Received August 9, 2021, accepted August 30, 2021, date of publication September 3, 2021, date of current version September 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3110255

A Cooperative Multi-Agent Reinforcement Learning Method Based on Coordination Degree

HAOYAN CUI AND ZHEN ZHANG
Shandong Key Laboratory of Industrial Control Technology, School of Automation, 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, in part by Qingdao Postdoctoral Applied Research Project with Name (AGV Road Network Design and Path Planning Method Based on Multi-Agent Reinforcement Learning), 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) has become a prevalent method for solving cooperative problems owing to its tractable implementation and task distribution. The goal of the MARL algorithms for fully cooperative scenarios is to obtain the optimal joint strategy that maximizes the expected common cumulative reward for all agents. However, to date, the analysis of MARL dynamics has focused on repeated games with few agents and actions. To this end, we propose a cooperative MARL algorithm based on the coordination degree (CMARL-CD) and analyze its dynamics in more general cases in which repeated games with more agents and actions are considered. Theoretical analysis shows that if the component action of every optimal joint action is unique, all optimal joint actions are asymptotically stable critical points. The CMARL-CD algorithm realizes coordination among agents without the need to estimate the global Q-value function. Each agent estimates the coordination degree of its own action, which represents the potential of being the optimal action. The efficacy of the CMARL-CD algorithm is studied through repeated games and stochastic games.

INDEX TERMS Multi-agent reinforcement learning, multi-agent system, reinforcement learning, independent learner.

I. INTRODUCTION

Reinforcement learning [1] (RL) is a prevalent method for optimizing an agent's behavior so that the best response from the environment can be obtained. Typically, RL is used to solve a Markov decision process (MDP). In an MDP, the state transition depends on a single agent's action. An agent can perceive states, execute an action, and receive a numerical reward from the environment. The goal is to obtain the maximum expected cumulative reward. However, some problems in the real world are naturally modeled as multi-agent systems (MASs), such as urban traffic signal control for multiple intersections [2], multiple automatic guided vehicle path planning [3], and mobile traffic for wireless networks [4]. In multi-agent settings, the state transition is determined by joint actions. The Markov property does not hold for any single agent in such settings, which is one of the major concerns when designing new multi-agent reinforcement learning (MARL) algorithms [5], [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Valentina E. Balas.

The goal of MARL is determined by the type of task. The goal in a general-sum game is to converge to some type of equilibrium [7] or socially optimal outcomes [8], [9]. This type of learning is known as equilibrium-based MARL (EMARL). The goal in a cooperative game is to maximize the expected common cumulative reward of all agents [10]. We name this type of learning cooperative MARL (CMARL). The goal in a zero-sum game is to maximize the expected reward of each agent [11], [12].

Non-stationarity and the rapidly growing joint action space are the two main challenges for MARL. First, for an independent learner, the individual Q-function of each agent is influenced by the other agents' actions. The analysis of independent learners focuses on repeated games with few agents and actions [13]–[18]. Second, to alleviate the non-stationarity problem, centralized learning is employed to estimate the global Q-function. In this framework, the joint action space grows exponentially with the number of agents, which affects the scalability of the JALs.

To this end, we propose a new CMARL algorithm known as CMARL based on the coordination degree (CMARL-CD). The CMARL-CD algorithm does not need to learn the global

Q-value function of the joint actions. Each agent records the maximal reward obtained in history and updates the coordination degree of its own action during the learning stage. The main contribution is the analysis of the CMARL-CD model in repeated games with more than two agents and actions. It has been proven that all optimal joint actions are asymptotically stable critical points if the component action of every optimal joint action is unique.

The remainder of this paper is organized as follows. Section II briefly reviews the different types of MARL algorithms. Section III presents the preliminaries. Section IV elaborates on the CMARL-CD algorithm, and provides theoretical analysis in repeated games. Section V studies the efficacy of CMARL-CD in two stochastic games: the distributed sensor network (DSN) task and the blood battlefield task. Finally, Section VI draws the conclusions.

II. RELATED WORK

Two characteristics of MARL algorithms are considered in this section. The first is whether the MARL algorithm belongs to JALs or independent learners. A JAL requires estimating the global Q-function of joint actions, while an independent learner requires each agent to estimate the individual Q-function of its own actions. The second characteristic, which is used to categorize the MARL algorithms reviewed in this paper, is whether the MARL algorithm belongs to CMARL or EMARL.

CMARL aims to optimize some performance index in a cooperative task. FMRQ [19], EAQR [20], and WRFMR [21] use the frequency of receiving the maximum reward. SOoN [22] utilizes the farsighted frequency together with the frequency used in FMRQ and EAQR. LA-OCA [23] is a learning automata-based algorithm that introduces a variable to indicate whether the maximum reward is achieved. LA-OCA has demonstrated excellent performance in some cooperative tasks. All of the above CMARL algorithms are independent learners.

Recently, deep learning has been incorporated into CMARL [24], [25]. One of the prevalent paradigms is centralized training with decentralized execution (CTDE), which attenuates both the problems of non-stationarity and the exponentially growing joint action space. MADDPG [26] uses decentralized critic networks for each agent, but the selected joint action is still required in centralized learning. COMA [27] uses a central critic network to estimate the global Q-function and uses distributed actor networks to select actions for each agent. COMA requires on-policy learning, which could be inefficient. To this end, a variety of Q-function decomposition methods have sprung up. Value decomposition networks (VDNs) [28] approximate the Q-function of joint actions by the sum of Q-functions of individual actions. Furthermore, QMIX [29] uses the mixing network to realize the individual-global-max (IGM) principle and account for the influence of the global state. To overcome the restriction on the structure of the critic used in QMIX, Qatten [30], QTRAN [31], and Q-value path decomposition (QPD) [32] have been proposed. Qatten uses multi-head attention to formulate the decomposition with theoretical foundations. QTRAN employs a gap function to satisfy the IGM and uses a fully centralized critic to guide the training of the individual Q-functions. QPD decomposes the Q-value function of joint actions along the state transition trajectories for credit assignments among agents and uses integrated gradients to approximate the Q-values.

The goal of most EMARL algorithms is to converge to an NE. Some EMARL algorithms attempt to accomplish this goal by employing the gradient method. These algorithms include but are not limited to infinitesimal gradient ascent (IGA) [33], WoLF-PHC [34], WPL [35], and PGA-APP [36]. Other EMARL algorithms have their own strategy updating rules. Nash-Q [37] searches for an NE in each state using quadratic programming. LRI [38], [39] is a learning automata-based algorithm. It has been proven that LRI converges to a pure NE in general-sum repeated games. Of the aforementioned EMARL algorithms, Nash-Q is a JAL, and the other EMARL algorithms are independent learners.

