Decentralized Multi-agent Reinforcement Learning with Multi-time Scale of Decision Epochs

Junjie Wu, *Student Member, IEEE,* Kuo Li, *Student Member, IEEE,* Qing-Shan Jia, *Senior Member, IEEE*

***Abstract*— Multi-agent reinforcement learning (MARL) has attracted more and more attention in recent years. It is now widely applied in various fields, including cyber physical systems, smart grid, finance, social network, and among others. The current researches on MARL mainly focus single-time scale, in which the agents have the same decision epoch. While in real applications, it is common that the agents make decisions by different frequencies. In addition, different agents may have separate roles in the system. In this paper, we propose a more general MARL framework by introducing multi-time scale of decision epochs. We assume that agents share information with their neighbors, including state, action, and reward. The global observability of state and action, which is a common assumption, is not required. We propose a decentralized Q-learning algorithm and a modified MADDPG algorithm to solve the problem. The main contributions of this paper are as follows. *First*, we formulate the multi-time scale multi-agent reinforcement learning (MTMARL) problem. This provides a general framework for the related systems and problems. *Second*, we provide a networked decentralized multi-time scale multi-agent Q-learning algorithm to solve the problem and prove its convergence. *Third*, we test the algorithm numerically. The results show that the proposed algorithm performs better than the previous QD-learning and is only slightly worse than the centralized algorithm.**

***Index Terms*— Multi-time scale, multiple decision epochs, multi-agent, reinforcement learning.**

I. Introduction

Reinforcement learning (RL) [1] enjoys significant advances in the past decade in various areas, including playing video games [2], game of Go [3], [4], autonomous driving [5], robotic control [6], [7], and among others. RL has been extended to multi-agent framework, where there are multiple agents in the system and each agent aims to optimize its policy to achieve maximum reward (or minimum cost) by interacting with other agents and environment [8], [9]. In recent years, multi-agent reinforcement learning (MARL) has achieved successes in more general applications, including cyber physical systems [10], smart grid [11], and social science [12]. MARL plays an increasingly important role in sequential decision-making problems.

With multiple agents interacting with each other, MARL is more challenging than the single agent RL. Some difficulties of MARL are as follows. *First*, divergent objectives. The agents may play different roles and have various objectives. The unaligned objectives result in the challenge of dealing with equilibrium points. *Second*, non-stationary. Each agent takes an action according to the local observation. Since the actions of other agents are not controllable, the system appears to be non-stationary in the perspective of each agent.

Some techniques are introduced to tackle the above difficulties. We categorize them into two groups, value sharing and parameter sharing, which will be reviewed in Section II. Many of the MARL algorithms require global observability of state and action of all the agents, which is often not practical in real applications. Some algorithms that use parameter sharing technique require that the value function or network have the same structure. Otherwise the parameters are not sharable. The current researches on MARL focus on single-time scale problem. The agents have the same decision frequencies. In many real problems, agents have different decision epochs. For example, the day-ahead and real time energy dispatching problem is inherently a multi-time scale problem [13]. Agents (micro-grid) have two decision frequencies, day (daily) and real time (maybe hourly).

In this paper, we extend the single-time scale MARL to multi-time scale. We allow agents to have different decision epochs and do not require global observability of states and actions, which is a common assumption in the existing literatures. We do not assume a central optimizer, and only require that the agents communicate with their neighbors through the communication network and share the local state, action and reward. Agents optimize their policies distributively. We propose an algorithm to solve the multi-time scale multi-agent problem. We also modify the previous MADDPG algorithm. By the modification, the critic network uses only the local information, including states and actions of neighbor agents. Our algorithm achieves a completely distributed format of learning.

We make the following contributions in this paper. *First*, we formulate the multi-time scale multi-agent reinforcement learning (MTMARL) problem. This provides a general framework for the related systems and problems. *Second*, we provide a networked decentralized multi-time scale multi-agent Q-learning (MTMAQL) algorithm to solve the problem and prove its convergence. *Third*, we test the algorithm numerically. The results show that the proposed algorithm performs better than the previous QD-learning and is only slightly worse than the centralized algorithm.

The remainder of this paper is organized as follows. We briefly review the related literature in section II, present the multi-time scale MARL formulation in section III, propose

This work is supported by the National Key Research and Development Program of China (2016YFB0901900), in part by the National Natural Science Foundation of China under grants (No. 61673229), and the 111 International Collaboration Program of China (No. BP2018006).

J. Wu, K. Li, and Q.-S. Jia are with the Center for Intelligent and Networked Systems, Department of Automation, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, P. R. China (e-mail: wujj16@mails.tsinghua.edu.cn, li-k19@mails.tsinghua.edu.cn, jiaqs@tsinghua.edu.cn)

a decentralized multi-time scale MARL algorithm and prove its convergence in section IV, show the numerical results in section V, and make a brief conclusion in section VI.

II. Literature Review

The previous reviews [8], [9], [14] categorize multi-agent reinforcement learning into three groups, fully cooperative, fully competitive, and a mix of the two, depending on the types of settings they address. When it comes to the learning techniques of MARL, the current researches can be classified into two groups, value sharing and parameter sharing. In MARL, each agent optimizes its own policy. The global optimum is usually not achievable without communications among agents. Because each agent observes only the local information and tends to optimize its own reward. And this may not be beneficial to other agents. Therefore, the information sharing among the agents is necessary.

