IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 16, 2020, accepted November 20, 2020, date of publication November 27, 2020,
date of current version December 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3040985

WRFMR: A Multi-Agent Reinforcement Learning Method for Cooperative Tasks

HUI LIU[1], ZHEN ZHANG[1], AND DONGQING WANG[2]
[1]School of Automation, Qingdao University, Qingdao 266071, China
[2]School of Electrical Engineering, Qingdao University, Qingdao 266071, China

Corresponding author: Zhen Zhang (tbsunshine8@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61903209, Grant 61873138, and Grant 61573205, in part by the Qingdao Postdoctoral Applied Research Project with Name (AGV Road Network Design and Path Planning Method Based on Multi-Agent Reinforcement Learning), in part by the Shandong Provincial Natural Science Foundation of China under Grant ZR2017PF005, and in part by the Science and Technology Support Plan for Youth Innovation of Universities in Shandong Province under Grant 2019KJN033.

ABSTRACT Multi-agent reinforcement learning (MARL) for cooperative tasks has been extensively studied in recent years. The balance of exploration and exploitation is crucial to MARL algorithms' performance in terms of the learning speed and the quality of the obtained strategy. To this end, we propose an algorithm known as the weighted relative frequency of obtaining the maximal reward (WRFMR), which uses a weight parameter and the action probability to balance exploration and exploitation and accelerate convergence to the optimal joint action. For the WRFMR algorithm, each agent needs to share the state and the immediate reward and does not need to observe the actions of the other agents. Theoretical analysis on the model of WRFMR in cooperative repeated games shows that each optimal joint action is an asymptotically stable critical point if the component action of every optimal joint action is unique. The box-pushing task, the distributed sensor network (DSN) task, and a strategy game known as blood battlefield are used for empirical studies. Both the DSN task and the box-pushing task involve full cooperation, while blood battle comprises both cooperation and competition. The simulation results show that the WRFMR algorithm outperforms the other algorithms regarding the success rate and the learning speed.

INDEX TERMS Multi-agent reinforcement learning, reinforcement learning, multi-agent system, repeated game.

I. INTRODUCTION

Reinforcement learning (RL) is a prevalent method to optimize a single agent's strategy in a Markov Decision Process (MDP). An agent can perceive the state with sensors, make decisions, and execute actions through actuators. Some tasks are naturally modeled as multi-agent systems (MASs) in which the Markov property still holds from the view of centralized learning [1]. However, the joint action space grows exponentially as the number of agents increases. Independent learning [2]–[4], which does not need any agent to observe the actions of the other agents, has been proposed to alleviate the dimension curse of the joint action space. In independent learning, each agent maintains a Q-value function that evaluates the benefit of its own action. In this article, we concern only independent learning algorithms.

The associate editor coordinating the review of this manuscript and approving it for publication was Ikramullah Lali.

The purpose of the MARL algorithms depends on the nature of the task. In zero-sum games, the goal is to maximize each agent's reward while thinking of the other agents in a pessimistic way [5]. In general-sum games, the goal is to converge to the Nash equilibrium (NE) [6], [7]. In fully-cooperative games, the goal is to maximize the sum of all agents' reward [8]–[12]. In addition, some algorithms can be applied to mixed tasks [13]–[16]. In this article, we focus on algorithms for fully cooperative tasks.

Two factors have to be considered when designing an independent MARL algorithm for fully cooperative tasks. First, the convergence to the optimal joint strategy is crucial for an effective algorithm. Most of the existing results on convergence analysis are limited to repeated games with two agents and two actions. Theoretical results of the convergence of MARL in repeated games with an arbitrary finite number of agents and actions are not much. Second, the learning speed is vital for an efficient algorithm. For an RL-based

algorithm, a well-designed exploration and exploitation policy can improve the learning speed. Exploitation is to use current information to generate a better solution. Exploration is to explore the search space more thoroughly to avoid falling into local optima. We propose an algorithm known as the weighted relative frequency of obtaining the maximal reward (WRFMR). The main contributions are as follows. First, the WRFMR algorithm does not need any agent to observe the actions of the other agents. Second, the decreasing weight parameter and the action probability are used to balance exploration and exploitation to improve the learning speed. Third, we analyze the characteristics of the WRFMR algorithm in repeated games with an arbitrary finite number of agents and actions. Theoretical analysis shows that each optimal joint action is an asymptotically stable critical point if the component action of every optimal joint action is unique. Empirical studies on repeated games and stochastic games are also presented. The efficacy of the WRFMR algorithm is studied through three fully cooperative tasks – the distributed sensor network (DSN) task, the box-pushing task, and a strategic game known as blood battlefield. The results show that the WRFMR algorithm outperforms the other algorithms in terms of the success rate and the learning speed. Joint action learner needs to estimate the Q-value of each joint action, while independent learner needs to estimate the Q-value of each component action.

A brief description of the other sections in this article is as follows. Section II reviews the related work on MARL algorithms. Section III introduces repeated games and stochastic games. Section IV elaborates the WRFMR algorithm in detail and presents a theoretical analysis of the characteristics of the algorithm in repeated games with an arbitrary number of agents and actions. Section V studies the efficacy of the WRFMR algorithm over the other MARL algorithms in three fully cooperative tasks – the distributed sensor network task, the box-pushing task, and a strategy game known as blood battlefield. Section VI gives the conclusion.

II. PREVIOUS WORK

In this section, the MARL algorithms for fully cooperative games and general-sum games, and multi-agent deep reinforcement learning (MDRL) algorithms are reviewed respectively. The MARL algorithms can belong to joint action learner or independent learner. For joint action learner, each agent can perceive the action of each of the other agents. For independent action learner, each agent cannot observe the actions of the other agents.

In fully cooperative games, the goal is to optimize the joint strategy to obtain the maximum sum of all agents' reward. Team Q-learning [17] avoids the coordination mechanism by assuming that all optimal joint actions are unique. Joint action learner (JAL) [18] learns the Q-value of each joint action, and needs each agent to construct models for its teammates to promote coordination. Optimal adaptive learning (OAL) [19] needs each agent to construct its teammates' models, and use the models to obtain the optimal joint action of each virtual game on the top of each stage of the stochastic game. The probability of maximum reward based on estimated gradient ascent (PMR-EGA) [20] uses the gradient of the probability of obtaining the maximum reward to each agent's strategy. The gradient information is estimated by the Q-value function of the joint actions. PMR-EGA has been proven to converge to the optimal joint action in repeated games with two optimal joint actions that have different component actions. Team Q-learning, JAL, OAL, and PMR-EGA belong to joint action learner. Q-learning with aggregation (QA-Learning) [21] reduces the complicacy tasks with large state space by decomposing the task into more manageable sub-tasks, and distributing agents between these sub-tasks, to promote efficiency and enhance parallelization. The frequency of the maximal reward Q-learning (FMRQ) [22] uses the frequency of obtaining the maximal reward to update the strategy of each agent. It uses the stability theory to analyze the convergence of the algorithm in some specific repeated games. QA-Learning and FMRQ belong to independent learner.

