MDPGT: Momentum-Based Decentralized Policy Gradient Tracking

Zhanhong Jiang[1], Xian Yeow Lee[2], Sin Yong Tan[2], Kai Liang Tan[2],
Aditya Balu[2], Young M. Lee[1], Chinmay Hegde[3], Soumik Sarkar[2]
[1]Johnson Controls Inc., 507 East Michigan St, Milwaukee, WI 53202,
[2]Iowa State University, Ames, IA 50010,
[3]New York University, 6 MetroTech Center, Brooklyn, NY 11201,
{zhanhong.jiang, young.m.lee}@jci.com, {xylee, tsyong98, kailiang, baditya, soumiks}@iastate.edu, {chinmay.h}@nyu.edu

Abstract

We propose a novel policy gradient method for multi-agent reinforcement learning, which leverages two different variance-reduction techniques and does not require large batches over iterations. Specifically, we propose a momentum-based decentralized policy gradient tracking (`MDPGT`) where a new momentum-based variance reduction technique is used to approximate the local policy gradient surrogate with importance sampling, and an intermediate parameter is adopted to track two consecutive policy gradient surrogates. `MDPGT` provably achieves the best available sample complexity of $\mathcal{O}(N^{-1}\epsilon^{-3})$ for converging to an ϵ-stationary point of the global average of N local performance functions (possibly nonconcave). This outperforms the state-of-the-art sample complexity in decentralized model-free reinforcement learning and when initialized with a single trajectory, the sample complexity matches those obtained by the existing decentralized policy gradient methods. We further validate the theoretical claim for the Gaussian policy function. When the required error tolerance ϵ is small enough, `MDPGT` leads to a linear speed up, which has been previously established in decentralized stochastic optimization, but not for reinforcement learning. Lastly, we provide empirical results on a multi-agent reinforcement learning benchmark environment to support our theoretical findings.

Introduction

Multi-agent reinforcement learning (MARL) is an emerging topic which has been explored both in theoretical (Nguyen et al. 2014; Zhang et al. 2018; Qu et al. 2019; Zhang et al. 2021b) and empirical settings (Helou, Kalathil, and Xie 2020; Mukherjee, Bai, and Chakrabortty 2020; Zhou et al. 2020). Several appealing applications of MARL can be seen in (Zhang, Yang, and Başar 2019; Nguyen, Nguyen, and Nahavandi 2020) and relevant references therein.

While MARL can primarily be cast into two different categories, i.e., cooperative (Li, Chen, and Chen 2020; Wang et al. 2020; Li et al. 2020) and competitive (Chen et al. 2020), our focus is in the cooperative setting; see (Wei et al. 2021) for details on the competitive setting. Cooperative MARL is typically modeled as a networked multi-agent Markov decision process (MDP) (Zhang, Yang, and Basar 2018; Chu, Chinchali, and Katti 2020; Zhang et al. 2018) in which the agents share a centralized reward function (Simões, Lau, and Reis 2020; Ackermann et al. 2019). However, in practice, this is not necessarily the case, and instead a more general yet challenging scenario is that agents have *heterogeneous* reward functions. Inherently, the ultimate goal in such a cooperative MARL setting is for agents to maximize the *global average* of local long-term returns. To address this problem, various algorithms have been proposed, including distributed-learning (Arslan and Yüksel 2016; Nguyen and Mukhopadhyay 2017) and distributed actor-critic (Li et al. 2020; Ryu, Shin, and Park 2020). More recent works have successfully showed finite-sample analysis for decentralized batch MARL (Zhang et al. 2021b) and leveraged advances in analysis of descent-ascent algorithms (Lu et al. 2021).

These preliminary attempts have facilitated the theoretical understanding of cooperative MARL by showing explicit sample complexity bounds, which match that of standard (vanilla) stochastic gradient descent (`SGD`). Additionally, recent works (Huang et al. 2020; Xu, Gao, and Gu 2019) in centralized RL have revealed that with simple *variance reduction* techniques, this sample complexity can be reduced to $\mathcal{O}(\epsilon^{-3})$ to reach an ϵ-stationary point (i.e., $\mathbb{E}[\|\nabla J(\mathbf{x})\|] \leq \epsilon$, where J is the return function and $\mathbf{x} \in \mathbb{R}^d$ is the decision variable to be optimized), which has been admitted as the best complexity in decentralized optimization (Das et al. 2020; Karimireddy et al. 2020). However, no similar matching bounds have yet been reported in the decentralized (cooperative MARL) setting. Hence, this motivates the question:

Can we achieve a sample complexity of $\mathcal{O}(\epsilon^{-3})$ in decentralized MARL via variance reduction?

In this paper, we answer this question affirmatively by proposing a variance-reduced policy gradient tracking approach, `MDPGT` (Algorithm 1), and analyzing it in Theorem 1. Additionally, we propose a variation (based on a different initialization) that enables state-of-the-art (SOTA) sample complexity for decentralized MARL (Zhang et al. 2021b; Lu et al. 2021). See Table 1 for SOTA comparisons. Specifically:

1. We propose `MDPGT`, in which we use a stochastic policy gradient surrogate, a convex combination of the vanilla stochastic policy gradient and an importance sampling-based stochastic recursive algorithm (`SARAH`) (Nguyen et al. 2017) for the local gradient update. Instead of directly applying the stochastic policy gradient surro-

gate in the parameter update, an intermediate parameter is adopted to track the difference between two consecutive stochastic policy gradient surrogates. For smooth nonconcave performance functions, we show that MDPGT with the mini-batch initialization can converge to an ϵ-stationary point in $\mathcal{O}(N^{-1}\epsilon^{-3})$ gradient-based updates which matches the best available known upper bounds (Huang et al. 2020).

2. We modify the initialization of the proposed algorithm MDPGT by using a single trajectory instead of a mini-batch of trajectories. Surprisingly, we find that *only one trajectory* results in a larger sampling complexity $\mathcal{O}(N^{-1}\epsilon^{-4})$, which, however, is the same as obtained by the SOTA (Zhang et al. 2021b; Lu et al. 2021) with a linear speed up when ϵ is sufficiently small. Additionally, our algorithm shows that when updating the policy parameter in MDPGT, the mini-batch size is $\mathcal{O}(1)$ instead of being ϵ-related (Xu, Gao, and Gu 2019; Qu et al. 2019), which can significantly improve practical efficiency.
3. To facilitate the theoretical understanding of MDPGT, we leverage a benchmark gridworld environment for numerical simulation and compare our proposed algorithm to a baseline decentralized policy gradient (DPG) and the momentum-based decentralized policy gradient (MDPG, described in the supplementary materials), which is a new variant created in this work for the purpose of empirical comparison. We show that our theoretical claims are valid based on the experiments.

