Communication-Efficient Distributed Reinforcement Learning

Communication-Efficient Distributed Reinforcement Learning

Tianyi Chen[⋆] **Kaiqing Zhang**[†] **Georgios B. Giannakis**[⋆] **Tamer Başar**[†]

[⋆] *University of Minnesota - Twin Cities, Minneapolis, MN 55455, USA*
[†] *University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA*
{chen3827,georgios}@umn.edu {kzhang66,basar1}@illinois.edu

Abstract

This paper deals with *distributed reinforcement learning* (DRL), which involves a central controller and a group of learners. In particular, two DRL settings encountered in several applications are considered: *multi-agent reinforcement learning* (RL) and *parallel RL*, where frequent information exchanges between the learners and the controller are required. For many practical distributed systems, however, such as those involving parallel machines for training deep RL algorithms, and multi-robot systems for learning the optimal coordination strategies, the overhead caused by these frequent communication exchanges is considerable, and becomes the bottleneck of the overall performance. To address this challenge, a novel policy gradient method is developed here to cope with such communication-constrained DRL settings. The proposed approach reduces the communication overhead without degrading learning performance by adaptively skipping the policy gradient communication during iterations. It is established analytically that i) the novel algorithm has convergence rate identical to that of the plain-vanilla policy gradient for DRL; while ii) if the distributed computing units are heterogeneous in terms of their reward functions and initial state distributions, the number of communication rounds needed to achieve a desirable learning accuracy is markedly reduced. Numerical experiments on a popular multi-agent RL benchmark corroborate the significant communication reduction attained by the novel algorithm compared to alternatives.

Keywords: Communication efficiency, distributed learning, multi-agent, reinforcement learning.

1. Introduction

Reinforcement learning (RL) involves a sequential decision-making procedure, where a learner takes (possibly randomized) actions in a stochastic environment over a sequence of time steps, and aims to maximize the long-term cumulative rewards received from the interacting environment. Generally modeled as a Markov decision process (MDP) (Sutton and Barto, 2018), the sequential decision-making process has been tackled by various RL algorithms, including Q-learning (Watkins and Dayan, 1992), policy gradient (PG) (Sutton et al., 2000), and actor-critic methods (Konda and Tsitsiklis, 2000). While these popular RL algorithms were originally developed for the single-learner task, a number of practical RL tasks such as autonomous driving (Shalev-Shwartz et al., 2016), robotics (Stone and Veloso, 2000) and video games (Tampuu et al., 2015), involve *multiple learners* operating in a distributed fashion. In this paper, we consider the distributed reinforcement learning (DRL) problem that covers two general RL settings: *multi-agent collaborative* RL and *parallel* RL. The DRL settings we consider include a central controller that coordinates the learning processes of all learners. The learners can be agents in the multi-agent collaborative RL, or, workers in the parallel RL. In the former setting, multiple agents aim to maximize the team-averaged long-term reward via collaboration in a common environment (Kar et al., 2013; Wai et al., 2018; Zhang et al., 2018c); while in the latter, multiple parallel machines are used for solving large-scale MDPs with larger computational power and higher data efficiency (Nair et al., 2015b; Mnih et al., 2016). Similar learning paradigms have been investigated in distributed *supervised learning* (Recht et al., 2011; Li et al., 2014), e.g., Federated Learning (McMahan and Ramage, 2017; McMahan et al., 2017).

To coordinate the distributed learners, the central controller must exchange information with all learners, by collecting their rewards and local observations, or, broadcasting the policy to them. This type of information exchange requires frequent communication between the controller and the learners. However, in many DRL applications, including cloud-edge AI systems (Stoica et al., 2017), autonomous driving (Shalev-Shwartz et al., 2016), power distribution systems (Zhang et al., 2018a), and other applications in IoT (Chen et al., 2018a), the communication is costly and the latency caused by frequent communication becomes the bottleneck of the overall performance. These considerations motivate well the development of communication-efficient approaches for latency-sensitive DRL tasks. Although there has been a surging interest in studying communication-efficient approaches for supervised learning (Alistarh et al., 2017; Jordan et al., 2018; Chen et al., 2018b), no prior work has focused on the DRL setting. In this context, our goal is to develop a simple yet general algorithm for solving various DRL problems, with provable convergence guarantees and reduced communication overhead.

1.1 Our contributions

Targeting a communication-efficient solver for DRL, we propose a new PG method that we term Lazily Aggregated Policy Gradient (LAPG). With judiciously designed communication trigger rules, LAPG is shown capable of: i) achieving the same order of convergence rate (thus iteration complexity) as vanilla PG under standard conditions; and, ii) reducing the communication rounds required to achieve a desirable learning accuracy, when the distributed agents are heterogeneous (meaning reward functions and initial states are not homogeneous). In certain learning settings, we show that LAPG requires only $\mathcal{O}(1/M)$ communication of PG with M denoting the number of learners. Empirically, we evaluate the performance of LAPG using neural network-parameterized policies on the popular multi-agent RL benchmark for the cooperative navigation task, and corroborate that LAPG can considerably reduce the communication required by PG.

1.2 Related work

PG methods. PG methods have been recognized as one of the most pervasive RL algorithms (Sutton and Barto, 2018), especially for RL tasks with large and possibly continuous state-action spaces. By parameterizing the infinite-dimensional policy with finite-dimensional vectors (Sutton and Barto, 2018), PG methods reduce the search for the optimal policy over functional spaces to that over parameter spaces. Early PG methods include the well-known REINFORCE algorithm (Williams, 1992), as well as the variance-reduced G(PO)MDP algorithm (Baxter and Bartlett, 2001). Both REINFORCE and G(PO)MDP are Monte-Carlo sampling-type algorithms that estimate the policy gradient using the rollout trajectory data. To further reduce the variance, a policy gradient estimate that utilizes Q-function approximation was developed in (Sutton et al., 2000), based on a policy gradient theorem derived therein. Recently, several PG variants have made significant progress in accelerating convergence (Kakade, 2002), reducing variance (Papini et al., 2018), handling continuous action spaces (Silver et al., 2014), and ensuring policy improvement (Papini et al., 2017), by employing deep neural networks as function approximators (Schulman et al., 2015; Lillicrap et al., 2016; Schulman et al., 2017). However, all these algorithms were developed for the single-learner setting.

DRL. DRL has been investigated in the regimes of both multi-agent RL and parallel RL. The studies of multi-agent RL can be traced back to (Claus and Boutilier, 1998) and (Wolpert et al., 1999), with applications to network routing (Boyan and Littman, 1994) and power network control (Schneider et al., 1999). All these works, however, rather heuristically build on the direct modification of Q-learning from a single- to multi-agent settings, without performance guarantees. The first DRL algorithm with convergence guarantees has been reported in (Lauer and Riedmiller, 2000), although tailored for the tabular multi-agent MDP setting. More recently, Kar et al. (2013) developed a distributed Q-learning algorithm, termed *QD-learning*, over networked agents that can only communicate with their neighbors. In the same setup, fully decentralized actor-critic algorithms with function approximation were developed in (Zhang et al., 2018b,c) to handle large or even continuous state-action spaces. From an empirical viewpoint, a number of deep multi-agent collaborative RL algorithms has also been developed (Gupta et al., 2017; Lowe et al., 2017; Omidshafiei et al., 2017). On the other hand, parallel RL, which can efficiently tackle the single-learner yet large-scale RL problem by exploiting parallel computation, has also drawn increasing attention in recent years. In particular, Li and Schuurmans (2011) applied the Map Reduce framework to parallelize batch RL methods, while Nair et al. (2015b) introduced the first massively distributed framework for RL. In (Mnih et al., 2016), asynchronous RL algorithms have also been introduced to solve large-scale MDPs. This parallelism was shown to stabilize the training process, and also benefit data efficiency (Mnih et al., 2016). Nonetheless, none of these algorithms has dealt with communication-efficiency in DRL.