In general-sum games, the goal is to converge to the Nash equilibrium. Some algorithms use the gradient information to update each agent's strategy, such as infinitesimal gradient ascent (IGA) [23], win or learn fast IGA (WoLF-IGA) [24], generalized IGA (GIGA) [25], and GIGA-WoLF [26]. Convergence with Model Learning and Safety (CMLES) [27] ensures targeted optimality for memory-bounded agents and safety for any other set of agents. These algorithms belong to JAL. The win or learn fast policy hill climbing (Wolf-PHC) [24] converges to the Nash equilibrium in two-agent-two-action repeated games by using the 'Win or Learn Fast' rule to update each agent's strategy. The exponential moving average (EMA) Q-learning [28] algorithm uses the exponential moving average mechanism to update each agent's strategy. The policy gradient ascent with approximate policy prediction (PGA-APP) [29] augments the basic gradient ascent method through approximate policy prediction. PGA-APP performs better than GIGA-WoLF in some stochastic games. The max or minimax Q-learning (M-Qubed) [30] balances best response, optimistic, and cautious learning biases to make profitable compromises in general-sum games. WoLF-PHC, EMA Q-learning, PGA-APP, and M-Qubed belong to independent learner. Table 1 and Table 2 show the classification of MARL algorithms according to two fundamental classes (independent learner and joint action learner) and the nature of the scenarios (cooperative scenarios and general-sum scenarios) respectively.

MDRL becomes an emerging research area in RL community. In multi-agent deep deterministic policy gradient (MADDPG) [31], each actor uses local observations to select actions and each critic uses the global state to evaluate the Q-value conditioned on the joint action. Counterfactual multi-Agent policy gradients (COMA) [35] uses a centralized critic and addresses the multi-agent credit assignment by using a counterfactual baseline. Value-decomposition networks (VDN) [32] uses a linear value-decomposition method where the global Q-functions is approximated by a

TABLE 1. Classification of MARL algorithms according to independent learner and joint action learner.

Category	Features
independent learner QA-Learning [21], FMRQ [22], EMA Q-learning [28], Wolf-PHC [24], PGA-APP [29], and M-Qubed [30]	Each agent cannot observe the actions of the other agents.
joint action learner Team Q-learning [17], JAL [18], OAL [19], PMR-EGA [20], IGA [23] WoLF-IGA [24], GIGA [25], GIGA-WoLF [26], and CMLES [27]	Each agent can perceive the action of each of the other agents.

TABLE 2. Classification of MARL algorithms according to cooperative scenarios and general-sum scenarios.

Category	Features
cooperative scenarios Team Q-learning [17], JAL [18], OAL [19] , PMR-EGA [20], QA-Learning [21], FMRQ [22], MADDPG [31], VDN [32], QMIX [33], and LDQN [34]	The goal is to maximize the global cumulative reward.
general-sum scenarios IGA [23], WoLF-IGA [24], GIGA [25], GIGA-WoLF [26], CMLES [27], EMA Q-learning [28], Wolf-PHC [24], PGA-APP [29], M-Qubed [30], and MADDPG [31]	The goal is to obtain some type of Nash equilibria.

sum of local Q-functions. QMIX [33] uses a mixing network to approximate the global Q-function by conflating the local Q-functions and the global state in a non-linear way. Lenient-DQN (LDQN) [34] applies leniency mechanism with decaying temperature values to regulate policy updates. To overcome the non-stationarity problem, fingerprint [36] disambiguates the age of the samples obtained from the replay memory applying a fingerprint.

We propose the WRFMR algorithm for cooperative agents. It has the following characteristics. First, compared with joint action learner such as OAL and PME-EGA, the WRFMR algorithm does not need each agent to observe the actions of the other agents and therefore mitigates the curse of dimensionality of the action space. Second, compared with FMRQ, the WRFMR algorithm uses the weight parameters and action probabilities to accelerate convergence to the optimal joint action.

III. PRELIMINARIES

A. STOCHASTIC GAMES

A Stochastic game is a tuple $< S, A_1, \ldots A_n, T, r_1, \ldots r_n >$, where n is the number of agents, S is the set of states, A_i is the set of agent $i's$ actions, T is the state transition function, and r_i is the immediate reward of agent i. The set of joint actions is denoted by $A = A_1 \times A_2, \ \ldots, \ \times A_n$ which consists of the actions of all agents. The state transition function $T : S \times A_1 \times A_2, \ \ldots, \ \times A_n \times S \rightarrow [0, 1]$ is probability distribution to transit to the next state s' given the current state s and the executed joint action a. The immediate reward of agent i $r_i : S \times A_1 \times A_2, \ \ldots, \ \times A_n \times S \rightarrow R$ is determined by the state s, the joint action a and the next state s'. The global immediate reward is the sum of each agent's immediate reward, and is denoted by $r = \sum_{i=1}^{n} r_i$. The goal of fully cooperative stochastic games is to maximize the following discounted cumulative reward at each time t

$$R(t) = r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \cdots = \sum_{k=0}^{K} \gamma^k r(t+k+1) \quad (1)$$

where γ is the discount factor and is used to weigh the importance of the future reward, K is the ending time of an episode, and $r(t+1)$ is the global immediate reward received at time $t+1$.

B. REPEATED GAMES

A repeated game is formed by a range of iterations of the same stage game. In each stage game, agents choose their own actions and a joint action is formed. According to the selected joint action, each agent will receive a local immediate reward. The global immediate reward is the sum of each agent's local immediate reward. In cooperative repeated games, each agent receives a global immediate reward in each stage game and optimizes its strategy to obtain the maximal global immediate reward. The strategy of an agent is pure if some action probability is one. Otherwise, the strategy is mixed. The joint strategy is pure if the strategy of each agent is pure. In this article, our aim is to obtain the optimal pure joint strategy for cooperative repeated games. The payoff matrix for a cooperative repeated game with two agents and three actions is shown in Fig.1. In the payoff matrix, each row represents an action of agent A, each column represents an action of agent B, and each numerical value represents a global immediate reward for both agents. If agent A selects the second action (the second row) and agent B selects the first action (the first column), both of them will receive a global immediate reward of 2. The goal of each game is to obtain the maximal global immediate reward marked with parentheses in Fig.1.

IV. WRFMR ALGORITHM FOR COOPERATIVE AGENTS

A. FORMULATION OF THE WRFMR ALGORITHM

The WRFMR algorithm is proposed to optimize performance indices of full collaboration tasks. For WRFMR, each agent needs to observe the states and the immediate rewards of the other agents. Each agent does not need to observe the actions of any other agent. The pseudo code of the WRFMR algorithm for repeated games is shown in Algorithm 1. Each

TABLE 3. Success rate in cooperative repeated games (runs = 100).

	2 agents	3 agents	4 agents	5 agents	6 agents	7 agents
2 actions	100%	100%	100%	100%	100%	100%
3 actions	100%	100%	100%	100%	100%	100%
4 actions	100%	100%	100%	100%	100%	100%
5 actions	100%	100%	100%	100%	100%	100%

	agent B		
agent A	**(6)**	2	1
	2	**(6)**	0
	1	3	5

FIGURE 1. The payoff matrix of a cooperative repeated game with two agents and three actions.