Related Works. Most previous decentralized MARL papers (Zhang et al. 2018; Suttle et al. 2020; Li et al. 2020; Chen et al. 2020; Bono et al. 2018) tend to focus on convergence to the optimal return. Exceptions include (Qu et al. 2019), where they proved non-asymptotic convergence rates with nonlinear function approximation using value propagation. This enables us to approximately derive the number of stochastic gradient evaluations. However, the algorithm involves the complex inner-outer structure and requires the size of the mini-batch to be $\sqrt{K}$, with K being the number of iterations, which may not be practically implementable. Zhang et al. (2021b) obtain $\mathcal{O}(\epsilon^{-4})$ for the cooperative setting by using gradient tracking (GT), which is a bias correction technique dedicated to decentralized optimization, but with several specifically imposed assumptions, such as stationary sample paths, which may not be realistic. Lu et al. (2021) also utilize GT but require dual parameter updates to achieve $\mathcal{O}(\epsilon^{-4})$; our approach is different and simpler. In this context, we mention that centralized counterparts of MARL (Huang et al. 2020; Xu, Gao, and Gu 2019; Papini et al. 2018) have achieved sample complexity of $\mathcal{O}(\epsilon^{-3})$. However, in both Xu, Gao, and Gu (2019) and Papini et al. (2018), the size of mini-batch is ϵ-related, which is more computationally sophisticated than those in both (Huang et al. 2020) and our proposed method. We provide additional discussion of related work in the supplementary materials.

Preliminaries

We first formulate MARL, followed by an overview of variance reduction techniques and decentralized policy gradients.

MARL Formulation

In this context, we consider a networked system involving multiple agents (say N) that *collaboratively* solve *dynamic* optimization problems. Specifically, the system can be quantified as a graph, i.e., $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, ..., N\}$ is the vertex set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. Throughout the paper, we assume that $\mathcal{G}$ is *static and undirected*, though in a few previous works (Lin et al. 2019; Suttle et al. 2020; Zhang et al. 2018) $\mathcal{G}$ could be directed. The goal of this work is to provide the rigorous theoretical analysis for our decentralized MARL algorithm, with the property of the graph not being the main focus. When a pair of agents i and j can communicate with each other, we have $(i, j) \in \mathcal{E}$. We also define the neighborhood of a specific agent i, $Nb(i)$, such that $Nb(i) \triangleq \{j | j \in \mathcal{V}, (i, j) \in \mathcal{E} \text{or } j = i\}$. Only agents in $Nb(i)$ are able to communicate with the agent i. We next present the definition of networked MARL on top of $\mathcal{G}$.

With multiple agents, the networked Markov decision process is thus characterized by a tuple $(\mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{V}}, \mathcal{P}, \{r^i\}_{i\in\mathcal{V}}, \mathcal{G}, \gamma)$, where $\mathcal{S}$ indicates a global state space shared by all agents in $\mathcal{V}$ with $|\mathcal{S}| < \infty$, $\mathcal{A}^i$ signifies the action space specified for agent i, and $\gamma \in (0, 1]$ is the discount factor. Moreover, in the cooperative MARL setting, the environment is driven by the *joint* action space instead of individual action spaces. Thus, $\mathcal{A} \triangleq \prod_{i\in\mathcal{V}} \mathcal{A}^i$ is defined as the joint action space over all agents in $\mathcal{V}$. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the probability function to transition the current state to the next state. $\{r^i\}_{i\in\mathcal{V}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the local reward function of agent i and $r^i \in [-R, R] (R > 0)$. Additionally, states and actions are assumed to be *globally* observable, while the rewards are only locally observable. Such an assumption corresponds to the definition of the cooperative MARL setting and has been generic in previous works (Zhang et al. 2018; Zhang, Yang, and Basar 2018; Zhang et al. 2021b).

We next describe how agents behave in such an environment. Suppose that the current state of the environment is $s_k \in \mathcal{S}$, where k is the time step. Each agent i chooses its own action $a^i_k \in \mathcal{A}^i$, based on the local policy, $\pi^i : \mathcal{S} \times \mathcal{A}^i \to [0, 1]$. For a parameterized policy, we denote by $\pi^i_{\mathbf{x}^i}(s, a^i)$, which indicates the probability of agent i choosing action a^i given the current state s and $\mathbf{x}^i \in \mathbb{R}^{d_i}$ here is the policy parameter. Stacking all local policy parameter together yields $\mathbf{x} = [(\mathbf{x}^1)^\top, (\mathbf{x}^2)^\top, ..., (\mathbf{x}^N)^\top]^\top \in \mathbb{R}^{\sum_{i\in\mathcal{V}} d_i}$. Hence, the joint policy can be denoted as $\pi_{\mathbf{x}} : \mathcal{S} \times \mathcal{A} \to [0, 1]$, where $\pi_{\mathbf{x}}(s, a) \triangleq \prod_{i\in\mathcal{V}} \pi^i_{\mathbf{x}^i}(s, a^i)$ and $a \in \mathcal{A}$. In this context, the decisions are decentralized due to locally observable rewards, locally evaluated policies and locally executed actions. To simplify the notations, we drop the $\mathbf{x}^i$ for $\pi^i_{\mathbf{x}^i}$ and $\mathbf{x}$ for $\pi_{\mathbf{x}}$ respectively for local and joint policies throughout the rest of the paper. With the joint policy π and the state transition function $\mathcal{P}$, the environment evolves from s to s' with the probability $\mathcal{P}(s'|s, a)$. Another assumption imposed in this paper for the policy function is that for all $i \in \mathcal{V}, s \in \mathcal{S}, a^i \in \mathcal{A}^i$, $\pi^i(s, a^i)$ is continuously differentiable w.r.t. all $\mathbf{x}^i \in \mathbb{R}^{d_i}$. Such an assumption will assist in characterizing the smoothness of the objective function.

The goal for each agent is to learn a local policy π^i_* such

Method	Complexity	Dec.	Var. Red.	Linear Speed Up	I.S.
MBPG (Huang et al. 2020)	$\mathcal{O}(\epsilon^{-3})$	✗	✓	✗	✓
Value Prop (Qu et al. 2019)	$\mathcal{O}(\epsilon^{-4})$	✓	✗	✗	✗
DCPG (Zeng et al. 2020)	$\mathcal{O}(\epsilon^{-4})$	✓	✗	✗	✗
Safe-Dec-PG (Lu et al. 2021)	$\mathcal{O}(\epsilon^{-4})$	✓	✓	✗	✗
DFQI (Zhang et al. 2021b)	$\mathcal{O}(\epsilon^{-4})$	✓	✓	✗	✗
Dec-TD(0)+GT (Lin and Ling 2021)	N/A	✓	✓	✗	✗
MDPGT (ours)	$\mathcal{O}(\epsilon^{-4})$	✓	✓	✓	✓
MDPGT-MI (ours)	$\mathcal{O}(\epsilon^{-3})$	✓	✓	✓	✓