Communication-efficient learning. Improving communication efficiency in generic distributed learning settings has attracted much attention recently, especially for supervised learning (Jordan et al., 2018; McMahan and Ramage, 2017). With their undisputed performance granted, available communication-efficient methods do not directly apply to DRL, because they are either non-stochastic (Chen et al., 2018b; Zhang and Lin, 2015), or, they are tailored for convex problems (Stich, 2018). Algorithms for nonconvex problems are available e.g., (Alistarh et al., 2017), but they are designed to minimize the required bandwidth per communication, not the rounds. Compared to communication-efficient supervised learning in (Chen et al., 2018b), the novelty of LAPG here lies in the fact that the gradient used in DRL is stochastic and biased, which requires new algorithmic design and more involved analysis. Another unique feature of RL is that the distribution used to sample data is a function of the time-varying policy parameters, which introduces non-stationarity. Therefore, communication-efficient DRL is a challenging task, and so far it has been an uncharted territory.

Notation. Bold lowercase letters denote column vectors, which are transposed by $(\cdot)^\top$. And $\|\mathbf{x}\|$ denotes the ℓ_2-norm of $\mathbf{x}$. Inequalities for vectors $\mathbf{x} > \mathbf{0}$ will be defined entrywise. Symbol $\mathbb{E}$ denotes expectation, and $\mathbb{P}$ stands for probability. In addition, $\Delta(\mathcal{S})$ denotes the distribution over space $\mathcal{S}$.

2. Distributed Reinforcement Learning

In this section, we present the essential background on DRL and the plain-vanilla PG methods that can be applied to solve the DRL tasks.

2.1 Problem statement

Consider a central controller, and a group of M distributed learners, belonging to a set $\mathcal{M} := \{1, \ldots, M\}$. Depending on the specific DRL setting to be introduced shortly, a learner can be either an agent in multi-agent collaborative RL, or, a worker in the parallel RL setup. As in conventional RL, the DRL task can be cast under the umbrella of MDP, described by the following sextuple

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \rho, \{\ell_m\}_{m\in\mathcal{M}}) \tag{1}$$

where $\mathcal{S}$ and $\mathcal{A}$ are, respectively, the state space and the action space for all learners; $\mathcal{P}$ is the space of the state transition kernels defined as mappings $\mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$; $\gamma \in (0, 1)$ is the discounting factor; ρ is the initial state distribution; and $\ell_m : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the loss (or the negative reward) for learner m.

In addition to the sextuple (1), the so-termed policy is another important component of MDPs. We consider the stochastic policy $\boldsymbol{\pi} : \mathcal{S} \to \Delta(\mathcal{A})$ that specifies a conditional distribution of all possible joint actions given the current state $\mathbf{s}$, where the probability density of taking the joint action $\mathbf{a}$ at a state $\mathbf{s}$ is denoted by $\boldsymbol{\pi}(\mathbf{a}|\mathbf{s})$. The commonly used Gaussian policy (Deisenroth, 2010) is a function of the state-dependent mean $\boldsymbol{\mu}(\mathbf{s})$ and covariance matrix $\boldsymbol{\Sigma}(\mathbf{s})$, given by $\boldsymbol{\pi}(\cdot\,|\mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{s}), \boldsymbol{\Sigma}(\mathbf{s}))$. Considering discrete time $t \in \mathbb{N}$ in an infinite horizon, a policy $\boldsymbol{\pi}$ can generate a trajectory of state-action pairs $\mathcal{T} := \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots\}$ with $\mathbf{s}_t \in \mathcal{S}$ and $\mathbf{a}_t \in \mathcal{A}$. In distributed RL, the objective is to find the optimal policy $\boldsymbol{\pi}$ that minimizes the infinite-horizon discounted long-term loss aggregated over all learners, that is

$$\min_{\boldsymbol{\pi}} \sum_{m\in\mathcal{M}} \mathcal{L}_m(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{L}_m(\boldsymbol{\pi}) := \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\pi})}\left[\sum_{t=0}^{\infty}\gamma^t \ell_m(\mathbf{s}_t,\mathbf{a}_t)\right] \tag{2}$$

where $\ell_m(\mathbf{s}_t,\mathbf{a}_t)$ and $\mathcal{L}_m(\boldsymbol{\pi})$ are respectively, the loss given the state-action pair $(\mathbf{s}_t,\mathbf{a}_t)$ and the cumulative loss for learner m. The expectation in (2) is taken over the random trajectory $\mathcal{T}$. Given a policy $\boldsymbol{\pi}$, the probability of generating trajectory $\mathcal{T}$ is given by

$$\mathbb{P}(\mathcal{T}|\boldsymbol{\pi}) = \mathbb{P}(\mathbf{s}_0,\mathbf{a}_0,\mathbf{s}_1,\mathbf{a}_1,\mathbf{s}_2,\mathbf{a}_2,\cdots|\boldsymbol{\pi}) = \rho(\mathbf{s}_0)\prod_{t=0}^{\infty}\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t)\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t) \tag{3}$$

where $\rho(\mathbf{s}_0)$ is the probability of the initial state being $\mathbf{s}_0$, and $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$ is the transition probability from the current state $\mathbf{s}_t$ to the next state $\mathbf{s}_{t+1}$ by taking action $\mathbf{a}_t$. Clearly, the trajectory $\mathcal{T}$ is determined by both the underlying MDP and the policy $\boldsymbol{\pi}$.

Depending on how different learners are coupled with each other, the generic DRL formulation (2) includes the two popular RL settings, as highlighted next.

