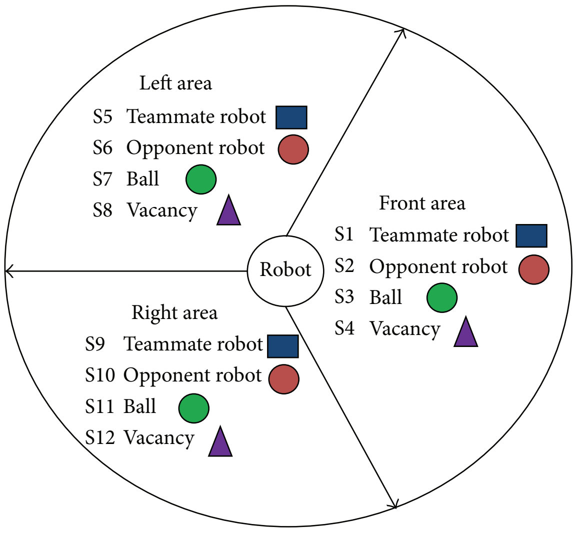

FIGURE 3: Environment states around soccer robot.

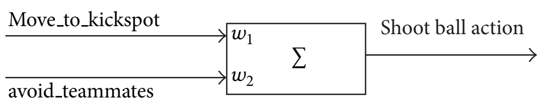

FIGURE 4: Motor schema based on shoot ball action.

of two primitive schemas: move_to_defended_goal and move_to_kickspot.

4.2. Reward Function for Soccer. As an instant evaluator the reward function for the action taken at a given state is important for reinforcement learning. Global reinforcement [35] refers to the case where a single reinforcement signal is simultaneously delivered to all robots. A potential problem with global reinforcement is the ambiguous assumption that the closet robot just happened to be near the goal while another soccer robot kicked the ball for a score from a distance. Two important factors should be considered: time and distance, and a modified reward function $r_i(s,a)$ for agent i from global reinforcement is as follows:

$$r_i(s,a) = \begin{cases} \eta^{\text{touch}} + \dfrac{1}{d} & \text{if the team scored at } t-1, \\ -\left(\eta^{\text{touch}} + \dfrac{1}{d}\right) & \text{if the opponent scored at } t-1, \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

where s denotes the soccer robot's state, a represents the joint action of all agents $(a_1,\ldots,a_n)$, touch is time in milliseconds since the soccer robot last touched the ball, and d represents the distance in meters between the ball and robot. η is a parameter value varying between 0.5 and 1 that indicates how quickly a potential reward should decay after the ball is touched, and in this paper η is set to be 0.7.

Table 1: Actions of soccer robot [37].

Action	Robot activity
Shoot ball	If robot is close to the ball and goal, this action is used to shoot the ball.
Chase ball	When robot is far away from ball, this action is given to go after the ball.
Goal keeping	Robot playing as a goal keeper gets this action to prevent losing point.

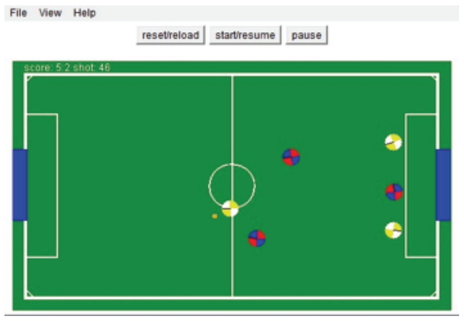

Figure 5: Simulated robot soccer.

5. Simulation

TeamBots as shown in Figure 5 is a Java-based assemble of application programs and java packages for multiagent mobile robotics research, where control system of a robot interacts with a well-defined sensor-actuator interface. The simulation proceeds in discrete steps. The robots process their sensor data in each step and then issue appropriate actuator commands. The simulation models physical interactions including robot, ball and wall collisions, sensors, and motor-driven actuators [24].

Two teams A and B of soccer robots are designed and each team is composed of three agents. Team A adopts Nash-Q learning with regret matching algorithm and Team B is equipped with original Nash-Q learning that learns Nash-Q equilibrium values by random action selection strategy. If a goal is kicked, the ball will be replaced to the center of the field without repositioning the agents and the match goes on. Historical data including scores, average reward, and the average number of policy changes are saved as the match proceeds. The agents preserve Nash-Q values learned between matches. No limited time is imposed on playing that the whole match is not over until a total of 10 points are completed. The simulation is composed of 100 10-point matches. The reward functions that the robots Team A and B adopt are the same as shown in (16).