agent selects an action according to:

$$p_j^i(t+1) = \frac{e^{Q_j^i(t)/T}}{\sum_{l=1}^{|A_i|} e^{Q_l^i(t)/T}} \tag{2}$$

where $p_j^i(t+1)$ represents the probability of agent i selecting its j-th action, $Q_j^i(t)$ is the Q-value of the j-th action of agent i, $|A_i|$ is the number of agent $i's$ actions, and T is the temperature parameter. After each game, the frequency of obtaining the maximal immediate reward and the Q-function of each agent will be updated. The Q-value updating rule is as follows:

$$Q_j^i(t+1) = Q_j^i(t) + \alpha[(1-\beta)u_j^i(t) - \beta p_j^i(t)] \tag{3}$$

where $\alpha \in (0,1)$ is the learning rate, $\beta \in [0, 0.5)$ is the weight parameter, and $u_j^i(t)$ is the relative frequency of agent i selecting its j-th action. The weight parameter and the action probability are used to balance exploration and exploitation. The relative frequency $u_j^i(t)$ is defined as follows:

$$u_j^i(t) = \frac{f_j^i(t)}{\sum_{k=1}^{|A_i|} f_k^i(t)} \tag{4}$$

where $f_j^i(t)$ represents the frequency of obtaining the maximal global immediate reward when agent i selects its j-th action. The value of the frequency is small during the early learning stage, so the relative frequency is used to speed up the learning process. The frequency of obtaining the maximal global immediate reward is estimated according to

$$f_j^i(t+1) = \begin{cases} (1-\alpha_h)f_j^i(t) + \alpha_h & \text{if } r_j^i(t) > r_{i\_\max}(t) \\ (1-\alpha_l)f_j^i(t) + \alpha_l & \text{if } r_j^i(t) = r_{i\_\max}(t) \\ (1-\alpha_l)f_j^i(t) & \text{if } r_j^i(t) < r_{i\_\max}(t) \end{cases} \tag{5}$$

Algorithm 1 The WRFMR Algorithm for Repeated Games

```
Initialize Q_j^i(t), u_j^i(t), and f_j^i(t) for agent i to zero, for i = 1, 2, ..., n, j = 1, 2, ..., |A_i|.
repeat
  for each agent i do
    Select an action according to (2).
  end for
  for each agent i do
    Observe the reward r_i.
    if r_i ≥ r_{i_max} then
      r_{i_max} = r_i.
    end if
    for each action j do
      Evaluate u_j^i(t) and f_j^i(t) according to (4)-(6).
      Update Q_j^i(t) according to (3).
    end for
  end for
until the predefined number of games have been played
return Q-value function for each agent
```

for the selected action a_j^i at time step t and

$$f_g^i(t+1) = \begin{cases} (1-\alpha_l)f_g^i(t) & \text{if } r_j^i(t) > r_{i\_\max}(t) \\ f_g^i(t) & \text{if } r_j^i(t) \le r_{i\_\max}(t) \end{cases} \tag{6}$$

for each of the action a_g^i $(g \neq j)$ at time step t. Among the above, a_j^i represents the j-th action of agent i, α_h and $\alpha_l(\alpha_l < \alpha_h)$ are learning rates, $r_j^i(t)$ is the immediate reward when agent i selects a_j^i, and $r_{i\_\max}(t)$ is the maximal immediate reward obtained by agent i in history.

B. ANALYSIS OF THE WRFMR ALGORITHM

Theorem 1: In a cooperative repeated game with n $(n \ge 2)$ agents and m $(m \ge 2)$ optimal joint actions, each agent adopts the WRFMR algorithm. If the component action of every optimal joint action is unique, then all of the m optimal joint actions are asymptotically stable critical points.

Proof: Let p_{ij} denote the probability of agent i selecting its corresponding component action of the j-th optimal joint action $i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$, Q_{ij} denote the Q-value of agent i's corresponding component action of the j-th optimal joint action. According to (2) and (3), the probabilities of the component actions that can never obtain the maximal global reward will gradually decrease to zero. The Q-value updating process of the

WRFMR algorithm can be approximated by the following differential equations when the value of α is infinitely small.

$$\dot{Q}_{ij} = \alpha[(1-\beta)\frac{\prod_{k=1(k\neq i)}^{n} p_{kj}}{\sum_{w=1}^{m}(\prod_{k=1(k\neq i)}^{n} p_{kw})} - \beta p_{ij}]. \tag{7}$$

According to the total derivative formula, the model of WRFMR algorithm can be obtained as follows:

$$\begin{aligned}
&\frac{dp_{ij}}{dt} \\
&= \sum_{h=1}^{m} \frac{\partial p_{ij}}{\partial Q_{ih}} \frac{dQ_{ih}}{dt} \\
&= \frac{\frac{1}{T} e^{\frac{Q_{ij}}{T}} \sum_{k=1(k\neq j)}^{m} e^{\frac{Q_{ik}}{T}}}{(\sum_{l=1}^{m} e^{\frac{Q_{il}}{T}})^2} \cdot \alpha[(1-\beta)\frac{\prod_{k=1(k\neq i)}^{n} p_{kj}}{\sum_{w=1}^{m}(\prod_{k=1(k\neq i)}^{n} p_{kw})} - \beta p_{ij}] \\
&\quad + \sum_{h=1(h\neq j)}^{m} \frac{-e^{\frac{Q_{ij}}{T}} \cdot \frac{1}{T} e^{\frac{Q_{ih}}{T}}}{(\sum_{l=1}^{m} e^{\frac{Q_{il}}{T}})^2} \cdot \alpha[(1-\beta)\frac{\prod_{k=1(k\neq i)}^{n} p_{kh}}{\sum_{w=1}^{m}(\prod_{k=1(k\neq i)}^{n} p_{kw})} - \beta p_{ih}] \\
&= \frac{1}{T}\alpha p_{ij}[(1-p_{ij})((1-\beta)\frac{\prod_{k=1(k\neq i)}^{n} p_{kj}}{\sum_{w=1}^{m}(\prod_{k=1(k\neq i)}^{n} p_{kw})} - \beta p_{ij}) \\
&\quad - \sum_{h=1(h\neq j)}^{m} p_{ih}((1-\beta)\frac{\prod_{k=1(k\neq i)}^{n} p_{kh}}{\sum_{w=1}^{m}(\prod_{k=1(k\neq i)}^{n} p_{kw})} - \beta p_{ih})].
\end{aligned} \tag{8}$$

Any critical point of the system described by (8) must satisfy:

$$\begin{aligned}
&\frac{1}{T} p_{ij}[(1-p_{ij})((1-\beta)\frac{\prod_{k=1(k\neq i)}^{n} p_{kj}}{\sum_{w=1}^{m}(\prod_{k=1(k\neq i)}^{n} p_{kw})} - \beta p_{ij}) - \sum_{h=1(h\neq j)}^{m} p_{ih} \\
&\qquad ((1-\beta)\frac{\prod_{k=1(k\neq i)}^{n} p_{kh}}{\sum_{w=1}^{m}(\prod_{k=1(k\neq i)}^{n} p_{kw})} - \beta p_{ih})] = 0.
\end{aligned} \tag{9}$$

It is obvious that any pure joint strategy is a critical point. By performing the following transformation:

$$p_{ij} = \begin{cases} \bar{p}_{ij} & \text{if } j \neq m \\ 1 - \sum_{l=1}^{m-1} \bar{p}_{il} & \text{if } j = m \,. \end{cases} \tag{10}$$