Multi-agent collaborative reinforcement learning. A number of important RL applications involve interaction between multiple heterogeneous but collaborative decision-makers (a.k.a. agents), such as those in controlling unmanned aerial vehicles, autonomous driving (Shalev-Shwartz et al., 2016), and many more emerging in the context of IoT (Chen et al., 2018a). In multi-agent collaborative RL, each agent m observes a global state $\mathbf{s}_t \in \mathcal{S}$ shared by all the agents, and takes an action $\mathbf{a}_{m,t} \in \mathcal{A}_m$ with the local action space denoted as $\mathcal{A}_m$. The local action of agent m is generated by a local policy $\boldsymbol{\pi}_m : \mathcal{S} \to \Delta(\mathcal{A}_m)$. While the local action spaces of different agents can be different, agents interact with a common environment that is influenced by the actions of all agents, where the joint action space can be written as $\mathcal{A} := \prod_{m\in\mathcal{M}}\mathcal{A}_m$. In other words, rather than any of the local actions $\mathbf{a}_{m,t}$, the joint action $(\mathbf{a}_{1,t},\ldots,\mathbf{a}_{M,t}) \in \mathcal{A}$ determines the transition probability to the next state $\mathbf{s}_{t+1}$ as well as the loss of each agent $\ell_m(\mathbf{s}_t,(\mathbf{a}_{1,t},\ldots,\mathbf{a}_{M,t}))$. As a consequence, the multi-agent collaborative RL problem can be viewed as an MDP characterized by the sextuple $\big(\mathcal{S},\prod_{m\in\mathcal{M}}\mathcal{A}_m,\mathcal{P},\gamma,\rho,\{\ell_m\}_{m\in\mathcal{M}}\big)$, which can be formulated in the following form

$$\min_{\boldsymbol{\pi}} \sum_{m\in\mathcal{M}} \mathcal{L}_m(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{L}_m(\boldsymbol{\pi}) := \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\pi})}\left[\sum_{t=0}^{\infty}\gamma^t \ell_m\big(\mathbf{s}_t,(\mathbf{a}_{1,t},\cdots,\mathbf{a}_{M,t})\big)\right] \tag{4}$$

where $\boldsymbol{\pi} := (\boldsymbol{\pi}_1\ldots\boldsymbol{\pi}_M)$ is a joint policy that concatenates all the local policies $\{\boldsymbol{\pi}_m\}_{m\in\mathcal{M}}$, and the expectation in $\mathcal{L}_m(\boldsymbol{\pi})$ is taken over all possible joint state-action trajectories, given by $\mathcal{T} := \{\mathbf{s}_0,(\mathbf{a}_{1,0},\ldots,\mathbf{a}_{M,0}),\mathbf{s}_1,(\mathbf{a}_{1,1},\ldots,\mathbf{a}_{M,1}),\mathbf{s}_2,(\mathbf{a}_{1,2},\ldots,\mathbf{a}_{M,2}),\ldots\}$. Replacing the action $\mathbf{a}_t$ in (2) by the joint action $(\mathbf{a}_{1,t},\cdots,\mathbf{a}_{M,t})$, the multi-agent collaborative RL problem can be viewed as an instance of DRL. Different from a single-agent MDP, the agents here are coupled by the state transition that depends on the joint action, and the local loss function that depends on the joint state.

Parallel reinforcement learning. Different from the multi-agent RL, the parallel RL is motivated by solving a large-scale single-agent RL task that needs to be run in parallel on multiple computing units (a.k.a. workers) (Nair et al., 2015a). The advantage of parallel RL is training time reduction and stabilization of the training processes (Mnih et al., 2016). Under such a setting, multiple workers typically aim to learn a common policy $\boldsymbol{\pi} : \mathcal{S} \to \Delta(\mathcal{A})$ for different instances of an *identical* MDP. By different instances of an identical MDP, we mean that each worker m aims to solve an independent MDP characterized by $(\mathcal{S}_m,\mathcal{A}_m,\mathcal{P}_m,\gamma,\rho_m,\ell_m)$. In particular, the local action and state spaces as well as the transition probabilities of the workers are the same; that is, $\mathcal{A}_m = \mathcal{A}$, $\mathcal{P}_m = \mathcal{P}$, and $\mathcal{S}_m = \mathcal{S}$, $\forall m \in \mathcal{M}$. However, the losses and the initial state distributions are different across workers, where the initial state distribution of worker m is ρ_m, and the loss of worker m is $\ell_m : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Nevertheless, they are quantities drawn from the same distributions, which satisfy $\mathbb{E}[\rho_m(\mathbf{s})] = \rho(\mathbf{s})$ and $\mathbb{E}[\ell_m(\mathbf{s}, \mathbf{a})] = \ell(\mathbf{s}, \mathbf{a})$ for any $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$. Therefore, the parallel RL can be written as follows

$$\min_{\boldsymbol{\pi}} \sum_{m \in \mathcal{M}} \mathcal{L}_m(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{L}_m(\boldsymbol{\pi}) := \mathbb{E}_{\mathcal{T}_m \sim \mathbb{P}(\cdot|\boldsymbol{\pi})} \left[\sum_{t=0}^{\infty} \gamma^t \ell_m(\mathbf{s}_{m,t}, \mathbf{a}_{m,t}) \right] \tag{5}$$

where $\mathbf{s}_{m,t} \in \mathcal{S}_m$, $\mathbf{a}_{m,t} \in \mathcal{A}_m$ are, respectively, the state and action of worker m, and $\boldsymbol{\pi}$ is the common policy to be learned. The expectation in $\mathcal{L}_m(\boldsymbol{\pi})$ is taken over all possible state-action trajectories of worker m, given by $\mathcal{T}_m := \{\mathbf{s}_{m,0}, \mathbf{a}_{m,0}, \mathbf{s}_{m,1}, \mathbf{a}_{m,1}, \mathbf{s}_{m,2}, \mathbf{a}_{m,2}, \ldots\}$. In contrast to the formulation of multi-agent RL in (4), the workers in parallel RL are not coupled by the joint state transition distributions or the loss functions, but they are rather intertwined by employing a common local policy.

2.2 Policy gradient methods

Policy gradient methods have been widely used in various RL problems with massive and possibly continuous state and action spaces. In those cases, tabular RL approaches are no longer tractable, and the intended solver typically involves function approximation. To overcome the inherent difficulty of learning a function, policy gradient methods restrict the search for the best performing policy over a class of parameterized policies. In particular, the policy $\boldsymbol{\pi}$ is usually parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$, which is denoted as $\boldsymbol{\pi}(\cdot|\mathbf{s}; \boldsymbol{\theta})$, or $\boldsymbol{\pi}(\boldsymbol{\theta})$ for simplicity. The commonly used Gaussian policy, for instance, can be parameterized as

$$\boldsymbol{\pi}(\,\cdot\,|\mathbf{s}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{s}; \boldsymbol{\theta}), \boldsymbol{\Sigma}) \tag{6}$$

where $\boldsymbol{\mu}(\mathbf{s}; \boldsymbol{\theta})$ is a general nonlinear mapping from $\mathcal{S}$ to $\mathcal{A}$ parameterized by $\boldsymbol{\theta}$. The mapping $\boldsymbol{\mu}(\mathbf{s}; \boldsymbol{\theta})$ can either be a deep neural network with the weight parameters $\boldsymbol{\theta}$, or, a linear function of $\boldsymbol{\theta}$ of the form $\boldsymbol{\mu}(\mathbf{s}; \boldsymbol{\theta}) = \boldsymbol{\Phi}(\mathbf{s})\boldsymbol{\theta}$, where $\boldsymbol{\Phi}(\mathbf{s})$ is the feature matrix corresponding to the state $\mathbf{s}$. Accordingly, the long-term discounted reward of a parametric policy per agent m is denoted by $\mathcal{L}_m(\boldsymbol{\theta}) := \mathcal{L}_m(\boldsymbol{\pi}(\boldsymbol{\theta}))$. Hence, the DRL problem (2) can be rewritten as the following parametric optimization problem