CMARL-CD distinguishes between the aforementioned cooperative independent learners as follows: First, compared with FMRQ, EAQR, and WRFMR, CMARL-CD does not use the frequency information that could influence the performance because of the estimation error introduced by it. Second, compared with LA-OCA, the analysis of the dynamics of CMARL-CD considers the normalization operation on the action probability.

III. PRELIMINARIES OF STOCHASTIC GAMES AND REPEATED GAMES

A. STOCHASTIC GAMES

In a stochastic game [40], [41], the state transition depends on the joint action. Let S represent the set of valid states, $A_i(s)$ represent the action set of agent i at state $s \in S$ for $i = 1, 2, \ldots, n$, and $A(s) = A_1(s) \times A_2(s), \ldots, \times A_n(s)$ represent the set of the joint actions at state $s \in S$. The probability of being at state s' if the joint action $a \in A$ is performed at state s is determined by the state transition T: $S \times A_1(s) \times A_2(s), \ldots, \times A_n(s) \times S \to [0,1]$. The reward function r_i: $S \times A_1(s) \times A_2(s), \ldots, \times A_n(s) \times S \to \mathbf{R}$ determines the immediate reward received by agent i. In a cooperative stochastic game, the global immediate reward is $r = \sum_{i=1}^{n} r_i$, and the goal is to maximize the global discounted cumulative reward, defined as follows:

$$\begin{aligned} R(t) &= r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \cdots \\ &= \sum_{k=0}^{K} \gamma^k r(t+k+1) \end{aligned} \tag{1}$$

where $\gamma \in (0, 1)$ is the discount factor and K is the ending time of an episode.

B. REPEATED GAMES

A repeated game [42], [43] is a one-stage game repeated by finite agents with finite actions. This study focuses on fully

		Agent 2	
Agent 1		**5**	2
		0	**5**

FIGURE 1. **The payoff matrix of a fully cooperative game.**

cooperative games. The payoff matrix determines the reward received by each agent. The payoff matrix of a two-agent-two-action cooperative game is shown in Fig. 1. If agent 1 chooses the first action (the first row) and agent 2 chooses the second action (the second column), both agents obtain a reward of 2. A strategy is the probability distribution on action selection. For a pure strategy, some action is always selected. For a mixed strategy, each action is assigned a probability.

Algorithm 1 CMARL-CD for Repeated Games

```
 1: Initialize the coordination degree of each agent and
    action, initialize $T$ with a positive value, and initialize
    the learning rate $\delta$ with a small positive value.
 2: Repeat for each game
 3:   For each agent $i$, do
 4:     Choose an action using $\boldsymbol{p}_i$.
 5:   End for each agent
 6:   For each agent $i$, do
 7:     Observe the reward $r_i$.
 8:     If $r_i \geq r_{i\_\max}$
 9:       $r_{i\_\max} = r_i$
10:       Update $\boldsymbol{c}_i$ according to (2)-(4).
11:       Update $\boldsymbol{p}_i$ according to (5).
12:     End if
13:   End for each agent
14: Until the strategy of each agent becomes pure.
15: Return $\boldsymbol{c}_i$ for each agent.
```

IV. COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING BASED ON COORDINATION DEGREE

A. FORMULATION OF THE ALGORITHM

To facilitate cooperation among agents, we propose the concept of coordination degree to evaluate the optimality of an action. In a fully cooperative repeated game, if the maximum global reward is obtained, the coordination degree of the selected action of each agent is increased, while the coordination degrees of the other actions are decreased. Otherwise, no updates are required. After the learning stage, each agent selects the action with the maximum coordination degree.

The pseudocode of CMARL-CD is shown in Algorithm 1. Each agent i updates its coordination degree $\boldsymbol{c}_i = (c_1^i, \ldots, c_{|A_i|}^i)$ according to:

$$c_g^i(k+1) = c_g^i(k) + \delta I_i(k), \quad \times \text{ if } a_g^i \text{ is selected in the } k\text{-th game} \tag{2}$$

$$c_j^i(k+1) = c_j^i(k) - \delta I_i(k), \quad \times \text{ for all } a_j^i \neq a_g^i \tag{3}$$

where a_j^i denotes the j-th action of agent i, c_j^i denotes the coordination degree of a_j^i, $I_i \in \{0, 1\}$ is an indicator variable, and $\delta \in (0, 1)$ is the learning rate. The value of I_i is determined by

$$I_i(k) = \begin{cases} 1 & \text{if } r_i(k) \geq r_{i\_\max}(k) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $r_i\ (k)$ is the global immediate reward received by agent i in the k-th game, and $r_{i\_\max}(k)$ is the maximal global immediate reward received by agent i by the k-th game. The learning rate δ should be set to a small positive value.

Then agent i's strategy $\boldsymbol{p}_i = (p_1^i, p_2^i, \ldots, p_{|A_i|}^i)$ is updated as follows:

$$p_j^i(k) = \frac{e^{\frac{c_j^i(k)}{T}}}{\sum_{h=1}^{|A_i|} e^{\frac{c_h^i(k)}{T}}} \tag{5}$$

where p_j^i is the probability of selecting the j-th action of agent i, and T is the temperature parameter that balances exploration and exploitation.

B. ANALYSIS OF CMARL-CD IN REPEATED GAMES

The updating rule of the coordination degrees is as follows:

$$c_j^i(k+1) = c_j^i(k) + \delta\Delta c_j^i(P(k), a(k), r(k)), \quad \times i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots |A_i| \tag{6}$$

where $P(k) = (\boldsymbol{p}_1(k), \boldsymbol{p}_2(k) \ldots, \boldsymbol{p}_n(k))$ is the joint strategy, $a(k)$ is the joint action, $r(k)$ is the global immediate reward, and $\Delta c_j^i(\ldots, \ldots, \ldots)$ represents the updating term of (2) and (3). According to Theorem 3.1 in [38], if δ is infinitely small, the CMARL-CD model in repeated games can be represented by

$$\frac{dc_j^i}{dt} = E[\Delta c_j^i(P(k), a(k), r(k) | P(k) = P], \quad \times i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots |A_i|. \tag{7}$$

The following theorem presents the characteristics of CMARL-CD in repeated games.

Theorem 1: Each agent uses the CMARL-CD algorithm to play a cooperative repeated game with n ($n \geq 2$) agents and m ($m \geq 2$) optimal joint actions. If the component action of every optimal joint action is unique, all the optimal joint actions are asymptotically stable critical points.