We obtain the following model from:

$$\begin{aligned}
&\frac{1}{T}\bar{p}_{ij}[(1-\bar{p}_{ij}) \\
&\quad (\frac{(1-\beta)\prod_{k=1(k\neq i)}^{n} \bar{p}_{kj}}{\sum_{w=1}^{m-1}(\prod_{k=1(k\neq i)}^{n} \bar{p}_{kw}) + \prod_{k=1(k\neq i)}^{n}(1-\sum_{l=1}^{m-1} \bar{p}_{kl})} - \beta\bar{p}_{ij}) \\
&\quad - \sum_{h=1(h\neq j)}^{m-1}(\bar{p}_{ih}(\frac{(1-\beta)\prod_{k=1(k\neq i)}^{n} \bar{p}_{kh}}{\sum_{w=1}^{m-1}(\prod_{k=1(k\neq i)}^{n} \bar{p}_{kw}) + \prod_{k=1(k\neq i)}^{n}(1-\sum_{l=1}^{m-1} \bar{p}_{kl})} - \beta\bar{p}_{ih})) \\
&\quad - (1-\sum_{l=1}^{m-1} \bar{p}_{il})(\frac{(1-\beta)\prod_{k=1(k\neq i)}^{n}(1-\sum_{l=1}^{m-1} \bar{p}_{kl})}{\sum_{w=1}^{m-1}(\prod_{k=1(k\neq i)}^{n} \bar{p}_{kw}) + \prod_{k=1(k\neq i)}^{n}(1-\sum_{l=1}^{m-1} \bar{p}_{kl})} \\
&\quad - \beta(1-\sum_{l=1}^{m-1} \bar{p}_{il}))] = 0 \\
&\quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m-1.
\end{aligned} \tag{11}$$

The Jacobin matrix $J \in R^{(m-1)n\times(m-1)n}$ corresponding to any of the m optimal joint actions is as follows:

$$J = \begin{bmatrix} \frac{1}{T}(2\beta-1) & 0 & \cdots & 0 \\ 0 & \frac{1}{T}(2\beta-1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{T}(2\beta-1) \end{bmatrix}. \tag{12}$$

It can be seen that all the eigenvalues of J are $(2\beta-1)/T$, since β is strictly less than 0.5, each of the m optimal joint actions is an asymptotically stable critical point.

C. EMPIRICAL STUDIES ON REPEATED GAMES

The convergence of the WRFMR algorithm in repeated games is verified experimentally in this section. The aim is to obtain the maximum global reward in repeated games with $n = 2, 3, 4, 5, 6, 7$ agents and $m = 2, 3, 4, 5$ actions. The results are averaged on 100 runs. The payoff matrix is randomly generated for each run under the assumption of Theorem 1. If each agent converges to a pure strategy and the joint strategy obtains the maximal global immediate reward, this run is successful. Each agent's strategy is considered to be a pure strategy if the probability of choosing some action is no less than 0.999. The WRFMR algorithm uses the parameters $T = 10$, $\alpha = 0.01$, $\alpha_l = 0.1$, $\alpha_h = 0.6$, and $\beta = 0.1$. As shown in Table 3, the success rate is 100% in all cases. The WRFMR algorithm converges to one of the optimal joint actions for all values of n and m. The simulation results are consistent with Theorem 1.

Algorithm 2 The WRFMR Algorithm for Stochastic Games

Initialize $Q_j^i(s)$, $u_j^i(s)$, and $f_j^i(s)$for agent i to zero, for $i = 1, 2, \ldots, n, j = 1, 2, \ldots, |Ai|$.
repeat
  repeat
    for each agent i **do**
      Select an action according to (13).
    end for
    for each agent i **do**
      Observe the next state s' and the immediate reward r_i.
      Record the tuple $< s, a_i, s', r_i >$.
    end for
  until the episode is ended
  for each agent i **do**
    for each visited state s in the last episode **do**
      Evaluate $R_i(s)$ by the recorded tuples and (1).
      if $R_i(s) \geq R_{i\_\max}(s)$ **then**
        $R_{i\_\max}(s) = R_i(s)$
      end if
      for each action j **do**
        Evaluate $u_j^i(s)$ and $f_j^i(s)$ according to (15)-(17).
        Update $Q_j^i(s)$ according to (14).
      end for
    end for
  end for
until the predefined number of episodes have been played
return Q-value function for each agent

D. WRFMR ALGORITHM FOR STOCHASTIC GAMES

The WRFMR algorithm can be extended to solve cooperative stochastic games. The pseudo-code is shown in Algorithm 2. Each agent selects an action according to:

$$p_j^i(s) = \frac{e^{Q_j^i(s)/T}}{\sum_{l=1}^{|A_i|} e^{Q_l^i(s)/T}} \tag{13}$$

where $p_j^i(s)$ represents the probability of agent i selecting its j-th action at state s, $Q_j^i(s)$ is the Q-value of the j-th action of agent i at state s. When an episode ends, the cumulative global reward in each visited state is evaluated by (1). Then the frequency of obtaining the maximal cumulative global reward and the Q-value of each visited state action pair (s, a_i) for $i = 1, 2, \ldots, n$ are updated. The Q-value updating rule is as follows:

$$Q_j^i(s) = Q_j^i(s) + \alpha[(1-\beta)u_j^i(s) - \beta p_j^i(s)] \tag{14}$$

where $u_j^i(s)$represents the relative frequency of agent i selecting its j-th action at state s, and is defined as follows:

$$u_j^i(s) = \frac{f_j^i(s)}{\sum_{k=1}^{|A_i|} f_k^i(s)} \tag{15}$$

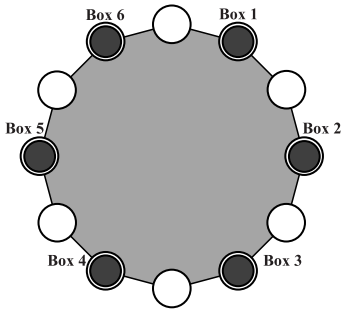

FIGURE 2. The 6-agent-12-vertex box-pushing task.

where $f_j^i(s)$ represents the frequency of obtaining the maximal cumulative reward when agent i selects its j-th action at state s. It is estimated according to

$$f_j^i(s) = \begin{cases} (1-\alpha_h)f_j^i(s) + \alpha_h & \text{if } R_j^i(s) > R_{i\_\max}(s) \\ (1-\alpha_l)f_j^i(s) + \alpha_l & \text{if } R_j^i(s) = R_{i\_\max}(s) \\ (1-\alpha_l)f_j^i(s) & \text{if } R_j^i(s) < R_{i\_\max}(s) \end{cases} \tag{16}$$

for the selected action a_j^i at state s and

$$f_g^i(s) = \begin{cases} (1-\alpha_l)f_g^i(s) & \text{if } R_j^i(s) > R_{i\_\max}(s) \\ f_g^i(s) & \text{if } R_j^i(s) \leq R_{i\_\max}(s) \end{cases} \tag{17}$$

for each of the action a_g^i $(g \neq j)$ at state s. Among the above, $R_j^i(s)$ is the cumulative global reward when agent i selects a_j^i at state s, and $R_{i\_\max}(s)$ is the maximal cumulative reward obtained by agent i at state s in history.

V. EMPIRICAL STUDIES FOR COOPERATIVE TASKS

In this section, the efficacy of the WRFMR algorithm is studied in three cooperative tasks – the box-pushing task, the DSN task, and a game known as blood battlefield. The differences of these tasks are as follows. First, the box-pushing task is a stochastic game with a deterministic transition function, while both the DSN task and blood battlefield are stochastic games with a probabilistic transition function. Second, the box-pushing task and the DSN task involve only cooperation, while the blood battlefield comprises both cooperation and competition.

A. TASK 1: BOX-PUSHING

The box-pushing task [37] is shown in Fig.2. Six boxes are allocated in six vertices of a polygon with a total number of 12 vertices. Each agent is responsible for pushing one box. Each agent can select to push the box to one of its adjacent vertices or stay still. The goal of this task is to coordinate the six agents to distribute the boxes evenly. At the beginning of each episode, the boxes locate in random positions. The state variables include five relative positions to box 1. The number of states is $C_{11}^5 \times 5! = 55440$. If all boxes are distributed evenly, each agent obtains a reward of 10, otherwise, each agent obtains a reward of -1.

