Computer Science and Information Systems 17(2):647–664 https://doi.org/10.2298/CSIS191220009G

Multi-Agent Cooperation Q-Learning Algorithm Based on Constrained Markov Game

Yangyang Ge[1], Fei Zhu[1,2], Wei Huang[1], Peiyao Zhao[1], and Quan Liu[1]

[1] School of Computer Science and Technology, Soochow University, Suzhou Jiangsu 215006, China
20184227043@stu.suda.edu.cn, {zhufei, huangwei}@suda.edu.cn, 20195427013@stu.suda.edu.cn, quanliu@suda.edu.cn
[2] Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, China

Abstract. Multi-Agent system has broad application in real world, whose security performance, however, is barely considered. Reinforcement learning is one of the most important methods to resolve Multi-Agent problems. At present, certain progress has been made in applying Multi-Agent reinforcement learning to robot system, man-machine match, and automatic, etc. However, in the above area, an agent may fall into unsafe states where the agent may find it difficult to bypass obstacles, to receive information from other agents and so on. Ensuring the safety of Multi-Agent system is of great importance in the above areas where an agent may fall into dangerous states that are irreversible, causing great damage. To solve the safety problem, in this paper we introduce a Multi-Agent Cooperation Q-Learning Algorithm based on Constrained Markov Game. In this method, safety constraints are added to the set of actions, and each agent, when interacting with the environment to search for optimal values, should be restricted by the safety rules, so as to obtain an optimal policy that satisfies the security requirements. Since traditional Multi-Agent reinforcement learning algorithm is no more suitable for the proposed model in this paper, a new solution is introduced for calculating the global optimum state-action function that satisfies the safety constraints. We take advantage of the Lagrange multiplier method to determine the optimal action that can be performed in the current state based on the premise of linearizing constraint functions, under conditions that the state-action function and the constraint function are both differentiable, which not only improves the efficiency and accuracy of the algorithm, but also guarantees to obtain the global optimal solution. The experiments verify the effectiveness of the algorithm.

Keywords: Markov game, Distributed perception, Multi-Agent cooperation, constrained Markov decision process.

1. Introduction

Multi-Agent System (MAS) is a combination of several sub-agents, which decomposes a large complex system into smaller and intercommunicating subsystems that are relatively manageable [1]. MAS is evolved from distributed artificial intelligence, it is applied in various areas such as intelligent robot, traffic control, distributed decision making,

business management, virtual reality and so on [2]. At present, a new mechanism that combines MAS and reinforcement learning is gradually considered to be a research hotspot [3].

Multi-Agent reinforcement learning (MARL) is to apply reinforcement learning algorithm to MAS [4]. Littman in 1900s put forward MARL with Markov Decision Process (MDP) being the contextual framework, which offered a simple mathematical framework for solving most of reinforcement learning problems [5]. MARL possesses certain properties, such as autonomy, distributivity, consistency and so on, and abilities such as learning, reasoning and self-organizing [6]. According to different learning objectives, MARL can be divided into full cooperation task, full competition task and hybrid task [7].

In full-cooperation stochastic game, agents are not making decisions independently, but are cooperating with each other trying to achieve a mutual goal in a parallel way. They share the same reward function and maximize it adopting greedy strategy [8]. Littman M proposed the Team-Q algorithm, which solved the cooperation problem among agents by hypothesizing an optimal union action [9]. Lauer M and Riedmiller M proposed the Distributed-Q algorithm, which solved the cooperation problem among agents without hypothesizing the coordination condition, with an infinite computational cost. It shares the same computation complexity with the single agent Q-learning algorithm [10]. This method, however, is suitable only for deterministic problems with non-negative reward functions. All the algorithms introduced above are limited, they all rely on the precise measurement of the state. Some of them need also the precise measurement of influence to an agent from other agents and may suffer from curse of dimensionality. On the other hand, they ignore the safety problem of agents and other constraint conditions.

In full-competition stochastic game, the agent maximizes its own reward while minimizes reward of its opponents [11]. Minimax-Q algorithm is a full-competition stochastic game that calculates policies and values through minimax principle [12].

In hybrid task, the reward function of the agent is not restricted [13], which is suitable for a selfish agent. In the game many algorithms are only for static tasks based on the concept of equilibrium in game theory [14]. In the process of a hybrid task, an agent needs to know the actions and rewards of other agents. The equilibrium selection problem arises when different agents obtain different policies. Nash Q-learning is a common method for solving this problem [15]. Correlation Equilibrium Q-learning (CE-Q) method solves the equilibrium problem with the concept of correlation equilibrium [16]. Asymmetric Q-learning solves the equilibrium problem using the leader-follower equilibrium. The follower needs not model the Q-table of the leader, but the leader should know how its followers choose their actions [17].

Traditional MARL ignores the safety problem of the agent which is inescapable in practical use. To solve the problem, in this paper we propose a Constrained Multi-Agent Cooperation Q-learning (CMACQ) algorithm based on constrained Markov game. Compared with traditional methods, this algorithm can ensure the safety of the agent, avoiding dangerous states and their consequences. In this method, before the agent executes an action, it determines all the safe executable actions based on the current state, and chooses the optimal one according to the greedy strategy as well as interactions with other agents. The algorithm ensures the safety of each agent when agents are cooperating to complete the task.

This paper is organized as follows. In section 2, we introduce the related concepts and studies. In section 3, we formalize our model and transform the model into a convex model by linearizing constraint functions, where we exploit the Lagrange multiplier method to obtain the safe action and choose the optimal safe action according to the greedy strategy as well as interactions among agents. In section 4, we compare CMACQ with MACQ in the firefighting-through-multi-agent-cooperation experiment and Deep Sea Fishing experiment. In section 5, a summary of CMACQ and a discussion of future work are presented.