At the beginning Q values of all robots were initialized with zero value. An important performance for robot soccer is measured as the scores difference D:

$$D = S_{\text{teamA}} - S_{\text{teamB}}, \tag{17}$$

where S_{teamA} denotes the scores of Team A and S_{teamB} is the scores of Team B. A negative value indicates that Team A lost the match, while positive values indicate that Team A won the match. Figure 6 shows the curves of scores difference D through which we know that robots of Team A found good strategy of joint action resulting in draw or scoring over 5 points after the 37th match and outperformed Team B from the 52th match to the end of simulation. It can be summarized that robots of Team A with action exploration strategy of regret matching have accumulated much experience by computing regret value for every action and gradually taking joint action improved after the 37th match. By online learning of regret matching, the joint actions of robots of Team A are gradually close to approach points of coarse correlated equilibrium, which greatly improved the offensive and defensive capabilities of the whole team.

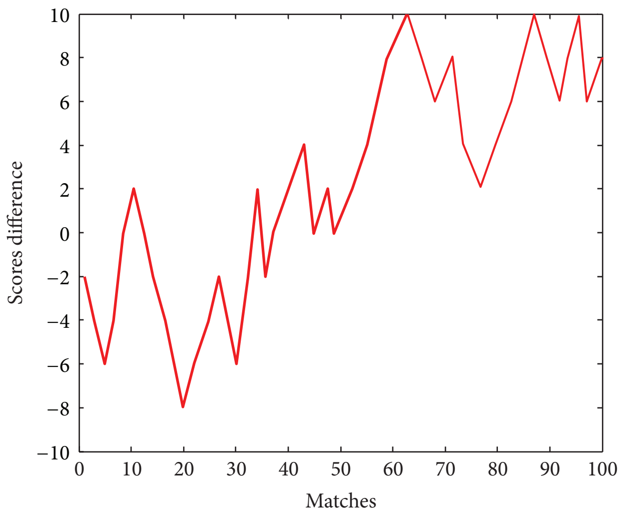

Figure 6: Scores difference D as the number of matches increases.

Through Figure 7, it may be concluded that the robots of Team A received positive rewards most of the matches. The average reward per match is increased as matches proceed when the robots obtained more experience of cooperating. It increases from approximately 3.8 to 8.7 in 100 rounds of continuous matches. Because a bigger average reward indicates that the robots have employed good cooperation strategies to kicking more goals, Figure 7 confirms that regret matching as action selection strategy is effective in helping the agents to improve the quality of tactics coordination in carrying out the cooperative attacking. Although for the initial learning phase (the former 37 matches) Team A has worse performance than Team B, as the matches proceed the performance of Team A becomes better and better as we expect before the simulation. For the latter 63 matches, the robots of Team A can quickly adapt themselves to the transition of environment state and coordinate their joint action reducing conflict with their own teammates and obtaining more and more positive rewards.

Learning rate is evaluated by monitoring the policy convergence which is tracked by recording the average number

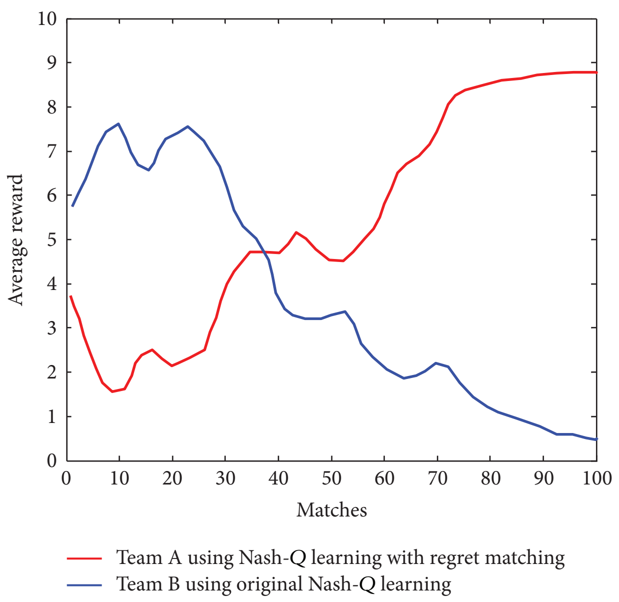

Figure 7: The average reward received by the robots in match.