The value sharing means that the agents share values like Q-factors or rewards with other agents. In [15], the authors propose a Q-factor sharing technique based on Q-learning. An additional term $(-\beta\sum_{j\in\mathbb{N}_i}(Q_i(s,a)-Q_j(s,a)))$ considering the neighborhood Q-factors is introduced to the update equation. In [16], a multi-agent DDPG algorithm is proposed. The actor is the same as single agent DDPG, each agent select action according to its own observation. The critic network scores the value of the current state according to the global observation of states and actions of all agents. Similar to [16], [17] proposed a Q-learning based algorithm with global observation. A Voting-Based method is studied in [18]. It avoids direct reward sharing by introducing primal and dual variables to each agent.

The parameter sharing means that the agents share parameters of the estimated Q-functions/networks (or value functions/networks). This requires the approximation functions or networks of the agents have the same structure. In [19], the authors proposed a parameter consensus step during the training. Each agent updates the estimated network by a weighted sum of the parameters of its own and neighbor agents'. The algorithm is guaranteed to converge when the value functions are approximated within the class of linear functions. Similarly, a consensus-like algorithm for updating agents policy parameters is proposed in [20]. In [21], under the assumption that the global objective is a convex combination of each agent's local objective, higher asymptotic performance is achieved by parameters sharing among neighbor agents.

The existing literatures focus on the single-time scale MARL. We extend MARL to multi-time scale framework, which makes it more flexible to be implemented. With value sharing technique, we propose a complete decentralized Q-learning algorithm which is guaranteed to converge.

III. Problem Formulation

We consider a multi-time scale framework, in which the agents have different decision frequencies. We consider the simple problem with bi-level time scale, slow time scale and fast time scale. It can be easily extended to the multi-time scale case. We formulate the problem in Markov decision process (MDP), which is characterized by a quintuple $M=\{\mathbb{S},\mathbb{A},P,R,\gamma\}$, with each element representing state space, action space, transition probability, reward function and discount factor.

A. System State

Denote the time intervals between adjacent decision epochs in the slow time scale as ΔT and fast time scale as Δt. Assume that ΔT is K times the value of Δt, i.e. $\Delta T/\Delta t = K$. Fig. 1 shows the relationship of decision epochs between the slow and fast time scale.

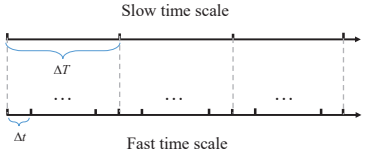

Fig. 1. Multi-time scale model

The state of slow time scale agent is denoted as $X^i_{t,k}$, where i is the index of the agent in slow time scale, $i=1,2,\ldots,N_s$, the lower script (t,k) means the k-th Δt-slot of the t-th ΔT-slot. Since the slow agents make decisions once every ΔT-slot, the states remain unchanged during decision making epochs, i.e. $X^i_{t,k}=X^i_{t,1},\forall k\in\{1,2,\ldots,K\}$. Similarly, the state of the fast time scale agent is denoted as $Y^j_{t,k}$, where j is the index of the agent in fast time scale, $j=1,2,\ldots,N_f$. Denote the set of slow and fast time scale agents as $\mathbb{N}_s$ and $\mathbb{N}_f$.

The state spaces of the slow and fast time scale are separately defined as $\mathbb{X}$ and $\mathbb{Y}$. We do not limit the state spaces of the two time scales to be identical.

B. Action and Policy

Define the action of the slow time scale agent as $b^i_{t,k}$ and the action space as $\mathbb{B}$. At the fast time scale, the action is defined as $c^j_{t,k}$ and the action space is $\mathbb{C}$. We do not restrict the action spaces of the two time scales to be identical.

A policy π is a mapping from the state to action $\pi: S\to a$. Denote the policy at the slow and fast time scale as π_s and π_f, and the corresponding policy space as Π_s and Π_f.

C. Reward and Objective Function

The reward function R_t is defined as the instant reward when an agent takes an action and transitions to a new state, i.e. $R_t=R(s,a,s')$. The reward functions in the slow and fast time scale are separately denoted as R^i_t and r^j_t.

The objective function is to maximize the discounted total reward of all the agents. Here γ_s and γ_f are the discount factor for the slow and fast time scale correspondingly. Both values range from $(0,1]$. And the relationship between the two is $\gamma_s=\gamma_f^K$.

$$\max_{\pi_s\in\Pi_s,\pi_f\in\Pi_f}\ \mathbf{E}\left[\sum_{t=0}^{T}(\gamma_s^t\sum_{i=1}^{N_s}R^i_t+\sum_{k=0}^{K-1}\gamma_f^{tK+k}\sum_{j=1}^{N_f}r^j_{tK+k})\right] \quad (1)$$

We aim to optimize the policies in both slow and fast time scale to achieve a maximal global reward.

IV. Main Results

In this section, we first introduce a method to transform the multi-time scale MARL to single-time scale one. By this transformation, the single-time scale algorithm can be implemented to tackle the problem. After that, we propose a Q-learning based multi-time scale MARL algorithm, which incorporates the Q-factor estimation in both time scales. We also mathematically prove that the proposed algorithm converges to global optimum.