Table 1: *Comparisons between existing and proposed approaches.*

[1] **Complexity**: Sampling complexity for achieving $\mathbb{E}[\|\nabla J(\mathbf{x})\|] \leq \epsilon$.
[2] **Linear Speed Up**: If an algorithm has $\mathcal{O}(1/\sqrt{K})$ convergence, then its sampling complexity of attaining an $\mathcal{O}(\epsilon)$ accurate solution is ϵ^{-2}. Similarly, $\mathcal{O}(1/\sqrt{NK})$ corresponds to $N^{-1}\epsilon^{-2}$, which is N times faster than the former. Typically, K has to satisfy a certain condition.
[3] MDPGT-MI is MDPGT with mini-batch initialization. We use this notation for conveniently classifying two different initialization approaches. In the rest of paper, we still adopt MDPGT to unify these two approaches.
[4] **Dec.**: decentralized. [5] **Var. Red.**: variance reduction. [4] **I.S.** importance sampling.

that the joint policy π_* is able to maximize the *global average* of expected cumulative discounted rewards, i.e.,

$$\pi_* = \text{argmax}_{\mathbf{x}\in\mathbb{R}^d} J(\mathbf{x}) \triangleq \frac{1}{N}\sum_{i\in\mathcal{V}}\mathbb{E}\left[\sum_{h=0}^{H}\gamma^h r_h^i\right], \quad (1)$$

where H is the horizon and $d = \sum_{i\in\mathcal{V}} d_i$. Several works (Zhang et al. 2018; Qu et al. 2019; Zhang et al. 2021b; Lin et al. 2019; Suttle et al. 2020) have made their attempts to resolve this optimization problem, leading to different algorithms. Since each agent only has access to local information, a communication protocol needs to be introduced in the system, as done in decentralized optimization. With that, well-known centralized policy-based algorithms can be extended as MARL algorithms. Nevertheless, one issue that has not been sufficiently explored is *the inherent policy gradient variance, which could even be more significant in the MARL algorithms.* Consequently, this work propose novel MARL algorithms to investigate how to reduce the policy gradient variance during the optimization.

Variance Reduction and Bias Correction

In stochastic optimization, variance reduction techniques have been well studied and applied to either centralized or decentralized gradient descent type of algorithms, such as SVRG (Johnson and Zhang 2013), SARAH (Nguyen et al. 2017), SPIDER (Fang et al. 2018), Hybrid-SARAH (Tran-Dinh et al. 2019) and STORM (Cutkosky and Orabona 2019). In another line of work, the GT technique (Pu and Nedić 2020; Sun, Daneshmand, and Scutari 2019) was proposed specifically for consensus-based decentralized optimization techniques to improve the convergence rate by tracking and correcting each agent's locally aggregated gradients. In our work, we leverage both Hybrid-SARAH and GT to reduce the policy gradient variance and correct the policy gradient bias respectively in the MARL and achieve the best convergence rate. Hybrid-SARAH performs with a trade-off parameter to balance the effect between vanilla stochastic gradient and SARAH. More detail on these techniques are elaborated in the supplementary materials.

So far, we are not aware of existing results that have successfully shown SARAH or Hybrid-SARAH type of variance reduction techniques well suited for *decentralized non-oblivious* learning problems, e.g., MARL. Consequently, the regular Hybrid-SARAH technique cannot be directly applied to MARL; we address this challenge in sections below.

Decentralized Policy Gradient

Given a time horizon H, we define a trajectory specifically for agent i, as $\tau^i \triangleq \{s_0, a_0^i, ..., s_{H-1}, a_{H-1}^i\}$ under any stationary policy. By following the trajectory τ^i, a cumulative discounted reward is given as $\mathcal{R}_i(\tau^i) \triangleq \sum_{h=0}^{H}\gamma^h r_h^i$ such that an individual return can be obtained as:

$$J_i(\mathbf{x}^i) \triangleq \mathbb{E}_{\tau^i\sim p_i(\tau^i|\mathbf{x}^i)}[\mathcal{R}_i(\tau^i)] = \int \mathcal{R}_i(\tau^i)p_i(\tau^i|\mathbf{x}^i)d\tau^i, \quad (2)$$

where $p_i(\tau^i|\mathbf{x}^i)$ is the probability distribution over τ^i that is equivalent to the following expression given the initial distribution $\rho_0^i = \rho^i(s_0)$. Without loss of generalization, we can assume that the initial distribution is identical for all agents, namely $\rho(s_0)$. Then, we have,

$$p_i(\tau^i|\mathbf{x}^i) = \rho_0(s_0)\prod_{h=0}^{H-1}\mathcal{P}(s_{h+1}|s_h, a_h^i)\pi^i(a_h^i|s_h). \quad (3)$$

For each agent i, the goal is to find an optimal policy π_*^i to maximize the return $J_i(\mathbf{x}^i)$. As discussed above, the underlying dynamic distribution results in a non-oblivious learning problem, which is more significant in MARL. To resolve this issue, the decentralized policy gradient is a decent choice. As background knowledge of MARL, we next present how to arrive at the local stochastic policy gradient, which will help characterize the analysis for the proposed algorithms.

Computing the gradient of $J_i(\mathbf{x}^i)$ w.r.t $\mathbf{x}^i$ yields the following formula:

$$\begin{aligned}\nabla J_i(\mathbf{x}^i) &= \int \mathcal{R}_i(\tau^i)\frac{\nabla p_i(\tau^i|\mathbf{x}^i)}{p_i(\tau^i|\mathbf{x}^i)}p_i(\tau^i|\mathbf{x}^i)d\tau^i \\ &= \mathbb{E}_{\tau^i\sim p_i(\tau^i|\mathbf{x}^i)}[\nabla \log p_i(\tau^i|\mathbf{x}^i)\mathcal{R}_i(\tau^i)]\end{aligned} \tag{4}$$

In practice, $p_i(\tau^i|\mathbf{x}^i)$ is typically unknown such that the accurate full policy gradient for agent i is difficult to obtain. Thus, similar to decentralized stochastic gradient descent (Jiang et al. 2017), we calculate the policy gradient by sampling a mini-batch of trajectories $\mathcal{B} = \{\tau^i_m\}_{m=1}^{|\mathcal{B}|}$ from the distribution $p_i(\tau^i|\mathbf{x}^i)$ such that

$$\hat{\nabla} J_i(\mathbf{x}^i) = \frac{1}{|\mathcal{B}|}\sum_{m\in\mathcal{B}} \nabla \log p_i(\tau^i_m|\mathbf{x}^i)\mathcal{R}_i(\tau^i_m). \tag{5}$$

In addition, combining Eq. 3, we can observe that $\nabla \log p_i(\tau^i_m|\mathbf{x}^i)$ is independent of the probability transition $\mathcal{P}$. Hence, Eq. 5 is written as

$$\begin{aligned}\hat{\nabla} J_i(\mathbf{x}^i) &= \frac{1}{|\mathcal{B}|}\sum_{m\in\mathcal{B}} \mathbf{g}_i(\tau^i_m|\mathbf{x}^i) \\ &= \frac{1}{|\mathcal{B}|}\sum_{m\in\mathcal{B}} \left(\sum_{h=0}^{H-1} \nabla_{\mathbf{x}^i}\log\pi^i(a_h^{i,m}, s_h^m)\right)\cdot \\ &\left(\sum_{h=0}^{H-1}\gamma^h r_h^i(a_h^{i,m}, s_h^m)\right)\end{aligned} \tag{6}$$

In the above equation, $\mathbf{g}_i(\tau^i|\mathbf{x}^i)$ is the unbiased estimate of $\nabla J_i(\mathbf{x}^i)$, i.e., $\mathbb{E}[\mathbf{g}^i(\tau^i|\mathbf{x}^i)] = \nabla J_i(\mathbf{x}^i)$. Some well-known policy gradient estimators can be obtained through Eq. 6, such as decentralized REINFORCE, which is the direct extension of its centralized version. We refer interested readers to (Huang et al. 2020) for more details.