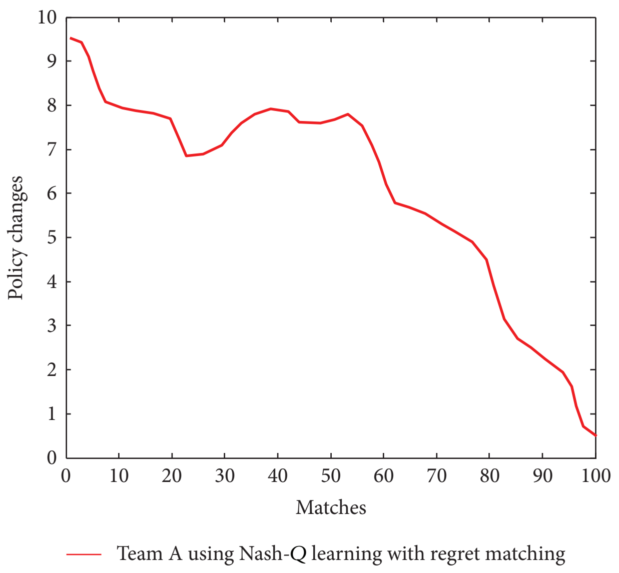

Figure 8: The average number of policy changes per match.

of policy changes for all agents of Team A. For example, an agent from Team A may have been following a strategy of goal keeping when opponents appearing in front area, the ball in front area and teammates in left and right, but owing to regret matching it switches to the chase ball action instead. Such alteration is viewed as policy change. The average number of policy changes for Team B is stochastic because of strategy of random action selection. So only the curve of policy changes of Team A is analyzed. The data plotted in Figure 8 shows good convergence for Team A using regret matching algorithm. The average number of policy changes per match dropped to 0.47 after 100 matches.

The number of policy changes for robots of Team A initially is high but decreases gradually in the latter matches. It can be noted that there is turnpoint at around the 52th match, from which the average number of policy changes of Team A monotonously diminished. An extended simulation shows that the average number of policy changes for Team A reached zero after 150 matches.

From Figures 6 to 8, it is clear that the new Nash-Q with regret matching learning algorithm has higher learning efficiency than the original Nash-Q learning algorithm in robot soccer. Regret matching can better harmonize the tradeoff between exploration and exploit such that the agent can reinforce the evaluation of the actions it already knows to be good but also explore new actions. In particular, the new algorithm Nash-Q learning with regret matching takes an average of 150 matches for completing policy convergence in order to find Nash-Q equilibrium values early.

6. Conclusion

This paper presents a new multiagent reinforcement learning approach combining Nash-Q learning with regret matching to increase the convergence rate of original Nash-Q learning algorithm that learns Nash-Q equilibrium values by random action selection in multiagent system. Regret matching which belongs to online learning as a branch of no regret learning algorithms can guarantee that the joint action will asymptotically converge to a set of points of coarse correlated equilibrium including Nash equilibrium points. So we investigate how to make improved action selection in original Nash-Q learning algorithm through regret matching. Robot soccer is adopted as platform to test the proposed approach. Compared to original Nash-Q learning, the results of experiments validate that Nash-Q learning with regret matching algorithm has better performance in terms of scores, average reward, and policy convergence for obtaining the Nash equilibrium policy.

References

[1] P. K. C. Wang, "Navigation strategies for multiple autonomous mobile robots moving in formation," *Journal of Robotic Systems*, vol. 8, no. 2, pp. 177–195, 1991.

[2] M. J. Matarić, "Reinforcement learning in the multi-robot domain," *Autonomous Robots*, vol. 4, no. 1, pp. 73–83, 1997.

[3] M. Tan, "Multi-agent reinforcement learning: Independent versus cooperative agents," in *Proceedings of the 10th International Conference on Machine Learning*, pp. 330–337, Morgan Kaufmann, 1993.

[4] T. Fujii, Y. Arai, H. Asama, and I. Endo, "Multilayered reinforcement learning for complicated collision avoidance problems," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2186–2191, May 1998.

[5] E. Uchibe, M. Nakamura, and M. Asada, "Co-evolution for cooperative behavior acquisition in a multiple mobile robot environment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 425–430, October 1998.

[6] K. H. Park, Y. J. Kim, and J. H. Kim, "Modular Q-learning based multi-agent cooperation for robot soccer," *Robotics and Autonomous Systems*, vol. 35, no. 2, pp. 109–122, 2001.