Proof: (i) Let a_{ij} be agent i's component action of the j-th optimal joint action, p_{ij} be the probability of selecting a_{ij}, and c_{ij} be the coordination degree of a_{ij} for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$. Because the component action of every optimal joint action is unique, (7) can be written as:

$$\begin{aligned} \frac{dc_{ij}}{dt} &= E[\Delta c_{ij}(P(k), a(k), r(k) | P(k) = P] \\ &= E[I_i | P, a_i = a_{ij}, r_i = r_{i\_\max}] \prod_{i=1}^{n} p_{ij} \end{aligned}$$

TABLE 1. Success rate in case 1.

	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%	98%
5 actions	100%	100%	100%	100%	100%	100%

$$\begin{aligned}
&+E[I_i|P, a_i = a_{ij}, r_i \neq r_{i\_\max}]p_{ij}(1-\prod_{h=1(h\neq i)}^{n} p_{hj})\\
&+\sum_{w\neq j} E[I_i|P, a_i = a_{iw}, r_i = r_{i\_\max}]\prod_{i=1}^{n} p_{iw}(-1)\\
&+\sum_{w\neq j} E[I_i|P, a_i = a_{iw}, r_i \neq r_{i\_\max}]p_{iw}\\
&\times(1-\prod_{h=1(h\neq i)}^{n} p_{hw})\\
=&\prod_{i=1}^{n} p_{ij}+0+\sum_{w=1(w\neq j)}^{m}(\prod_{i=1}^{n} p_{iw}(-1))+0\\
=&\prod_{i=1}^{n} p_{ij}-\sum_{w=1(w\neq j)}^{m}\prod_{i=1}^{n} p_{iw}, \quad i=1,2,\ldots,n,\\
&\times j=1,2,\ldots m. \qquad (8)
\end{aligned}$$

The coordination degree of the component action that does not constitute any optimal joint action decreases over time. According to (5), the probabilities of such actions decrease to zero over time. According to (5) and (8), we have:

$$\begin{aligned}
\frac{\mathrm{d}p_{ij}}{\mathrm{d}t} &= \sum_{h=1}^{m}\frac{\partial p_{ij}}{\partial c_{ih}}\frac{\mathrm{d}c_{ih}}{\mathrm{d}t}\\
&= \sum_{h=1}^{m}\left[\frac{\frac{1}{T}\mathrm{e}^{\frac{c_y}{T}}\sum_{k=1}^{m}\mathrm{e}^{\frac{c_k}{T}}-\mathrm{e}^{\frac{c_y}{T}}\frac{1}{T}\mathrm{e}^{\frac{c_y}{T}}}{\left(\sum_{k=1}^{m}\mathrm{e}^{\frac{c_{ik}}{T}}\right)^2}\right.\\
&\quad\left.\times\left(\prod_{i=1}^{n}p_{ih}-\sum_{w=1(v\neq h)}^{m}\prod_{i=1}^{n}p_{iv}\right)\right]\\
&= \frac{\frac{1}{T}\mathrm{e}^{\frac{c_j}{T}}\sum_{k=1}^{m}\mathrm{e}^{\frac{c_k}{T}}}{\left(\sum_{k=1}^{m}\mathrm{e}^{\frac{c_i}{T}}\right)^2}\left(\prod_{i=1}^{n}p_{ij}-\sum_{n=1(w\neq j)}^{m}\prod_{i=1}^{n}p_{iw}\right)\\
&\quad+\sum_{h=1(h\neq j)}^{m}\left[\frac{-\mathrm{e}^{\frac{c_y}{T}}\cdot\frac{1}{T}\mathrm{e}^{\frac{c_h}{T}}}{\left(\sum_{k=1}^{m}\mathrm{e}^{\frac{c_{ik}}{T}}\right)^2}\right.\\
&\quad\left.\times\left(\prod_{i=1}^{n}p_{ih}-\sum_{w=1(w\neq h)}^{m}\prod_{i=1}^{n}p_{iw}\right)\right]
\end{aligned}$$

$$\begin{aligned}
&= \frac{2}{T}p_{ij}\left[\left(\prod_{i=1}^{n}p_{ij}\right)(1-p_{ij})\right.\\
&\quad\left.-\sum_{h=1(h\neq j)}^{m}\left(p_{ih}^2\prod_{l=1(l\neq i)}^{n}p_{lh}\right)\right] \qquad (9)
\end{aligned}$$

Any critical point of (9) must satisfy

$$\frac{2}{T}p_{ij}[(\prod_{i=1}^{n}p_{ij})(1-p_{ij})-\sum_{h=1(h\neq j)}^{m}(p_{ih}^2\prod_{l=1(l\neq i)}^{n}p_{lh})]=0. \qquad (10)$$

It is clear that all joint actions are critical points. It is noted that $\sum_{j=1}^{m}p_{ij} = 1, i = 1, 2, \ldots, n$ when the probabilities of the actions that do not constitute any optimal joint actions decrease to zero. Thus, we perform the transformation as follows:

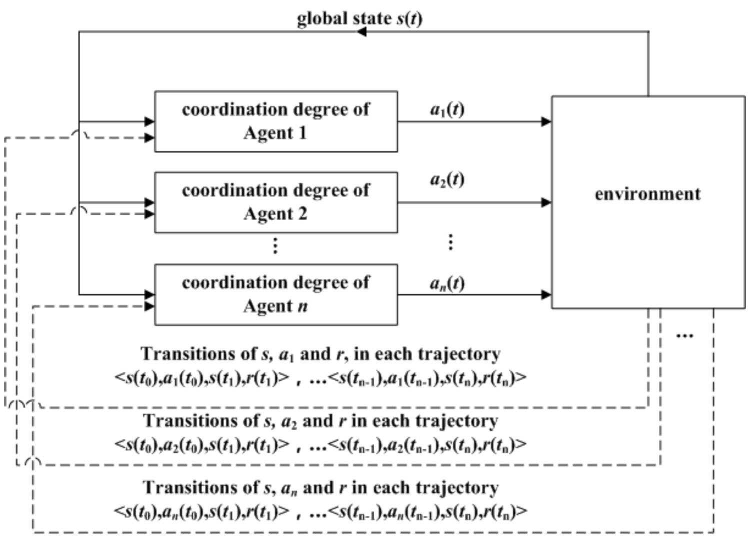
$$p_{ij}=\begin{cases}\bar{p}_{ij} & \text{if } j\neq m\\ 1-\sum_{h=1}^{m-1}\bar{p}_{ih} & \text{if } j=m\end{cases}. \qquad (11)$$