A. Multi-time Scale to Single-time Scale

First, we transform the state and action spaces of the agents in different time scales to be identical. Note that the state spaces of slow and fast time scales are separately $\mathbb{X}$ and $\mathbb{Y}$. We combine the two state spaces to obtain an augmented common state space $\mathbb{S}$, i.e. $\mathbb{S}=\mathbb{X}\cup\mathbb{Y}$. Similarly, we obtain an augmented common action space $\mathbb{A}$, i.e. $\mathbb{A}=\mathbb{B}\cup\mathbb{C}$.

Second, we transform the multi-time scale framework into a flat single-time scale one featured by the fast time scale. To do this, we project the slow time scale to the fast. For the slow time scale agent, the time interval between two adjacent decision epoch is ΔT and $\Delta T = K\Delta t$. This means that the slow time scale agents do not do anything at every point of Δt-slot between the ΔT-slot. To fit the MDP in slow time scale to the fast time scale MDP, we restrict the action of slow time scale agents between the ΔT-slot to be empty.

By performing the above procedure, the transformation is done. The original multi-time scale problem is now a flat single-time scale problem. The time intervals between adjacent decision epochs of all the agent now are Δt. This formulation is the same as proposed in the existing literatures. The algorithms and results can be directly extended to our problem. We introduce the transformed MDP as follows.

1) System State: The state of agent is represented as

$$S_t^i = (X_t^i, Y_t^i), i = 1,2,\ldots,N_s+N_f, t = 1,2,\ldots,TK.$$

Specially, the state of the original slow time scale agent is $S_t^i = (X_t^i, \mathbf{0}), i \in \mathbb{N}_s$ and the state of the original fast time scale agent is $S_t^j = (\mathbf{0}, Y_t^j), j \in \mathbb{N}_f$.

2) Action and Policy: The action is represented as

$$a_t^i = (b_t^i, c_t^i), i = 1,2,\ldots,N_s+N_f, t = 1,2,\ldots,TK.$$

Specially, the action of the original slow time scale agent is $a_t^i = (b_t^i, \mathbf{0}), i \in \mathbb{N}_s$ and the state of the original fast time scale is $a_t^j = (\mathbf{0}, c_t^j), j \in \mathbb{N}_f$.

3) Reward and Objective Function: Since the agents in slow time scale make decisions at every ΔT time step. We restrict the action between two adjacent decision epochs to be no-action. The original slow time scale agent when $k \neq 0$ is penalized if the action is not empty.

The objective function is now represented as follows.

$$\max_{\pi\in\Pi} \quad \mathbf{E}\left[\sum_{t=0}^{TK}\gamma_f^t \sum_{i=1}^{N_s+N_f}(R_t^i + r_t^j)\right] \tag{2}$$

Transform the multi-time to single-time scale is an indirect way to solve the problem. When the state and action spaces of agents are different, the augmented space enlarges the search space, which makes it harder to obtain a good policy.

B. MTMAQL Algorithm

We now propose a direct way to solve the multi-time scale multi-agent RL problem. We do not require the state/action or the structure of the agents' neural networks to be identical. The global observability of agents' state and action is not required, either. On the other hand, we assume that the agents can distribute rewards and share state/action to their neighbors through the communication network. Note that the information sharing is localized. The agents can learn their policies in a complete decentralized format.

We first propose consensus reward. The consensus reward for each agent is a weighted sum of the rewards obtained by itself and its neighbor agents:

$$\hat{r}_t^i = \sum_{j\in\mathbb{L}_i} w(j,i) r_t^j + r_t^i, \tag{3}$$

where $\mathbb{L}_i$ is the set of neighbor agents of agent i, $w(j,i)$ is the weight factor of agent j to i. The weight factor matrix W is decided by the topology of the communication network. Denote d_i as the number of neighbors of agent i, i.e. $d_i = ||L_i||$. Denote C as the adjacency matrix of the agents, where $c(i,j)=1$ if agent i and j are neighbors, otherwise $c(i,j)=0$. Specially, $c(i,i)=0$. Let $D = \text{diag}(d_1^{-1}, d_2^{-1}, \ldots, d_N^{-1})$. Note that $\mathbf{d} = (d_1, d_2, \ldots, d_N)^\top = C\mathbf{1}$, where $\mathbf{1} = (1,1,\ldots,1)^\top$. Let I be the identity matrix. The weight factor matrix is

$$W = I + DC. \tag{4}$$

Remember that the agents of slow time scale can only make decision and execute actions at every $K\Delta t$. To make the reward of slow time scale agents sharable to their neighbor agents in fast time scale throughout the time, we introduce a projected reward simply by dividing the reward by K:

$$\bar{r}^i = \frac{r_t^i}{K}, i \in \mathbb{N}_s. \tag{5}$$

More generally, if the decision period of an agent is p, the projected reward is defined as $\bar{r} = r_t/p$. By this definition, the projected reward for fast time scale agent is $\bar{r} = r_t/1 = r_t$.