2. Related Work

2.1. Reinforcement Learning

In reinforcement learning tasks, the agent detects the environment and takes actions to obtain the largest long-term cumulative reward which is a valuable encouraging signal [18]. Markov Decision Process framework is used to solve most of the reinforcement learning problems, which is denoted by a tetrad (*S, A, P, R*) [19] where *S* is the set of states which contains finite numbers of elements, *A* is the set of actions, *P* is the state transition probability and *R* is the reward function. In MDP, the state transition probability contains actions, which is expressed as follows [20]:

$$P_{ss'}^{a} = P\left[s_{t+1} = s' \mid s_t = s, a_t = a \right] \tag{1}$$

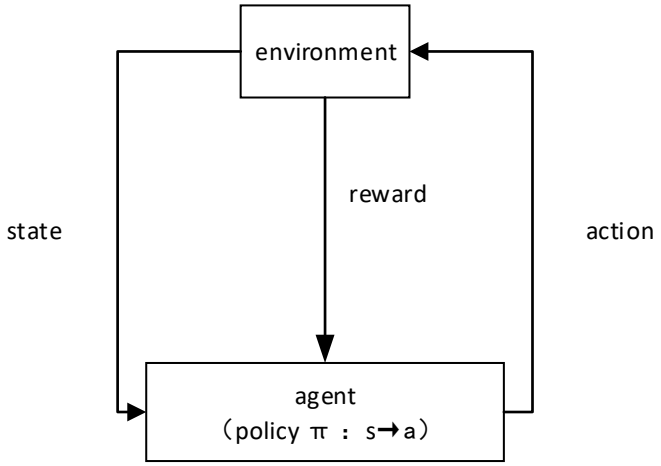

Fig. 1. The illustration of reinforcement learning.

At time *t*, the agent is in current state s_t, chooses action *a* according to policy π, receives a feedback from the environment and proceeds to the next state s_{t+1}, and obtains the reward r_t. The policy π, which is divided into deterministic policy and non-deterministic

policy, denotes the mapping from s_t to a_t. In reinforcement learning, the cumulative discounted reward is defined as [19]:

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{2}$$

where the discounted factor $\gamma \in (0,1]$, R_t is the reward value from time t to T. The reinforcement learning model is showed as figure 1 [19].

The state-action value function, $Q^{\pi}(s,a)$, is used to evaluate policies, which denotes the sum of cumulative rewards when the agent chooses action a_t according to the policy π under the current state s_t. The function [19] is shown as:

$$Q^{\pi}(s,a) = E\left[R_t \mid s_t = s, a_t = a, \pi\right] \tag{3}$$

As the iteration continues, the state-action value will converge to be optimal. Although the optimal policy may not be unique, the optimal state-action value is unique, as shown in equation (4) [19]:

$$Q^{*}(s,a) = \max_{\pi} E\left[R_t \mid s_t = s, a_t = a, \pi\right] \tag{4}$$

The state-value function $V(s)$ denotes the expectation of all state-action value functions when the agent follows policy π under the state s_t, which is calculated as equation (5) [19]. As the policy iteration continues, the state-value function converges to a unique optimal one. The optimal state-value function is obtained by equation (6) [19]:

$$V^{\pi}(s) = E\left[R_t \mid s_t = s, \pi\right] \tag{5}$$

$$V^{*}(s) = \max_{\pi} E\left[R_t \mid s_t = s, \pi\right] \tag{6}$$

Q-Learning algorithm [21] is a typical off-policy algorithm and is one of the most widely used reinforcement learning method. The algorithm defines a Q function, and updates this function using equation (7) [22] and equation (8) [22], that is, TD error, and finally obtains the converged optimal state-action value:

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q_t\left(s_{t+1}, a'\right) - Q_t\left(s_t, a_t\right) \tag{7}$$

$$Q_{t+1}\left(s_t, a_t\right) = Q_t\left(s_t, a_t\right) + \alpha_t \delta_t \tag{8}$$

where t denotes time step, α_t is learning rate, δ_t is temporal difference error [22], a' is the action taken at next state s_{t+1}. When t tends to infinity, the optimal control policy is obtained [23], [24].

2.2. Multi-Agent Cooperation

With rapid development in areas such as sensor technology, wireless communication technology and computer vision, intelligent robot system was transformed from stand-alone system into multicomputer system. Multi-Agent system is widely applied in which the cooperation between agents played an important role [25]. Therefore, to study the cooperation between two agents is a key procedure when developing Multi-Agent systems [26]. At present, the research is divided into two categories: one is to apply the research of Multi-body behaviors to the cooperation of agents; the other is to concentrating on programming and solving of the problem [27], [28]. Allen et al studied the effect of minor changes on social evolution by assessing the cooperation tendencies

in many different population structures [29]; Mcavoy studied the effect of Evolutionary Game Dynamics on changing the scale of agents by public good game [30]; Engesser et al applied the cognitive programming to Multi-Agent system and solved program tasks in a decentralized way, by expanding the cognitive programming framework and the perspective conversion [31].

Reinforcement Learning is one of algorithms to solve cooperation problems among agents which treats Cooperative Game as the core issue in the cooperation research. Agents cooperate by delivering message with each other to achieve the mutual objective [32]. In game mechanism, mechanisms that promote cooperation are divided into categories of strong-weak reciprocity, network reciprocity and group selection [33], [34]. Elise discussed the internal mechanism of strong reciprocal behaviors: when treachery appeared in an agent group, the strong reciprocal individual would conduct a altruism punishment to the individual who betrayed others, which made betraying individual to be more cooperative [35]; Perolat et al modeled subjects who occupied public resources using Markov Observation Model. The model revealed relations among exclusiveness, sustainability and inequality and demonstrated the solution, which improved efficiency of resource management [36]; Tuyls et al put forward the LDQN algorithm, which imported toleration policy to Deep Q-network that updated passive policy with leniency methods, thus to improve convergence and stability of Multi-Agent cooperation algorithm [37]; Hwang et al combined the multi-agent cooperative Q-learning algorithm with Stochastic recording real-valued unit to solve the problem that the agent is prone to fall into local solution [38].

2.3. Constrained Markov Game

Reinforcement learning, based on MDP, consists of an agent and several states. While MARL is a game of stochastic multi-player cooperation game, including more than one player and state, based on which a Markov game (stochastic) is defined [39]. Markov game is expressed as a tuple $(n,S,A_1,\cdots,A_n,T,\gamma,R_1,\cdots,R_n)$ [40], where n is the number of players, S is the set of states. The state here is referred as the union state of all players at a certain time instant. $T:S\times A_1\times\cdots\times A_n\times S\rightarrow[0,1]$ is the transition function, $A_i(i=1,\cdots,n)$ is the action set of player i, $\gamma\in[0,1]$ is the discounted factor, $R_i: S\times A_1\times\cdots\times A_n\times S\rightarrow\mathbb{R}$ is the reward function of player i. In a stochastic game, the transition function probability of next states is determined the current state and the action taken by the player. A reward function $R_i(s,a_1,\cdots,a_n,s')$ denotes the reward obtained by the player after the player takes the union action $(a_1,\cdots,a_n)$ and transfers from state s to state s'. Similar with MDP, the stochastic game also possesses markov property [41], that is, the next state and reward of the player depends only on the current state and current action of all players. Multi-Agent reinforcement learning model is shown as figure 2 [42].