Our Proposed Approach: MDPGT

Hybrid Importance Sampling SARAH

In this subsection, we propose a hybrid importance sampling version of SARAH, termed HIS-SARAH, for decentralized policy gradient updates. First, we define the importance sampling weight (Metelli et al. 2020) as follows:

$$\upsilon(\tau|\mathbf{x}', \mathbf{x}) = \frac{p(\tau|\mathbf{x}')}{p(\tau|\mathbf{x})} = \prod_{h=0}^{H-1}\frac{\pi_{\mathbf{x}'}(a_h|s_h)}{\pi_{\mathbf{x}}(a_h|s_h)}. \tag{7}$$

As mentioned in the last section, due to the non-oblivious learning problem, $\mathbb{E}_{\tau\sim p(\tau|\mathbf{x})}[\mathbf{g}(\tau|\mathbf{x}) - \mathbf{g}(\tau|\mathbf{x}')] \neq \nabla J(\mathbf{x}) - \nabla J(\mathbf{x}')$. With Eq. 7 we have $\mathbb{E}_{\tau\sim p(\tau|\mathbf{x})}[\mathbf{g}(\tau|\mathbf{x}) - \upsilon(\tau|\mathbf{x}', \mathbf{x})\mathbf{g}(\tau|\mathbf{x}')] = \nabla J(\mathbf{x}) - \nabla J(\mathbf{x}')$, which has been analyzed in (Huang et al. 2020) for centralized policy optimization methods and will be a key relationship in our proof. We denote by $\mathbf{u}^i$ the stochastic policy gradient surrogate for agent i. Thus, applying Eq. 7 in a decentralized manner for Hybrid-SARAH (See Supplementary materials for definition) gives the following update law at a time step k:

$$\begin{aligned}\mathbf{u}_k^i &= \beta\mathbf{g}_i(\tau_k^i|\mathbf{x}_k^i) + (1-\beta)[\mathbf{u}_{k-1}^i + \mathbf{g}_i(\tau_k^i|\mathbf{x}_k^i) \\ &- \upsilon_i(\tau_k^i|\mathbf{x}_{k-1}^i, \mathbf{x}_k^i)\mathbf{g}_i(\tau_k^i|\mathbf{x}_{k-1}^i)].\end{aligned} \tag{8}$$

Algorithm 1: MDPGT

```
Result: $\tilde{\mathbf{x}}_K$ chosen uniformly random from $\{\mathbf{x}_k^i, i\in\mathcal{V}\}_{k=1}^K$
Input: $\mathbf{x}_0^i = \bar{\mathbf{x}}_0 \in \mathbb{R}^d, \eta \in \mathbb{R}^+, \beta \in (0,1), \mathbf{W} \in \mathbb{R}^{N\times N}, \mathbf{v}_0^i = \mathbf{0}_d, \mathbf{u}_{-1}^i = \mathbf{0}_d, K, \mathcal{B} \in \mathbb{Z}^+, k = 1$
Initialize the local policy gradient surrogate by sampling
  a trajectory $\tau_0^i$ from $p_i(\tau^i|\mathbf{x}_0^i)$ : $\mathbf{u}_0^i = \mathbf{g}_i(\tau_0^i|\mathbf{x}_0^i)$, or by
  sampling a mini-batch of trajectories $\{\tau_0^{i,m}\}_{m=1}^{|\mathcal{B}|}$ from
  $p_i(\tau^i|\mathbf{x}_0^i)$ : $\mathbf{u}_0^i = \frac{1}{|\mathcal{B}|}\sum_{m=1}^{|\mathcal{B}|}\mathbf{g}_i(\tau_0^{i,m}|\mathbf{x}_0^i)$
Initialize the local policy gradient tracker: $\mathbf{v}_1^i = \sum_{j\in Nb(i)}\omega_{ij}\mathbf{v}_0^j + \mathbf{u}_0^i - \mathbf{u}_{-1}^i$
Initialize the local estimate of the policy network parameter:
  $\mathbf{x}_1^i = \sum_{j\in Nb(i)}\omega^{ij}(\mathbf{x}_0^j + \eta\mathbf{v}_1^j)$
while $k < K$ do
    for each agent do
        Sample a trajectory $\tau_k^i$ from $p_i(\tau^i|\mathbf{x}_k^i)$ and compute
          the local policy gradient surrogate using Eq. 8
        Update the local policy gradient tracker $\mathbf{v}_{k+1}^i = \sum_{j\in Nb(i)}\omega_{ij}\mathbf{v}_k^j + \mathbf{u}_k^i - \mathbf{u}_{k-1}^i$
        Update the local estimate of the policy network parameters $\mathbf{x}_{k+1}^i = \sum_{j\in Nb(i)}\omega_{ij}(\mathbf{x}_k^j + \eta\mathbf{v}_{k+1}^j)$
    end
    $k = k + 1$
end
```

The second term on the right hand side of Eq. 8 differ in the extra importance sampling weight compared to Eq. 13 in the supplementary materials. Intuitively, $\upsilon_i(\tau_k^i|\mathbf{x}_{k-1}^i, \mathbf{x}_k^i)$ resolves the non-stationarity in the MARL and retains the regular variance reduction property of HIS-SARAH as applied in supervised learning problems. Clearly, each $\mathbf{u}_k^i$ is a *conditionally* biased estimator $\nabla J_i(\mathbf{x}_k^i)$, i.e., $\mathbb{E}[\mathbf{u}_k^i] \neq \nabla J_i(\mathbf{x}_k^i)$ typically. Nevertheless, it can be shown that $\mathbb{E}[\mathbf{u}_k^i] = \mathbb{E}[\nabla J_i(\mathbf{x}_k^i)]$, which implies that $\mathbf{u}_k^i$ acts as a *surrogate* for the underlying exact full policy gradient. Therefore, $\mathbf{u}_k^i$ will be called directly the stochastic policy gradient surrogate for the rest of analysis. With Eq. 8 in hand, we now are ready to present the algorithmic framework in the following.

Algorithmic Framework

We first present MDPGT (in Algorithm 1), which only takes a trajectory to initialize the policy gradient surrogate, leading to significant randomness due to the conditionally biased estimator property at the starting point of optimization, but still retaining the same sampling complexity as compared to the SOTA of MARL. To have a better initialization, we also present another way of initialization by sampling a mini-batch of trajectories from the distribution (in blue in Algorithm 1). Surprisingly, we will see that with a proper size of mini-batch initialization, the sampling complexity of our proposed approach complies with the best result in centralized RL, which improves the SOTA of MARL.