By information sharing among neighborhood agents, the observation of each agent consists of the state of itself and also the states and actions of its neighbor agents. Denote the observation of agent i as o^i, and

$$o^i = (S^i, S^j), \forall j \in \mathbb{L}_i. \tag{6}$$

The locally observed action of agent i is denoted as $\mathbf{a}^i$, and

$$\mathbf{a}^i = (a^i, a^j), \forall j \in \mathbb{L}_i, \tag{7}$$

For any agent i, the objective function is (we neglect the agent index in the following for simplicity)

$$\max\ J(o_0) = \mathbf{E}\left[\sum_{t=0}^{\infty}\gamma^t \hat{r}_t^i\right]. \tag{8}$$

The value function for an observation o under a policy π is

$$V^\pi(o) = J^\pi(o, \mathbf{a}). \tag{9}$$

The corresponding Q-function Q^{π} is

$$Q^{\pi}(o,\mathbf{a}) = \sum_{o'} P_{\mathbf{a}}(o,o') \left[\hat{r}(o,\mathbf{a},o') + \gamma \max_{o'} V^{\pi}(o') \right], \quad (10)$$

where $P_{\mathbf{a}}(o,o')$ is the transition probability that observation o transitions to o' when action $\mathbf{a}$ is taken.

The networked Q-learning based algorithm for multi-time scale multi-agent reinforcement learning is in **Algorithm 1**.

Algorithm 1 The networked multi-time scale multi-agent Q-learning algorithm (MTMAQL)

initialize the network Q-tables of slow and fast time scale agents Q_s^i and Q_f^j and initialize the learning rate α
for episode=1,M **do**
  initialize system state S_0
  $k \leftarrow 0$
  for t=0,T **do**
    for each agent **do**
      select action a_t^i based on $\arg_a \max Q(o^i,\mathbf{a}^i)$ with ε-greedy ($a_t = \emptyset$ for slow time scale agents if $k \neq 0$)
      execute actions and observe the new state S_{t+1} and reward r_t^i
      calculate the projected reward $\bar{r}^i$
      distribute the projected rewards to neighbors
    end for
    if $k = 0$ **then**
      for slow time scale agents **do**
        calculate the consensus reward by
$$\hat{r}_t^i = \sum_{j=1}^{N} w(j,i)\bar{r}_t^i$$
        calculate the error:
$$\delta = \hat{r}_t^i + \gamma \max_{\mathbf{a}'} Q_s^i(o_{t+1},\mathbf{a}') - Q_s^i(o_t,\mathbf{a}_t)$$
        update the Q-factor by:
$$Q_s^i(o_t,\mathbf{a}_t) \leftarrow Q_s^i(o_t,\mathbf{a}_t) + \alpha\delta$$
      end for
    end if
    for fast time scale agents **do**
      calculate the consensus reward for each agent:
$$\hat{r}_t^i = \sum_{j=1}^{N} w(j,i)\bar{r}_t^i$$
      calculate $\delta = \hat{r}_t^i + \gamma \max_{\mathbf{a}'} Q_f^i(o_{t+1},\mathbf{a}') - Q_f^i(o_t,\mathbf{a}_t)$
      update the Q-factor by:
$$Q_f^i(o_t,\mathbf{a}_t) \leftarrow Q_f^i(o_t,\mathbf{a}_t) + \alpha\delta$$
    end for
    $k \leftarrow k+1$
    if $k = K$ **then**
      $k \leftarrow 0$
    end if
  end for
end for

C. Convergence of MTMAQL

Assumption 1: The learning rate for both slow and fast time scale agents follows the constraints $\alpha \in (0,1)$, $\alpha \ll 1$.

Assumption 2: The reward function $r_t^i, \forall t, i$ is bounded.

Assumption 3: The markov chains in both slow and fast time scale are ergodic.

Assumption 3 means that all observation action pair $(o^i,\mathbf{a}^i)$ can be visited infinitely often.

Theorem 1: Under *Assumption* 1, 2 and 3, Algorithm 1 converges w.r.1 to optimal Q-functions in both slow and fast time scale.

We use the results (Theorem 2) in [22] to establish Theorem 1. For details of the proof for Theorem 2, please refer to [22].

Theorem 2: When the state space is finite, a random iterative process $\Delta_{n+1}(x) = (1-\alpha_n(x))\Delta_n(x) + \beta_n(x)F_n(x)$ converges to zero w.p.1 under the following assumptions:

1) $\sum_n \alpha_n(x) = \infty$, $\sum_n \beta_n(x) = \infty$, $\sum_n \beta_n^2(x) < \infty$, and $\mathbf{E}\{\beta_n(x)|P_n\} \leq \mathbf{E}\{\alpha_n(x)|P_n\}$ uniformly w.p.q.

2) $||\mathbf{E}\{F_n(x)|P_n\}||_W \leq \gamma||\Delta_n||_W$, where $\gamma \in (0,1)$.

3) $\mathrm{Var}\{F_n(x)|P_n\} \leq c(1+||\Delta_n||_W)^2$, c is some constant.

Here $P_n = \{\Delta_n, \Delta_{n-1}, \ldots, F_{n-1}, \ldots, \alpha_{n-1}, \ldots, \beta_{n-1}, \ldots\}$ stands for the past at step n. $F_n(x)$,$\alpha_n(x)$,$\beta_n(x)$ are allowed to depend on the past insofar as the above conditions remain valid. The notation $||\cdot||_W$ refers to some weighted maximum norm.

We now prove Theorem 1 as follows.