$$\min_{\boldsymbol{\theta}} \sum_{m \in \mathcal{M}} \mathcal{L}_m(\boldsymbol{\theta}) \quad \text{with} \quad \mathcal{L}_m(\boldsymbol{\theta}) := \mathbb{E}_{\mathcal{T} \sim \mathbb{P}(\,\cdot\,|\boldsymbol{\theta})} \left[\sum_{t=1}^{\infty} \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{7}$$

where the probability distribution of a trajectory $\mathcal{T}$ under the policy $\boldsymbol{\pi}(\boldsymbol{\theta})$ is denoted as $\mathbb{P}(\cdot|\boldsymbol{\theta})$. The search for an optimal policy can thus be performed by applying the gradient descent-type iterative methods to the parameterized optimization problem (7). By virtue of the *log-trick*, the gradient of each learner's cumulative loss $\mathcal{L}_m(\boldsymbol{\theta})$ in (7) can be written as (Baxter and Bartlett, 2001)

$$\textbf{Policy gradient} \qquad \nabla \mathcal{L}_m(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{T} \sim \mathbb{P}(\,\cdot\,|\boldsymbol{\theta})} \left[\sum_{t=0}^{\infty} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right]. \tag{8}$$

When the MDP model (1) is unknown, or, the expectation in (8) is computationally difficult to obtain, the stochastic estimate of the policy gradient (8) is often used, that is

$$\textbf{G(PO)MDP gradient} \qquad \hat{\nabla} \mathcal{L}_m(\boldsymbol{\theta}) = \sum_{t=0}^{\infty} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \tag{9}$$

which was first proposed in (Baxter and Bartlett, 2001), and is abbreviated as G(PO)MDP policy gradient. The G(PO)MDP policy gradient is an unbiased estimator of the policy gradient, while the latter incurs lower variance than other estimators, e.g., REINFORCE (Williams, 1992). In our ensuing algorithm design and performance analysis, we will leverage the G(PO)MDP gradient.

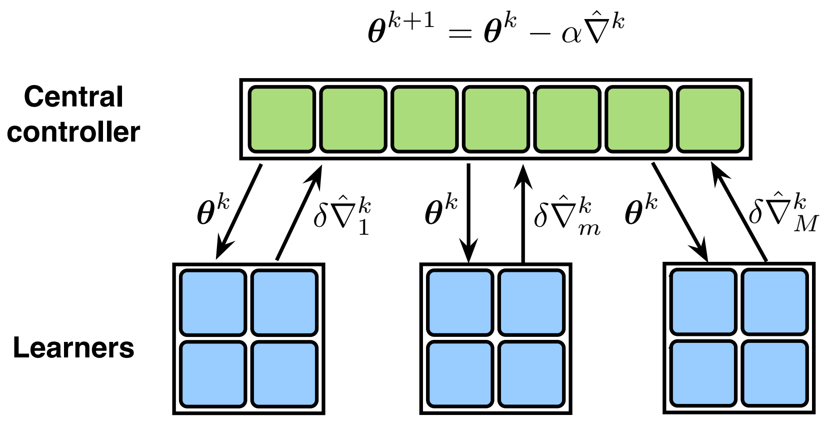

Figure 1: LAPG for communication-efficient distributed reinforcement learning.

Nonetheless, the variance of G(PO)MDP gradient can be high in general, and thus requires small stepsizes and sufficiently many iterations to guarantee convergence. For the plain-vanilla PG method in DRL, a number of needed iterations result in high communication overhead, since all learners' gradients need to be uploaded at each iteration in order to form the gradient for the collective objective in (2). This motivates the development of communication-efficient algorithms to be introduced next.

3. Communication-Efficient Policy Gradient Approach

Before introducing our approach, we first revisit the popular G(PO)MDP gradient method for solving (7) in the DRL setting: At iteration k, the central controller broadcasts the current policy parameter $\boldsymbol{\theta}^k$ to *all* learners; every learner $m \in \mathcal{M}$ computes an approximate policy gradient via

$$\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big) := \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau^{n,m}|\mathbf{s}_\tau^{n,m};\boldsymbol{\theta}^k)\right)\gamma^t\ell_m(\mathbf{s}_t^{n,m},\mathbf{a}_t^{n,m}) \tag{10}$$

where $\mathcal{T}_T^{n,m} := (\mathbf{s}_0^{n,m}, \mathbf{a}_0^{n,m}, \mathbf{s}_1^{n,m}, \mathbf{a}_1^{n,m}, \ldots, \mathbf{s}_T^{n,m}, \mathbf{a}_T^{n,m})$ is the nth T-slot trajectory (a.k.a. episode) generated at learner m; every learner m then uploads $\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big)$ to the central controller; and once receiving gradients from all learners, the controller updates the policy parameters via

PG iteration
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha\hat{\nabla}_{\rm PG}^k \quad \text{with} \quad \hat{\nabla}_{\rm PG}^k := \sum_{m\in\mathcal{M}}\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k) \tag{11}$$

where α is a stepsize, and $\hat{\nabla}_{\rm PG}^k$ is an aggregated policy gradient with each component received from each learner. The policy gradient in (10) is a mini-batch G(PO)MDP gradient computed by learner m using N batch trajectories $\{\mathcal{T}_T^{n,m}\}_{n=1}^N$ over T time slots. To implement the mini-batch PG update (11), however, the controller has to communicate with *all* learners to obtain fresh $\{\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big)\}$.

In this context, the present paper puts forth a new policy gradient-based method for DRL (as simple as PG) that can *skip* communication at certain rounds, which explains the name **L**azily **A**ggregated **Policy G**radient (**LAPG**). With derivations postponed until later, we introduce the LAPG iteration for the DRL problem (7) that resembles the PG update (11), given by

LAPG iteration
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha\hat{\nabla}^k \quad \text{with} \quad \hat{\nabla}^k := \sum_{m\in\mathcal{M}}\hat{\nabla}_{N,T}\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^k\big) \tag{12}$$

Algorithm 1 PG for DRL

```
1: Input: Stepsize $\alpha > 0$, $N$, and $T$.
2: Initialize: $\boldsymbol{\theta}^1$.
3: for $k = 1, 2, \ldots, K$ do
4:     Controller broadcasts $\boldsymbol{\theta}^k$ to all learners.
5:     for learner $m = 1, \ldots, M$ do
6:         Learner $m$ computes $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$.
7:         Learner $m$ uploads $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$.
8:     end for
9:     Controller updates via (11).
10: end for
```

Algorithm 2 LAPG for DRL

```
1: Input: Stepsize $\alpha > 0$, $\{\xi_d\}$, $N$ and $T$.
2: Initialize: $\boldsymbol{\theta}^1, \hat{\nabla}^0, \{\hat{\boldsymbol{\theta}}_m^0, \forall m\}$.
3: for $k = 1, 2, \ldots, K$ do
4:     Controller broadcasts the policy parameters.
5:     for learner $m = 1, \ldots, M$ do
6:         Learner $m$ computes $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$.
7:         if learner $m$ violates the condition then
8:             Learner $m$ uploads $\delta\hat{\nabla}_m^k$.
9:                 $\triangleright$ Save $\hat{\boldsymbol{\theta}}_m^k = \boldsymbol{\theta}^k$ at learner $m$
10:        else
11:            No actions at learner $m$.
12:        end if
13:    end for
14:    Controller updates the global policy.
15: end for
```