A brief outline of Algorithm 1 is as follows. The initialization of the policy gradient surrogate $\mathbf{u}_0^i$ can either be based on only a trajectory sampled from $p_i(\tau^i|\mathbf{x}_0^i)$ or a mini-batch.

Subsequently, the policy gradient tracker and network parameters are initialized based on $\mathbf{u}_0^i$. The core part of the algorithm consists of each individual update for $\mathbf{u}_k^i, \mathbf{v}_k^i$, and $\mathbf{x}_k^i$. By controlling the value of β in Eq. 8, `MDPGT` can degenerate to either vanilla decentralized policy gradient (with $\beta = 1$) or decentralized version of `SRVR-PG` (Xu, Gao, and Gu 2019) (with $\beta = 0$), both with the gradient tracking step. In our work, to emphasize the impact of the trade-off on the policy gradient surrogate, we keep $\beta \in (0, 1)$, which makes β act more closely as the momentum coefficient in accelerated `SGD` algorithms (Singh et al. 2020).

We emphasize that we are unaware of theoretical results for decentralized `SRVR-PG`. Hence, the proof techniques presented in this paper can also apply to this case. Another implication from Algorithm 1 is that at the beginning of each time step k, only one trajectory is required for computing the policy gradient, allowing for the batch size to be *independent* of ϵ, i.e., $\mathcal{O}(1)$, where we omit the number of agents N when considering the whole networked system.

Theoretical Convergence

In this section, we present an analysis of MDPGT. Most of the assumptions below are mild, and standard in the decentralized optimization and RL literature. Due to space limitations, we defer auxiliary lemmas and proofs to the supplementary materials.

Assumption 1. *Gradient and Hessian matrix of function* $\log\pi^i(a^i|s)$ *are bounded, i.e., there exist constants* $C_g, C_h > 0$ *such that* $\|\nabla\log\pi^i(a^i|s)\| \leq C_g$ *and* $\|\nabla^2\log\pi^i(a^i|s)\| \leq C_h$*, for all* $i \in \mathcal{V}$.

Note that we skip the subscript $\mathbf{x}^i$ at π^i for the notation simplicity. In this context, we did not impose the bounded policy gradient assumption, though it can be derived based on the above assumption, which has been adopted in centralized RL algorithms (Zhang et al. 2021a; Huang et al. 2020; Xu, Gao, and Gu 2019). Additionally, it also helps derive the smoothness of $J_i(\mathbf{x}^i)$ that has typically been exposed as an assumption in decentralized learning/optimization literature.

Assumption 2. *The mixing matrix* $\mathbf{W} \in \mathbb{R}^{N\times N}$ *is doubly stochastic such that* $\lambda \triangleq \|\mathbf{W} - \mathbf{P}\| \in [0, 1)$*, where* λ *signifies the second largest eigenvalue to measure the algebraic connectivity of the graph, and* $\mathbf{P} = \frac{1}{N}\mathbf{1}^\top\mathbf{1}$ *and* $\mathbf{1}$ *is a column vector with each entry being 1.*

Assumption 3. *Variance of importance sampling weight* $\upsilon_i(\tau^i|\mathbf{x}_1, \mathbf{x}_2)$ *is bounded, i.e., there exists a constant* $\mathcal{M} > 0$ *such that* $\mathbb{V}(\upsilon_i(\tau^i|\mathbf{x}_1, \mathbf{x}_2)) \leq \mathcal{M}$*, for any* $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{d_i}$ *and* $\tau^i \sim p_i(\tau^i|\mathbf{x}_1)$*, for all* $i \in \mathcal{V}$.

Assumption 2 is generic in most previous works on decentralized optimization, though such a property has been relaxed in some works (Nedić and Olshevsky 2014). However, we have not been aware of any existing works in MARL doing such a relaxation and its investigation can be of independent interest. Assumption 3 is specifically introduced for importance sampling-based methods. Such an assumption is critical to construct the relationship between $\mathbb{V}(\upsilon_i(\tau^i|\mathbf{x}_1, \mathbf{x}_2))$ and $\|\mathbf{x}_1 - \mathbf{x}_2\|^2$, through which the impact of the variance of importance sampling on the convergence can be explicitly quantified. Another typical assumption is for the bounded variance of stochastic gradient such that $\mathbb{E}[\|\mathbf{g}_i(\tau^i|\mathbf{x}^i) - \nabla J_i(\mathbf{x}^i)\|^2] \leq \sigma_i^2$. However, under MARL setting, such a result can be derived from Assumption 1 and we present the formal result in Lemma 1. In this context, we also have $\bar{\sigma}^2 = \frac{1}{N}\sum_{i=1}^N \sigma_i^2$, for all $i \in \mathcal{V}$. The explicit expression of $\bar{\sigma}^2$ is given in the supplementary materials.

Main Results

We present the main results to show specifically the convergence rates for `MDPGT` when it is initialized by a mini-batch of trajectories. We denote by $L > 0$ the *smoothness constant* and $G > 0$ the *upper bound* of $\|\mathbf{g}_i(\tau^i|\mathbf{x}^i)\|$ for all $i \in \mathcal{V}$. We further define a constant $C_v > 0$ such that $\mathbb{V}(\upsilon_i(\tau^i|\mathbf{x}_1, \mathbf{x}_2)) \leq C_v^2\|\mathbf{x}_1 - \mathbf{x}_2\|^2$. The explicit expressions of these constants are derived in lemmas in the supplementary materials. Note that in our work, G is not directly assumed, but instead derived based on Assumption 1.

Theorem 1. *Let Assumptions 1,2 and 3 hold. Let the momentum coefficient* $\beta = \frac{96L^2+96G^2C_v}{N}\eta^2$*. If* `MDPGT` *is initialized by a mini-batch of trajectories with the size being* $\mathcal{B}$ *and the step size satisfies the following condition*

$$0 < \eta \leq \min\left\{\frac{(1-\lambda^2)^2}{\lambda\sqrt{12844L^2 + 9792G^2C_v}}, \frac{\sqrt{N(1-\lambda^2)\lambda}}{31\sqrt{L^2+G^2C_v}}, \frac{1}{6\sqrt{6(L^2+G^2C_v)}}\right\},$$

then the output $\tilde{\mathbf{x}}_K$ *satisfies: for all* $K \geq 2$*:*

$$\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|^2] \leq \frac{4(J^* - J(\bar{\mathbf{x}}_0))}{\eta K} + \frac{4\bar{\sigma}^2}{N|\mathcal{B}|\beta K} + \frac{8\beta\bar{\sigma}^2}{N} + \frac{34\lambda^2\|\nabla\mathbf{J}(\bar{\mathbf{x}}_0)\|^2}{KN(1-\lambda^2)^3} + \frac{68\lambda^2\bar{\sigma}^2}{(1-\lambda^2)^3|\mathcal{B}|K} + \frac{204\lambda^2\beta^2\bar{\sigma}^2}{(1-\lambda^2)^3}, \tag{9}$$

where J^* *is the upper bound of* $J(\mathbf{x})$ *and* $\|\nabla\mathbf{J}(\bar{\mathbf{x}}_0)\|^2 \triangleq \sum_{i=1}^N \|\nabla J_i(\bar{\mathbf{x}}_0)\|^2$.