Proof: For any agent i, the objective function is

$$\max \ J(o_0) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t^i \right].$$

The optimal value function for an observation o is

$$V^{\star}(o) = \max_a J(o,\mathbf{a}).$$

The corresponding optimal Q-function $Q^{\star}$ is

$$Q^{\star}(o,\mathbf{a}) = \sum_{o'} P_{\mathbf{a}}(o,o') \left[\hat{r}(o,\mathbf{a},o') + \gamma V^{\star}(o') \right],$$

where $P_{\mathbf{a}}(o,o')$ is the transition probability that observation o transitions to o' when action $\mathbf{a}$ is taken. The optimal Q-function is a fixed point of a contraction operator $\mathbf{H}$, defined for a generic function q as

$$(\mathbf{H}q)(o,\mathbf{a}) = \sum_{o'} P_{\mathbf{a}}(o,o') \left[\hat{r}(o,\mathbf{a},o') + \gamma \max_{\mathbf{a}'} q(o',\mathbf{a}') \right].$$

It can be proved that the operator $\mathbf{H}$ is contractive.

$$\begin{aligned}
&||\mathbf{H}q_1 - \mathbf{H}q_2||_{\infty} = \max_{o,\mathbf{a}} \sum_{o'} P_{\mathbf{a}}(o,o') \\
&\left| \hat{r}(o,\mathbf{a},o') + \gamma \max_{\mathbf{a}'} q_1(o',\mathbf{a}') - \hat{r}(o,\mathbf{a},o') - \gamma \max_{\mathbf{a}'} q_2(o',\mathbf{a}') \right| \\
&= \max_{o,\mathbf{a}} \sum_{o'} \gamma P_{\mathbf{a}}(o,o') \left| \max_{\mathbf{a}'} q_1(o',\mathbf{a}') - \max_{\mathbf{a}'} q_2(o',\mathbf{a}') \right| \\
&\leq \max_{o,\mathbf{a}} \sum_{o'} \gamma P_{\mathbf{a}}(o,o') \max_{\mathbf{a}'} |q_1(o',\mathbf{a}') - q_2(o',\mathbf{a}')| \\
&\leq \max_{o,\mathbf{a}} \sum_{o'} \gamma P_{\mathbf{a}}(o,o') ||q_1(o',\mathbf{a}') - q_2(o',\mathbf{a}')||_{\infty} \leq \gamma ||q_1 - q_2||_{\infty}.
\end{aligned} \quad (11)$$

The networked multi-time multi-agent Q-learning uses the following update rule:

$$Q(o_t,\mathbf{a}_t) = Q(o_t,\mathbf{a}_t) + \alpha\left[\hat{r}_t + \gamma\max_{\mathbf{a}'} Q(o_{t+1},\mathbf{a}') - Q(o_t,\mathbf{a}_t)\right]$$
$$= (1-\alpha)Q(o_t,\mathbf{a}_t) + \alpha\left[\hat{r}_t + \gamma\max_{\mathbf{a}'} Q(o_{t+1},\mathbf{a}')\right], \tag{12}$$

where α is learning rate. Under Assumption 1, the *Condition 1)* in Theorem 1 is satisfied.

Subtracting $Q^\star(o,\mathbf{a})$ from both sides of (12), we get

$$Q(o_t,\mathbf{a}_t) - Q^\star(o_t,\mathbf{a}) = (1-\alpha)\left[Q(o_t,\mathbf{a}_t) - Q^\star(o_t,\mathbf{a})\right] + \alpha\left[\hat{r}_t + \gamma\max_{\mathbf{a}'} Q(o_{t+1},\mathbf{a}') - Q^\star(o_t,\mathbf{a})\right]. \tag{13}$$

Let $\Delta(o_t,\mathbf{a}_t) = Q(o_t,\mathbf{a}_t) - Q^\star(o_t,\mathbf{a})$ and (13) yields

$$\Delta(o_t,\mathbf{a}_t) = (1-\alpha)\Delta(o_t,\mathbf{a}_t) + \alpha\left[\hat{r}_t + \gamma\max_{\mathbf{a}'} Q(o_{t+1},\mathbf{a}') - Q^\star(o_t,\mathbf{a})\right]. \tag{14}$$

Let $F_t(o_t,\mathbf{a}) = \hat{r}_t(o_t,\mathbf{a},o_{t+1}) + \gamma\max_{\mathbf{a}'} Q(o_{t+1},\mathbf{a}') - Q^\star(o_t,\mathbf{a})$. Then we have

$$\mathbf{E}\left[F_t(o_t,\mathbf{a})\right] = \sum_{o_{t+1}} P_{\mathbf{a}}(o_t,o_{t+1}) \left[\hat{r}_t(o_t,\mathbf{a},o_{t+1}) + \gamma\max_{\mathbf{a}'} Q(o_{t+1},\mathbf{a}') - Q^\star(o_t,\mathbf{a})\right]. \tag{15}$$
$$= (\mathbf{H}Q)(o_t,\mathbf{a}) - Q^\star(o_t,\mathbf{a})$$

Note that $Q^\star(o_t,\mathbf{a})$ is a fixed point of operator $\mathbf{H}$, i.e. $Q^\star = \mathbf{H}Q^\star$, and $\mathbf{H}$ is a contraction operator in the sup-norm (from (11)), we have

$$||\mathbf{E}\left[F_t(o_t,\mathbf{a})\right]||_\infty = ||(\mathbf{H}Q)(o_t,\mathbf{a}) - (\mathbf{H}Q^\star)(o_t,\mathbf{a})||_\infty \leq \gamma||Q(o_t,\mathbf{a}) - Q^\star(o_t,\mathbf{a})||_\infty = \gamma||\Delta(o_t,\mathbf{a})||_\infty \tag{16}$$

Thus, *Condition 2)* in Theorem 1 is satisfied.