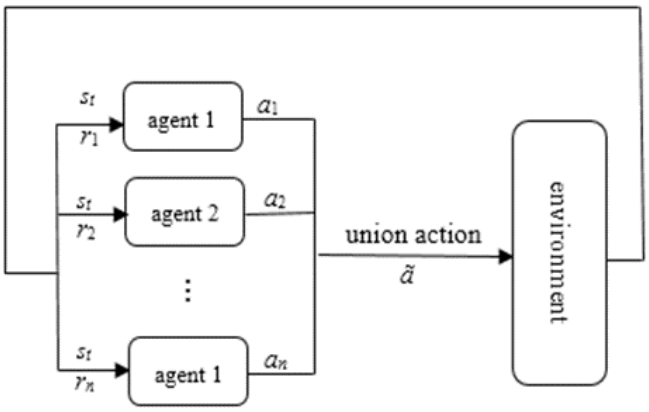

Fig. 2. Model of multi-agent reinforcement learning

Constrained Markov game adds constraint function to the Markov game, which is denoted by a tuple (n,S,A_1,$\cdots$,A_n,T,γ,R_1,$\cdots$,R_n,C), where C is the constraint function set. In constrained MDP model, the goal of the agent is changed into maximizing the reward function of the whole plot under condition that the agent satisfies the constraints. The constraint function set is shown as follows:

$$C = \left\{ c_{ij} : S \times A_i \times \cdots \times A_n \to \mathbb{R} \mid j = 1 \cdots k \right\} \tag{9}$$

where k is constant and denotes the number of constraint functions.

2.4. Lagrange Multiplier Method

Adding constraints to the objective function can ensure the safety of the agent in the learning process. Lagrange multiplier method is to find optimal solution in the case of equality constraint. Optimization problem with inequality constraint can be solved using Lagrange multiplier method and Karush Kuhn-Tucker conditions (KKT) [43]. KKT is a sufficient and necessary condition only when the model is convex, otherwise it is only the necessary condition when used to decide whether the solution obtained by Lagrange multiplier method is optimal [45].

$$\begin{aligned} &\max\ g(y) \\ s.t.\ \ &c_j(y) = C_j,\ \ j = 1, 2, \cdots k' \\ &c_j(y) \le C_j,\ \ j = k' + 1, \cdots, k \end{aligned} \tag{10}$$

The above model is the optimization problem with inequality constraints, where $c_j(y) = C_j$ and $c_j(y) \le C_j$ are abstract expressions of constraint function, γ is discount factor, $0<\gamma<1$, k is constant and denotes the number of constraint funtions. k' is constant and denotes the number of equality constraint functions. The number of inequality constraint functions is denoted by k-k'. The optimal solution satisfies $\lambda=0$ or $c_j(y) - C_j = 0,\ j = k' + 1, \cdots, k$, such that when y satisfies the strict inequality, the

constraint functions are not effective. Only when the constraint functions are equality constraints can the constraint functions be effective. Therefore, the optimization problem with inequality constraints will be turned into the problem with equality constraints and then be solved using Lagrange multiplier method, which simplifies the problem [44]. The model is shown as follows:

$$\max L(y,\lambda) = g(y) - \lambda_j (c_j(y) - C_j) - \lambda_j^{'} (c_j^{'}(y) - C_j^{'}) \tag{11}$$

where the independent variable y is a local optimal solution when satisfying $\nabla_y L(y,\lambda) = 0$ and $\nabla_\lambda L(y,\lambda) = 0$ [45], which is obtained using gradient descent method. When the model is convex, the local optimal solution is the same as the global optimal solution. λ_j is lagrange undetermined multipliers, which denote changes of objective functions in accordance with constraint functions. Since the optimal solution satisfies the constraint $c_j(y) - C_j = 0$, the value of λ_j will not affect the final solution.

3. Constrained Multi-Agent Cooperation Q-learning

Traditional Multi-Agent learning algorithms ignore the safety issues for simplicity while the safety issues have been proved to be crucial for completing the task. To handle the safety problem of Multi-Agent system, we propose Constrained Multi-Agent Cooperation Q-learning algorithms that encode the safety requirement as constraints to ensure a stable Multi-Agent system.

3.1. Model Design

CMACQ model is described by the tuple (n,S,A_1,$\cdots$,A_n,T,γ,R_1,$\cdots$, R_n, C) which is shown as figure 3. In the current state, the state-value function of agent i is:

$$\begin{aligned} V_{\pi^i}(s) &= E[R_t \mid s_t = s] \\ &= E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid s_t = s\right], i = 1,\cdots,n, \forall s \in S \\ s.t.\, c_j(s_t, a_t^i) &= C_j,\ j = 1,\cdots,k^{'},\ i = 1,\cdots,n \\ c_j(s_t, a_t^i) &\le C_j,\ j = 1,\cdots,k,\ i = 1,\cdots,n \end{aligned} \tag{12}$$

Given the current state and the union action of all players, the state-value function of agent i is:

$$\begin{aligned} Q_{\pi^i}(s, a^1, \cdots, a^n) &= E\left[R_t \mid s_t = s, a_t^1 = a^1, \cdots, a_t^n = a^n\right] \\ &= E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid s_t = s, a_t^1 = a^1, \cdots, a_t^n = a^n\right] \\ & i = 1,\cdots,n,\ \forall s \in S \end{aligned}$$

$$s.t.\, c_j\left(s_t, a_t^i\right) = C_j,\ j = 1,\cdots,k',\ i = 1,\cdots,n \tag{13}$$
$$c_j\left(s_t, a_t^i\right) \le C_j,\ j = 1,\cdots,k,\ i = 1,\cdots,n$$

In the constrained case, when the optimal policy $\pi_*^i, i = 1,\cdots,n$ is obtained, the corresponding optimal state-value function and optimal state-action value function are obtained at the same time, defined as equation (14) and (15).

$$V_{\pi_*^i}(s) = \max_{\pi^i} V_{\pi^i}(s),\ \forall s \in S \tag{14}$$

$$Q_{\pi_*^i}(s, a^1,\cdots,a^n) = \max_{\pi^i} Q_{\pi^i}(s, a^1,\cdots,a^n), \forall s \in S \tag{15}$$

Index set {1,2,⋯ ,k'} and {k'+1,k'+2,⋯ ,k} denote the equality constraint index set and inequality constraint index set respectively. State s_t denotes the state of the agent at time t, a_t^i denotes the action chosen by the agent i under the state s_t.