The rules of the box-pushing task are as follows. First, all agents push the boxes simultaneously. Second, if two or more boxes collide with each other, they will stay still. Otherwise, they move successfully. A collision occurs when a box moves to another box that chooses to stay still or has failed to move, two boxes move to the same positions, or two adjacent boxes move towards each other.

FMRQ [22], WoLF-PHC [24], EMA Q-learning [28], and EAQR [12] are selected as comparison algorithms. The results are averaged on 100 runs and each run includes L learning episodes and 300,000 evaluation episodes. For the WRFMR algorithm, the discount factor γ is set to 0.9, the learning rates $\alpha = 0.01$, $\alpha_l = 0.15$, $\alpha_h = 0.6$, and the temperature parameter T follows:

$$T = \begin{cases} 3 & 1 \le n < 0.4L \\ 1 & 0.4L \le n \le L \end{cases}$$

where n represents the current episodes, and L represents the total number of episodes. The weight parameter β is as follows:

$$\beta = \begin{cases} 0.4 & 1 \le n < 0.2L \\ 0.1 & 0.2L \le n < 0.4L \\ 0 & 0.4L \le n \le L. \end{cases}$$

For EMA Q-learning, $\gamma = 0.9$, $\eta_w = 0.1$, $\eta_l = 0.001\eta_w$, $\varepsilon = 0.6$, and α follows:

$$\alpha = \alpha_{ini} - \frac{\alpha_{ini} n}{1.05L} \tag{18}$$

where $\alpha_{ini} = 0.2$. For WoLF-PHC, $\gamma = 0.9$, $\delta_w = 0.003$, $\delta_l = 0.01$, $\varepsilon = 0.1$, and α follows (18) with $\alpha_{ini} = 0.6$. For FMRQ $\alpha = 0.5$, $\gamma = 1.0$, and T follows:

$$T = \begin{cases} 0.7 & 1 \le n < L/6 \\ 0.6 & L/6 \le n < L/5 \\ 0.5 & L/5 \le n < L/3 \\ 0.2 & L/3 \le n < L/2 \\ 0.1 & L/2 \le n < 0.8L \\ 0.05 & 0.8L \le n \le L. \end{cases}$$

| For EAQR, $\gamma = 0.9$, $\alpha = 0.7$, and ε follows:

$$\varepsilon = \begin{cases} 0.9 & 1 \le n < 0.2L \\ 0.8 & 0.2L \le n < 0.4L \\ 0.7 & 0.4L \le n < 0.6L \\ 0.6 & 0.6L \le n < 0.8L \\ 0.5 & 0.8L \le n \le L. \end{cases}$$

The performance metrics include the average success rate, the average number of steps, and the standard deviation. A successful episode uses minimal steps to complete the task. We develop a program to obtain the actual minimal number of steps in an episode. The average number of steps and the standard deviation are shown in Table 4. Take the WRFMR algorithm and $L = 5{,}000{,}000$ for example, 1.55 | 0.01 represents an average number of steps of 1.55 and a standard deviation of 0.01. The performance of all algorithms improves as L increases. The WRFMR algorithm and WoLF-PHC perform well for all different values of L. WoLF-PHC uses the smallest number of average steps and standard deviation among all algorithms when L is 5,000,000. As L increases, the WRFMR algorithm outperforms the WoLF-PHC in terms of average steps. After 10,000,000 learning episodes, the WRFMR algorithm uses 1.48 average steps per episode. The average steps per episode in box-pushing task are shown in Fig.3. In the initial learning stage, the learning speed of WRFMR is lower than that of WoLF-PHC, EMA and EAQR. However, after four million episodes, only WRFMR continuously improves its performance. The average steps for WRFMR drop off at 4000000-th learning episode, because the joint strategy becomes more greedy as T varies at that episode.

TABLE 4. Average steps and standard deviation for the 6-agent-12-vertex box pushing task (runs = 100).

	L=5,000,000	L=7,500,000	L=10,000,000
WRFMR	**1.55 | 0.011**	**1.49 | 0.005**	**1.48 | 0.003**
FMRQ	1.56 | 0.012	1.52 | 0.009	1.50 | 0.006
EAQR	1.57 | 0.016	1.53 | 0.013	1.50 | 0.009
WoLF-PHC	1.53 | 0.004	1.50 | 0.003	1.49 | 0.003
EMA Q-learning	1.88 | 0.023	1.76 | 0.020	1.68 | 0.015

TABLE 5. Success rate for the 6-agent-12-vertex box pushing task (runs = 100).

	L=5,000,000	L=7,500,000	L=10,000,000
WRFMR	**93.52%**	**96.86%**	**98.30%**
FMRQ	94.70%	95.90%	96.71%
EAQR	93.47%	95.58%	96.73%
WoLF-PHC	92.74%	95.72%	96.86%
EMA Q-learning	71.18%	79.45%	85.13%

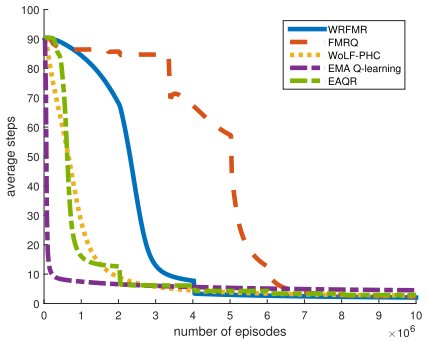

FIGURE 3. Learning performance on average steps per episode in box-pushing (runs = 100).

Table 5 shows the success rate. The success rates of FMRQ with $L = 5{,}000{,}000$ is higher than the success rate of other algorithms. The WRFMR algorithm has the highest learning speed. It obtains the highest success rate when $L = 10{,}000{,}000$. EMA Q-learning obtains the lowest success rate for all values of L.

B. TASK 2: DISTRIBUTED SENSOR NETWORK

The goal of the DSN [38] task is to coordinate the sensors to capture two targets in minimal time steps. Fig.4 shows a DSN

TABLE 6. The action sets of the twelve sensors.

SENSOR ID	AVAILABLE ACTIONS
Sensor 1	focus on cell 1 and no focus
Sensor 2	focus on cell 1, cell 2, and no focus
Sensor 3	focus on cell 2, cell 3, and no focus
Sensor 4	focus on cell 3 and no focus
Sensor 5	focus on cell 1, cell 4, and no focus
Sensor 6	focus on cell 1, cell 2, cell 4, cell 5, and no focus
Sensor 7	focus on cell 2, cell 3, cell 5, cell 6, and no focus
Sensor 8	focus on cell 3, cell 6, and no focus
Sensor 9	focus on cell 4 and no focus
Sensor 10	focus on cell 4, cell 5, and no focus
Sensor 11	focus on cell 5, cell 6, and no focus
Sensor 12	focus on cell 6 and no focus

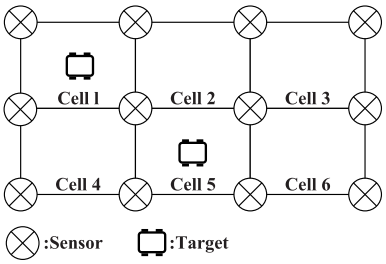

FIGURE 4. The DSN task with 12 sensors and 2 targets.

with twelve sensors and two targets. Each sensor is viewed as an agent. The action set of each sensor is shown in Table 6. The number of joint actions is 291,600. At each step, each target has equal probability to move to one of four directions (up, down, left, and right), or stay still. If a target tries to move out of the grid or move to a cell that has been occupied by another target, it fails and stays still. Each cell can be occupied by at most one target.