Under Assumption 2, the reward function r is bounded. Since the weight matrix W is bounded, the consensus reward $\hat{r}$ is bounded. Thus we have

$$\mathbf{var}\left[F_t(o_t,\mathbf{a})\right] = \mathbf{E}\left[(F_t(o_t,\mathbf{a}) - \mathbf{E}\left[F_t(o_t,\mathbf{a})\right])^2\right]$$
$$= \mathbf{E}\left[(\hat{r}_t(o_t,\mathbf{a},o_{t+1}) + \gamma\max_{\mathbf{a}'} Q(o_{t+1},\mathbf{a}') - (\mathbf{H}Q)(o_t,\mathbf{a}))^2\right]$$
$$= \mathbf{var}\left[\hat{r}_t(o_t,\mathbf{a},o_{t+1}) + \gamma\max_{\mathbf{a}'} Q(o_{t+1},\mathbf{a}')\right]$$
$$\leq c(1+||\Delta_t||_W^2), \tag{17}$$

where c is some constant. Thus *Condition 3)* in Theorem 1 is satisfied.

Up to now, we have proved that for each agent i, Δ_t^i converges to zero w.p.1, which means Q_t^i converges to its optimal value $Q^{i\star}$ w.p.1.

Note that the the objective function for each agent i is now $J(o_0) = \mathbf{E}\left[\sum_{t=0}^{\infty}\gamma^t\hat{r}_t^i\right]$. We need to prove that objectives of $\mathbf{E}\left[\sum_i\sum_{t=0}^{\infty}\gamma^t\hat{r}_t^i\right]$ and $\mathbf{E}\left[\sum_i\sum_{t=0}^{\infty}\gamma^t r_t^i\right]$ are equivalent.

Denote $\mathbf{r}_t = (r_t^1, r_t^2, \ldots, r_t^N)^\top$. Replace $\hat{r}_t^i$ with (3), we have

$$\mathbf{E}\left[\sum_i\sum_{t=0}^{\infty}\gamma^t\hat{r}_{i,t}\right] = \mathbf{E}\left[\sum_i\sum_{t=0}^{\infty}\gamma^t(\textstyle\sum_{j\in\mathbb{N}_i} w(j,i)r_t^j + r_t^i)\right]$$
$$= \mathbf{E}\sum_{t=0}^{\infty}\mathbf{r}_t^\top W\mathbf{1} = \mathbf{E}\left[\sum_{t=0}^{\infty}\mathbf{r}_t^\top(I+DC)\mathbf{1}\right]$$
$$= \mathbf{E}\sum_{t=0}^{\infty}\mathbf{r}_t^\top I\mathbf{1} + \mathbf{E}\sum_{t=0}^{\infty}\mathbf{r}_t^\top DC\mathbf{1} \tag{18}$$
$$= \mathbf{E}\sum_i\sum_{t=0}^{\infty}\gamma^t r_t^i + \mathbf{E}\sum_{t=0}^{\infty}\mathbf{r}_t^\top D\mathbf{d}$$
$$= \mathbf{E}\sum_i\sum_{t=0}^{\infty}\gamma^t r_t^i + \mathbf{E}\sum_{t=0}^{\infty}\mathbf{r}_t^\top\mathbf{1} = 2\times\mathbf{E}\left[\sum_i\sum_{t=0}^{\infty}\gamma^t r_t^i\right]$$

This means that objectives of $\mathbf{E}\left[\sum_i\sum_{t=0}^{\infty}\gamma^t\hat{r}_t^i\right]$ and $\mathbf{E}\left[\sum_i\sum_{t=0}^{\infty}\gamma^t r_t^i\right]$ are equivalent. This completes the proof for Theorem 1. ■

D. Networked MADDPG Algorithm

In the previous subsection, we propose a tabular-based Q-learning algorithm for MTMARL. In this subsection, we modify the MADDPG algorithm [16] to solve multi-time scale MARL with continuous state and action space. Based on communication network, the agents can observe states and actions of their neighbors. The critic network does not rely on the global observation of states of all the agents. Each agent maintains its own critic network with only local information, including neighbors' states and actions. By this modification, complete distributed optimization can be achieved.

Denote $\mathbb{D}_t$ as the set of agents that make decision and execute action at time t. The networked multi-time scale multi-agent DDPG (n-MADDPG) algorithm is proposed in Algorithm 2. The n-MADDPG algorithm are different from MADDPG in three aspects. *First*, it introduces the multi-time scale of decision epochs. *Second*, it considers the reward sharing between neighbor agents. *Third*, the input of the critic network of each agent consists of only local information, it does not require global observability of states and actions of all the agents. By n-MADDPG, complete decentralized optimization can be achieved.

V. Numerical Experiments

A. Performance of MTMAQL

We implement the proposed algorithm in two examples. The first example is the shared EVs scheduling problem [23], where the agents (**50** shared EVs) have the same state and action space. The roles of the EVs are identical. The other example is the speaker listener problem [24]. The two agents (speaker and listener) have different state and action space and they play different roles. The speaker aims to learn to communicate the right goal to the listener. While the listener should learn to move to the right goal (position). Since the proposed algorithm is tabular-based Q-learning. We compare the performance of this algorithm with the tabular-based QD-learning and centralized Q-learning.

