Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Finite-Sample Analysis For Decentralized Cooperative Multi-Agent Reinforcement Learning From Batch Data

Kaiqing Zhang * **Zhuoran Yang** ** **Han Liu** *** **Tong Zhang** ****
Tamer Başar*

* *Department of Electrical and Computer Engineering & Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA (e-mail: {kzhang66,basar1}@illinois.edu)*
** *Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ 08544, USA (e-mail: zy6@princeton.edu)*
*** *Department of Electrical Engineering and Computer Science and Statistics, Northwestern University, Evanston, IL 60208, USA (e-mail: hanliu@northwestern.edu)*
**** *Department of Computer Science & Engineering, the Hong Kong University of Science and Technology, Hong Kong, China (e-mail: tongzhang@ust.hk)*

Abstract: In contrast to its great empirical success, theoretical understanding of multi-agent reinforcement learning (MARL) remains largely underdeveloped. As an initial attempt, we provide a finite-sample analysis for decentralized cooperative MARL with networked agents. In particular, we consider a team of cooperative agents connected by a time-varying communication network, with no central controller coordinating them. The goal for each agent is to maximize the long-term return associated with the team-average reward, by communicating only with its neighbors over the network. A batch MARL algorithm is developed for this setting, which can be implemented in a decentralized fashion. We then quantify the estimation errors of the action-value functions obtained from our algorithm, establishing their dependence on the function class, the number of samples in each iteration, and the number of iterations. This work appears to be the first finite-sample analysis for decentralized cooperative MARL from batch data.

Keywords: Reinforcement Learning; Finite-Sample Analysis; Networked Systems; Multi-Agent Systems; Decentralized Optimization

1. INTRODUCTION

There is an increasing interest in investigating both the empirical (Foerster et al., 2016; Lowe et al., 2017; Lanctot et al., 2017) and theoretical (Perolat et al., 2016; Zhang et al., 2018e,d; Doan et al., 2019a; Srinivasan et al., 2018; Zhang et al., 2019b) performance of multi-agent reinforcement learning (MARL). See Busoniu et al. (2008); Zhang et al. (2019a) for comprehensive surveys on MARL and its applications in various multi-agent systems, including distributed control, telecommunications, and economics.

Among various settings, *cooperative* MARL is one of the most commonly studied one. Cooperative MARL is usually modeled as either a multi-agent Markov decision process (MDP) (Boutilier, 1996), or a team Markov game (Wang and Sandholm, 2003), where the agents are assumed to share a common reward function. A more general while challenging setting considers heterogeneous reward functions that are private to each individual agent, while the collective goal is to maximize the average of the long-term return among all agents (Kar et al., 2013; Zhang et al., 2018e; Suttle et al., 2019; Doan et al., 2019a). This setting makes it nontrivial to design *decentralized* MARL algorithms, in which agents make globally optimal decisions when no central controller exists to coordinate them. Instead, agents are connected by a possibly time-varying communication network to exchange information with each other. This decentralized protocol with networked agents finds broad applications in practical multi-agent systems, including unmanned (aerial) vehicles (Alexander and Murray, 2004; Zhang et al., 2018a), smart power grid (Zhang et al., 2018b,c), and robotics (Corke et al., 2005). There exist several theoretical efforts that study MARL in this setting (Kar et al., 2013; Zhang et al., 2018e,d; Lee et al., 2018), which, however, are restricted to the *asymptotic* regime, i.e., the convergence is guaranteed only as the number of iterations increases to infinity. Concurrent to

⋆ K. Zhang and T. Başar are supported in part by the US Army Research Laboratory (ARL) Cooperative Agreement W911NF-17-2-0196, and in part by the Office of Naval Research (ONR) MURI Grant N00014-16-1-2710. Z. Yang is supported by Tencent PhD Fellowship. Due to space limitation, some definitions and proof details are deferred to the accompanying complete report (Zhang et al., 2018f).

the preparation of the present work, Doan et al. (2019a,b) carried out a *finite-time* performance analysis for temporal difference (TD) learning under this setting, which focused on the *policy evaluation* task, and the algorithms with *streaming data*.

Complementary to Doan et al. (2019a,b), we focus here on finite-sample analysis for the more challenging *control* task, with *batch data* sets. In particular, our analysis follows the significant lines of work on the finite-sample analyses for *batch RL* algorithms (Munos and Szepesvári, 2008; Antos et al., 2008a,b; Yang et al., 2018; Fan et al., 2019), using tools from approximate dynamic programming and statistical learning theory. This type of analysis is favored over other existing finite-sample analyses for (single-agent) RL, e.g., Kakade et al. (2003); Strehl et al. (2009); Jiang et al. (2016); Bhandari et al. (2018); Srikant and Ying (2019), which are restricted to either tabular or linear function approximation settings, or for policy evaluation tasks only. Batch RL analyses can accommodate general function classes, which is especially necessary in the multi-agent setting as the joint action space grows exponentially with the number of agents. Moreover, batch RL algorithms enjoy the advantages of off-policy exploration (Antos et al., 2008a,b). It is thus imperative to study the finite-sample performance of batch MARL algorithms, which is still missing in the literature.

In this work, we propose a decentralized fitted Q-iteration algorithm with value function approximation for decentralized cooperative MARL with networked agents, and establish a finite-sample performance analysis. Specifically, using batch data, a team of networked agents aims to estimate the global action-value (Q-value) function corresponding to the team-average reward in a decentralized fashion, via variants of the fitted Q-iteration algorithm (Riedmiller, 2005). We then establish the statistical error of the Q-value function returned by the algorithm, measured by the ℓ_2-distance between the estimated and the optimal Q-functions. Interestingly, we show that the statistical error can be decomposed into a sum of three error terms that reflect the effects of the function class, number of samples in each fitted Q-iteration, and the number of iterations. This work appears to be the first finite-sample analysis for decentralized cooperative MARL with batch data.

Notation. For a measurable space with domain $\mathcal{S}$, we denote the set of measurable functions on $\mathcal{S}$ that are bounded by V in absolute value by $\mathcal{F}(\mathcal{S}, V)$. Let $\mathcal{P}(\mathcal{S})$ be the set of all probability measures on $\mathcal{S}$. For any $\nu \in \mathcal{P}(\mathcal{S})$ and any measurable function $f\colon \mathcal{S} \to \mathbb{R}$, we denote by the ℓ_2-norm of f with respect to measure ν. We use $a \vee b$ to denote $\max\{a, b\}$ for any $a, b \in \mathbb{R}$, and define the set $[K] = \{1, 2, \cdots, K\}$.

2. PROBLEM FORMULATION

Consider a team of N cooperative agents, denoted by $\mathcal{N} = [N]$, that operate in a common environment. In the decentralized setting, there exists no central controller that is able to either collect rewards or make decisions for the agents. Alternatively, to foster collaboration, agents are assumed to exchange information via a possibly time-varying communication network, denoted by a graph $G_\tau = (\mathcal{N}, E_\tau)$, where the edge set E_τ denotes the set of communication links at time $\tau \in \mathbb{N}$. Formally, we define the following model of networked multi-agent MDP (M-MDP).