At the beginning of an episode, each target has an energy value of three. The energy of a target is reduced by one if the target is focused by at least three sensors. If a target's energy is decreased to zero, it is captured and wiped out from the cells. The state variables contain the number of uncaptured targets and the positions of both the targets. The state space contains 43 elements. If both the targets are captured or 300-time steps have elapsed, an episode ends.

The reward function is defined as follows. If a target is captured, each of the three sensors involved in the capture is rewarded by 10. If four sensors capture the target, the sensors with the largest indexes receive a reward of 0. The action of focus produces an immediate reward of -1, and no focus produces an immediate reward of 0. For the DSN task, the actual maximal cumulative reward is 42, and the minimal number of time steps is 3. A success is obtained if a cumulative reward of 42 is obtained in an evaluation episode.

The results are averaged on 100 runs, and each run includes L learning episodes and 50,000 evaluation episodes. The WRFMR algorithm uses the parameters $\gamma = 0.9$, $\alpha = 0.01$, $\alpha_l = 0.1$, $\alpha_h = 0.8$, $T = 8$, and β follows:

$$\beta = \begin{cases} 0.1 & 1 \le n < 0.2L \\ 0 & 0.2L \le n \le L. \end{cases}$$

For FMRQ, $\gamma = 0.9$, $\alpha = 0.2$, T follows:

$$T = \begin{cases} 0.02 & 1 \le n < 0.2L \\ 0.015 & 0.2L \le n < 0.4L \\ 0.01 & 0.4L \le n < 0.6L \\ 0.005 & 0.6L \le n < 0.8L \\ 0.002 & 0.8L \le n \le L. \end{cases}$$

For EMA Q-learning, $\gamma = 0.9$, $\eta_w = 0.1$, $\eta_l = 0.001\eta_w$, $k = 2$, $\varepsilon = 0.2$, and α follows (18) with $\alpha_{ini} = 0.7$. For WoLF-PHC, $\gamma = 0.9$, $\delta_w = 0.003$, $\delta_l = 0.01$, $\varepsilon = 0.2$, and α follows (18) with $\alpha_{ini} = 0.7$. For EAQR, $\gamma = 0.9$, $\alpha = 0.2$, α follows (18), and ε follows:

$$\varepsilon = \begin{cases} 0.9 & 1 \le n < 0.2L \\ 0.8 & 0.2L \le n < 0.4L \\ 0.7 & 0.4L \le n < 0.6L \\ 0.6 & 0.6L \le n < 0.8L \\ 0.5 & 0.8L \le n \le L. \end{cases}$$

Table 7 shows the success rate. Compared with the other algorithms, the WRFMR algorithm obtains a higher success rate. After 3,000,000 episodes, the WRFMR algorithm obtains a success rate of 100%. The success rate of FMRQ rises as L increases, but it learns slower than the WRFMR algorithm. Besides, the learning speed is a great advantage of the WRFMR algorithm. The average number of steps per episode in the DSN task is shown in Fig.5. We can see that WRFMR converges more quickly than other algorithms. The average steps for WRFMR did not drop off at 4000000-th learning episode, because the optimal joint strategy has been obtained before T varies.

The average cumulative reward is shown in Table 8. All the algorithms except WoLF-PHC obtain more cumulative reward as L increases. The WRFMR algorithm is the only one that obtains the optimal average cumulative reward of 42, which is consistent with the results of the success rate and the worst case is presented in Table 9.

The average number of steps is presented in Table 10, and the worst case is presented in Table 11. The WRFMR algorithm overwhelms the other algorithms in terms of the steps. After 3,000,000 episodes, the WRFMR algorithm obtains 3.01 steps to capture both the targets.

To verify the statistical significance of the results in Table 8 and Table 10, we conduct a one-way analysis of variance (ANOVA). As shown in Table 12, p-values are less than 0.05, and F is greater than F-critical, which indicates that there a statistically significant difference between the results of all groups. In addition, we conduct a one-tailed t-test at 0.05 significance level for average steps and average cumulative reward. The p-value for t-test between WRFMR and each of the other algorithms is presented in Table 13. A p-value

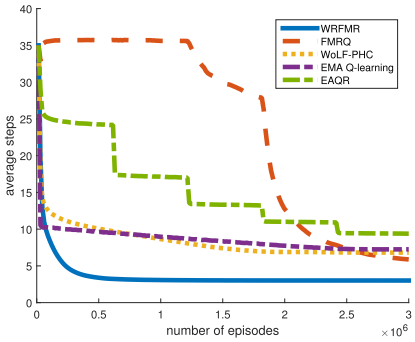

FIGURE 5. Learning performance on average steps per episode in DSN (runs = 100).

TABLE 7. Success rate for the DSN task (runs = 100).

	L=1,000,000	L=2,000,000	L=3,000,000
WRFMR	**98.48%**	**99.55%**	**100%**
FMRQ	36.77%	55.63%	70.66%
EAQR	63.17%	69.39%	70.87%
WoLF-PHC	13.25%	12.87%	12.05%
EMA Q-learning	0.16%	0.32%	0.62%

TABLE 8. Average cumulative reward and standard deviation for the DSN task (runs = 100).

	L=1,000,000	L=2,000,000	L=3,000,000
WRFMR	**41.96 | 0.10**	**41.99 | 0.06**	**42 | 0**
FMRQ	40.00 | 0.89	40.50 | 0.87	41.05 | 0.71
EAQR	41.85 | 0.12	41.97 | 0.05	41.99 | 0.02
WoLF-PHC	39.59 | 0.63	39.52 | 0.65	39.42 | 0.73
EMA Q-learning	33.78 | 1.28	34.98 | 1.12	35.49 | 1.17

TABLE 9. Minimal cumulative reward for the DSN task (runs = 100).

	L=1,000,000	L=2,000,000	L=3,000,000
WRFMR	**41.51**	**41.70**	**42**
FMRQ	36.41	37.38	37.92
EAQR	41.50	41.79	41.87
WoLF-PHC	37.50	37.98	37.60
EMA Q-learning	30.78	32.18	32.21

less than 0.05 indicates that there a statistically significant difference between the WRFMR algorithm and the other algorithms.

The comparison MARL algorithms did not obtain the best performance for several reasons. FMRQ and EAQR learn slowly because it consumes too much time to obtain the estimate of the probability of obtaining the maximum reward. Its poor performance in average steps might be because the punishment induced by the discount factor is not enough. As for WoLf-PHC and EMA Q-learning, they might converge to the NE at each stage, but the NE is not the optimal one that obtains the maximal cumulative reward.

C. TASK 3: BLOOD BATTLEFIELD

Blood battlefield is a strategic game that involves both cooperation and competition. In this game, each player commands a team to fight against another player who commands another team. Each team has two gunners and four riflemen. The goal is to eliminate all the units of the opponent team and survive the battle.

TABLE 10. Average steps and standard deviation for the DSN task (runs = 100).