Fig. 2 shows the learning curves during the training process. The three algorithms converge to the same level of mean episode reward. The centralized Q-learning converges

Algorithm 2 The networked multi-agent DDPG (n-MADDPG) algorithm (Multi-time scale version)

for episode=1,M **do**
  initialize a random process $\mathcal{N}$ for action exploration
  observe initial sate $\mathbf{x}$
  for t=0,T **do**
    for each agent $i \in \mathbb{D}_t$, select action $a^i = \mu_{\theta_i}(o^i) + \mathcal{N}_t$ w.r.t the current policy and exploration
    execute actions $a = (a_1, \ldots, a_N)$ and observe reward $\mathbf{r}$ and new state $\mathbf{x}'$
    calculate the projected reward $\bar{r}^i$
    calculate the consensus reward by
$$\hat{r}_t^i = \sum_{j=1}^{N} w(j,i)\bar{r}_t^i$$
    store $(\mathbf{x}, a, \hat{\mathbf{r}}, \mathbf{x}'^i)$ in replay buffer D
    for agent $i \in \mathbb{D}_t$ **do**
      sample a random minibatch of W samples $(\mathbf{x}^j, a, \hat{\mathbf{r}}^j, \mathbf{x}'^j)$ from D
      set $y^j = \hat{r}^j + \gamma Q_i^{\mu}(o'^j, \mathbf{a}')|_{\mathbf{a}'=\mu(o'^j)}$
      update critic by minimizing the loss:
$$\mathscr{L} = \frac{1}{W}\sum_j (y^j - Q_i^{\mu}(o^j, \mathbf{a}^j))^2$$
      update actor using the sampled policy gradient:
$$\nabla_{\theta_i} J \approx \frac{1}{W}\sum_j \nabla_{\theta_i}\mu_i(o^j)\nabla_{\mathbf{a}_i} Q_i^{\mu}(o^j, \mathbf{a}^j)|_{\mathbf{a}_i=\mu_i(o^j)}$$
    end for
    update target network parameters for each agent i:
$$\theta_i' \leftarrow \alpha\theta_i + (1-\alpha)\theta_i'$$
  end for
end for

faster than the others. The performance of the proposed algorithm and QD-learning are close. This means that MTMAQL performs well when the state/action spaces of agents are the same and agents play identical roles.

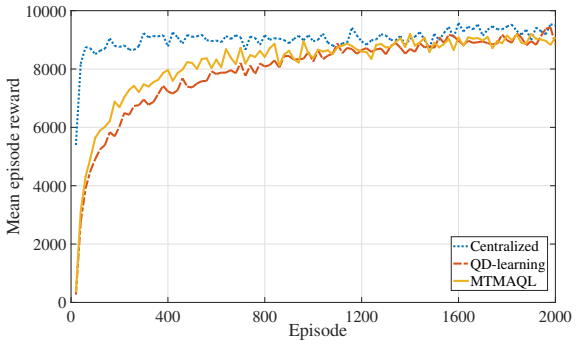

Fig. 2. Shared EVs scheduling

To transform the original Speaker listener problem to multi-time scale, the speaker is assumed to have slower decision frequency. The speaker sends out a message to the listener every three time steps. We compare our algorithm with the centralized Q-learning and QD-learning. Fig. 3 shows the learning curves. The MTMAQL algorithm converges with slower rate than the centralized Q-learning. It performs much better than the QD-learning, which fails to converge to the optimal value. The results show that the proposed MTMAQL algorithm performs well even when the agents' state and action spaces are different.

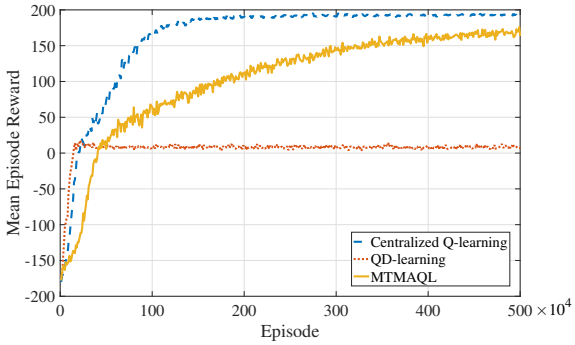

Fig. 3. Speaker listener problem

B. Performance of n-MADDPG

In this subsection, we test the n-MADDPG algorithm in two scenarios proposed in [24], namely simple spread and simple adversary. There are 3 agents in each scenario. We set the decision epoch of one agent (the agent with only one neighbor) as 3 time steps. The communication network among agents are shown in Fig.4. The learning curves are shown in Fig.5 and 6 correspondingly. The results show that the n-MADDPG algorithm achieves almost the same performance as the original one. Note that the agents in newly proposed algorithm use only the local information shared through the communication network. It learns agent policies with a more flexible decentralized format.