Definition 1. (Networked Multi-Agent MDP). A networked multi-agent MDP is characterized by a tuple

$$(\mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, P, \{R^i\}_{i\in\mathcal{N}}, \{G_\tau\}_{\tau\geq 0}, \gamma),$$

where $\mathcal{S}$ is the global state space shared by all the agents in $\mathcal{N}$, and $\mathcal{A}^i$ is the set of actions that agent i can choose from. Letting $\mathcal{A} = \prod_{i=1}^N \mathcal{A}^i$ denote the joint action space of all agents, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the probability distribution of the next state, and $R^i : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathbb{R})$ is the distribution of local reward function of agent i, both of which depending on the joint actions a and the global state s. $\gamma \in (0, 1)$ is the discount factor. $\mathcal{S}$ is a compact subset of $\mathbb{R}^d$ which can be infinite, $\mathcal{A}$ has finite cardinality $A = |\mathcal{A}|$, and the rewards have absolute values uniformly bounded by $R_{\max}$. At time[1] τ, the agents are connected by the communication network G_τ. The states and the joint actions are globally observable while the rewards are observed only locally.

By this definition, agents observe the global state s_t and perform joint actions $a_t = (a_t^1, \ldots, a_t^N) \in \mathcal{A}$ at time t. In consequence, each agent i receives an instantaneous reward $r_t^i \sim R^i(\cdot \,|\, s_t, a_t)$. Then, the environment evolves to a new state following $s_{t+1} \sim P(\cdot \,|\, s_t, a_t)$. We refer to this model as a *decentralized* one because each agent makes *individual* decisions based on the *local* information acquired from the network. In particular, we assume that given the current state, each agent i chooses actions independently of others, following its own policy $\pi^i : \mathcal{S} \to \mathcal{P}(\mathcal{A}^i)$. Thus, the joint policy of all agents, denoted by $\pi\colon \mathcal{S} \to \mathcal{P}(\mathcal{A})$, satisfies $\pi(a \,|\, s) = \prod_{i\in\mathcal{N}} \pi^i(a^i \,|\, s)$ for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

The goal of the cooperative agents is to maximize the *global average* of the cumulative discounted reward obtained by all agents over the network, which can be written as

$$\max_{\pi} \quad \frac{1}{N} \sum_{i\in\mathcal{N}} \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t \cdot r_t^i \right).$$

Accordingly, under any joint policy π, the action-value function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ can be defined as

$$Q_\pi(s, a) = \frac{1}{N} \sum_{i\in\mathcal{N}} \mathbb{E}_{a_t\sim\pi(\cdot \,|\, s_t)} \left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t^i \,\middle|\, s_0 = s, a_0 = a \right].$$

Notice that since $r_t^i \in [-R_{\max}, R_{\max}]$ for any $i \in \mathcal{N}$ and $t \geq 0$, $|Q_\pi| \leq R_{\max}/(1-\gamma)$ for any policy π. We let $Q_{\max} = R_{\max}/(1-\gamma)$ for notational convenience. Thus we have $Q_\pi \in \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max})$ for any π. We refer to Q_π as *global Q-function* hereafter. For notational convenience, under joint policy π, we define the operator $P_\pi : \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max}) \to \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max})$ and the Bellman operator $\mathcal{T}_\pi : \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max}) \to \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max})$ that correspond to the globally averaged reward as follows

$$\begin{aligned} (P_\pi Q)(s, a) &= \mathbb{E}_{s'\sim P(\cdot \,|\, s,a), a'\sim\pi(\cdot \,|\, s')}\big[Q(s', a')\big], \qquad (1)\\ (\mathcal{T}_\pi Q)(s, a) &= \overline{r}(s, a) + \gamma \cdot (P_\pi Q)(s, a), \end{aligned}$$

where $\overline{r}(s, a) = \sum_{i\in\mathcal{N}} r^i(s, a) \cdot N^{-1}$ denotes the globally averaged reward with $r^i(s, a) = \int r R^i(dr \,|\, s, a)$. Note that

[1] Note that this time index τ can be different from the time index t for the M-MDP, e.g., it can be the index of the algorithm updates. See examples in §3.

the action-value function Q_π is the unique fixed point of $\mathcal{T}_\pi$. Similarly, we also define the optimal Bellman operator corresponding to the averaged reward $\overline{r}$ as

$$(\mathcal{T}Q)(s,a) = \overline{r}(s,a) + \gamma \cdot \mathbb{E}_{s' \sim P(\cdot \,|\, s,a)} \big[\max_{a' \in \mathcal{A}} Q(s',a') \big].$$

Given a vector of Q-functions $\mathbf{Q} \in [\mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max})]^N$ with $\mathbf{Q} = [Q^i]_{i \in \mathcal{N}}$, we also define the *average Bellman operator* $\widetilde{\mathcal{T}} : [\mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max})]^N \to \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max})$ as

$$(\widetilde{\mathcal{T}}\mathbf{Q})(s,a) = \frac{1}{N} \sum_{i \in \mathcal{N}} (\mathcal{T}^i Q^i)(s,a) \quad \text{with} \tag{2}$$

$$\mathcal{T}^i Q^i = r^i(s,a) + \gamma \cdot \mathbb{E}_{s' \sim P(\cdot \,|\, s,a)} \left[\frac{1}{N} \cdot \sum_{i \in \mathcal{N}} \max_{a' \in \mathcal{A}} Q^i(s',a') \right].$$

Note that $\widetilde{\mathcal{T}}\mathbf{Q} = \mathcal{T}Q$ if $Q^i = Q$ for all $i \in \mathcal{N}$.

In addition, for any action-value function $Q \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, one can define the *one-step* greedy policy π_Q to be the deterministic policy that chooses the action with the largest Q-value, i.e., for any $s \in \mathcal{S}$, it holds that

$$\pi_Q(a \,|\, s) = 1 \;\; \text{if} \;\; a = \operatorname*{argmax}_{a' \in \mathcal{A}} \; Q(s,a').$$

For more than one action a' that maximizes the $Q(s,a')$, we break the tie randomly. Furthermore, we can define an operator $\mathcal{G}$, which generates the average greedy policy of a vector of Q-functions, i.e., $\mathcal{G}(\mathbf{Q}) = N^{-1} \sum_{i \in \mathcal{N}} \pi_{Q^i}$, where π_{Q^i} denotes the greedy policy with respect to Q^i.

3. ALGORITHM

In this section, we introduce the decentralized cooperative MARL algorithm using batch data, which builds on the single-agent fitted-Q iteration algorithm (Riedmiller, 2005). In particular, all agents have access to a dataset $\mathcal{D} = \{(s_t, \{a_t^i\}_{i \in \mathcal{N}}, s_{t+1})\}_{t=1,\cdots,T}$ that records the transition of the multi-agent system along the trajectory under a fixed joint behavior policy. The local reward function, however, is decentralized, and only available to each agent itself, as agents may have privacy concerns and are not willing to share their preferences.