Then we can get

$$\begin{aligned}
&\frac{2}{T}\bar{p}_{ij}[(\prod_{i=1}^{n}\bar{p}_{ij})(1-\bar{p}_{ij})-\sum_{h=1(h\neq j)}^{m-1}(\bar{p}_{ih}^2\prod_{l=1(l\neq i)}^{n}\bar{p}_{lh})\\
&\quad-(1-\sum_{h=1}^{m-1}\bar{p}_{ih})^2\prod_{l=1(l\neq i)}^{n}(1-\sum_{h=1}^{m-1}\bar{p}_{lh})]=0,\\
&\quad\times i=1,2,\ldots,n,\ j=1,2,\ldots m-1. \qquad (12)
\end{aligned}$$

The stability of each of the m optimal joint actions is determined by the eigenvalues of the following Jacobin matrix $J\in R^{(m-1)n\times(m-1)n}$:

$$J=\begin{bmatrix}-\frac{2}{T} & 0 & 0 & \cdots & 0\\ 0 & -\frac{2}{T} & 0 & \cdots & 0\\ 0 & 0 & -\frac{2}{T} & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & -\frac{2}{T}\end{bmatrix}. \qquad (13)$$



global state $s(t)$
coordination degree of Agent 1
coordination degree of Agent 2
coordination degree of Agent n
$a_1(t)$
$a_2(t)$
$a_n(t)$
environment
Transitions of s, a_1 and r, in each trajectory
$<s(t_0),a_1(t_0),s(t_1),r(t_1)>$, $\ldots <s(t_{n-1}),a_1(t_{n-1}),s(t_n),r(t_n)>$
Transitions of s, a_2 and r in each trajectory
$<s(t_0),a_2(t_0),s(t_1),r(t_1)>$, $\ldots <s(t_{n-1}),a_2(t_{n-1}),s(t_n),r(t_n)>$
Transitions of s, a_n and r in each trajectory
$<s(t_0),a_n(t_0),s(t_1),r(t_1)>$, $\ldots <s(t_{n-1}),a_n(t_{n-1}),s(t_n),r(t_n)>$

FIGURE 2. The framework of the CMARL-CD algorithm for fully cooperative stochastic games (The arrows with solid lines indicate the execution process and the arrows with dotted lines indicate the training process).

TABLE 2. Success rate in case 2.

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

This shows that each eigenvalue of J is $-\frac{2}{T}$. Thus all the optimal joint actions are asymptotically stable critical points of (9). ■

C. EMPIRICAL STUDIES IN REPEATED GAMES

The efficacy of CMARL-CD in repeated games with n agents and m actions is investigated by empirical studies. Two situations are considered.

Case 1: The component action of every optimal joint action is unique.

Case 2: The component action of at least one optimal joint is not unique.

The simulation is performed for 50 runs. A game with a random payoff matrix is played repeatedly in each run. If the strategy of each agent becomes pure, and the joint action can obtain the maximum reward, a successful run is achieved. The learning ends if for each agent, the probability of selecting some action is greater than 0.999. The temperature T is 1.0, and δ is 0.04.

In case 1, each game contains m optimal joint actions. It is shown in Tab. 1 that the success rate is 98% when $n = 7, m = 4$, and 100% otherwise. The reason for failure to obtain 100% when $n = 7$, $m = 4$ is that the maximal global reward has never been obtained during the learning stage, and the joint strategy converges to the local optimum.

In case 2, each game contains $[0.1m^n]$ optimal joint actions, where [ ] returns a round integer. It is shown in Tab. 2 that a success rate of 100% is obtained in all games.

The empirical results show that the CMARL-CD algorithm can converge to one of the optimal joint actions in cooperative repeated games.

D. CMARL-CD FOR STOCHASTIC GAMES

CMARL-CD can be applied to stochastic games. The framework is illustrated in Fig. 2. Each agent receives the global state as input, evaluates the coordination degree of each of its actions, and executes an action, which is indicated by the arrows with solid lines. Each agent independently updates the coordination degree using each trajectory, as indicated by the arrows with dotted lines. The pseudocode is presented in Algorithm 2. The coordination degree of agent $\boldsymbol{ic}_i = (c^i_1(s), \ldots, c^i_{|A_i|}(s))$ is updated according to

$$c^i_g(s) = c^i_g(s) + \delta I_i(s), \quad \times \text{ if } a^i_g \text{ is selected at state } s \tag{14}$$

$$c^i_j(s) = c^i_j(s) - \delta I_i(s), \quad \times \text{ for all } a^i_j \neq a^i_g \tag{15}$$

where $c^i_j(s)$ denotes the coordination degree of a^i_j at state s, and $I_i(s) \in \{0, 1\}$ is an indicator variable. The value of $I_i(s)$ is determined by

$$I_i(s) = \begin{cases} 1 & \text{if } R_i(s) \geq R_{i\_\max}(s) \\ 0 & \text{otherwise} \end{cases}. \tag{16}$$

where $R_i(s)$ is the global cumulative reward obtained by agent i from state s, and $R_{i\_\max}(s)$ is the maximal global cumulative reward in history. To maintain exploration, we confine the value of $c^i_j(s)$ within $[c_{\min}, c_{\max}]$, where $c_{\min}$ is positive.

The probability of selecting action a_j^i at state s is updated as follows.

$$p_j^i(s) = \frac{e^{\frac{c_j^i(s)}{T}}}{\sum_{h=1}^{|A_i|} e^{\frac{c_h^i(s)}{T}}}. \tag{17}$$

V. EMPIRICAL STUDIES FOR STOCHASTIC GAMES

The efficacy of CMARL-CD in stochastic games is studied empirically through the DSN task and the blood battlefield task. The states and the actions of both the tasks take discrete values. The tasks differ in that the first task is fully cooperative, while the second task involves competition between two teams of cooperative agents. CMARL-CD is compared with LA-OCA, VDN, and QMIX to demonstrate its efficacy. For fairness, global state information is used during the learning stage for all the algorithms.