Theorem 1 depicts that when $K \to \infty$, `MDPGT` enables convergence to a *steady-state error* in a sublinear rate $\mathcal{O}(1/K)$ if η and β are selected properly, i.e.,

$$\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|^2] \leq \frac{8\beta\bar{\sigma}^2}{N} + \frac{204\lambda^2\beta^2\bar{\sigma}^2}{(1-\lambda^2)^3}. \tag{10}$$

By observing Eq. 10, the steady-state error is determined by the number of agents, the variance of stochastic policy gradient, and the spectral gap of the graph $1 - \lambda$. Increasing the number of agents leads to a small error bound. Though different network topologies imply different error bounds, the higher order term of β can reduce the impact of the spectral gap on the error bound. Another suggestion from Eq. 10 is that η and β can be reduced to make the steady-state error arbitrarily small, while in return this can affect the speed of convergence. Surprisingly, even though we have to adopt the bounded stochastic policy gradient derived from Assumption 1 for analysis, the error bound in Eq. 10 only depends heavily on the variance, which is inherently consistent with most conclusions from decentralized optimization in literature without the bounded stochastic gradient assumption. While J^* is essentially correlated to the upper bound of reward R, in this context, we still adopt the implicit J^* for convenience. We next provide the analysis for the non-asymptotic behavior, defining appropriately η, β, and $|\mathcal{B}|$.

Corollary 1. *Let $\eta = \frac{N^{2/3}}{8LK^{1/3}}, \beta = \frac{DN^{1/3}}{64L^2K^{2/3}}, |\mathcal{B}| = \lceil \frac{K^{1/3}}{N^{2/3}} \rceil$ in Theorem 1. We have,*

$$\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|^2] \leq \underbrace{\frac{256L^3D(J^* - J(\bar{\mathbf{x}}_0)) + 2048L^4\bar{\sigma}^2 + D^2\bar{\sigma}^2}{8L^2D(NK)^{2/3}}}_{T_1} + \underbrace{\frac{34\lambda^2\|\nabla \mathbf{J}(\bar{\mathbf{x}}_0)\|^2}{KN(1-\lambda^2)^3} + \frac{\lambda^2\bar{\sigma}^2(51D^2 + 69632N^{2/3}L^4)}{1024(1-\lambda^2)^3K^{4/3}L^4}}_{T_2}, \tag{11}$$

for all

$$K \geq \max\left\{\frac{N^2D^{1.5}}{512L^3}, \frac{29791\sqrt{N}(L^2 + G^2C_v)^{1.5}}{512L^3\lambda^3(1-\lambda^2)^{1.5}}, \frac{(12844L^2 + 9792G^2C_v)^{1.5}N^2\lambda^3}{512L^3(1-\lambda^2)^6}\right\},$$

where $D = 96L^2 + 96G^2C_v$.

Remark 1. *An implication from Corollary 1 is that at the early stage of optimization, before T_1 in Eq. 11 dominates, the complexity is tightly related to the algebraic connectivity of the network topology in T_2, which is measured by the spectral gap $1 - \lambda$. However, T_2 is in a large order of $1/K$. As the optimization moves towards the latter stage where T_1 dominates, the overall complexity is independent of the network topology.*

For the ease of exposition, with Corollary 1, when K is sufficiently large, it is an immediate consequence as $\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|^2] \leq \mathcal{O}((NK)^{-2/3})$. Thus, for achieving $\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|] \leq \epsilon$, the following relationship is obtained: $\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|] = \sqrt{(\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|])^2} \leq \sqrt{\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|^2]} \leq \epsilon$. Combining the last two inequalities results in the ultimate sampling complexity, i.e., $\mathcal{O}(N^{-1}\epsilon^{-3})$, which exhibits *linear speed up*. More importantly, this is N times smaller than the sampling complexity of the centralized momentum-based policy gradient methods (Huang et al. 2020) that performs on a single node. However, we have known from Corollary 1 that typically K has to be large, which will in the linear speed up regime reduce η and β.

We also investigate a worse initialization with only a single trajectory sampled from $p_i(\tau^i|\mathbf{x}_0^i)$. However, without a mini-batch initialization, the eventual sampling complexity is $\mathcal{O}(N^{-1}\epsilon^{-4})$ (see results in the supplementary materials). Though variance reduction techniques have not reduced the order of ϵ^{-1}, compared to the SOTA approaches, the linear speed up still enables the complexity to be N times smaller than that in (Xu, Gao, and Gu 2019; Huang et al. 2020). Additionally, different from traditional decentralized learning problems, MARL has more significant variances in the optimization procedure due to the non-oblivious characteristic. Using just a single trajectory for each agent to initialize is can be a poor scheme, but the adopted variance reduction techniques can successfully maintain the SOTA sampling complexity in a decentralized setting. Please refer to the supplementary materials for formal results and proof.

Implication for Gaussian Policy. We study the sample complexity when the policy function $\pi^i(a^i|s)$ of each agent is explicitly a Gaussian distribution. For a bounded action space $\mathcal{A}^i \subset \mathbb{R}$, a Gaussian policy parameterized by $\mathbf{x}_i$ is defined as

$$\pi^i(a^i|s) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{((\mathbf{x}^i)^\top\phi_i(s) - a^i)^2}{2\xi^2}\right), \tag{12}$$

where ξ^2 is a constant standard deviation parameter and $\phi_i(s) : \mathcal{S} \to \mathbb{R}^{d_i}$ is a mapping from the state space to the feature space. Thus, the following formal result can be obtained. A more formal analysis and proof can be seen in the supplementary materials.

Corollary 2. *Let $\pi^i(a^i|s)$ be defined as a Gaussian distribution in Eq. 12 with $|a^i| \leq C_a$, where $C_a, C_f > 0$, and $\|\phi_i(s)\| \leq C_f$, and $\eta, \beta, |\mathcal{B}|$ be defined as in Corollary 1. The sampling complexity of attaining the accuracy $\mathbb{E}[\|\nabla J(\tilde{\mathbf{x}}_K)\|] \leq \epsilon$ is $\mathcal{O}((1-\gamma)^{-4.5}N^{-1}\epsilon^{-3})$.*

Numerical Experiments and Results

To validate our proposed algorithm, we performed experiments on a cooperative navigation multi-agent environment that has been commonly used as a benchmark in several previous works (Qu et al. 2019; Zhang et al. 2018; Lu et al. 2021). Our platform for cooperative navigation is derived off the particle environment introduced by (Lowe et al. 2017). In our modification, all agents are initialized at random locations with a specific goal in a 2D grid world. Each agent observes the combined position and velocity of itself and all other agents. The agents are capable of moving up, down, left or right with the objective of navigating to their respective goals. The reward function of each agent is defined as the negative euclidean distance of the agent to the goal. Additionally, a penalty of -1 is imposed whenever the agent collides with other agents. All agent's policy is represented by a 3-layer dense neural network with 64 hidden units and $tanh$ activation functions. The agents were trained for 50k episodes with a horizon of 50 steps and discount factor of 0.99. For brevity, we present numerical results in only one environment setting with five agents. Additional results with different number of agents and a simplified environment and computing infrastructure details are available in the supplementary materials[1].