At iteration k, each agent i maintains an estimate of the globally averaged Q-function, denoted by $\widetilde{Q}_k^i$. Then, agent i samples local reward $\{r_t^i\}_{t=1,\cdots,T}$ along the trajectory $\mathcal{D}$, and calculates the local target data $\{Y_t^i\}_{t=1,\cdots,T}$ following $Y_t^i = r_t^i + \gamma \cdot \max_{a \in \mathcal{A}} \widetilde{Q}_k^i(s_{t+1}, a)$. With the local data available, all agents hope to cooperatively find a common estimate of the global Q-function, by solving the following least-squares fitting problem

$$\min_{f \in \mathcal{H}} \quad \frac{1}{N} \sum_{i \in \mathcal{N}} \frac{1}{T} \sum_{t=1}^{T} \big[Y_t^i - f(s_t, a_t) \big]^2, \tag{3}$$

where $\mathcal{H} \subseteq \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max})$ denotes the function class used for Q-function approximation. The exact solution to (3), denoted by $\widetilde{Q}_{k+1}$, can be viewed as an improved estimate of the global Q-function, which can be used to generate the targets for the next iteration $k+1$. However, in practice, since agents have to solve (3) in a distributed fashion, with a finite number of iterations of any distributed optimization algorithm, the estimate at each agent may not reach an exact consensual value. Instead, each agent i may have an estimate $\widetilde{Q}_{k+1}^i$ that is different from the exact solution $\widetilde{Q}_{k+1}$, as well as different from each other. This mismatch will then propagate to the next iteration since agents can only use the local $\widetilde{Q}_{k+1}^i$ to generate the target for iteration $k+1$. This is in fact one of the departures of our finite-sample analysis that exists for MARL from the analyses for the single-agent setting (Munos and Szepesvári, 2008; Lazaric et al., 2010). After K iterations, each agent i finds the local greedy policy with respect to $\widetilde{Q}_K^i$ and the local estimate of the global Q-function. To obtain a consistent joint greedy policy, all agents output the average of their local greedy policies, i.e., output $\pi_K = \mathcal{G}(\widetilde{\mathbf{Q}}_K)$. The proposed decentralized algorithm for cooperative MARL is summarized in Algorithm 1.

When a parametric function class is considered, we denote $\mathcal{H}$ by $\mathcal{H}_\Theta$, where $\mathcal{H}_\Theta = \{f(\cdot,\cdot;\theta) \in \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max}) : \theta \in \mathbb{R}^d\}$. In this case, (3) becomes a vector-valued optimization problem with a separable objective function among the agents. For notational convenience, we introduce $g^i(\theta) = T^{-1} \cdot \sum_{t=1}^{T} \big[Y_t^i - f(s_t, a_t; \theta) \big]^2$; then, (3) can be written as

$$\min_{\theta \in \mathbb{R}^d} \quad \frac{1}{N} \sum_{i \in \mathcal{N}} g^i(\theta). \tag{4}$$

Since each agent i has access to only its own $g^i(\theta)$, it needs to exchange local information over the network G_τ to solve (4), which entails a decentralized/distributed optimization algorithm. Note that problem (4) is nonconvex with respect to θ when $\mathcal{H}_\Theta$ is a nonlinear function class, e.g., deep neural networks, which makes computation of the exact minimum of (4) intractable. Moreover, even if $\mathcal{H}_\Theta$ is a linear function class, which turns (4) into a convex problem, with only a finite number of steps in practice, decentralized optimization algorithms can at best converge to a neighborhood of the global minimizer. Thus, the aforementioned mismatch between $\widetilde{Q}_k^i$ and $\widetilde{Q}_k$ is inevitable for our finite iteration analysis.

There exists a rich family of decentralized or consensus optimization algorithms for networked agents that can solve the vector-valued optimization problem (4). Since we consider a more general setting with a time-varying communication network, several recent work (Zhu and Martínez, 2013; Nedic et al., 2017; Tatarenko and Touri, 2017; Hong and Chang, 2017) may apply. For the case when the overall objective function is strongly-convex, Nedic et al. (2017) has proposed an algorithm, named *DIGing*, that is guaranteed to achieve a geometric/linear convergence rate. Thus, we use *DIGing* as one possible algorithm to solve (4). In particular, each agent i maintains two vectors in *DIGing*, i.e., the solution estimate $\theta_l^i \in \mathbb{R}^d$, and the average gradient estimate $\gamma_l^i \in \mathbb{R}^d$, at iteration l. Each agent exchanges these two vectors to the neighbors over the time-varying [2] network $\{G_l\}_{l \geq 0}$, weighted by some consensus matrix $\mathbf{C}_l = [c_l(i,j)]_{N \times N}$ that respects the topology of the graph G_l. Details on choosing the consensus matrix $\mathbf{C}_l$ will be provided in §4. The updates of the *DIGing* algorithm are summarized in Algorithm 2. If $\mathcal{H}_\Theta$ represents a linear function class, then (4) can be

[2] Note that here we allow the communication graph to be time-varying even within each iteration k of Algorithm 1. Thus, we use l as the time index used in the decentralized optimization algorithm instead of τ, the general time index in Definition 1.

Algorithm 1 Decentralized Fitted Q-Iteration Algorithm for Cooperative MARL

Input: Function class $\mathcal{H}$, trajectory data $\mathcal{D} = \big\{\big(s_t, \{a_t^i\}_{i\in\mathcal{N}}, s_{t+1}\big)\big\}_{t=1,\cdots,T}$, number of iterations K, number of samples n, the initial estimator vector $\widetilde{\mathbf{Q}}_0 = [\widetilde{Q}_0^i]_{i\in\mathcal{N}}$.
for $k = 0, 1, 2, \ldots, K-1$ **do**
  for agent $i \in \mathcal{N}$ **do**
    Sample $r_t^i \sim R^i(\cdot \,|\, s_t, a_t)$ and compute local target $Y_t^i = r_t^i + \gamma \cdot \max_{a\in\mathcal{A}} \widetilde{Q}_k^i(s_{t+1}, a)$, for all data $\big(s_t, \{a_t^i\}_{i\in\mathcal{N}}, s_{t+1}\big) \in \mathcal{D}$.
  end for
  Solve (3) for all agents $i \in \mathcal{N}$, by decentralized optimization algorithms, e.g., by **Algorithm 2**, if $\mathcal{H}$ is a parametric function class $\mathcal{H}_\Theta$.
  Update the estimator $\widetilde{Q}_{k+1}^i$ for all agents $i \in \mathcal{N}$.
end for
Output: The vector of estimator $\widetilde{\mathbf{Q}}_K = [\widetilde{Q}_K^i]_{i\in\mathcal{N}}$ of Q^* and joint greedy policy $\pi_K = \mathcal{G}(\widetilde{\mathbf{Q}}_K)$.