Algorithm 2 CMARL-CD for Stochastic Games

```
1: Initialize the coordination degree of each agent and
   action, and initialize the learning rate $\delta$ to a small positive
   number.
2: Repeat
3:   Repeat
4:     For each agent $i$, do
5:       Observe state $s$.
6:       Choose an action $a_i$ using $\boldsymbol{p}_i(s)$.
7:     End for each agent
8:     For each agent $i$, do
9:       Observe $s'$ and $r$.
10:      Record the tuple $< s, a_i, s', r >$.
11:    End for each agent
12:  Until the episode is over
13:  For each agent $i$, do
14:    For each state $s$ visited in the previous episode
15:      Update $R_i(s)$ according to (1) and the recorded
         tuples.
16:      If $R_i(s) \geq R_{i\_\max}(s)$
17:        $R_{i\_\max}(s) = R_i(s)$
18:        Update $\boldsymbol{c}_i(s)$ according to (14)-(16).
19:        Update $\boldsymbol{p}_i(s)$ according to (17).
20:      End if
21:    End for each visited state
22:  End for each agent
23: Until the specified number of episodes have elapsed
24: Return $\boldsymbol{c}_i(s)$ for each agent $i$.
```

A. TASK 1: DISTRIBUTED SENSOR NETWORK

The DSN task [44] requires the sensors (agents) to cooperate to capture the targets. As shown in Fig. 3, 12 sensors are distributed within a grid. The two targets walk randomly within the six cells. Both the targets' number and positions can be sensed by all the sensors. The sensors must cooperate at each of the 42 states (excluding one absorbing state). Each sensor can choose to do nothing or focus on one of its adjacent cells. The joint action space contains 291,600 elements. All the sensors execute their actions simultaneously. Then, the targets move sequentially at each time step. If a target moves to a cell that is not empty, it fails to move. If three or four sensors focus on a target, a hit is made. If a target receives three hits, it is captured. The captured target does not occupy any cells. An episode lasts 40 time steps unless both the targets are captured.

The reward assignment obeys the following rules. If a target is captured, a reward of 10 is obtained by each sensor that involves the capture. If the target is captured by four sensors,

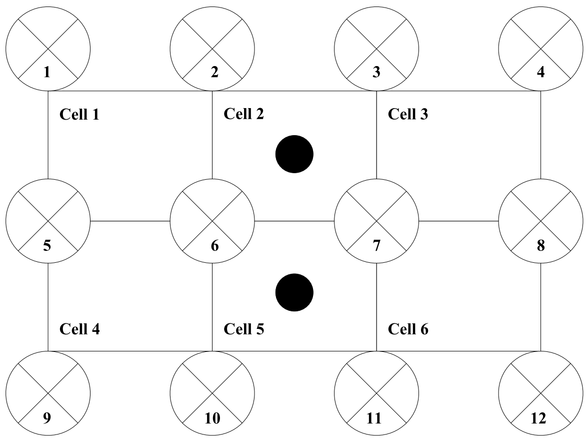


FIGURE 3. A distributed sensor network with 12 sensors and two targets.

TABLE 3. Success rate for the DSN task.

	$L = 100{,}000$	$L = 300{,}000$	$L = 500{,}000$
CMARL-CD	**90.90%**	**100%**	**100%**
LA-OCA	46.57%	89.16%	95.08%
VDN	0.40%	5.49%	7.85%
QMIX	3.29%	6.88%	6.42%

TABLE 4. Average cumulative reward for the DSN task.

	$L = 100{,}000$	$L = 300{,}000$	$L = 500{,}000$
CMARL-CD	**41.90 | 0.10**	**42 | 0**	**42 | 0**
LA-OCA	39.92 | 0.72	41.63 | 0.42	41.80 | 0.30
QMIX	39.49 | 1.05	40.81 | 0.70	41.17 | 0.44
VDN	36.75 | 0.78	39.51 | 0.98	40.63 | 0.57

TABLE 5. Average steps for the DSN task.

	$L = 100{,}000$	$L = 300{,}000$	$L = 500{,}000$
CMARL-CD	**3.06 | 0.07**	**3 | 0**	**3 | 0**
LA-OCA	3.39 | 0.15	3.11 | 0.10	3.05 | 0.07
QMIX	6.30 | 0.49	5.09 | 0.12	5.07 | 0.10
VDN	6.48 | 0.25	5.29 | 0.10	5.09 | 0.10

TABLE 6. Minimum cumulative reward for the DSN task.

	$L = 100{,}000$	$L = 300{,}000$	$L = 500{,}000$
CMARL-CD	**41.60**	**42**	**42**
LA-OCA	38.00	40.11	40.66
QMIX	37.60	38.52	40.01
VDN	35.98	37.35	39.23

TABLE 7. Maximum steps for the DSN task.

	$L = 100{,}000$	$L = 300{,}000$	$L = 500{,}000$
CMARL-CD	**3.20**	**3**	**3**
LA-OCA	3.68	3.33	3.23
QMIX	7.74	5.37	5.25
VDN	7.39	5.60	5.33

only the sensors with the top three indices are rewarded. Focusing on one cell is rewarded by -1, and doing nothing is rewarded by 0. Each sensor shares its immediate reward with the other agents at each step. The optimal joint strategy can obtain a cumulative reward of 42 in three steps.

The simulation is performed for 50 runs, each of which contains L episodes for learning and 50,000 episodes for evaluation. In the learning stage, all agents' strategies are updated. In the evaluation stage, all agents' strategies are fixed.

The CMARL-CD algorithm uses the parameters $\delta = 0.05$, $T = 1.0$, $c_{\min} = 0.5$, $c_{\max} = 3.0$, $\gamma = 0.9$, and the initial values of the coordination degrees of all state-action pairs $c^i_j(s) = 1.5$. To increase exploration near the local optima, the CMARL-CD algorithm resets the coordination degrees of all actions at state s to 1.5, and sets δ to 0.5 when the obtained cumulative reward from state s is larger than the maximum cumulative reward in history. The LA-OCA algorithm uses the parameters in [23].

The parameter settings for QMIX and VDN are as follows. Each agent network is an MLP with one hidden layer of 49 neurons. The size of the replay buffer is 40000, and the size of each batch is 400. Parameter updating begins after one batch of tuples is available. The estimation networks are updated after every 200 time steps using the Adam optimizer with an initial learning rate of 0.001. The target networks are cloned from the estimation networks after every 2000 time steps. The ε-greedy policy is used to select an action during the learning stage. The exploration rate ε follows

$$\varepsilon = \begin{cases} \varepsilon_{\text{ini}} - \dfrac{(\varepsilon_{\text{ini}} - \varepsilon_{\text{fin}})}{0.8L} n & 1 \leq n \leq 0.8L \\ 0.01 & 0.8L < n \leq L \end{cases} \tag{18}$$

where $\varepsilon_{\text{ini}} = 1.0$ and $\varepsilon_{\text{fin}} = 0$ are two constants, and n is the number of elapsed episodes. For QMIX, the mixing network contains one hidden layer of 70 neurons with exponential linear unit (ELU).

The success rate, cumulative reward, and the number of steps are selected as performance indices. If a cumulative reward of 42 is obtained in three steps in an evaluation episode, success is obtained. The success rate in a run is defined as the number of successful evaluation episodes divided by the total number of evaluation episodes (50,000).