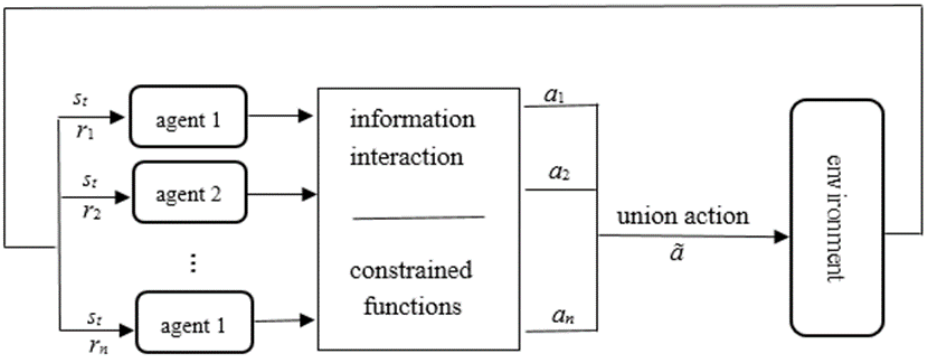

Fig. 3. Model of constrained multi-agent reinforcement learning

In CMACQ, by adding constraint conditions, the state set can be divided into safe state set and unsafe state set and the action set into safe and unsafe ones. By doing this, the safety problem of the agent can be solved at the beginning of the schedule and can also decide whether a state is safe or not using constrained conditions. Feasible region of CMACQ is shown as:

$$\overline{S} = \left\{ s \mid \overline{c_j}(s) = C_j, i = 1,\cdots,k'; \overline{c_j}(s) \le C_j,\ j = k'+1,\cdots,k \right\} \tag{16}$$

$$\overline{A}^i = \left\{ \begin{matrix} a^i \mid c_j(s,a^i) = C_j,\ j = 1,\cdots,k'; c_j(s,a^i) \le C_j, \\ j = k'+1,\cdots,k, i = 1,\cdots,n \end{matrix} \right\} \tag{17}$$

$\overline{c}_j(s_t) \le C_j$ and $c_j(s_t,a_t^i) \le C_j$ are standard forms of inequality constraints concerning state set and action set. If some states satisfies the forms $\overline{c}_j(s_t) \ge C_j$ and $c_j(s_t,a_t^i) \ge C_j$, these forms can be turned into standard forms by multiplying both sides of inequations with -1. For inequality constraints, if there exists $j_0 \in \{k'+1,\cdots,k\}$, such that $\overline{c}_{j_0}(s_t) < \overline{C}_{j0}$ and $c_{j0}(s_t,a_t^i) < C_{j0}$, the i_0th constraint in state s_t is not effective, and can be removed. The effective constraint set is denoted as ξ. Under the effective constraints, CMACQ model can be described as follows:

$$\arg\max_{a_{t+1}^i} Q^i\left(s_{t+1}, a_{t+1}^1, \cdots, a_{t+1}^n\right) \leftarrow Q^i\left(s_t, a_t^1, \cdots, a_t^n\right)$$
$$+\alpha\left(R_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}^1, \cdots, a_{t+1}^n\right) - Q\left(s_t, a_t^1, \cdots, a_t^n\right)\right)$$
$$s.t.\, c_j\left(s_t, a_t^i\right) = C_j,\ j \in \xi, i = 1, \cdots, n \tag{18}$$

The above learning model includes only equality model so as to simplify the model and improve the performance of the algorithm.

3.2. Model Solution

CMACQ algorithm based on constrained Markov game ensures the safety of all agents by adding multi-dimensional constraints, which makes traditional solutions no more suitable. To solve CMACQ model accurately and effectively, in this paper we adopt Lagrange multiplier method to decide all the safe and optimal actions for the agent under the current state. Lagrange multiplier method requires that the objective function and the constraint function are first order continuous differentiable. The objective function satisfies the condition when time t is continuous. The constraint function is not necessarily first order continuous differentiable, which can be solved by linearization of the constrained function. Since the next state of agent is decided by the current state and union action of all agents, it can be concluded that:

$$s_t^{'} = f(s_t, a_t^1, \cdots, a_t^n) \tag{19}$$
$$\overline{c_j}(s^{'}{}_t) \doteq c_j\left(s_t, a_t^i\right), i = 1, \cdots, n,\ \ j \in \xi \tag{20}$$

When solving the model, to ensure that the solution is globally optimal, the objective function and constraint function need to be convex. It can be known from the model that the objective function is convex while the constraint function is not. Therefore, the constraint function is linearized. Since linear function is always convex, so is the linearized constraint function. By doing this the globally optimal solution is guaranteed. Linear approximation of constraint function is shown as:

$$c_j(s_t, a_t^i) \approx \overline{c_j}(s_t) + g\left(s_t; \omega_j\right)^T a_t^i, i = 1, \cdots, n,\ \ j \in \xi \tag{21}$$

In the above equation, $g(s_t;\omega_j)$ takes st as input, and output a vector sharing the same dimension with a_t^i, which can be obtained by solving the following equation:

$$\arg\min_{\omega_j} \sum_{(s_t, a_t^1, \cdots, a_t^n, s_t^{'}) \in D} \left(\overline{c}_j(s_t^{'}) - \left(\overline{c}_j(s_t) + g\left(s_t; \omega_j\right)^T a_t^i\right)\right)^2$$
$$D = \left\{\left(s_t, a_t^1, \cdots, a_t^n, s_t^{'}\right)\right\}, i = 1, \cdots, n,\ \ j \in \xi \tag{22}$$

where set D is composed of tuples (s_t,a_t^1,$\cdots$,a_t^n,s_t') denoting that agent i, under the current state s_t, executes action a_t^i and switches to next state s_t'. Optimal solution of the objective function is contained in set D.