Algorithm 2 *DIGing*: A Decentralized Optimization Algorithm for Solving (4)

Input: Parametric function class $\mathcal{H}_\Theta$, stepsize $\alpha > 0$, initial consensus matrix $\mathbf{C}_0 = [c_0(i,j)]_{N\times N}$, local target data $\{Y_t^i\}_{t=1,\cdots,T}$, initial parameter $\theta_0^i \in \mathbb{R}^d$, and initial vector $\gamma_0^i = \nabla g^i\big(\theta_0^i\big)$ for all agent $i \in \mathcal{N}$.
for $l = 0, 1, 2, \ldots, L-1$ **do**
  for agent $i \in \mathcal{N}$ **do**
    $\theta_{l+1}^i = \sum_{j\in\mathcal{N}} c_l(i,j) \cdot \theta_l^j - \alpha \cdot \gamma_l^i$
    $\gamma_{l+1}^i = \sum_{j\in\mathcal{N}} c_l(i,j) \cdot \gamma_l^j + \nabla g^i\big(\theta_{l+1}^i\big) - \nabla g^i\big(\theta_l^i\big)$
  end for
end for
Output: The vector of functions $[\widetilde{Q}^i]_{i\in\mathcal{N}}$ with $\widetilde{Q}^i = f\big(\cdot, \cdot; \theta_L^i\big)$ for all agent $i \in \mathcal{N}$.

strongly-convex under mild conditions. In this case, one can quantitatively characterize the mismatch between the global minimizer of (4) and the output of Algorithm 2 after a finite number of iterations, thanks to the linear convergence rate of the algorithm.

For a general nonlinear function class $\mathcal{H}_\Theta$, the algorithms for nonconvex decentralized optimization (Zhu and Martínez, 2013; Hong et al., 2016; Tatarenko and Touri, 2017) can be applied. Nonetheless, the mismatch between the algorithm output and the global minimizer is very difficult to quantify, which is a fundamental issue in general nonconvex optimization problems.

4. THEORETICAL RESULTS

We first introduce the finite-sample performance bound for the proposed algorithms with general function approximation, followed by a more concrete bound when linear function approximation is used.

4.1 Using General Function Approximation

We start with several standard assumptions. The function class $\mathcal{H}$ used for action-value function approximation greatly influences the performance of the algorithm. Here we use the concept of *pseudo-dimension* (Munos and Szepesvári, 2008; Antos et al., 2008a,b) to capture the capacity of function classes.

Assumption 2. (Function Classes Capacity). Let $V_{\mathcal{H}^+}$ denote the *pseudo-dimension* of a function class $\mathcal{H}$, i.e., the VC-dimension of the subgraphs of functions in $\mathcal{H}$. Then the function class $\mathcal{H}$ used in Algorithm 1 has finite pseudo-dimension, i.e., $V_{\mathcal{H}^+} < \infty$.

In our decentralized setting, each agent may not have access to the simulators for the overall MDP model transition. Thus, the data $\mathcal{D}$ have to be collected from an actual trajectory of the networked M-MDP, under some joint behavior policy of all agents. Note that the behavior policy of other agents are not required to be known in order to generate such a sample path. Our assumption regarding the sample path is as follows.

Assumption 3. (Sample Path). The transition data $\mathcal{D} = \{(s_t, \{a_t^i\}_{i\in\mathcal{N}}, s_{t+1})\}_{t=1,\cdots,T}$ are collected from a sample path of the networked M-MDP under some stochastic behavior policy. Moreover, the process $\{(s_t, a_t)\}$ is stationary, i.e., $(s_t, a_t) \sim \nu \in \mathcal{P}(\mathcal{S}\times\mathcal{A})$), and exponentially β-mixing [3] with a rate defined by $(\overline{\beta}, g, \zeta)$.

The mixing property of the random process basically means that the *future* of the process depends weakly on the *past*, which allows us to derive tail inequalities for certain empirical processes. Note that Assumption 3 is standard in the literature (Antos et al., 2008b; Lazaric et al., 2010) for finite-sample analyses of batch RL using a single trajectory data. Also, the mixing coefficients do not need to be known when implementing the algorithms.

In addition, we also make the following standard assumption on the *concentrability coefficient* of the networked M-MDP as in Munos and Szepesvári (2008); Antos et al. (2008b). The definitions of concentrability coefficients follow from those in Munos and Szepesvári (2008); Perolat et al. (2015), which can also be found in the complete report (Zhang et al., 2018f, Definition A.2).

Assumption 4. (Concentrability Coefficient). Let ν be the stationary distribution of the samples $\{(s_t, a_t)\}$ in $\mathcal{D}$ from the networked M-MDP in Assumption 3. Let μ be a fixed distribution on $\mathcal{S}\times\mathcal{A}$. We assume that there exist constants $\phi_{\mu,\nu}^{\mathrm{MDP}} < \infty$ such that

$$(1-\gamma)^2 \cdot \sum_{m\geq 1} \gamma^{m-1} \cdot m \cdot \kappa^{\mathrm{MDP}}(m; \mu, \nu) \leq \phi_{\mu,\nu}^{\mathrm{MDP}}, \quad (5)$$

where κ^{MDP} is the concentrability coefficient.

The concentrability coefficient measures the similarity between ν and the distribution of the future states of the networked M-MDP when starting with some distribution μ. The boundedness of the concentrability coefficient can be interpreted as the controllability of the underlying system, and holds in a large class of regular MDPs. See more interpretations on concentrability coefficients in Munos and Szepesvári (2008); Perolat et al. (2015).

As mentioned in §3, in practice, at iteration k of Algorithm 1, with a finite number of iterations of the decentralized optimization algorithm, the output $\widetilde{Q}_k^i$ would be different

[3] See Definition A.1 in the appendix of (Zhang et al., 2018f) on β-mixing and exponentially β-mixing of a stochastic process.

from the exact minimizer of (3). Thus, we make the following assumption on this one-step computation error.

Assumption 5. (Decentralized Computation Error). At iteration k of Algorithm 1, the computation error from solving (3) is uniformly bounded, i.e., there exists $\epsilon_k^i > 0$, such that for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, it holds that $|\widetilde{Q}_k^i(s,a) - \widetilde{Q}_k(s,a)| \leq \epsilon_k^i$, where $\widetilde{Q}_k$ is the exact minimizer of (3) and $\widetilde{Q}_k^i$ is the output of the decentralized optimization algorithm at agent $i \in \mathcal{N}$.

The computation error generally comes from two sources: 1) the error caused by finiteness of the number of iterations of the decentralized optimization algorithm; 2) the error caused by the nonconvexity of (4) with nonlinear parametric function class $\mathcal{H}_\Theta$. Note that the error is always bounded for function class $\mathcal{H} \subset \mathcal{F}(\mathcal{S} \times \mathcal{A}, Q_{\max})$ with bounded absolute values. Moreover, the error can be further quantified when $\mathcal{H}_\Theta$ is a linear function class, as to be detailed in §4.2.

Now we are ready to lay out the finite-sample error bounds for decentralized cooperative batch MARL.