The success rates are listed in Tab. 3. It can be seen that CMARL-CD achieves the highest learning speed and highest success rate. Both QMIX and VDN achieve a success rate of less than 8% for all values of L. After $L = 100{,}000$ learning episodes, LA-OCA achieves a success rate of 46.57%, while CMARL-CD achieves a success rate of 90.90%. Although LA-OCA, QMIX, and VDN perform better given more learning episodes, they obtain a lower success rate when CMARL-CD has already obtained a 100% success rate after 300,000 learning episodes. These results indicate that CMARL-CD learns much faster than the other algorithms.

The average cumulative reward and steps are listed in Tab. 4 and Tab. 5 respectively. Taking the item of CMARL-CD with $L = 100{,}000$ in Tab. 4 as an example, '41.90|0.10' represents an average cumulative reward of 41.90 and a standard deviation of 0.10. CMARL-CD obtains the maximum cumulative reward and uses fewer time steps than any of the other algorithms for each value of L. Both QMIX and VDN obtain high cumulative rewards with $L = 500{,}000$. However, most of the time, they fail to capture the targets in three steps, which leads to a low success rate. The worst case is shown in Tab. 6 and Tab. 7. It can be seen that CMARL-CD is more reliable than the others.

To verify the effectiveness of CMARL-CD, we visualize the joint strategy obtained by CMARL-CD with $L = 500{,}000$, as shown in Fig. 4. Each sensor selects an action with the maximal coordination degree. It can be seen that the optimal joint action is selected under each of the 42 states (not including the absorbing state).

B. TASK 2: BLOOD BATTLEFIELD

Blood battlefield, which is developed by us, is a strategy game. The player needs to command a troop to fight against its opponent who owns a troop.

Each side has four marines and two gunners. The property values are presented in Fig. 5. All units cannot move, just like the units in Hearthstone that have been deployed on the battlefield. Unlike most turn-based strategy games, the units on both sides take actions at the same time in each turn. Every live unit must attack a live opponent unit in each turn and take damage afterward. The true damage depends on the attacker's attack damage (AD) and hit rate (HR). For example, a gunner with 2 HP (hit point) was attacked by a gunner and a marine on the other side. The marine was missed and the gunner hit successfully. The true damage received by the target was $0 + 2 = 2$. The HP became $2 - 2 = 0$. A dead unit will never become a target. A game ends with one side beating

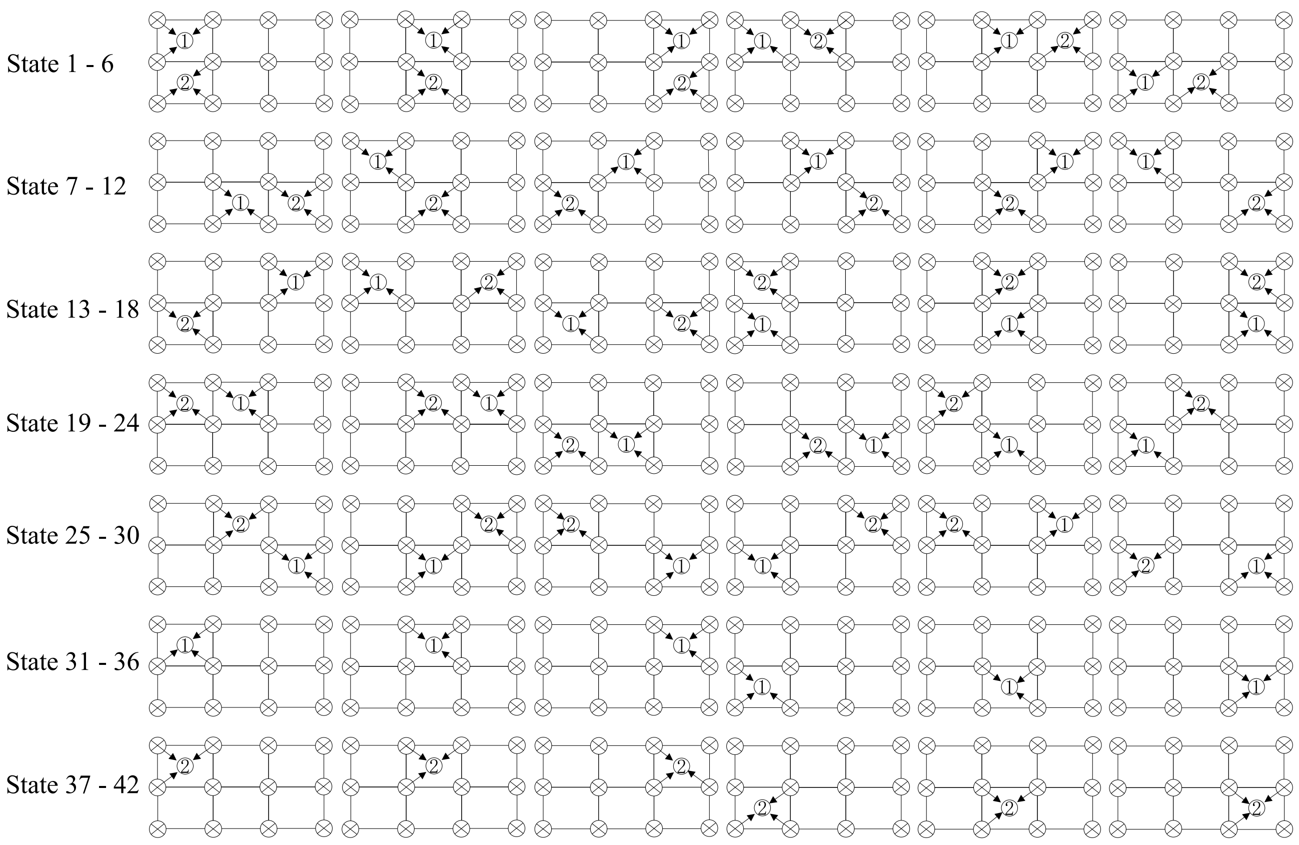


FIGURE 4. **The obtained optimal joint strategy for the DSN task by using the CMARL-CD algorithm (after 500,000 learning episodes).**

the other side or a tie. A tie appears if no side wins the game in 100 turns.

The state vector is $\boldsymbol{s} = [hp_1, \ldots hp_6, s_1, \ldots s_6]^T$ where hp_i is the HP value of opponent i, and $s_j \in \{\text{ALIVE}, \text{DESTROYED}\}$ is the state of its j-th teammate (including itself). The state space contains $4^4 \times 7^2 \times 2^5 = 401,408$ elements. Each unit can select only a live opponent unit as its target. The reward assignment obeys the following rules: if one opponent unit is destroyed, all units of the other side (alive or not) are rewarded with 2; the winner obtains a reward of 10 and the loser obtains a reward of -10, and a tie brings a reward of 0 with both sides.