After implementing linear approximation, the learning model is shown as:

$$\begin{aligned}&\arg\max_{a_{t+1}^i} Q^i\left(s_{t+1}, a_{t+1}^1, \cdots, a_{t+1}^n\right) \leftarrow Q^i\left(s_t, a_t^1, \cdots, a_t^n\right) \\ &\quad + \alpha\left(R_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}^1, \cdots, a_{t+1}^n\right) - Q\left(s_t, a_t^1, \cdots, a_t^n\right)\right) \\ &s.t. c_j(s_t, a_t^i) \approx \overline{c_j}(s_t) + d\left(s_t; \omega_j\right)^T a_t^i = C_j,\ j \in \xi, i = 1, \cdots, n\end{aligned} \tag{23}$$

Utilizing Lagrange multiplier method, the following equations should be solved:

$$\begin{aligned}&\arg\max_{a^*}\{Q^i\left(s_{t+1}, a_{t+1}^1, \cdots, a_{t+1}^n\right) \leftarrow Q^i\left(s_t, a_t^1, \cdots, a_t^n\right) \\ &+\alpha\left(R_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}^1, \cdots, a_{t+1}^n\right) - Q\left(s_t, a_t^1, \cdots, a_t^n\right)\right) \\ &-\sum_{j \in \xi} \lambda_j \left(\overline{c_j}(s_t) + d\left(s_t; \omega_j\right)^T a_t^i - C_j\right)\}, i = 1, \cdots, n,\ \ j \in \xi\end{aligned} \tag{24}$$

To avoid falling into locally optimal solution, the linearized constrained function is adopted to guarantee the solution in globally optimal.

3.3. Algorithm Description

In CMACQ, agents cooperate with each other to achieve the goal. During the learning process, under the condition that the constraints are satisfied, the agent chooses an action based on its current state and on observing the action-value function. The model ensures the safety of all agents through constraint function. That a state is safe means that all agents are safe.

Algorithm 1: Constrained Multi-Agent Cooperation Q-Learning

Input: state set *S*, union action set *A*, and reward function

Output: safety state sequence and safety union action corresponding to each safety state sequence after training

1: Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, $\gamma \in (0,1]$

2: **Initialize:**

a)state-action value function $Q^i\left(s, a^1, \cdots, a^n\right), \forall s \in S, a^i \in A^i, i = 1, \cdots, n$

b)Lagrange multiplier $\lambda_j,\ j = 1, \cdots, k, k \in N^+$

c)parameter ω_j

d) $D = \left\{\left(s, a^1, \cdots, a^n, s^{'}\right)\right\}, \forall s, s^{'} \in S, a^i \in A^i, i = 1, \cdots, n$

3: **Loop forever (**for each episode):

4: Initialize initial state *s*

5: **Loop forever (**for each step of episode):

6: Obtain $g\left(s; \omega_j\right)$ by solving the formula

$$\omega_j^* \leftarrow \arg\min_{\omega_j} \sum_{(s, a^1, \cdots, a^n, s^{'}) \in D} \left(\overline{c_j}(s^{'}) - \left(\overline{c_j}(s) + g\left(s; \omega_j\right)^T a^i\right)\right)^2, \forall a^i \in A^i, i = 1, \cdots, n$$

7: The constraint function is approximated linearly: $c_j \leftarrow \overline{c_j}(s) + g(s, \omega_j^*)^T a^i, \forall a^i \in A^i, i = 1, \cdots, n$

8: The Lagrange multiplier method is used to obtain the action $a_*^i \leftarrow \arg\max_{a^i} \{Q^i(s, a^1, \cdots, a^n) - \sum_{j \in \xi \cup \zeta'} \lambda_j (c_j - C_j)\}, \forall a^i \in A^i, i = 1, \cdots, n$

9: Agent i take action $a_*^i, i = 1, \cdots, n$, observe $r^i, i = 1, \cdots, n$, next state s',

10: $Q^i(s, a_*^1, \cdots, a_*^n) \leftarrow Q^i(s, a_*^1, \cdots, a_*^n) + \alpha \left[R_{t+1} + \gamma \max_{a^1, \cdots, a^n} Q^i(s', a_*^1, \cdots, a_*^n) - Q^i(s, a_*^1, \cdots, a_*^n) \right]$

11: Agent $i, i = 1 \cdots, n$ moves to the next state: $s \leftarrow s'$

12: **Until** s is terminal

Each s of state set S represents a union state $s=(s_1,s_2,\ldots,s_n)$, which is different from action set. In $a^i \in A^i$, a^i represents available actions of agent i, and A^i represents all the available actions. According to steps 6-8 in the above algorithm, the constraint function is solved using Lagrange multiplier method so as to decide all the safe and executable actions that each agent can choose under the condition that the long-term cumulative reward is obtained. λ_j is Lagrange multiplier. Step 6 and step 7 describe the linear approximation constraint function to ensure that the constraint functions are differentiable. If the initial constraint function is differentiable, then these two steps are skipped. In step 8, each agent uses Lagrange multiplier to work out the optimal action value under constraints. In step 9, each agent is in safe state after choosing the safe and optimal action. CMACQ ensures the safety of each agent under the condition that the globally optimal solution is obtained.

4. Experiment and Analysis

CMACQ algorithm based on constrained Markov game is suitable for determining a policy for multiple agents under a constrained condition and obtaining the optimal long-term cumulative reward. The constrained algorithm introduced in this paper is mainly to solve safety problem of multiple agents. In the experiment, a dangerous state is defined as a state that causes great damage when chosen by the agent.

To simplify the solution procedure, if problem is discrete and state set and action set are relatively small, the next state can be judged to be safe or not through only the constraint function and can decide all suitable actions for the agent. If the state set and action set are large and are even continuous, the safe state is calculated through Lagrange multiplier method.

To ensure the safety of each agent, CMACQ adopts constraint function to prevent each agent form falling into a dangerous state. The distance between the agent's current state and dangerous state is measured by Manhattan distance [46]. To ensure the safety of the agent, the Manhattan distance between the agent's next state and the dangerous state should be equal or greater than 1. The distance described above is defined as safe distance d, as showed below:

$$d(s_{t+1}, \tilde{S}) > 0 \tag{25}$$

where $\tilde{S}$ is the set of all dangerous states, s_{t+1} is the next state for the agents.

4.1. Firefighting through Multi-Agent Cooperation Experiment

The environment of the firefighting through multi-agent cooperation experiment is a 10×10 grid world with 3 agents starting respectively at (0,9), (9,0), (9,9). In the experiment, agents cooperate with each other to complete firefighting missions, 9 origins of fire are (1,4), (3,3), (4,9), (9,3), (8,9), (7,3), (4,6), (5,2), (9,7). Each agent carries the fire-fighting equipment, and enters into a fire point, quells the fire and receives a reward of 20. Agents starting from (0, 9) and (9, 0) are able to quell four firing places while the agent starting from (9,9) is able to quell 3 firing places. When firefighting materials are used up, agents stop firefighting and go back to starting points. When all 9 firing points are quelled, a whole plot is terminated. Agents cooperate with each other to find out firing points that are not yet quelled. In the process, the agent may encounter 3 dangerous states, located at (4, 3), (8, 2), (5, 6). When the agent enters into a dangerous state, it suffers from permanent damage and receives a reward of -100. To avoid that agents take random walks in the grid world, agents receive a reward of -1 when entering into a new state other than the firing state and dangerous state. By doing this, agents are able to find the shortest path toward firing place in the shortest period of time.