Theorem 6. (Finite-sample Error Bounds). Let $\{\widetilde{\mathbf{Q}}_k\}_{0\leq k\leq K}$ be the estimator vectors generated from Algorithm 1, and $\pi_K = \mathcal{G}(\widetilde{\mathbf{Q}}_K)$ be the joint average greedy policy with respect to the estimator vector $\widetilde{\mathbf{Q}}_K$. Let Q_{π_K} be the Q-function corresponding to π_K, Q^* be the optimal Q-function, and $\widetilde{R}_{\max} = (1+\gamma)Q_{\max} + R_{\max}$. Also, recall that $A = |\mathcal{A}|$, $N = |\mathcal{N}|$, and $T = |\mathcal{D}|$. Then, under Assumptions 2-5, for any fixed initial distribution $\mu \in \mathcal{P}(\mathcal{S} \times \mathcal{A})$ and $\delta \in (0,1]$, there exist constants K_1 and K_2 with

$$K_1 = K_1\big(V_{\mathcal{H}^+}\log(T), \log(1/\delta), \log(\widetilde{R}_{\max}), V_{\mathcal{H}^+}\log(\overline{\beta})\big),$$
$$K_2 = K_2\big(V_{\mathcal{H}^+}\log(T), V_{\mathcal{H}^+}\log(\overline{\beta}), V_{\mathcal{H}^+}\log[\widetilde{R}_{\max}(1+\gamma)], V_{\mathcal{H}^+}\log(Q_{\max}), V_{\mathcal{H}^+}\log(A)\big),$$

and $\Lambda_T(\delta) = K_1 + K_2 \cdot N$, such that with probability at least $1-\delta$

$$\|Q^* - Q_{\pi_K}\|_\mu \leq \underbrace{C_{\mu,\nu}^{\mathrm{MDP}} \cdot \left\{\frac{\Lambda_T(\delta/K)[\Lambda_T(\delta/K)/b \vee 1]^{1/\zeta}}{T/(2048 \cdot \widetilde{R}_{\max}^4)}\right\}^{\frac{1}{4}}}_{\text{Estimation error}}$$
$$+ \underbrace{C_{\mu,\nu}^{\mathrm{MDP}} \cdot E(\mathcal{H})}_{\text{Approximation error}} + \underbrace{\sqrt{2}\gamma \cdot C_{\mu,\nu}^{\mathrm{MDP}} \cdot \overline{\epsilon} + \frac{2\sqrt{2}\gamma}{1-\gamma} \cdot \overline{\epsilon}_K}_{\text{Decentralized computation error}}$$
$$+ \frac{4\sqrt{2} \cdot Q_{\max}}{(1-\gamma)^2} \cdot \gamma^{K/2},$$

where $\overline{\epsilon}_K = [N^{-1} \cdot \sum_{i\in\mathcal{N}} (\epsilon_K^i)^2]^{1/2}$, and

$$C_{\mu,\nu}^{\mathrm{MDP}} = \frac{4\gamma \cdot \left(\phi_{\mu,\nu}^{\mathrm{MDP}}\right)^{1/2}}{\sqrt{2}(1-\gamma)^2}, \quad E(\mathcal{H}) = \sup_{\mathbf{Q}\in\mathcal{H}^N} \inf_{f\in\mathcal{H}} \|f - \widetilde{\mathcal{T}}\mathbf{Q}\|_\nu,$$
$$\overline{\epsilon} = \max_{0\leq k\leq K-1} \left[\frac{1}{N}\sum_{i\in\mathcal{N}} (\epsilon_k^i)^2\right]^{1/2}.$$

Moreover, $\phi_{\mu,\nu}^{\mathrm{MDP}}$, given in (5), is a constant that only depends on the distributions μ and ν.

Proof First, we quantify the propagation of one-step errors as Algorithm 1 proceeds in the following theorem.

Theorem 7. (Error Propagation). Under Assumptions 4 and 5, for any fixed distribution $\mu \in \mathcal{P}(\mathcal{S} \times \mathcal{A})$, we have

$$\|Q^* - Q_{\pi_K}\|_\mu \leq \frac{4\sqrt{2} \cdot Q_{\max}}{(1-\gamma)^2} \cdot \gamma^{K/2}$$
$$+ \underbrace{C_{\mu,\nu}^{\mathrm{MDP}} \cdot \|\varrho\|_\nu}_{\text{Statistical error}} + \underbrace{\sqrt{2}\gamma \cdot C_{\mu,\nu}^{\mathrm{MDP}} \cdot \overline{\epsilon} + \frac{2\sqrt{2}\gamma}{1-\gamma} \cdot \overline{\epsilon}_K}_{\text{Decentralized computation error}}$$

where we define

$$\|\varrho\|_\nu = \max_{k\in[K]} \|\widetilde{\mathcal{T}}\widetilde{\mathbf{Q}}_{k-1} - \widetilde{Q}_k\|_\nu,$$

and $\overline{\epsilon}_K, C_{\mu,\nu}^{\mathrm{MDP}}$, and $\overline{\epsilon}$ are as defined in Theorem 6.

Due to space limitation, the proof of Theorem 7 is deferred to (Zhang et al., 2018f, §5.1), which mainly consists of three steps. First, the recursion between the errors of the exact minimizer of (3) at consecutive iterations with respect to the optimal Q-function, i.e., the recursion between $Q^* - \widetilde{Q}_{k+1}$ and $Q^* - \widetilde{Q}_k$, is established. This further gives a multi-step error propagation formula, as a function of the one-step approximation error of the fitting problem (3), and of the computation error due to finite-iteration of decentralized optimization algorithms. Second, the error between Q^* and Q_{π_k}, the output Q-function estimate after k iterations, is established, by connecting it to $Q^* - \widetilde{Q}_k$ and the decentralized computation error. Third, the weighted norm of $\|Q^* - Q_{\pi_K}\|_\mu$ under some probability distribution $\mu \in \mathcal{P}(\mathcal{S} \times \mathcal{A})$ is upper-bounded, leading to the desired result. Note that by the theorem, both the one-step statistical error and the decentralized computation error will propagate, which constitute the fundamental error that will not vanish even when the iteration $K \to \infty$.

Now it suffices to characterize the one-step statistical error $\|\varrho\|_\nu$. The following theorem establishes a high probability bound for this statistical error.

Theorem 8. (One-step Statistical Error). Let $\mathbf{Q} = [Q^i]_{i\in\mathcal{N}} \in \mathcal{H}^N$ be a vector of real-valued random functions (may be dependent on the sample path); let $(s_t, \{a_t^i\}_{i\in\mathcal{N}}, s_{t+1})$ be the samples from the trajectory data $\mathcal{D}$, and $\{r_t^i\}_{i\in\mathcal{N}}$ be the rewards sampled from $(s_t, \{a_t^i\}_{i\in\mathcal{N}}, s_{t+1})$. Define $Y_t^i = r_t^i + \gamma \cdot \max_{a\in\mathcal{A}} Q^i(s_{t+1}, a)$, and f' by

$$f' \in \operatorname*{argmin}_{f\in\mathcal{H}} \frac{1}{N}\sum_{i\in\mathcal{N}} \frac{1}{T}\sum_{t=1}^{T} \big[Y_t^i - f(s_t, a_t)\big]^2. \tag{6}$$

Then, under Assumptions 2 and 3, for $\delta \in (0,1]$, $T \geq 1$, there exists some $\Lambda_T(\delta)$ as defined in Theorem 6, such that with probability at least $1-\delta$,

$$\|f' - \widetilde{\mathcal{T}}\mathbf{Q}\|_\nu^2 \leq \inf_{f\in\mathcal{H}} \|f - \widetilde{\mathcal{T}}\mathbf{Q}\|_\nu^2 + \sqrt{\frac{\Lambda_T(\delta)[\Lambda_T(\delta)/b \vee 1]^{\frac{1}{\zeta}}}{T/(2048 \cdot \widetilde{R}_{\max}^4)}}. \tag{7}$$

The proof of Theorem 8 can be found in (Zhang et al., 2018f, §5.2), which integrates the proof ideas of one-step statistical error in Munos and Szepesvári (2008) and Antos et al. (2008b). Essentially, it establishes how the probability of the fitting error relies on the pseudo-dimension of the function class $V_{\mathcal{H}^+}$, and the number of samples T by the concentration inequality for β-mixing sequences. Similar to the existing results in the single-agent setting (e.g., Lemma 10 in Antos et al. (2008b)), the one-step statistical error consists of two parts, the approximation error depending on the richness of the function class $\mathcal{H}$, and the estimation error that vanishes with the number of samples T.