The game involves both coordination and competition. The units of each side need to coordinate to eliminate the opponent units to survive the war. Because the game is not a sequential decision problem, each player needs to consider only its own fire deployment. Four algorithms including CMARL-CD, LA-OCA, QMIX, and VDN are compared in a tournament.

The CMARL-CD algorithm uses the parameters $c_{\max} = 2.5$ and $T = 0.8$. The other parameters are the same as those used in the DSN task. The LA-OCA algorithm uses the same parameters as those used in the DSN task. The parameter settings for QMIX and VDN are as follows: Each agent network is an MLP with one hidden layer of 49 neurons. The size of the replay buffer is 100000, and the size of each

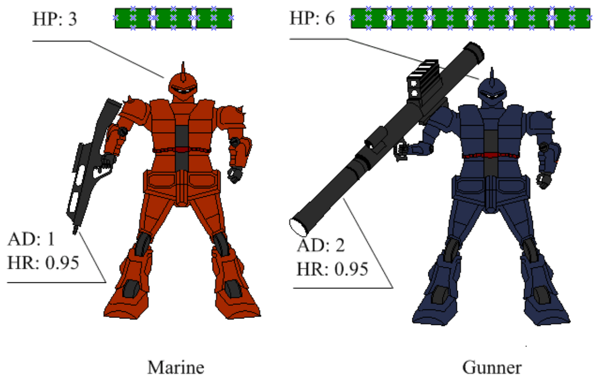


FIGURE 5. **Two kinds of units in blood battlefield: Marine and gunner.**

batch is 720. Parameters updating begins after one batch of tuples is available. The estimation networks are updated after every 240 time steps using the Adam optimizer with an initial learning rate of 0.001. The target networks are cloned from the estimation networks after every 2400 time steps. The ε-greedy policy is used during the learning stage. The exploration rate ε is annealed from 1.0 to 0.0 with L increases. For QMIX, the mixing network contains one hidden layer of 64 neurons with ELUs.

The tournament has 30 rounds. Each round includes $4 + 4 \times (4 - 1)/2 = 10$ matches. In each match, one

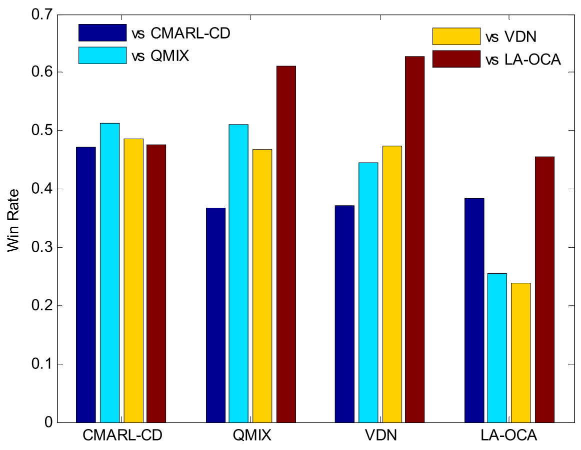


FIGURE 6. The win rate of each algorithm.

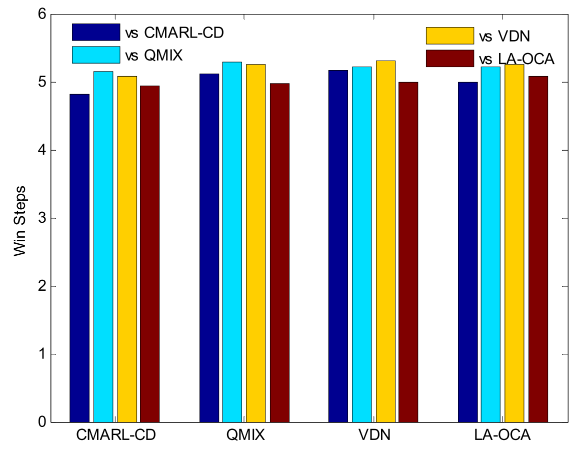


FIGURE 7. The win steps of each algorithm.

algorithm plays against another algorithm (which could be itself) for 500,000 episodes in the learning stage, and then plays another 500,000 episodes in the evaluation stage. Fixed learned strategies are used in the evaluation stage. All results are averaged over 30 rounds.

Fig. 6 shows the win rate of each algorithm (a tie does not count in the win rate, which explains that the sum of the win rates of the two opponents is not 100%). CMARL-CD has a higher win rate against any of the other algorithms. Both QMIX and VDN have a great advantage over LA-OCA, but they have a win rate of less than 40% against CMARL-CD.

Fig. 7 shows the win steps of each algorithm. The win-step of an algorithm is the average number of steps used in each winning episode. Fewer win-steps indicate that the algorithm learns a better strategy to defeat its opponent. It can be seen that CMARL-CD has fewer win-steps compared to the other algorithms.

VI. CONCLUSION

This paper proposes an MARL algorithm known as CMARL-CD for fully cooperative scenarios. Empirical studies support the theoretical analysis of repeated games. The simulation results on the DSN task and blood battlefield empirically demonstrate that the CMARL-CD algorithm can converge to the optimal joint strategy for stochastic games. In the future, we will study the dynamics of the CMARL-CD algorithm in stochastic games, and incorporate deep learning into CMARL-CD to improve scalability.

REFERENCES

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

[2] H. Ge, Y. Song, C. Wu, J. Ren, and G. Tan, "Cooperative deep Q-learning with Q-value transfer for multi-intersection signal control," *IEEE Access*, vol. 7, pp. 40797–40809, 2019.

[3] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-AGV systems applied to industrial warehouses," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 922–934, Jul. 2015.

[4] M. Yan, G. Feng, J. Zhou, and S. Qin, "Smart multi-RAT access based on multiagent reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4539–4551, May 2018.

[5] H. M. Schwartz, "Learning in muitiplayer stochastic games," in *Multi-Agent Machine Learning: A Reinforcement Approach*. Hoboken, NJ, USA: Wiley, 2014, pp. 73–143.

[6] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[7] 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.

[8] C. Zhang, X. Li, J. Hao, S. Chen, K. Tuyls, W. Xue, and Z. Feng, "SA-IGA: A multiagent reinforcement learning method towards socially optimal outcomes," *Auto. Agents Multi-Agent Syst.*, vol. 33, no. 4, pp. 403–429, Jul. 2019.