In the experiment, step size α is set uniformly to 0.5, discounted factor γ is set to 1. ε-greedy method is adopted to explore actions in the training process so as to avoid the locally optimal solution. Policy parameter ε is set to 0.1. The experiment is independently operated for 50 times with 500 plots for each operation.

Figure 4 demonstrates performance of CMACQ. The result shows that agent 3 possesses the highest firefighting speed and cumulative reward for each plot. Agent 2 and agent 1 possess similar cumulative rewards with agent 2 being slightly better which is due to the distribution of firing points and the number of firefighting materials carried by the agent. The overall distribution is closer to the starting point of agent 3 and agent 3 carries only 3 units of firefighting materials, which are 1 unit lesser than those of agent 1 and agent 2, Therefore agent 3 completes the goal in the shortest period of time. Agent 3 need not wander in the grid to search for firing points such that it receives the highest long-term cumulative reward for each plot.

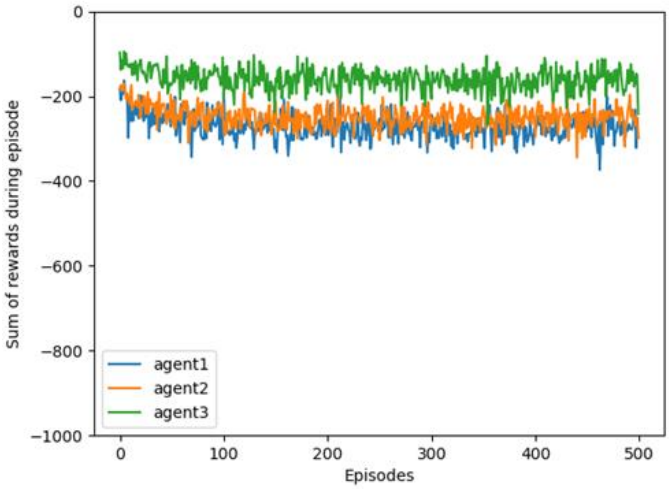

Figure. 4. Performance of CMACQ.

Figure 5 shows long-term cumulative rewards for each plot without safety constraints. The order of rewards for each agent is similar with that of Figure 4, but each reward in Figure 5 is lower than that of Figure 4. The lower rewards are because that agents fall into dangerous states and receive a reward of -100.

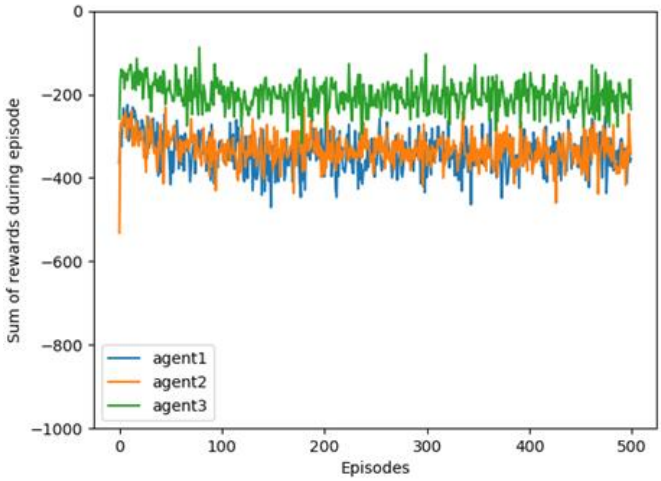

Figure. 5. Performance of MACQ

Figure 6 compares the average long-term cumulative rewards per plot between CMACQ and MACQ [38]. The result shows that CMACQ behaves better than MACQ and that CMACQ enables the agent to avoid dangerous states.

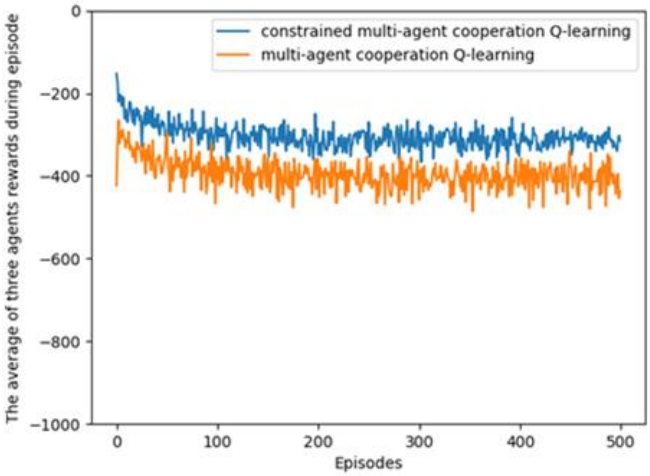

Figure. 6. Average long-term cumulative rewards per plot of two algorithms.

4.2. Deep Sea Fishing Experiment

The environment of Deep Sea Fishing experiment is a 12×12 grid word with 2 agents and agents work together to catch fish while steering clear of obstacles in the sea. Agent 1 starts at (0,0) and agent 2 starts at (0,11). In the process, there are 6 dangerous states, located at (3,0), (8,1), (10,4), (11,7), (4,8), (7,11). Dangerous states simulate rocks on the bottom of the sea, sea grass and so on. In the experiment, agents need to avoid hitting rocks and getting entangled in sea grass in order to avoid irreversible damage, continue to fish and return from the sea. States of shoal of fish locate at (7,0), (5,2), (10,5), (4,6), (7,8), (11,9), (5,11). There are one unit of fish in each state of shoal of fish. Resources carried by one agent that can catch 4 units of fish. If one agent catches 4 units of fish, it can no longer fish and leave the sea.

In the experiment, the agent receives a reward of 20 when it catches 1 unit of fish. When the agent enters into a dangerous state, it suffers from irreversible damage and receives a reward of -100. To avoid that the agent takes random walks in the grid world, agents receive a reward of -1 when entering into a new state other than the state of shoal of fish and dangerous state. By doing this, agents are able to find the shortest path toward shoal of fish location in the shortest period of time.

In the experiment, step size α is set uniformly to 0.5, discounted factor γ is set to 1. The ε-greedy method is adopted to explore actions in the training process so as to avoid the locally optimal solution. Policy parameter ε is set to 0.1. The experiment is independently operated for 50 times with 500 plots for each operation.