By replacing $\mathbf{Q}$ by $\widetilde{\mathbf{Q}}_{k-1}$ and f' by $\widetilde{Q}_k$, the results in Theorem 8 can characterize the one-step statistical error $\|\widetilde{\mathcal{T}}\widetilde{\mathbf{Q}}_{k-1} - \widetilde{Q}_k\|_\nu$. Together with Theorem 7, we arrive at the main results in Theorem 6. ■

Theorem 6 establishes a high probability bound on the error of the output value function Q_{π_K} obtained from Algorithm 1 after K iterations. The finite-sample error is controlled by three fundamental terms: 1) the approximation error that depends on the richness of the function class $\mathcal{H}$, i.e., how well $\mathcal{H}$ preserves the average Bellman operator $\widetilde{\mathcal{T}}$; 2) the estimation error incurred by the fitting step (1), which vanishes with increasing number of samples T; 3) the computation error in solving the fitting problem (3) in a decentralized way with a finite number of updates. After suppressing constants and logarithmic terms, the estimation error has the form

$$\left\{\frac{[V_{\mathcal{H}^+}(N+1)\log(T) + V_{\mathcal{H}^+}N\log(A) + \log(K/\delta)]^{1+1/\zeta}}{T}\right\}^{\frac{1}{4}}. \tag{8}$$

Compared with the existing results in the single-agent setting, e.g., (Antos et al., 2008b, Theorem 4), our results have an additional dependence on $O(N\log(A))$, where $N = |\mathcal{N}|$ is the number of agents in the team and $A = |\mathcal{A}|$ is cardinality of the joint action set. This dependence on N is due to the fact that the target data used in the fitting step are collections of local target data from N agents; while the dependence on $\log(A)$ characterizes the difficulty of estimating Q-functions, each of which has A choices to find the maximum given any state s. Similar terms of order $\log(A)$ also show up in the single-agent setting (Antos et al., 2008a,b), which is induced by the capacity of the action space. In addition, a close examination of the proof shows that the *effective dimension* (Antos et al., 2008b) is $(N+1)V_{\mathcal{H}^+}$, which is because we allow N agents to have their own estimates of Q-functions, each of which lies in the function class $\mathcal{H}$ with pseudo-dimension $V_{\mathcal{H}^+}$. We note that it is possible to sharpen the dependence of the rate and the effective dimension on N via different proof techniques from here, which is left as our future work.

4.2 Using Linear Function Approximation

With linear function approximation, the finite-sample bound can be made more concrete. Specifically, we quantify the one-step computation error bound in Assumption 5, after L iterations of the decentralized optimization algorithm that solves (3). We first make the following assumption on the features of the linear function class.

Assumption 9. The function class $\mathcal{H}_\Theta$ used in Algorithm 1 is a parametric linear function class, i.e., $\mathcal{H}_\Theta \subset \mathcal{F}(\mathcal{S}\times\mathcal{A}, Q_{\max})$ and $\mathcal{H}_\Theta = \{f(s,a;\theta) = \theta^\top\varphi(s,a) : \theta \in \mathbb{R}^d\}$, where for any $(s,a)\in\mathcal{S}\times\mathcal{A}$, $\varphi(s,a)\in\mathbb{R}^d$ is the feature vector. Moreover, let $\mathbf{M}^{\mathrm{MDP}} = T^{-1}\cdot\sum_{t=1}^T \varphi(s_t,a_t)\varphi^\top(s_t,a_t)$ with $\{(s_t,a_t)\}_{t\in[T]}$ being samples from the data set $\mathcal{D}$, and assume that the matrix $\mathbf{M}^{\mathrm{MDP}}$ is full rank.

The assumption on the rank of the matrix $\mathbf{M}^{\mathrm{MDP}}$ ensures that the least-squares problem (3) is strongly-convex, which allows the *DIGing* algorithm to achieve the geometric convergence rate, even over time-varying communication networks. We note that this assumption can be readily satisfied in practice. Let $\varphi(s,a) = [\varphi_1(s,a),\cdots,\varphi_d(s,a)]^\top$. Then, if one follows the conventional RL literature (Tsitsiklis and Van Roy, 1997; Geramifard et al., 2013) to assume that the functions $\{\varphi_1(s,a),\cdots,\varphi_d(s,a)\}$ are linearly independent, then with a rich enough $\mathcal{D}$, one can easily find $T \gg d$ samples from $\mathcal{D}$, such that the matrix $[\varphi(s_1,a_1),\cdots,\varphi(s_d,a_d)]^\top$ is full-rank, i.e., has rank d. Then, with some algebra (see Lemma B.1 in Zhang et al. (2018f)), one can show that $\mathbf{M}^{\mathrm{MDP}}$ is also full-rank.

Moreover, we make the following assumption on the time-varying consensus matrix $\mathbf{C}_l$ used in the *DIGing* algorithm (see also Assumption 1 in Nedic et al. (2017)).

Assumption 10. (Consensus Matrix $\{\mathbf{C}_l\}_{l\geq 0}$). For any $l = 0,1,\cdots$, the consensus matrix $\mathbf{C}_l = [c_l(i,j)]_{N\times N}$ satisfies the following relations:

1) (Decentralized property) If $i\neq j$, and edge $(j,i)\notin E_l$, then $c_l(i,j) = 0$;

2) (Double stochasticity) $\mathbf{C}_l\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^\top\mathbf{C}_l = \mathbf{1}^\top$;

3) (Joint spectrum property) There exists a positive integer B such that $\chi < 1$, where

$$\chi = \sup_{l\geq B-1}\sigma_{\max}\Big\{\mathbf{C}_l\mathbf{C}_{l-1}\cdots\mathbf{C}_{l-B+1} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top\Big\}$$

for all $l = 0,1,\cdots$, and $\sigma_{\max}(\cdot)$ denotes the largest singular value of a matrix.

Assumption 10 is standard and is satisfied by many matrix sequences in decentralized optimization. Specifically, condition 1) imposes the physical connection constraint of the network; condition 2) ensures that the convergent vector is consensual for all agents; condition 3) regards the connectivity of the time-varying graph $\{G_l\}_{l\geq 0}$. For more discussions on these, see (Nedic et al., 2017, Section 3).