Table 1: A comparison of PG and LAPG for DRL.

where each policy gradient $\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)$ is either $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$, when $\hat{\boldsymbol{\theta}}_m^k = \boldsymbol{\theta}^k$, or an outdated policy gradient that has been computed using an old copy $\hat{\boldsymbol{\theta}}_m^k \neq \boldsymbol{\theta}^k$. Instead of requesting fresh batch policy gradients from every learner in (11), our idea here is to obtain $\hat{\nabla}^k$ by refining the previous aggregated gradient $\hat{\nabla}^{k-1}$; e.g., using only the new gradients from the learners in $\mathcal{M}^k$, while reusing the outdated gradients from the remaining learners. Therefore, with $\hat{\boldsymbol{\theta}}_m^k := \boldsymbol{\theta}^k$, $\forall m \in \mathcal{M}^k$, $\hat{\boldsymbol{\theta}}_m^k := \hat{\boldsymbol{\theta}}_m^{k-1}$, $\forall m \notin \mathcal{M}^k$, LAPG in (12) is equivalent to

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \sum_{m\in\mathcal{M}} \hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \alpha \sum_{m\in\mathcal{M}^k} \left(\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k) - \hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})\right) \tag{13a}$$

$$:= \boldsymbol{\theta}^k - \alpha\hat{\nabla}^{k-1} - \alpha \sum_{m\in\mathcal{M}^k} \delta\hat{\nabla}_m^k \tag{13b}$$

where $\delta\hat{\nabla}_m^k := \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k) - \hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})$ denotes the *innovation* between two evaluations of $\hat{\nabla}_{N,T}\mathcal{L}_m$ at the current policy parameter $\boldsymbol{\theta}^k$ and the old copy $\hat{\boldsymbol{\theta}}_m^{k-1}$. Note that the old copies for evaluating policy gradient at each learner can be different here, depending on the most recent iteration that each learner uploads its fresh batch policy gradient.

To this point, a myopic approach to minimizing per-iteration communication is to include as few learners in $\mathcal{M}^k$ as possible. However, it will turn out that such a simple selection will lead to many more iterations that may in turn increase the total number of needed communication rounds. A more principled way is to guide the communication selection according to learners' optimization progress. The first step of implementing such principle is to characterize the optimization progress as follows.

Lemma 1 (LAPG descent lemma) *Suppose $\mathcal{L}(\boldsymbol{\theta}) := \sum_{m\in\mathcal{M}} \mathcal{L}_m(\boldsymbol{\theta})$ is L-smooth, and $\boldsymbol{\theta}^{k+1}$ is generated by running one-step LAPG iteration (12) given $\boldsymbol{\theta}^k$. If the stepsize is selected such that $\alpha \leq 1/L$, then the objective values satisfy*

$$\mathcal{L}(\boldsymbol{\theta}^{k+1}) - \mathcal{L}(\boldsymbol{\theta}^k) \leq -\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3\alpha}{2}\left\|\sum_{m\in\mathcal{M}_c^k} \delta\hat{\nabla}_m^k\right\|^2 + \frac{3\alpha}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2$$
$$+\frac{3\alpha}{2}\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \tag{14}$$

where $\delta\hat{\nabla}_m^k$ *is defined in* (13)*, and* $\mathcal{M}_c^k$ *is the set of learners that* do not *upload at iteration* k*.*

Proof: A complete proof can be found in Appendix B. ∎

In Lemma 1, the first term on the right hand side of (14) drives the descent in the objective of DRL, while the error induced by skipping communication (the second term), the variance of stochastic policy gradient (the third term), as well as the finite-horizon gradient approximation error (the fourth term) increase the DRL objective thus impede the optimization progress. Intuitively, the error induced by skipping communication should be properly controlled so that it is small or even negligible relative to the magnitude of policy gradients that drives the optimization progress, and also the variance of policy gradients that originally appears in the PG-type algorithms (Papini et al., 2018).

To account for these error terms in the design of our algorithm, we first approximate $\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\|^2$ by $\sum_{d=1}^{D}\frac{\xi_d}{\alpha^2}\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2$, where $\{\xi_d\}_{d=1}^D$ are pre-selected weights, and then quantify the variance of using mini-batch policy gradient estimation through the following lemma.

Lemma 2 (PG concentration) *Under Assumptions 1 and 2, there exists a constant* V_m *depending on* $G,\gamma,\bar{\ell}_m$ *such that given* K *and* $\delta\in(0,1)$*, with probability at least* $1-\delta/K$*, for any* $\boldsymbol{\theta}$ *we have that*

$$\left\|\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta})-\nabla_T\mathcal{L}_m(\boldsymbol{\theta})\right\|^2\leq\frac{2\log(2K/\delta)V_m^2}{N}:=\sigma^2_{m,N,\delta/K}\tag{15}$$

where $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta})$ *and* $\nabla_T\mathcal{L}_m(\boldsymbol{\theta})$ *are the batch stochastic policy gradient* (10)*, and the full policy gradient for the* T*-slot truncated objective* (7)*, namely,* $\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\sum_{t=1}^{T}\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]$.

Proof: A complete proof can be found in Appendix C. ∎

Building upon Lemmas 1 and 2, we will include the learner m in $\mathcal{M}^k$ of (13) only if its current policy gradient has enough innovation relative to the most recently uploaded one; that is, it satisfies

LAPG condition
$$\left\|\delta\hat{\nabla}_m^k\right\|^2\geq\frac{1}{\alpha^2M^2}\sum_{d=1}^{D}\xi_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2+6\sigma^2_{m,N,\delta/K}\tag{16}$$

where $\{\xi_d\}_{d=1}^D$ are constant weights, and $\sigma^2_{m,N,\delta/K}$ is the variance of the policy gradient in (15). The values of $\{\xi_d\}$ and D are hyper-parameters and can be optimized case-by-case, while the variance $\sigma^2_{m,N,\delta/K}$ can be estimated on-the-fly in simulations. In a nutshell, a comparison of PG and LAPG for solving the DRL problem (7) is summarized in Table 1.

Regarding our proposed LAPG method, two remarks are in order.

LAPG implementation. By recursively updating the lagged gradients in (12) and the lagged condition in (16), implementing LAPG is as simple as PG. The only additional complexity comes from storing the most recently uploaded policy gradient $\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)$ and checking the LAPG communication condition (16). Despite its simplicity, we will further demonstrate that using lagged policy gradients in DRL can cut down a portion of unnecessary yet costly communication among learners.

Beyond LAPG. Compared with existing efforts for improving the performance of PG in single-agent RL settings such as the trust region PG (Schulman et al., 2015), the deterministic PG (Silver et al., 2014), and the variance-reduced PG (Papini et al., 2018), LAPG is not orthogonal to any of them. Instead, LAPG points out an alternate direction for improving communication efficiency of solving DRL, and can be combined with these methods to develop even more powerful DRL schemes. Extension to the actor-critic version of LAPG is also possible to accelerate and stablize the learning processes.

4. Main Results

In this section, we present the main theorems quantifying the performance of LAPG. Before that, we introduce several assumptions that serve as stepping stones for the subsequent analysis.