	L=1,000,000	L=2,000,000	L=3,000,000
WRFMR	**3.01 | 0.04**	**3.01 | 0.03**	**3.01 | 0.03**
FMRQ	6.17 | 1.02	6.48 | 1.03	6.03 | 0.73
EAQR	3.50 | 0.20	3.45 | 0.19	3.43 | 0.18
WoLF-PHC	5.73 | 0.41	5.67 | 0.45	5.63 | 0.43
EMA Q-learning	6.00 | 0.03	6.00 | 0.03	6.00 | 0.02

TABLE 11. Maximal steps for the DSN task (runs = 100).

	L=1,000,000	L=2,000,000	L=3,000,000
WRFMR	**3.20**	**3.20**	**3.20**
FMRQ	8.91	10.90	8.11
EAQR	4.07	3.98	3.83
WoLF-PHC	6.24	6.13	6.13
EMA Q-learning	6.13	6.13	6.12

The property values of both the units are presented in Table 14. A live unit can choose to attack only one live opponent unit at each step. The damage caused by a unit is determined by the unit's AD (attack damage) and HR (hit rate). All units of both sides act simultaneously at each step. If the HP (hit point) value of a unit is below zero, the unit is wiped out. The player wins the battle if at least one unit survives and all units of the other player are eliminated. A tie happens if all units of both sides are eliminated at the same step. An episode ends if one player wins the game, a tie happens, or 100 time steps have elapsed.

The state vector is expressed as $\mathbf{s} = [h_1, \ldots h_6, k_1, \ldots k_6]^{\mathrm{T}}$, where h_i represents the HP of the i-th opponent unit, and $k_l \in$ {live,dead}represents the state of its l-th teammate. The number of states is $4^4 \times 7^2 \times 2^6 = 802,816$. The number of available actions for each unit depends on the number of live opponent units. The reward function is described as follows: if a unit is eliminated, the opponent player receives a reward of 2; when an episode ends, the winner receives a reward of 10 while the loser receives a reward of 10. If a tie occurs or 100 time steps have elapsed, both players receive a reward of 0. The WRFMR algorithm uses the parameters $\alpha = 0.01$, $\alpha_l = 0.05$, $\alpha_h = 0.8$, $T = 0.5$, $\gamma = 1$, and β follows:

$$\beta = \begin{cases} 0.4 & 1 \le n < 0.2L \\ 0 & 0.2L \le n \le L \end{cases}$$

FMRQ and EAQR use more memory than our computing resources and cannot perform this task on our computer, so the approximate frequency of obtaining maximal cumulative reward is also used for FMRQ and EAQR to avoid recording large amounts of data. For FMRQ, $\alpha = 0.1$, $\alpha_l = 0.1$, $\alpha_h = 0.8$, $T = 0.6$, and $\gamma = 1$. For EMA Q-learning, $\alpha = 0.1$, $\varepsilon = 0.2$, $\eta_w = 0.1$, $\eta_l = 0.001\eta_w$, and $\gamma = 0.9$. For WoLF-PHC, $\delta_w = 0.003$, $\delta_l = 0.01$, $\gamma = 0.9$, $\alpha = 0.1$, and

TABLE 12. Results of one-way ANOVA analysis for average steps and average cumulative reward for DSN task.

	P-value	F	F-critical
Average steps	1.0562e-271	1449.99	4.40
Average cumulative reward	1.4816e-278	1553.63	4.40

TABLE 13. Results of one-tailed T-Test (0.05 significance) for average steps and average cumulative reward for the DSN task.

	WRFMR vs FMRQ	WRFMR vs WoLF-PHC	WRFMR vs EMA	WRFMR vs EAQR
Average steps	1.6842e-99	1.9557e-129	0	2.7870e-60
Average cumulative reward	1.1005e-29	1.0568e-87	4.7307e-123	0.0132

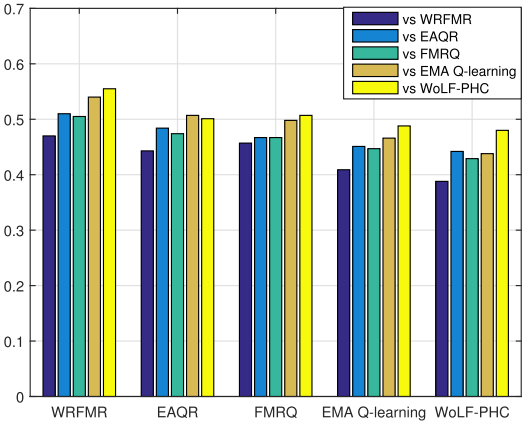

FIGURE 6. Win rate of each algorithm (after 1,000,000 learning episodes).

TABLE 14. Properties of units in blood battlefield.

	AD	HP	HR
riflemen	1	3	0.9
gunner	2	6	0.7

$\varepsilon = 0.2$. For EAQR, $\gamma = 1$, $\alpha = 0.1$, $\alpha_l = 0.1$, $\alpha_h = 0.6$, and $\varepsilon = 0.3$.

The results are averaged on 100 runs, and each run includes 1,000,000 learning episodes and 1,000,000 evaluation episodes. Fig.6 shows the win rates of four algorithms against each other. The ties are not considered. The WRFMR algorithm gains the highest win rate against FMRQ, EMA Q-learning, WoLF-PHC, and EAQR. The WRFMR algorithm performs slightly better than FMRQ. However, its win rate against FMRQ is apparently higher than the win rate of FMRQ against it. The win rates of both EMA Q-learning and WoLF-PHC are not high because the Nash equilibrium is not the optimal strategy in this game.

To verify the efficacy of the WRFMR algorithm, we visualize a match between it (team 1) and FMRQ (team 2). As shown in Fig.7, the actions of the units are indicated by arrows. The arrow with a solid line represents a hit from team 1, and the arrow with a dotted line represents a hit from team 2. For example, the leftmost rifleman of team 2 is attacked by two riflemen and a gunner of team 1 at step 0, and only one rifleman hits successfully. Thus the true damage taking by the target is $0 + 0 + 2 = 2$. The target's HP is

FIGURE 7. A game played by the WRFMR algorithm (team 1) and the FMRQ algorithm (team 2) (after 1,000,000 learning episodes).

reduced by 2. At each step, several units of team 1 concentrate fire to attack the same opponent unit. By contrast, the fire of team 2 is scattered on more opponent units. Team 2 is outnumbered after step 2, and is finally defeated by team 1 at step 5. The WRFMR algorithm performs well in all three stochastic games, which indicates that the WRFMR algorithm can converge to one of the optimal cumulative global rewards in many cases.

VI. CONCLUSION

This article aims to solve the coordination problem in fully cooperative MASs. We propose the WRFMR algorithm to obtain the maximal global reward. The decreasing weight parameter and the action probability are used to balance exploration and exploitation to improve the learning speed. Theoretical analysis on the model of the WRFMR algorithm for cooperative repeated games indicates that if the

component action of every optimal joint action is unique, then all optimal joint actions are asymptotically stable critical points. The efficacy of the WRFMR algorithm is also studied empirically through three cooperative tasks. The results show that the WRFMR algorithm performs better than other algorithms in the box-pushing task, the DSN task, and blood battlefield in terms of the success rate and the learning speed. In the future, the convergence of the algorithm will be further studied.