Efficacy of MDPGT. Figure 1 illustrates the average training rewards obtained by the five agents in the cooperative navigation gridworld environment. As observed, both `MDPG` and `MDPGT` significantly outperforms the baseline, denoted as `DPG`. Comparing `MDPG` with `MDPGT`, we observe that while both algorithms initially have similar performance,

[1]Codes to reproduce results are also available at:https://github.com/xylee95/MD-PGT

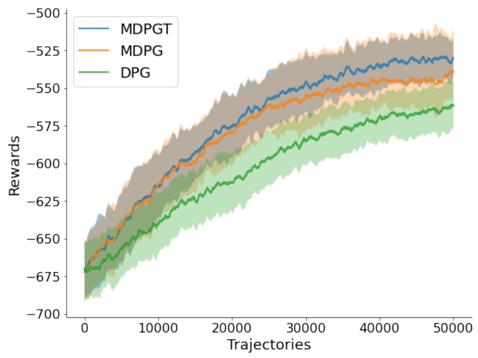

Figure 1: Average rewards of `MDPGT`($\beta = 0.5$), `MDPG` and `DPG` in a cooperative navigation task for five agents (averaged across five random seeds). The solid lines denote the mean and shaded regions denote the standard deviation of rewards.

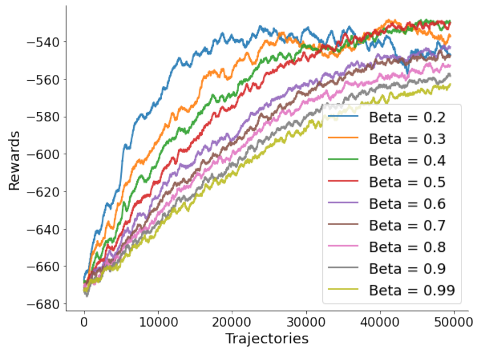

Figure 2: Ablation study illustrating the effect of various momentum coefficients, β on the performance of `MDPGT` for five agents in the cooperative navigation environment.

`MDPGT` begins to outperform `MDPG` around the 15k iteration. Additionally, when we compare the standard deviation of the rewards, shown as shaded regions, we observe that standard deviation of `MDPGT` is also smaller than the standard deviation of `MDPG`. In summary, these results validate our theoretical findings that utilizing gradient tracking as bias correction technique does improve the performance of the algorithm. Additionally, the improvement in terms of sampling complexity over `DPG` is empirically evident through the result.

Effect of Momentum Coefficient. Next, we perform an additional ablation study to investigate the effect of the momentum coefficient β on the performance of `MDPGT`. As shown in Figure 2, we see that the choice of momentum coefficient does indeed have an effect on the performance. A β that is low can induce a faster convergence rate, but at the cost of a higher fluctuations in rewards, as seen by $\beta =$ 0.2 and 0.3. Conversely, a high β value will cause the surrogate to degenerate into vanilla policy gradients and reflects a similar performance as `DPG`, which matches the implication by Eq. 10. Ultimately, for this environment, $\beta = 0.4$ and 0.5 offers the perfect balance between convergence rate and stability/variance of the training. Hence, β can be viewed as hyper-parameter which can be tuned to trade off between optimizing for convergence versus training stability.

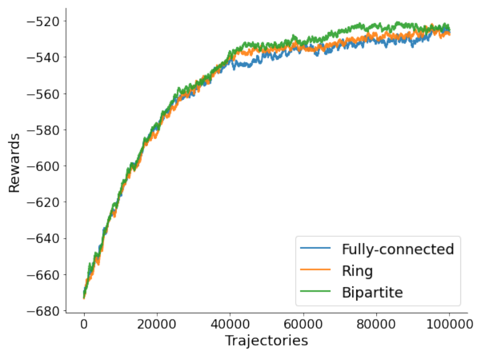

Figure 3: Experiment results for five agents in the cooperative navigation environment to compare the effects of different network topologies. $\beta = 0.5$ for all experiments shown.

Effect of Different Topologies. Finally, we provide evidence which confirms the fact that our proposed method holds under various networks topologies. To test our hypothesis, we train five agents in the same cooperative navigation environment using three different network topologies: fully-connected, ring and bi-partite topology. As seen in Figure 3, the five agents achieves similar rewards despite communicating via different network topologies. This validates our claim in Remark 1.

Conclusions

This paper proposes a novel MARL algorithm that involves variance reduction techniques to reduce the sampling complexity of decentralized policy-based methods. Specifically, we developed the algorithmic framework and analyzed it in a principled manner. An importance sampling-based stochastic recursive momentum is presented as the policy gradient surrogate, which is taken as input to a policy gradient tracker. Through theoretical analysis, we find that the proposed method can improve the sampling efficiency in the decentralized RL settings compared to SOTA methods. To the best of our knowledge, this is the first work that achieve the best available rate, $\mathcal{O}(\epsilon^{-3})$, for generic (possibly non-concave) performance functions. Empirical results have shown the superiority of the proposed `MDPGT` over the baseline decentralized policy gradient methods. Future research directions include: 1) testing on more complex decentralized environments to reveal potentially novel and interesting results; 2) extending the proposed method to model-based decentralized RL settings to improve further the sampling efficiency; 3) theoretically analyzing the robustness of the proposed method under adversarial attacks.

Acknowledgements

This work was partly supported by the National Science Foundation under grant numbers CNS-1932033, CNS-1845969, and CCF-2005804.

References

Ackermann, J.; Gabler, V.; Osa, T.; and Sugiyama, M. 2019. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465*.

Arslan, G.; and Yüksel, S. 2016. Decentralized Q-learning for stochastic teams and games. *IEEE Transactions on Automatic Control*, 62(4): 1545–1558.

Bono, G.; Dibangoye, J. S.; Matignon, L.; Pereyron, F.; and Simonin, O. 2018. Cooperative multi-agent policy gradient. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 459–476. Springer.

Chen, B.; Xu, M.; Liu, Z.; Li, L.; and Zhao, D. 2020. Delay-Aware Multi-Agent Reinforcement Learning for Cooperative and Competitive Environments. *arXiv e-prints*, arXiv–2005.

Chu, T.; Chinchali, S.; and Katti, S. 2020. Multi-agent reinforcement learning for networked system control. *arXiv preprint arXiv:2004.01339*.

Cutkosky, A.; and Orabona, F. 2019. Momentum-based variance reduction in non-convex sgd. *arXiv preprint arXiv:1905.10018*.