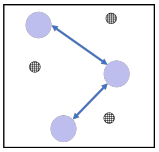

Fig. 4. Communication network among agents

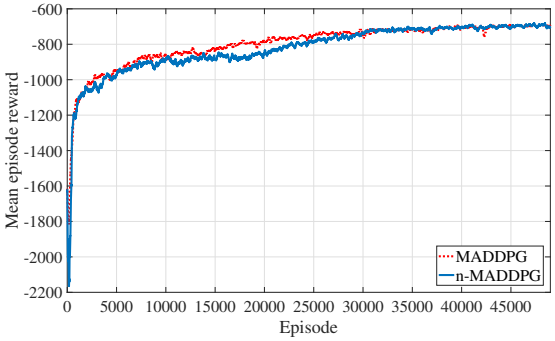

Fig. 5. Simple spread problem: 3 agents

We further test n-MADDPG in simple spread problem with 10 agents. We can see from Fig.7 that the n-MADDPG achieves almost the same performance with MADDPG, and can even converge faster in the early training period.

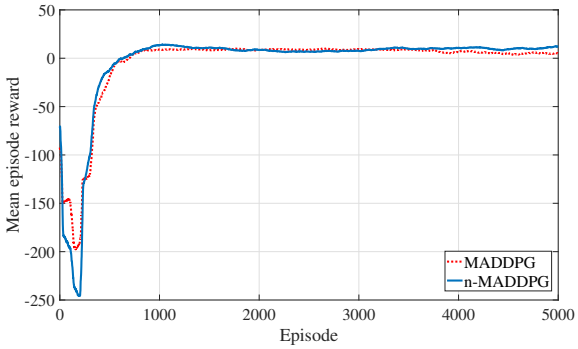

Fig. 6. Simple adversary problem

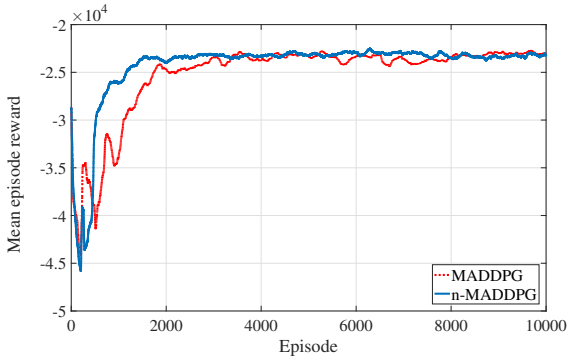

Fig. 7. Simple spread problem: 10 agents

VI. CONCLUSIONS

We formulate the multi-agent reinforcement learning in a multi-time scale framework, which is more flexible than the single-time version. We propose a tabular-based Q-learning algorithm for optimizing multi-time scale multi-agent policies, which is proved to converge to global optimum. We also modify the previous MADDPG algorithm. By our algorithms, the global observability of states and actions is not required. The central optimizer is not required either. With only the locally shared information, each agent optimizes its own policy distributively. A complete decentralized reinforcement learning is achieved. We compare our algorithm with the previous algorithms, including QD-learning, centralized Q-learning and MADDPG. Numerical results show the advantages of our proposed algorithm. We hope our work may shed some light on multi-time scale multi-agent reinforcement learning.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[5] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[7] S. James and E. Johns, "3d simulation for robot arm control with deep q-learning," *arXiv preprint arXiv:1609.03759*, 2016.

[8] L. Bu, R. Babu, B. De Schutter *et al.*, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[9] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.

[10] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer Networks*, vol. 101, pp. 158–168, 2016.

[11] T. Long, J.-X. Tang, and Q.-S. Jia, "Multi-scale event-based optimization for matching uncertain wind supply with ev charging demand," in *the 13th IEEE Conference on Automation Science and Engineering (CASE)*. Xi'an, China, Aug. 21-23, 2017, pp. 847–852.

[12] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," *arXiv preprint arXiv:1702.03037*, 2017.

[13] Z. Bao, Q. Zhou, Z. Yang, Q. Yang, L. Xu, and T. Wu, "A multi time-scale and multi energy-type coordinated microgrid scheduling solutionpart i: Model and methodology," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2257–2266, 2014.

[14] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications," *arXiv preprint arXiv:1812.11794*, 2018.

[15] S. Kar, J. M. Moura, and H. V. Poor, "*QD*-learning: A collaborative distributed strategy for multi-agent reinforcement learning through *Consensus + Innovations*," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1848–1862, 2013.

[16] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[17] H. Saad, A. Mohamed, and T. ElBatt, "Distributed cooperative q-learning for power allocation in cognitive femtocell networks," in *2012 IEEE Vehicular Technology Conference*. IEEE, 2012, pp. 1–5.

[18] Y. Xu, Z. Deng, M. Wang, W. Xu, A. M.-C. So, and S. Cui, "Voting-based multi-agent reinforcement learning," *arXiv preprint arXiv:1907.01385*, 2019.

[19] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv preprint arXiv:1802.08757*, 2018.

[20] P. Pennesi and I. C. Paschalidis, "A distributed actor-critic algorithm and applications to mobile sensor network coordination problems," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 492–497, 2010.

[21] S. V. Macua, A. Tukiainen, D. G.-O. Hernández, D. Baldazo, E. M. de Cote, and S. Zazo, "Diff-dac: Distributed actor-critic for average multitask deep reinforcement learning," *arXiv preprint arXiv:1710.10363*, 2017.

[22] T. Jaakkola, M. I. Jordan, and S. P. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in neural information processing systems*, 1994, pp. 703–710.

[23] J. Wu and Q.-S. Jia, "A q-learning method for scheduling shared evs under uncertain user demand and wind power supply," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. Copenhagen, Demark, 2018, pp. 601–607.

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.