Now we are ready to present the following corollary on the finite-sample error bound of Algorithm 1, when Algorithm 2 and linear function approximation are used.

Corollary 11. (Finite-sample Error Bounds With Linear Function Approximation) Suppose Assumptions 2-5, and 9-10 hold, and Algorithm 2 is used in the fitting step (3) for decentralized optimization. Then, for any $\delta\in(0,1]$, $\epsilon > 0$, and fixed initial distribution $\mu\in\mathcal{P}(\mathcal{S}\times\mathcal{A})$, there exist integers K, T, and L, where K is linear in $\log(1/\epsilon)$, $\log[1/(1-\gamma)]$, and $\log(Q_{\max})$; T is polynomial in $1/\epsilon$, $\gamma/(1-\gamma)$, $1/\widetilde{R}_{\max}$, $\log(1/\delta)$, $\log(\overline{\beta})$, and $N\log(A)$; and L is linear in $\log(1/\epsilon)$, $\log[\gamma/(1-\gamma)]$, such that

$$\|Q^* - Q_{\pi_K}\|_\mu \leq C^{\mathrm{MDP}}_{\mu,\nu}\cdot E(\mathcal{H}) + \epsilon$$

holds with probability at least $1-\delta$.

Proof The proof proceeds by controlling the three error terms in the bound in Theorem 6 (except the inherent approximation error) by $\epsilon/3$ for any $\epsilon > 0$. In particular, to show the first argument for the cooperative setting, let

$$\frac{4\sqrt{2}\cdot Q_{\max}}{(1-\gamma)^2}\cdot\gamma^{K/2} \leq \frac{4\sqrt{2}\cdot Q_{\max}}{(1-\gamma)^2}\cdot\frac{(1-\gamma)^2\epsilon}{12\sqrt{2}\cdot Q_{\max}} = \frac{\epsilon}{3};$$

we then immediately obtain that K is linear in $\log(1/\epsilon)$, $\log[1/(1-\gamma)]$, and $\log(Q_{\max})$. Letting the estimation error be controlled by $\epsilon/3$, we have

$$C_{\mu,\nu}^{\mathrm{MDP}} \cdot \left\{ \frac{\Lambda_T(\delta/K)[\Lambda_T(\delta/K)/b \vee 1]^{1/\zeta}}{T/(2048 \cdot \widetilde{R}_{\max}^4)} \right\}^{1/4}$$
$$= \frac{4\gamma \cdot \left(\phi_{\mu,\nu}^{\mathrm{MDP}}\right)^{1/2}}{\sqrt{2}(1-\gamma)^2} \cdot \left\{ \frac{\Lambda_T(\delta/K)[\Lambda_T(\delta/K)/b \vee 1]^{1/\zeta}}{T/(2048 \cdot \widetilde{R}_{\max}^4)} \right\}^{1/4} \leq \frac{\epsilon}{3}.$$

By definition of Λ_T in Theorem 6, we obtain that T is polynomial in $1/\epsilon$, $\gamma/(1-\gamma)$, $1/\widetilde{R}_{\max}$, $\log(1/\delta)$, $\log(\overline{\beta})$, and $N\log(A)$. For the decentralized computation error, let

$$\sqrt{2}\gamma \cdot C_{\mu,\nu}^{\mathrm{MDP}} \cdot \bar{\epsilon} + \frac{2\sqrt{2}\gamma}{1-\gamma} \cdot \bar{\epsilon}_K$$
$$= \sqrt{2}\gamma \cdot \frac{4\gamma \cdot \left(\phi_{\mu,\nu}^{\mathrm{MDP}}\right)^{1/2}}{\sqrt{2}(1-\gamma)^2} \cdot \bar{\epsilon} + \frac{2\sqrt{2}\gamma}{1-\gamma} \cdot \bar{\epsilon}_K \leq \frac{\epsilon}{3}. \quad (9)$$

Note that under Assumptions 9 and 10, we can apply (Nedic et al., 2017, Theorem 10), which shows that there exist constants $\lambda \in [0,1)$ and $C_0 > 0$, such that at each iteration k of Algorithm 1,

$$\sqrt{\sum_{i\in\mathcal{N}} \|\theta_{k,l}^i - \theta_k^*\|^2} \leq C_0 \cdot \lambda^l, \quad (10)$$

where θ_k^* corresponds to the exact solution to (3) at this iteration k, i.e., $\widetilde{Q}_k = (\theta_k^*)^\top \varphi$, and $\theta_{k,l}^i$ represents the estimate of θ_k^* of agent i at iteration l of Algorithm 2. Thus, if Algorithm 2 terminates after L iterations, we have $\widetilde{Q}_k^i = (\theta_{k,L}^i)^\top \varphi$, where we recall that $\widetilde{Q}_k^i$ denotes the output of the decentralized optimization step in Algorithm 1. Since the features φ are uniformly bounded, we obtain from (10) that there exists a constant $C_1 > 0$, such that for any $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\sqrt{\frac{1}{N}\sum_{i\in\mathcal{N}} |\widetilde{Q}_k^i(s,a) - \widetilde{Q}_k(s,a)|^2}$$
$$\leq C_1 \sqrt{\frac{1}{N}\sum_{i\in\mathcal{N}} \|\theta_{k,L}^i - \theta_k^*\|^2} \leq \frac{C_1 C_0}{\sqrt{N}} \cdot \lambda^L.$$

Thus, we can choose $C_1C_0/\sqrt{N} \cdot \lambda^L$ to bound the decentralized optimization $\bar{\epsilon}_k$. These arguments apply to all iterations $k \in [K]$, which means that we can bound both $\bar{\epsilon} = \max_{0\leq k\leq K-1}$ and $\bar{\epsilon}_K$ in (9) by $C_2 \cdot \lambda^L$ for some constant C_2. Therefore, we conclude from (9) that the number of iterations L is linear in $\log(1/\epsilon)$, $\log[\gamma/(1-\gamma)]$. This completes the proof. ■

Corollary 11 shows that Algorithm 1 is efficient with the aid of Algorithm 2 under some mild assumptions, in the sense that finite number of samples and iterations, which scale at most polynomially with the problem parameters, are needed to achieve arbitrarily small Q-value errors, provided the inherent approximation error is small.

We note that if the full-rank condition in Assumption 9 does not hold, the fitting problem (3) is simply convex. Then, over time-varying communication network, it is also possible to establish convergence rate of $O(1/l)$ using the proximal-gradient consensus algorithm (Hong and Chang, 2017). We will forgo a detailed discussion on various decentralized optimization algorithms since it is beyond the scope of this paper.

5. CONCLUSIONS

In this paper, we have provided a finite-sample analysis for decentralized cooperative multi-agent RL from batch data. Specifically, we have developed a decentralized fitted-Q iteration algorithm for cooperative MARL with networked agents, and quantified how the performance bound of the output action-value depends on the function class, the number of samples in each iteration, and the number of iterations. We believe that these theoretical results provide useful insights into the fundamental performance of MARL algorithms implemented with finite samples and finite computation iterations in practice. One interesting future research direction is to extend the finite-sample analysis to more general MARL settings, e.g., general-sum Markov games. It would also be promising to sharpen the bounds we obtained, to better understand and improve both the sample and computation efficiency of MARL algorithms.