Figure 7 shows the experimental results and compares long-term cumulative rewards for each plot solved by CMACQ and MACQ respectively. The experimental results show that CMACQ behaves better than MACQ, because CMACQ enables the agent to avoid dangerous states and ensures the safety of the agent. In the same algorithm, the

difference between the long-term cumulative rewards per plot of agent 1 and those of agent 2 is small, because the danger state and the state of shoal of fish are uniform distribution, while agent 1 and agent 2 enter the experimental environment from the upper left corner and upper right corner respectively.

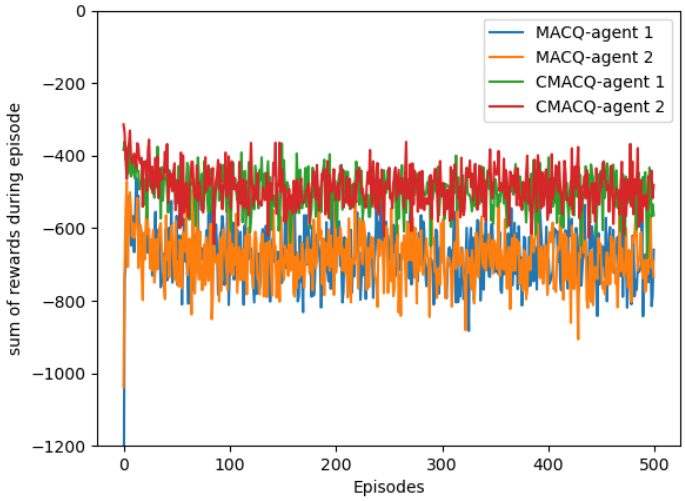

Figure. 7. Comparison of results of Deep Sea Fishing experiment of two algorithms.

The reason why the experimental result of CMACQ algorithm are better than that of MACQ algorithm is that the CMACQ algorithm can avoid the agents from entering the dangerous state and causing irreversible damage, and the safety of agents is guaranteed by the constraint conditions.

5. Conclusion

Multi-Agent system is widely applied in real world in areas such as robot system, distributed decision, traffic control, business management and so on. Reinforcement learning algorithm is a key method for solving Multi-Agent problem. However, traditional algorithms ignore the safety problem of Multi-Agent system. The agent may fall into dangerous states and suffer from great damage. To solve this problem, in this paper we introduce CMACQ algorithm based on constrained Markov game and test this method through the firefighting cooperation experiment. The result shows that CMACQ is able to handle the safety problem.

The CMACQ algorithm presented in this paper guarantees the safety of Multi-Agent system; it can also be applied to the problems where resource or cost is constrained and the area of Multi-Agent cooperation, such as robot system and traffic control, where agents work with each other and are under certain constraints. In future work we will adopt constrained algorithm to solve problems concerning limited resource, minimization of the cost and multiple objectives etc.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (61303108), The Natural Science Foundation of Jiangsu Higher Education Institutions of China (17KJA520004), Suzhou Key Industries Technological Innovation-Prospective Applied Research Project (SYG201804), the Program of the Provincial Key Laboratory for Computer Information Processing Technology (Soochow University) (KJS1524), and a Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

1. Buşoniu, L., Babuška, R., Schutter, B. D.: Multi-agent Reinforcement Learning: An Overview. Innovations in Multi-Agent Systems and Applications, Vol. 38, No. 2, 156-172. (2010)
2. Babuš ka, R., buşoniu, L., and De Schutter, B.: Reinforcement learning for multi-agent systems. IEEE International Conference on Emerging Technologies and Factory Automation. IEEE, 1-7. (2006)
3. Dimeas, Hatziargyriou.: Multi-agent reinforcement learning for microgrids. Power & Energy Society General Meeting. IEEE, 25-29. (2010)
4. Zhang, K., Yang, Z., Liu, H., et al.: Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. Proceedings of the 35th international Conference on Machine Learning. ICML 2018, Stockholmsmässan, Stockholm, Sweden, 1-10. (2018)
5. Littman, M. L.: Markov games as a framework for multi-agent reinforcement learning. New Brunswick: Machine Learning Proceedings, 157-163. (1994)
6. Foerster, J., Assael, I., De Freitas, N., et al.: Learning to communicate with deep multi-agent reinforcement learning. Advances in Neural Information Processing Systems. Spain: NIPS Press, 2137-2145. (2016)
7. Leibo, J., Zambaldi, V., Lanctot, M., et al.: Multi-agent reinforcement learning in sequential social dilemmas. Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems. Singapore: AAMAS Press, 464-473. (2017)
8. Lowe, R., Wu, Y., et al.: Multi-agent actor-critic for mixed cooperative-competitive environment. Advances in Neural Information Processing Systems. Los Angeles: NIPS Press, 6379-6390. (2017)
9. Littman, M.: Value-function reinforcement learning in Markov games. Cognitive Systems Research, Vol. 2, No. 1, 55-66. (2001)
10. Lauer, M., Riedmiller, M.: An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems. Seventeenth International Conference on Machine Learning. Stanford: Morgan Kaufmann Press, 535-542. (2000)
11. Lanctot, M., Zambaldi, V., Gruslys, A., et al.: A unified game-theoretic approach to multi-agent reinforcement learning. Advances in Neural Information Processing Systems. Los Angeles: NIPS Press, 4190-4203. (2017)
12. Bing, S., Zhu, H., Wang, J., et al.: Optimize Pricing Policy in Evolutionary Market with Multiple Proactive Competing Cloud Providers. IEEE International Conference on Tools with Artificial Intelligence. (2017)
13. Kofinas, P., Dounis, A. I., Vouros, G. A.: Fuzzy Q-Learning for multi-agent decentralized energy management in microgrids. Applied Energy, Vol. 219, No. 3, 53-67. (2018)
14. Vidhate, D. A., Kulkarni, P.: Cooperative multi-agent reinforcement learning models (CMRLM) for intelligent traffic control. 2017 1st International, Conference on Intelligent Systems and Information Management (ICISIM). India: IEEE Press, 325-331. (2017)
15. Hu, Junling, Wellman, et al.: Nash q-learning for general-sum stochastic games. Journal of Machine Learning Research, Vol. 4, No. 4, 1039-1069. (2003)