[9] J. Jiang and Z. Lu, "Learning fairness in multi-agent systems," 2019, *arXiv:1910.14472*. [Online]. Available: http://arxiv.org/abs/1910.14472

[10] 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.

[11] Y. Zhu and D. Zhao, "Online minimax Q network learning for two-player zero-sum Markov games," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 14, 2020, doi: 10.1109/TNNLS.2020.3041469.

[12] 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.

[13] K. Tuyls and A. Nowé, "Evolutionary game theory and multi-agent reinforcement learning," *Knowl. Eng. Rev.*, vol. 20, no. 1, pp. 63–90, Mar. 2005.

[14] K. Tuyls and S. Parsons, "What evolutionary game theory tells us about multiagent learning," *Artif. Intell.*, vol. 171, no. 7, pp. 406–416, May 2007.

[15] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers, "Evolutionary dynamics of multi-agent learning: A survey," *J. Artif. Intell. Res.*, vol. 53, pp. 659–697, Aug. 2015.

[16] M. M. Kaisers and K. Tuyls, "Frequency adjusted multi-agent Q-learning," in *Proc. AAMAS*, Toronto, ON, Canada, 2010, pp. 309–316.

[17] A. Kianercy and A. Galstyan, "Dynamics of Boltzmann Q-learning in two-agent two-action games," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 85, no. 4, pp. 1145–1154, 2012.

[18] M. Babes, M. Wunder, and M. L. Littman, "Q-learning in two-player two-action games," in *Proc. AAMAS*, Budapest, Hungary, 2009, pp. 1–6.

[19] 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, Jun. 2017.

[20] Z. Zhang and D. Wang, "EAQR: A multiagent Q-learning algorithm for coordination of multiple agents," *Complexity*, vol. 2018, Aug. 2018, Art. no. 7172614.

[21] H. Liu, Z. Zhang, and D. Wang, "WRFMR: A multi-agent reinforcement learning method for cooperative tasks," *IEEE Access*, vol. 8, pp. 216320–216331, 2020.

[22] L. Matignon, G. J. Laurent, and N. L. Fort-Piat. (2009). *Coordination of Independent Learners in Cooperative Markov Games*. [Online]. Available: http://hal.archives-ouvertes.fr/docs/00/37/08/89/PDF/Rapport-1.pdf

[23] Z. Zhang, D. Wang, and J. Gao, "Learning automata-based multi-agent reinforcement learning for optimization of cooperative tasks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 7, 2020, doi: 10.1109/TNNLS.2020.3025711.

[24] T. Thi Nguyen, N. Duy Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications," 2018, *arXiv:1812.11794*. [Online]. Available: http://arxiv.org/abs/1812.11794

[25] P. Hernandez-Leal, B. Kartal, and E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auto. Agents Multi-Agent Syst.*, vol. 33, pp. 750–797, Oct. 2019.

[26] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.

[27] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatc, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[28] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. ICAAMS*, Stockholm, Sweden, 2018, pp. 2085–2087.

[29] 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. 4292–4301.

[30] Y. Yang, J. Hao, B. Liao, K. Shao, G. Chen, W. Liu, and H. Tang, "Qatten: A general framework for cooperative multiagent reinforcement learning," 2020, *arXiv:2002.03939*. [Online]. Available: http://arxiv.org/abs/2002.03939

[31] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. ICML*, Long Beach, CA, USA, 2019, pp. 5887–5896.

[32] Y. Yang, J. Hao, G. Chen, H. Tang, Y. Chen, Y. Hu, C. Fan, and Z. Wei, "Q-value path decomposition for deep multiagent reinforcement learning," in *Proc. ICML*, 2020, pp. 10706–10715.

[33] 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.

[34] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artif. Intell.*, vol. 136, no. 2, pp. 215–250, Apr. 2002.

[35] S. Abdallah and V. Lesser, "A multiagent reinforcement learning algorithm with non-linear dynamics," *J. Artif. Intell. Res.*, vol. 33, pp. 521–549, Dec. 2008.

[36] C. Zhang and V. R. Lesser, "Multi-agent learning with policy prediction," in *Proc. AAAI*, Atlanta, GA, USA, 2010, pp. 927–934.

[37] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochasticgames," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.

[38] P. S. Sastry, V. V. Phansalkar, and M. A. L. Thathachar, "Decentralized learning of Nash equilibria in multi-person stochastic games with incomplete information," *IEEE Trans. Syst., Man Cybern.*, vol. 24, no. 5, pp. 769–777, May 1994.

[39] P. Vrancx, K. Verbeeck, and A. Nowe, "Decentralized learning in Markov games," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 38, no. 4, pp. 976–981, Aug. 2008.

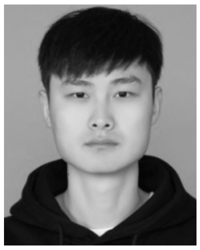
[40] T. E. S. Raghavan, "Stochastic games—An overview," in *Stochastic Games and Related Topics*, T. E. S. Raghavan, T. S. Ferguson, T. Parthasarathy, O. J. Vrieze, Eds. Springer, 1991, pp. 1–9.

[41] O. J. Vrieze, "Stochastic games and stationary strategies," in *Stochastic Games Application*, A. Neyman and S. Sorin, Eds. Springer, 2003, pp. 37–50.

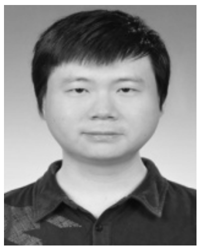
[42] H. Peters, "Repeated games," in *Game Theory*. Springer, 2008, pp. 101–110.

[43] S. N. Durlauf and L. E. Blume, "Repeated games," in *Game Theory*. Basingstoke, U.K.: Macmillan, 2010, pp. 286–299.

[44] A. Syed, S. Koenig, and M. Tambe, "Preprocessing techniques for accelerating the DCOP algorithm ADOPT," in *Proc. AAMAS*, Utrecht, The Netherlands, 2005, pp. 1041–1048.



HAOYAN CUI received the B.S. degree from Qingdao University, Qingdao, China, in 2020. Since 2020, he has been a Graduate Student with Shandong Key Laboratory of Industrial Control Technology, Department of Automation, School of Automation, Qingdao University. His current research interests include reinforcement learning and deep learning.



ZHEN ZHANG received the B.S. degree from China University of Petroleum, Dongying, China, in 2006, the M.S. degree in control theory and control engineering from 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 Shandong Key Laboratory of Industrial Control Technology, 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 August 2018 to February 2019. He has been an Associate Professor with the School of Automation, Qingdao University, since November 2019. His current research interests include reinforcement learning and intelligent optimization methods.