Assumption 1: *For each state-action pair* $(\mathbf{s}, \mathbf{a})$*, the loss* $\ell_m(\mathbf{s}, \mathbf{a})$ *is bounded as* $\ell_m(\mathbf{s}, \mathbf{a}) \in [0, \bar{\ell}_m]$*, and thus for each parameter* $\boldsymbol{\theta}$*, the per-learner cumulative loss is bounded as* $\mathcal{L}_m(\boldsymbol{\theta}) \in [0, \bar{\ell}_m/(1-\gamma)]$*.*

Assumption 2: *For each state-action pair* $(\mathbf{s}, \mathbf{a})$*, and any policy parameter* $\boldsymbol{\theta} \in \mathbb{R}^d$*, there exist constants* G *and* F *such that*

$$\|\nabla \log \boldsymbol{\pi}(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})\| \leq G \quad \text{and} \quad \left| \frac{\partial^2}{\partial \theta_i \partial \theta_j} \log \boldsymbol{\pi}(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta}) \right| \leq F \tag{17}$$

where θ_i and θ_j denote, respectively, the ith and jth entries of $\boldsymbol{\theta}$.

Assumption 1 requires boundedness of the instantaneous loss and thus the discounted cumulative loss, which is natural and commonly assumed in analyzing RL algorithms, e.g., (Papini et al., 2018; Zhang et al., 2018c; Baxter and Bartlett, 2001). Assumption 2 requires the score function and its partial derivatives to be bounded, which can be also satisfied by a wide range of stochastic policies, e.g., parameterized Gaussian policies (Papini et al., 2018). As we will see next, Assumptions 1 and 2 are sufficient to guarantee the smoothness of the objective function in (7).

Lemma 3 (smoothness in cumulative losses) *Under Assumptions 1 and 2, for any policy parameter* $\boldsymbol{\theta} \in \mathbb{R}^d$*, the accumulated loss* $\mathcal{L}_m(\boldsymbol{\theta})$ *for worker* m *is* L_m*-smooth, that is*

$$\|\nabla \mathcal{L}_m(\boldsymbol{\theta}_1) - \nabla \mathcal{L}_m(\boldsymbol{\theta}_2)\| \leq L_m \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| \quad \text{with} \quad L_m := \left(F + G^2 + \frac{2\gamma G^2}{1-\gamma}\right) \frac{\gamma \bar{\ell}_m}{(1-\gamma)^2} \tag{18}$$

where $\bar{\ell}_m$ *is the upper bound of the instantaneous loss in Assumption 1, and* F*,* G *are constants bounding the score function in* (17)*. Likewise, the overall accumulated loss* $\mathcal{L}(\boldsymbol{\theta})$ *is* L*-smooth, that is*

$$\|\nabla \mathcal{L}(\boldsymbol{\theta}_1) - \nabla \mathcal{L}(\boldsymbol{\theta}_2)\| \leq L \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| \quad \text{with} \quad L := \left(F + G^2 + \frac{2\gamma G^2}{1-\gamma}\right) \frac{\gamma \sum_{m \in \mathcal{M}} \bar{\ell}_m}{(1-\gamma)^2}. \tag{19}$$

Proof: A complete proof can be found in Appendix D. ■

The smoothness of the objective function is critical in the convergence analyses of many nonconvex optimization algorithms. Building upon Lemma 3, the subsequent analysis critically builds on the following Lyapunov function:

$$\mathbb{V}^k := \mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*) + \frac{3}{2\alpha} \sum_{d=1}^{D} \sum_{\tau=d}^{D} \xi_\tau \left\| \boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d} \right\|^2 \tag{20}$$

where $\boldsymbol{\theta}^*$ is the minimizer of (2), and $\alpha, \{\xi_\tau\}$ are constants that will be determined later.

For the DRL problem in (7), LAPG can guarantee the following convergence result.

Theorem 1 (iteration complexity) *Under Assumptions 1 and 2, if the stepsize* α *and the parameters* $\{\xi_d\}$ *in the LAPG condition* (16) *are chosen such that*

$$\alpha \leq \frac{\left(1 - 3\sum_{d=1}^{D} \xi_d\right)}{L} \tag{21}$$

and the constants T*,* K*, and* N *are chosen to satisfy*

$$T = \mathcal{O}(\log(1/\epsilon)), \quad K = \mathcal{O}(1/\epsilon), \quad \text{and} \quad N = \mathcal{O}(\log(K/\delta)/\epsilon) \tag{22}$$

then with probability at least $1-\delta$, *the iterates* $\{\boldsymbol{\theta}^k\}$ *generated by LAPG satisfy*

$$\frac{1}{K}\sum_{k=1}^{K}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \le \frac{2}{\alpha K}\mathbb{V}^1 + 3\sigma_T^2 + 21\sigma_{N,\delta/K}^2 \le \epsilon \tag{23}$$

where σ_T *and* $\sigma_{N,\delta/K}$ *are some constants depending on* $T, N, F, G, \gamma, \{\bar{\ell}_m\}$.

Proof: A complete proof can be found in Appendix E. ■

Theorem 1 asserts that even with the adaptive communication rules, LAPG can still achieve sublinear convergence to the stationary point of (7).

Regarding the communication complexity, it would be helpful to first estimate each learner's frequency of activating the communication condition (16). Ideally, we want those learners with a small reward (thus a small smoothness constant) to communicate with the controller less frequently. This intuition will be formally captured in the next lemma.

Lemma 4 (lazy gradient communication) *Under Assumptions 1 and 2, define the task hardness of every learner* m *as* $\mathbb{H}(m) := L_m^2/L^2$. *If the constants* $\{\xi_d\}$ *in the communication condition* (16) *are chosen to be* $\xi_D \le \ldots \le \xi_1$, *and the hardness of the learner* m *satisfies*

$$\mathbb{H}(m) \le \frac{\xi_d}{3d\alpha^2 L^2 M^2} := \gamma_d \tag{24}$$

then it uploads to the controller at most $1/(d+1)$ *fraction of time, with probability at least* $1-2\delta$.

Proof: A complete proof can be found in Appendix F. ■

Lemma 4 implies that the communication frequency of each learner is proportional to its task hardness. In addition, choosing larger trigger constants $\{\xi_d\}$ and a smaller stepsize α will reduce the communication frequencies of all learners. However, such choice of parameters will generally require many more iterations to achieve a desirable accuracy. To formally characterize the overall communication overhead of solving DRL, we define the communication complexity of solving the DRL problem (2) as the number of needed uploads to achieve ϵ-policy gradient error; e.g., $\min_{k=1,\cdots,K}\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\|^2 \le \epsilon$.

Building upon Theorem 1 and Lemma 4, the communication complexity is established next.