[7] Y. Ma, Z. Cao, X. Dong, C. Zhou, and M. Tan, "A multi-robot coordinated hunting strategy with dynamic alliance," in *Proceedings of the Chinese Control and Decision Conference (CCDC '09)*, pp. 2338–2342, chn, June 2009.

[8] J. H. Kim and P. Vadakkepat, "Multi-agent systems: a survey from the robot-soccer perspective," *Intelligent Automation and Soft Computing*, vol. 6, no. 1, pp. 3–18, 2000.

[9] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[10] M. E. Harmon and S. S. Harmon, *Reinforcement Learning: A Tutorial*, Wright Lab, Wright-Patterson AFB, Ohio, USA, 1997.

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

[12] Y. Wang, *Cooperative and intelligent control of multi-robot systems using machine learning [thesis]*, The University of British Columbia, 2008.

[13] Y. Duan, B. X. Cui, and X. H. Xu, "A multi-agent reinforcement learning approach to robot soccer," *Artificial Intelligence Review*, vol. 38, no. 3, pp. 193–211, 2012.

[14] F. Michaud and M. J. Matarić, "Learning from history for behavior-based mobile robots in non-stationary conditions," *Machine Learning*, vol. 31, no. 1–3, pp. 141–167, 1998.

[15] M. Asada, E. Uchibe, and K. Hosoda, "Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development," *Artificial Intelligence*, vol. 110, no. 2, pp. 275–292, 1999.

[16] M. Wiering, R. Sałustowicz, and J. Schmidhuber, "Reinforcement learning soccer teams with incomplete world models," *Autonomous Robots*, vol. 7, no. 1, pp. 77–88, 1999.

[17] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the 11th International Conference on Machine Learning*, pp. 157–163, 1994.

[18] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 1039–1069, 2004.

[19] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, pp. 322–328, 2001.

[20] A. Greenwald and K. Hall, "Correlated-Q learning," in *Proceedings of the 20th International Conference on Machine Learning*, pp. 242–249, August 2003.

[21] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: theoretical framework and an algorithm," in *Proceedings of the 15th International Conference on Machine Learning*, pp. 242–250, 1998.

[22] J. Nash, "Non-cooperative games," *The Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[23] M. Bowling, "Convergence and no-regret in multiagent learning," *Advances in Neural Information Processing Systems*, vol. 17, pp. 209–216, 2005.

[24] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in cooperative multi-agent systems," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 326–331, AAAI Press, MIT Press, Menlo Park, Calif, USA, 1999.

[25] S. Hart and A. Mas-Colell, "A simple adaptive procedure leading to correlated equilibrium," *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2000.

[26] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "Gambling in a rigged casino: the adversarial multi-armed bandit problem," in *Proceedings of the 36th IEEE Annual Symposium on Foundations of Computer Science*, pp. 322–331, October 1995.

[27] M. Zinkevich, "Online convex programming and generalized ininitesimal gradient ascent," in *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, Washington, DC, USA, 2003.

[28] J. R. Marden, *Learning in Large-Scale Games and Cooperative Control*, University of California, Los Angeles, Los Angeles, Calif, USA, 2007.

[29] F. Thusijsman, *Optimality and Equilibrium in Stochastic Games*, Centrum voor Wiskunde en Informatica, 1992.

[30] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team markov games," *Advances in Neural Information Processing Systems*, vol. 15, pp. 1571–1578, 2002.

[31] E. Yang and D. Gu, "Multiagent reinforcement learning for multi-robot systems: a survey," University of Essex Technical Report CSM-404, Department of Computer Science, 2004.

[32] H. P. Young, *Strategic Learning and Its limIts*, Oxford University Press, New York, NY, USA, 2004.

[33] D. Monderer and L. S. Shapley, "Fictitious play property for games with identical interests," *Journal of Economic Theory*, vol. 68, no. 1, pp. 258–265, 1996.

[34] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Autonomous Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[35] T. Balch, *Behavioral divesity in learning robot teams [thesis]*, Georgia Institute of Technology, 1998.

[36] R. C. Arkin, "Cooperation without communication: multiagent schemabased robot navigation," *Journal of Robotic Systems*, vol. 9, no. 3, pp. 351–364, 1992.

[37] T. Y. Tsou, C. H. Liu, and Y. T. Wang, "Term formation control for soccer robot systems," in *Proceeding of the IEEE International Conference on Networking, Sensing and Control*, pp. 1121–1125, March 2004.