REFERENCES

[1] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.
[2] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *Knowl. Eng. Rev.*, vol. 27, no. 1, pp. 1–31, Feb. 2012.
[3] A. R. Tavares and A. L. C. Bazzan, "Independent learners in abstract traffic scenarios," *RITA*, vol. 19, no. 2, pp. 13–33, 2012.
[4] Z. Kaifu, "Learn to coordinate with generic non-stationary opponents," in *Proc. 5th IEEE Int. Conf. Cognit. Informat.*, Beijing, China, vol. 1, Jul. 2006, pp. 558–565.
[5] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. ICML*, New Brunswick, NJ, USA, vol. 157, 1994, pp. 157–163.
[6] K. G. Vamvoudakis, F. L. Lewis, and G. R. Hudas, "Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality," *Automatica*, vol. 48, no. 8, pp. 1598–1611, 2012.
[7] D. Zhao, Q. Zhang, D. Wang, and Y. Zhu, "Experience replay for optimal control of nonzero-sum game systems with unknown dynamics," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 854–865, Mar. 2016.
[8] H. Kazmi, J. Suykens, A. Balint, and J. Driesen, "Multi-agent reinforcement learning for modeling and control of thermostatically controlled loads," *Appl. Energy*, vol. 238, pp. 1022–1035, Mar. 2019.
[9] H. T. Rauf, S. Malik, U. Shoaib, M. N. Irfan, and M. I. Lali, "Adaptive inertia weight bat algorithm with sugeno-function fuzzy search," *Appl. Soft Comput.*, vol. 90, May 2020, Art. no. 106159.
[10] R. Cui, C. Yang, Y. Li, and S. Sharma, "Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 6, pp. 1019–1029, Jun. 2017.
[11] Z. Zhang and D. Zhao, "Clique-based cooperative multiagent reinforcement learning using factor graphs," *IEEE/CAA J. Automatica Sinica*, vol. 1, no. 3, pp. 248–256, Jul. 2014.
[12] Z. Zhang and D. Wang, "EAQR: A multiagent Q-Learning algorithm for coordination of multiple agents," *Complexity*, vol. 2018, pp. 1–14, Aug. 2018.
[13] V. Conitzer and T. Sandholm, "AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents," *Mach. Learn.*, vol. 67, nos. 1–2, pp. 23–43, May 2007.
[14] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. IAT*, Halifax, Canada, 1994, pp. 336–342.
[15] B. Banerjee and J. Peng, "Adaptive policy gradient in multiagent learning," in *Proc. 2nd Int. Joint Conf. Auto. Agents Multiagent Syst.*, Melbourne, VIC, Australia, 2003, pp. 686–692.
[16] N. Suematsu and A. Hayashi, "A multiagent reinforcement learning algorithm using extended optimal response," in *Proc. 1st Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, Bologna, Italy, 2002, pp. 370–377.
[17] M. L. Littman, "Value-function reinforcement learning in Markov games," *Cognit. Syst. Res.*, vol. 2, no. 1, pp. 55–66, Apr. 2001.
[18] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proc. AAAI*, Menlo Park, CA, USA, 1998, pp. 746–752.
[19] V. Kononen, "Asymmetric multiagent reinforcement learning," in *Proc. IEEE/WIC Int. Conf. Intell. Agent Technol.*, Vancouver, BC, Canada, 2003, pp. 1603–1610.
[20] Z. Zhang, D. Wang, D. Zhao, Q. Han, and T. Song, "A gradient-based reinforcement learning algorithm for multiple cooperative agents," *IEEE Access*, vol. 6, pp. 70223–70235, 2018.
[21] B. H. Abed-alguni, S. K. Chalup, F. A. Henskens, and D. J. Paul, "A multi-agent cooperative reinforcement learning model using a hierarchy of consultants, tutors and workers," *Vietnam J. Comput. Sci.*, vol. 2, no. 4, pp. 213–226, Nov. 2015.
[22] Z. Zhang, D. Zhao, J. Gao, D. Wang, and Y. Dai, "FMRQ—A Multiagent reinforcement learning algorithm for fully cooperative tasks," *IEEE Trans. Cybern.*, vol. 47, no. 6, pp. 1367–1379, Apr. 2017.
[23] S. P. Singh, M. J. Kearns, and Y. Mansour, "Nash convergence of gradient dynamics in general-sum games," in *Proc. UAI*, Stanford, CA, USA, 2000, pp. 541–548.
[24] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artif. Intell.*, vol. 136, no. 2, pp. 215–250, Apr. 2002.
[25] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. ICML*, Washington, DC, USA, 2003, pp. 928–935.
[26] M. Bowling, "Convergence and no-regret in multiagent learning," in *Proc. NIPS*, Vancouver, BC, Canada, 2004, pp. 209–216.
[27] D. Chakraborty and P. Stone, "Convergence, targeted optimality, and safety in multiagent learning," in *Proc. ICML*, Haifa, Israel, 2010, pp. 215–250.
[28] M. D. Awheda and H. M. Schwartz, "Exponential moving average based multiagent reinforcement learning algorithms," *Artif. Intell. Rev.*, vol. 45, no. 3, pp. 299–332, Mar. 2016.
[29] C. Zhang and V. Lesser, "Multi-agent learning with policy prediction," in *Proc. AAAI*, Atlanta, GA, USA, 2010, pp. 927–934.
[30] J. W. Crandall and M. A. Goodrich, "Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning," *Mach. Learn.*, vol. 82, no. 3, pp. 281–314, Mar. 2011.
[31] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, Los Angeles, CA, USA, 2017, pp. 6379–6390.
[32] P. Sunehag, G. Lever, A. Gruslys, W. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning," in *Proc. AAAI*, Stockholm, Sweden, 2017, pp. 2085–2087.
[33] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. ICML*, Stockholm, Sweden, 2018, pp. 1–14.
[34] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," in *Proc. AAMAS*, Stockholm, Sweden, 2018, pp. 1–9.
[35] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI*, San Francisco, CA, USA, 2017, pp. 1–10.
[36] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. ICML*, Sydney, NSW, Australia, 2017, pp. 1–10.
[37] G. Chen, W. Cao, X. Chen, and M. Wu, "Multi-agent Q-learning with joint state value approximation," in *Proc. CCC*, Yantai, China, 2011, pp. 4878–4882.
[38] S. Ali, S. Koenig, and M. Tambe, "Preprocessing techniques for accelerating the DCOP algorithm ADOPT," in *Proc. 4th Int. Joint Conf. Auto. Agents Multiagent Syst.*, Utrecht, The Netherlands, 2005, pp. 1041–1048.

HUI LIU received the B.S. degree from the School of Mechanical and Electrical Engineering, Dezhou University, Dezhou, China, in 2018. He is currently pursuing the M.S. degree with the School of Automation, Qingdao University, Qingdao, China. His research interests include reinforcement learning, multi-agent reinforcement learning, and game theory.

ZHEN ZHANG received the B.S. degree from the China University of Petroleum, Dongying, China, in 2006, the M.S. degree in control theory and control engineering from the Dalian University of Technology, Dalian, China, in 2009, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2013. He has been with the School of Automation, Qingdao University, Qingdao, China, since 2013. He was a Visiting Scholar with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, from 2018 to 2019. He has been an Associate Professor with the School of Automation, Qingdao University, since 2019. His current research interests include reinforcement learning and intelligent optimization methods.

DONGQING WANG received the Ph.D. degree from the College of Automation Engineering, Tianjin University, Tianjin, China, in 2006. She was with the School of Electrical Engineering, Qingdao University, Qingdao, China, since 1988. She was a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Tennessee, Knoxville, USA, from 2004 to 2005. She has been a Professor with the School of Electrical Engineering, Qingdao University, since 2011. Her research interests are stochastic systems, system identification, and system modeling and control.