Theorem 2 (communication complexity) *Under Assumptions 1 and 2, define* $\Delta\mathbb{C}(h;\{\gamma_d\})$ *as*

$$\Delta\mathbb{C}(h;\{\gamma_d\}) := \sum_{d=1}^{D}\Big(\frac{1}{d}-\frac{1}{d+1}\Big)h\left(\gamma_d\right) \tag{25}$$

where h *is the cumulative density function of the learners' task hardness, given by*

$$h(\gamma) := \frac{1}{M}\sum_{m\in\mathcal{M}}\mathbb{1}(\mathbb{H}(m)\le\gamma). \tag{26}$$

With the communication complexity of LAPG and PG denoted as $\mathbb{C}_{\mathrm{LAPG}}(\epsilon)$ *and* $\mathbb{C}_{\mathrm{PG}}(\epsilon)$, *if the parameters are chosen as in* (22), *with probability at least* $1-4\delta$, *we have that*

$$\mathbb{C}_{\mathrm{LAPG}}(\epsilon) \le (1-\Delta\mathbb{C}(h;\{\gamma_d\}))\frac{\mathbb{C}_{\mathrm{PG}}(\epsilon)}{(1-3\sum_{d=1}^{D}\xi_d)}. \tag{27}$$

Choosing the parameters as in Theorem 1, and if for the heterogeneity function $h(\gamma)$ *there exists* γ' *such that* $\gamma' < \frac{h(\gamma')}{(D+1)DM^2}$, *then we have that* $\mathbb{C}_{\mathrm{LAPG}}(\epsilon) < \mathbb{C}_{\mathrm{PG}}(\epsilon)$.

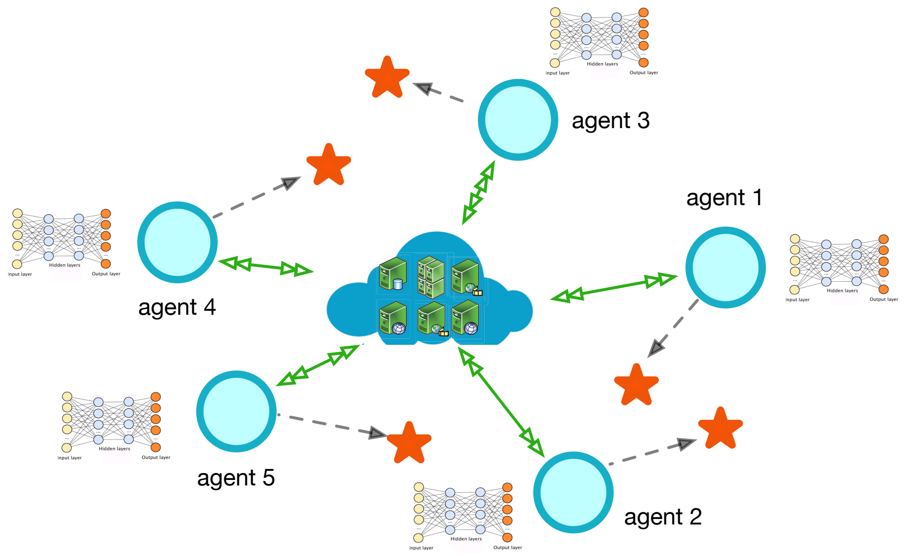

Figure 2: Multi-agent cooperative navigation task used in the simulation. Specifically, the blue circles represent the agents, the stars represent the landmarks, the green arrows represent the agent-cloud communication links, and the gray arrows direct the target landmark each agent aims to cover.

Proof: A complete proof can be found in Appendix G. ■

By carefully designing our communication selection rule, Theorem 2 demonstrates that the overall communication of LAPG is less than that of PG, provided that the reward functions (thus the smoothness constants) of each learner are very heterogeneous. Consider the extreme case where $L_m = \mathcal{O}(1), \forall 1, \ldots, M-1$, and $L_M = L = \mathcal{O}(M^2)$. One can easily verify that using proper parameters, LAPG only requires $\mathcal{O}(1/M)$ number of communication rounds of plain-vanilla PG.

While the improved communication complexity in Theorem 2 builds on slightly restrictive dependence on the problem parameters, the LAPG's empirical performance gain over PG goes far beyond the worst-case theoretical analysis presented. As the subsequent numerical tests will confirm, LAPG remains operational in a broader DRL setting where $h(\cdot)$ may not satisfy the condition in Theorem 2.

5. Numerical Tests

To validate the theoretical results, this section reports the empirical performance of LAPG in the multi-agent RL task, as an example of DRL. All experiments were performed using Python 3.6 on an Intel i7 CPU @ 3.4 GHz (32 GB RAM) desktop. Throughout this section, we consider the simulation environment of the *Cooperative Navigation* task in (Lowe et al., 2017), which builds on the popular OpenAI Gym paradigm (Brockman et al., 2016). In this RL environment, M agents aim to reach a set of M landmarks through physical movement, which is controlled by a set of five actions {*stay, left, right, up, down*}. Agents are connected to a remote central coordinator, and are rewarded based on the proximity of their position to the one-to-one associated landmark; see the depiction in Figure 2.

In the simulation, we modify the environment in (Lowe et al., 2017) as follows: i) we assume the state is globally observable, i.e., the position and velocity of other agents in a two-dimensional grid are observable to each agent; and, ii) each agent has a certain target landmark to cover, and the individual reward is determined by the proximity to that certain landmark, as well as the penalty of collision with other agents. In this way, the reward function varies among agents, and the individual reward of an agent also depends on the other agents' movement, which is consistent with the multi-agent

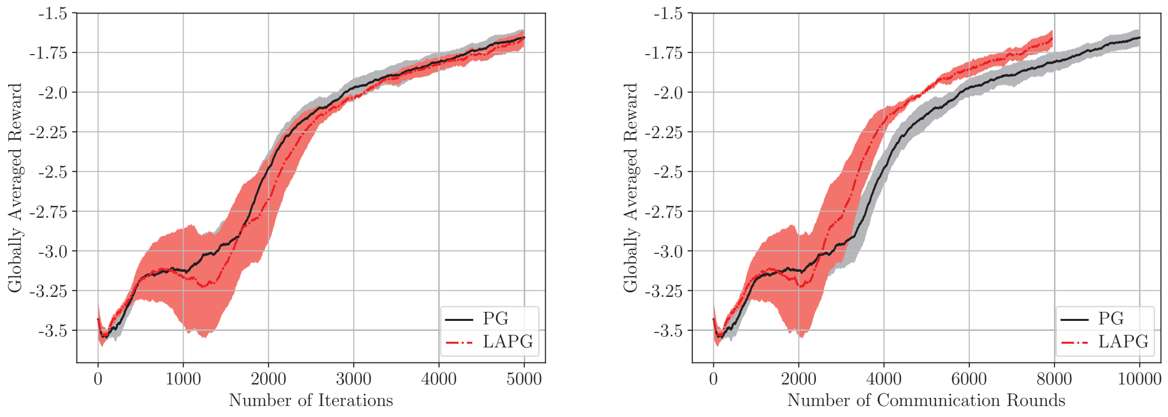

Figure 3: Iteration and communication complexity in a heterogeneous environment (Non momentum). The shaded region in all the figures represents the globally averaged reward distribution of each scheme within the half standard deviation of the mean.