Das, R.; Acharya, A.; Hashemi, A.; Sanghavi, S.; Dhillon, I. S.; and Topcu, U. 2020. Faster Non-Convex Federated Learning via Global and Local Momentum. *arXiv preprint arXiv:2012.04061*.

Fang, C.; Li, C. J.; Lin, Z.; and Zhang, T. 2018. Spider: Near-optimal non-convex optimization via stochastic path integrated differential estimator. *arXiv preprint arXiv:1807.01695*.

Helou, R. E.; Kalathil, D.; and Xie, L. 2020. Fully Decentralized Reinforcement Learning-based Control of Photovoltaics in Distribution Grids for Joint Provision of Real and Reactive Power. *arXiv preprint arXiv:2008.01231*.

Huang, F.; Gao, S.; Pei, J.; and Huang, H. 2020. Momentum-based policy gradient methods. In *International Conference on Machine Learning*, 4422–4433. PMLR.

Jiang, Z.; Balu, A.; Hegde, C.; and Sarkar, S. 2017. Collaborative deep learning in fixed topology networks. *arXiv preprint arXiv:1706.07880*.

Johnson, R.; and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26: 315–323.

Karimireddy, S. P.; Jaggi, M.; Kale, S.; Mohri, M.; Reddi, S. J.; Stich, S. U.; and Suresh, A. T. 2020. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*.

Li, M.-L.; Chen, S.; and Chen, J. 2020. Adaptive learning: A new decentralized reinforcement learning approach for cooperative multiagent systems. *IEEE Access*, 8: 99404–99421.

Li, W.; Jin, B.; Wang, X.; Yan, J.; and Zha, H. 2020. F2A2: Flexible Fully-decentralized Approximate Actor-critic for Cooperative Multi-agent Reinforcement Learning. *arXiv preprint arXiv:2004.11145*.

Lin, Q.; and Ling, Q. 2021. Decentralized TD (0) with Gradient Tracking. *IEEE Signal Processing Letters*.

Lin, Y.; Zhang, K.; Yang, Z.; Wang, Z.; Başar, T.; Sandhu, R.; and Liu, J. 2019. A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 5562–5567. IEEE.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NIPS)*.

Lu, S.; Zhang, K.; Chen, T.; Basar, T.; and Horesh, L. 2021. Decentralized Policy Gradient Descent Ascent for Safe Multi-Agent Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 8767–8775.

Metelli, A. M.; Papini, M.; Montali, N.; and Restelli, M. 2020. Importance Sampling Techniques for Policy Optimization. *Journal of Machine Learning Research*, 21(141): 1–75.

Mukherjee, S.; Bai, H.; and Chakrabortty, A. 2020. Model-free decentralized reinforcement learning control of distributed energy resources. In *2020 IEEE Power & Energy Society General Meeting (PESGM)*, 1–5. IEEE.

Nedić, A.; and Olshevsky, A. 2014. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3): 601–615.

Nguyen, D. T.; Yeoh, W.; Lau, H. C.; Zilberstein, S.; and Zhang, C. 2014. Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28.

Nguyen, L. M.; Liu, J.; Scheinberg, K.; and Takáč, M. 2017. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, 2613–2621. PMLR.

Nguyen, T.; and Mukhopadhyay, S. 2017. Selectively decentralized Q-learning. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 328–333. IEEE.

Nguyen, T. T.; Nguyen, N. D.; and Nahavandi, S. 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9): 3826–3839.

Papini, M.; Binaghi, D.; Canonaco, G.; Pirotta, M.; and Restelli, M. 2018. Stochastic variance-reduced policy gradient. In *International Conference on Machine Learning*, 4026–4035. PMLR.

Pu, S.; and Nedić, A. 2020. Distributed stochastic gradient tracking methods. *Mathematical Programming*, 1–49.

Qu, C.; Mannor, S.; Xu, H.; Qi, Y.; Song, L.; and Xiong, J. 2019. Value propagation for decentralized networked deep multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09326*.

Ryu, H.; Shin, H.; and Park, J. 2020. Multi-agent actor-critic with hierarchical graph attention network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 7236–7243.

Simões, D.; Lau, N.; and Reis, L. P. 2020. Multi-agent actor centralized-critic with communication. *Neurocomputing*, 390: 40–56.

Singh, N.; Data, D.; George, J.; and Diggavi, S. 2020. SQuARM-SGD: Communication-Efficient Momentum SGD for Decentralized Optimization. *arXiv preprint arXiv:2005.07041*.

Sun, Y.; Daneshmand, A.; and Scutari, G. 2019. Convergence rate of distributed optimization algorithms based on gradient tracking. *arXiv preprint arXiv:1905.02637*.

Suttle, W.; Yang, Z.; Zhang, K.; Wang, Z.; Başar, T.; and Liu, J. 2020. A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning. *IFAC-PapersOnLine*, 53(2): 1549–1554.

Tran-Dinh, Q.; Pham, N. H.; Phan, D. T.; and Nguyen, L. M. 2019. Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization. *arXiv preprint arXiv:1905.05920*.

Wang, X.; Wang, C.; Li, X.; Leung, V. C.; and Taleb, T. 2020. Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet of Things Journal*, 7(10): 9441–9455.

Wei, C.-Y.; Lee, C.-W.; Zhang, M.; and Luo, H. 2021. Last-iterate Convergence of Decentralized Optimistic Gradient Descent/Ascent in Infinite-horizon Competitive Markov Games. *arXiv preprint arXiv:2102.04540*.

Xu, P.; Gao, F.; and Gu, Q. 2019. Sample efficient policy gradient methods with recursive variance reduction. *arXiv preprint arXiv:1909.08610*.

Zeng, S.; Anwar, A.; Doan, T.; Romberg, J.; and Raychowdhury, A. 2020. A decentralized policy gradient approach to multi-task reinforcement learning. *arXiv preprint arXiv:2006.04338*.

Zhang, J.; Ni, C.; Yu, Z.; Szepesvari, C.; and Wang, M. 2021a. On the Convergence and Sample Efficiency of Variance-Reduced Policy Gradient Method. *arXiv preprint arXiv:2102.08607*.

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

Zhang, K.; Yang, Z.; and Başar, T. 2019. Decentralized Multi-Agent Reinforcement Learning with Networked Agents: Recent Advances. *arXiv preprint arXiv:1912.03821*.

Zhang, K.; Yang, Z.; Liu, H.; Zhang, T.; and Basar, T. 2018. Fully decentralized multi-agent reinforcement learning with networked agents. In *International Conference on Machine Learning*, 5872–5881. PMLR.

Zhang, K.; Yang, Z.; Liu, H.; Zhang, T.; and Basar, T. 2021b. Finite-sample analysis for decentralized batch multi-agent reinforcement learning with networked agents. *IEEE Transactions on Automatic Control*.

Zhou, P.; Chen, X.; Liu, Z.; Braud, T.; Hui, P.; and Kangasharju, J. 2020. DRLE: Decentralized Reinforcement Learning at the Edge for Traffic Light Control. *arXiv preprint arXiv:2009.01502*.