REFERENCES

Alexander, F.J. and Murray, R.M. (2004). Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9), 1465–1476.

Antos, A., Szepesvári, C., and Munos, R. (2008a). Fitted Q-iteration in continuous action-space MDPs. In *Advances in Neural Information Processing Systems*, 9–16.

Antos, A., Szepesvári, C., and Munos, R. (2008b). Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1), 89–129.

Bhandari, J., Russo, D., and Singal, R. (2018). A finite time analysis of temporal difference learning with linear function approximation. In *Conference On Learning Theory*, 1691–1692.

Boutilier, C. (1996). Planning, learning and coordination in multi-agent decision processes. In *Conference on Theoretical Aspects of Rationality and Knowledge*, 195–210.

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008.*

Corke, P., Peterson, R., and Rus, D. (2005). Networked robots: Flying robot navigation using a sensor net. *Robotics Research*, 234–243.

Doan, T., Maguluri, S., and Romberg, J. (2019a). Finite-time analysis of distributed TD (0) with linear function approximation on multi-agent reinforcement learning. In *International Conference on Machine Learning*, 1626–1635.

Doan, T.T., Maguluri, S.T., and Romberg, J. (2019b). Finite-time performance of distributed temporal difference learning with linear function approximation. *arXiv preprint arXiv:1907.12530.*

Fan, J., Yang, Z., Xie, Y., and Wang, Z. (2019). A theoretical analysis of deep Q-learning. *arXiv preprint arXiv:1901.00137.*

Foerster, J., Assael, Y.M., de Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2137–2145.

Geramifard, A., Walsh, T.J., Tellex, S., Chowdhary, G., Roy, N., How, J.P., et al. (2013). A tutorial on linear function approximators for dynamic programming and reinforcement learning. *Foundations and Trends® in Machine Learning*, 6(4), 375–451.

Hong, M. and Chang, T.H. (2017). Stochastic proximal gradient consensus over random networks. *IEEE Transactions on Signal Processing*, 65(11), 2933–2948.

Hong, M., Luo, Z.Q., and Razaviyayn, M. (2016). Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1), 337–364.

Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R.E. (2016). Contextual decision processes with low Bellman rank are PAC-learnable. *arXiv preprint arXiv:1610.09512*.

Kakade, S.M. et al. (2003). *On the sample complexity of reinforcement learning*. Ph.D. thesis, University of London London, England.

Kar, S., Moura, J.M., and Poor, H.V. (2013). QD-learning: A collaborative distributed strategy for multi-agent reinforcement learning through Consensus + Innovations. *IEEE Transactions on Signal Processing*, 61(7), 1848–1862.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Perolat, J., Silver, D., Graepel, T., et al. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, 4193–4206.

Lazaric, A., Ghavamzadeh, M., and Munos, R. (2010). Finite-sample analysis of LSTD. In *International Conference on Machine Learning*, 615–622.

Lee, D., Yoon, H., and Hovakimyan, N. (2018). Primal-dual algorithm for distributed reinforcement learning: Distributed GTD2. *arXiv preprint arXiv:1803.08031*.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*.

Munos, R. and Szepesvári, C. (2008). Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May), 815–857.

Nedic, A., Olshevsky, A., and Shi, W. (2017). Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4), 2597–2633.

Perolat, J., Scherrer, B., Piot, B., and Pietquin, O. (2015). Approximate dynamic programming for two-player zero-sum Markov games. In *International Conference on Machine Learning*, 807–814.

Perolat, J., Strub, F., Piot, B., and Pietquin, O. (2016). Learning Nash equilibrium for general-sum Markov games from batch data. In *International Conference on Artificial Intelligence and Statistics*, 232–241.

Riedmiller, M. (2005). Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, 317–328.

Srikant, R. and Ying, L. (2019). Finite-time error bounds for linear stochastic approximation and TD learning. In *Conference on Learning Theory*, 2803–2830.

Srinivasan, S., Lanctot, M., Zambaldi, V., Pérolat, J., Tuyls, K., Munos, R., and Bowling, M. (2018). Actor-critic policy optimization in partially observable multi-agent environments. In *Advances in Neural Information Processing Systems*, 3422–3435.

Strehl, A.L., Li, L., and Littman, M.L. (2009). Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10(Nov), 2413–2444.

Suttle, W., Yang, Z., Zhang, K., Wang, Z., Başar, T., and Liu, J. (2019). A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning. *arXiv preprint arXiv:1903.06372*.

Tatarenko, T. and Touri, B. (2017). Non-convex distributed optimization. *IEEE Transactions on Automatic Control*, 62(8), 3744–3757.

Tsitsiklis, J.N. and Van Roy, B. (1997). Analysis of temporal-diffference learning with function approximation. In *Advances in Neural Information Processing Systems*, 1075–1081.

Wang, X. and Sandholm, T. (2003). Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *Advances in Neural Information Processing Systems*, 1603–1610.

Yang, Z., Zhang, K., Hong, M., and Başar, T. (2018). A finite sample analysis of the actor-critic algorithm. In *Proceedings of IEEE Conference on Decision and Control*, 2759–2764.

Zhang, K., Lu, L., Lei, C., Zhu, H., and Ouyang, Y. (2018a). Dynamic operations and pricing of electric unmanned aerial vehicle systems and power networks. *Transportation Research Part C: Emerging Technologies*, 92, 472–485.

Zhang, K., Shi, W., Zhu, H., and Başar, T. (2018b). Distributed equilibrium-learning for power network voltage control with a locally connected communication network. In *IEEE Annual American Control Conference*, 3092–3097. IEEE.

Zhang, K., Shi, W., Zhu, H., Dall'Anese, E., and Başar, T. (2018c). Dynamic power distribution system management with a locally connected communication network. *IEEE Journal of Selected Topics in Signal Processing*, 12(4), 673–687.

Zhang, K., Yang, Z., and Başar, T. (2018d). Networked multi-agent reinforcement learning in continuous spaces. In *Proceedings of IEEE Conference on Decision and Control*, 2771–2776.

Zhang, K., Yang, Z., and Başar, T. (2019a). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*.

Zhang, K., Yang, Z., and Başar, T. (2019b). Policy optimization provably converges to Nash equilibria in zero-sum linear quadratic games. In *Advances in Neural Information Processing Systems*.

Zhang, K., Yang, Z., Liu, H., Zhang, T., and Başar, T. (2018e). Fully decentralized multi-agent reinforcement learning with networked agents. In *International Conference on Machine Learning*, 5867–5876.

Zhang, K., Yang, Z., Liu, H., Zhang, T., and Başar, T. (2018f). Finite-sample analyses for decentralized batch multi-agent reinforcement learning with networked agents. *arXiv preprint arXiv:1812.02783*.

Zhu, M. and Martínez, S. (2013). An approximate dual subgradient algorithm for multi-agent non-convex optimization. *IEEE Transactions on Automatic Control*, 58(6), 1534–1539.