16. Greenwald, A., Hall, K., Serrano, R.: Correlated Q-learning. ICML. Washington: ICML Press, 242-249. (2003)
17. Kononen, V.: Asymmetric multi agent reinforcement learning. International Conference on Intelligent Agent Technology. Canada: IEEE Press, 336-342. (2003)
18. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature, Vol. 518, No. 7540, 529-533. (2015)
19. Sutton, Richard, S. and Barto, Andrew, G.: Reinforcement learning: An introduction. MIT press Cambridge. (2018)
20. Garc, Javier, A., Ndez, F. A.: Comprehensive survey on safe reinforcement learning. JMLR.org, 1437-1480. (2015)
21. C. J. C. H. Watkins, and Dayan, P.: Q-learning. Machine Learning, Vol. 8, 279-292. (1992)
22. Schaul, T., Quan., Antonoglou, I., et al.: Prioritized Experience Replay. Computer Science, 1-21. (2016)
23. Golden, R. M.: Adaptive Learning Algorithm Convergence in Passive and Reactive Environments. Neural Computation, 1-28. (2018)
24. Luo, B., Liu, D., Huang, T., et al.: Model-Free Optimal Tracking Control via Critic-Only Q-Learning. IEEE Transactions on Neural Networks and Learning Systems, 1-11. (2016)
25. Langergraber, K. E., Watts, D. P., Vigilant, L., et al.: Group augmentation, collective action, and territorial boundary patrols by male c himpanzees. Proc Natl Acad Sci U S A, Vol. 114, No. 28, 7337-7342. (2017)
26. Ma, Y., Lu, J., Shi, L.: Diversity of neighborhoods promotes cooperation in evolutionary social dilemmas. Physica AStatistical Mechanics & Its Applications, Vol. 468, 212-218. (2017)
27. Rand, D. G., Dreber, A., Ellingsen, T., et al.: Positive interactions promote public cooperation [J]. Science, Vol. 325, No. 5945, 1272-5. (2009)
28. Milinski, M., Semmann, D., Krambeck, H. J.: Reputation helps solve the 'tragedy of the commons'. Nature, Vol. 415, No. 6870, 424-6. (2002)
29. Allen, B., Lippner, G., Chen, Y. T., et al.: Evolutionary dynamics on any population structure. Nature, Vol. 544, No. 7649, 227-230. (2017)
30. Mcavoy, A., Fraiman, N., Hauert, C., et al.: Public goods games in populations with fluctuating size. Theoretical Population Biology, Vol. 121, 72-84. (2018)
31. Engesser, T., Bolander, T., Mattm, R., et al.: Cooperative Epistemic Multi-Agent Planning with Implicit Coordination. Distributed and Multi-Agent Planning. (2015).
32. Matignon, L.， Laurent, G. J.， Fort-Piat, N. L.: Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System (IROS). IEEE， San Diego, California, 64-69. (2007)
33. Coultas, J. C., Leeuwen, E. J. C. V.: Conformity: Definitions, Types, and Evolutionary Grounding. Evolutionary Perspectives on Social Psychology, 189-202. (2016)
34. Gintis, H.: Strong reciprocity and human sociality. Journal of Theoretical Biology, Vol. 206, No. 2, 169-179. (2000)
35. Seip, E. C., Dijk, W. W. V., Rotteveel, M.: Anger motivates costly punishment of unfair behavior. Motivation & Emotion, Vol. 38, No. 4, 578-588. (2014)
36. Perolat, J., Leibo, J. Z., Zambaldi, V., et al.: A multi-agent reinforcement learning model of common-pool resource appropriation. Advances in Neural Information Processing Systems. Los Angeles: NIPS Press, 3643-3652. (2017)
37. Palmer, G., tuyls, K., Bloembergen, D., et al.: Lenient multi-agent deep reinforcement learning. Proceedings of the 17th International Conference on Autonomous Agent and Multi-Agent Systems. Swede: AAMAS press, 443-451. (2018)
38. Hwang K., Lin Y. and Lo C.: Multi-Agent Cooperation by Q-Learning in Continuous Action Domain. 2008 First International Conference on Intelligent Networks and Intelligent Systems. China, 111-114. (2008)

39. Chalmers, R., Scheidt, D., Neighoff, T., Witwicki, S., and Bamberger, R.: Cooperating unmanned vehicles. AIAA 1st Intelligent System Technical Conference. (2004)
40. Ying-ying, D., Yan, H., and Jing-ping, J.: Self-organizing multi-robot system based on personality evolution. IEEE International Conference on Systems, Man, and Cybernetics. (2002)
41. Sidney, G.: Analysis and Design of Swarm Based Robots Using Game Theory. Ph. D. Thesis, Ottawa, ON: Carleton University. (2009)
42. Nowé A., Vrancx P., De Hauwere YM.: Game Theory and Multi-agent Reinforcement Learning. In: Wiering M., van Otterlo M. (eds) Reinforcement Learning. Adaptation, Learning, and Optimization, Springer, Berlin, Heidelberg, vol. 12, 441-470. (2012)
43. Martyna, J.: Power Allocation in Cognitive Radio with Distributed Antenna System. Lecture Notes in Computer Science, 745-754. (2017)
44. Kuan, T. W., Wang, J. F., and Wang, J. C., et al.: VLSI Design of an SVM Learning Core on Sequential Minimal Optimization Algorithm. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 20, No. 4, 673-683. (2012)
45. Farina, F., Garulli, A., and Giannitrapani, A., et al.: Asynchronous Distributed Method of Multipliers for Constrained Nonconvex Optimization. 2018 European Control Conference. Limassol, Cyprus, Vol. 103, 243-253. (2018)
46. Blackburn, S. R., Homberger, C., Winkler, P.: The minimum Manhattan distance and minimum jump of permutations. Journal of Combinatorial Theory, Series A, Vol. 161, 364-386. (2019)

Yangyang Ge is a postgraduate of School of Computer Science and Technology, Soochow University. Her main research interests include safe reinforcement learning. E-mail: 20184227043@stu.suda.edu.cn.

Fei Zhu (corresponding author) got Ph.D. and associate professor in School of Computer Science and Technology, Soochow University. He is a member of China Computer Federation. His main research interests include reinforcement learning, text mining, and pattern recognition. E-mail: zhufei@suda.edu.cn. He is the corresponding author of this paper.

Wei Huang is associate professor in School of Computer Science and Technology, Soochow University. Her main research interests include reinforcement learning and pattern recognition. E-mail: huangwei@suda.edu.cn.

Peiyao Zhao is a postgraduate student in the Soochow University. His main research interest is reinforcement learning. He programmed the algorithms and implemented the experiments. E-mail: 20195427013@stu.suda.edu.cn.

Quan Liu got Ph.D. and professor in School of Computer Science and Technology, Soochow University. His main research interests include intelligence information processing, automated reasoning and machine learning. E-mail: quanliu@suda.edu.cn.

Received: December 20, 2019; Accepted: May 02, 2020