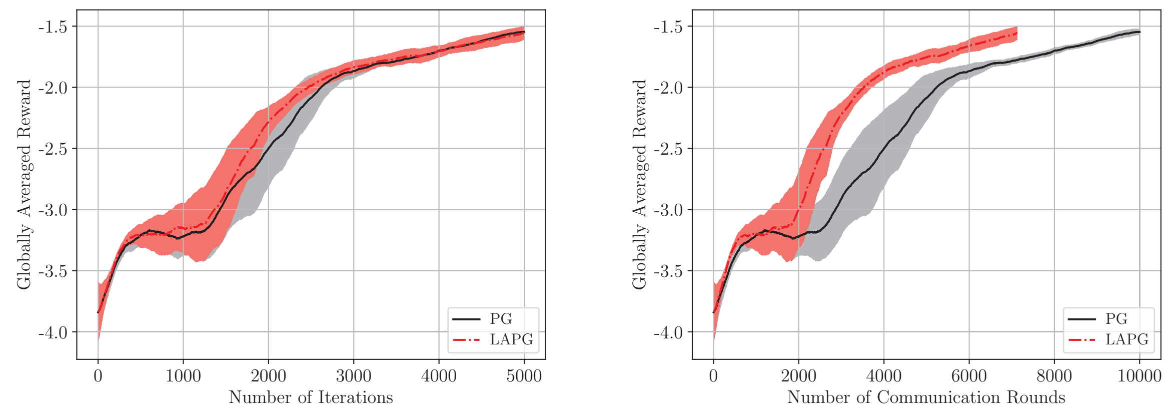

Figure 4: Iteration and communication complexity in a heterogeneous environment (Momentum).

RL formulation (4). The reward is further scaled by different positive coefficients, representing the heterogeneity (e.g., different priority) of different agents. The collaborative goal of the agents is to maximize the network averaged long-term reward so as to reduce distances to the landmark and avoid collisions. We implement LAPG using G(PO)MDP gradient estimators, and compare it with the G(PO)MDP-based PG method. The discounting factor in the cumulative loss is $\gamma = 0.99$ in all the tests. For each episode, both algorithms terminate after $T = 20$ iterations.

In the first test with $M = 2$ agents, the targeted local policy of each agent $\boldsymbol{\pi}_m(\boldsymbol{\theta}_m)$ is parameterized by a three-layer neural network, where the first and the second hidden layers contain 30 and 10 neural units with ReLU as the activation function, and the output layer is the softmax operator. We run in total $N = 10$ batch episodes in each Monte Carlo run, and report the globally averaged reward from 10 Monte Carlo runs. LAPG and PG are first implemented using gradient descent update for the heterogeneous (scaled reward) case. As shown in Figure 3, LAPG converges within the same number of iterations as PG, and the communication reduction is observable. To accelerate the training of neural networks used in policy parameterization, both LAPG and PG are implemented using heavy-ball based

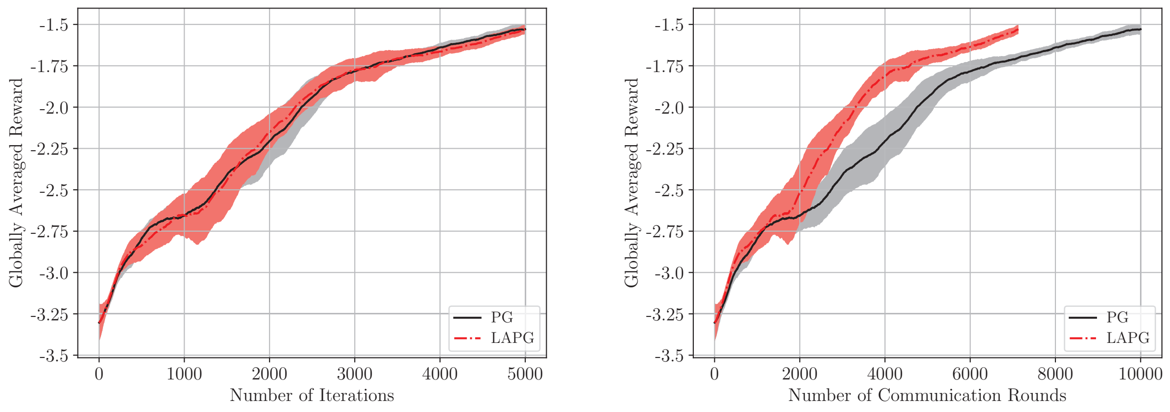

Figure 5: Iteration and communication complexity in a homogeneous environment (Momentum).

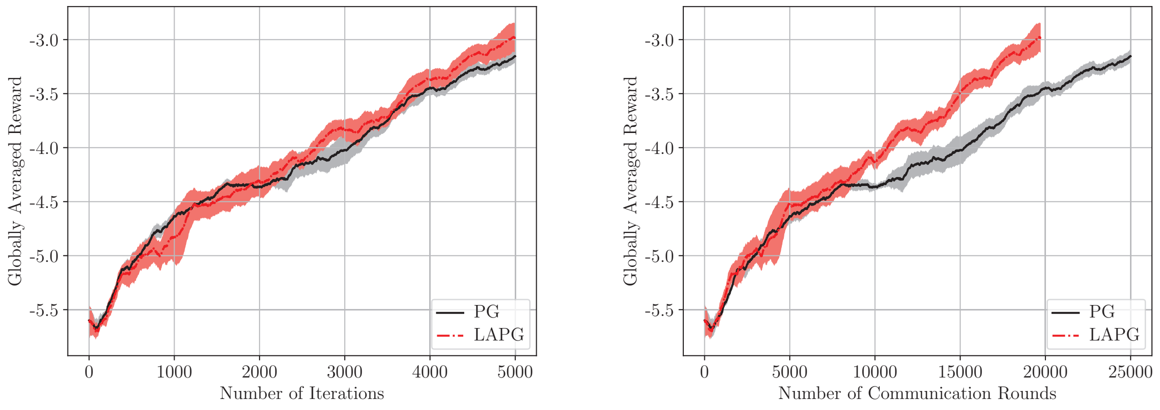

Figure 6: Iteration and communication complexity in a five-agent environment (RELU activation).

momentum update thereafter, where the stepsize and the momentum factor are set as 0.01 and 0.6, respectively. The corresponding performance is reported in Figure 4 for the heterogeneous case and in Figure 5 for the homogeneous (non-scaled reward) case. Clearly, in both Figures 4 and 5, our LAPG converges within the same number of iterations as the PG algorithm. When it comes to the number of communication rounds, LAPG requires significantly smaller amount than PG in both homogeneous and heterogeneous cases. With momentum update, the performance gain of LAPG is larger than that without momentum update, which is partially due to that both algorithms converge faster in this case.

In the second test with $M = 5$, the targeted policy is again parameterized by a three-layer neural network. For this larger multi-agent RL task, we use a larger network to characterize the optimal policy, where the first and the second hidden layers contain 50 and 20 neural units. To reduce the runtime, we only run in total $N = 8$ batch episodes in each Monte Carlo run, and report the globally averaged reward from 5 Monte Carlo runs in Figure 6 for the heterogeneous case. It is shown in Figure 6 that LAPG successfully converges using the same number of iterations as PG, but it requires fewer number of communication rounds than PG. The performance gain is sizable in terms of communication.

Since RELU-based activation functions may introduce certain nonsmoothness in the resultant state-to-action distribution mapping, the performance is also evaluated using the softplus activation,

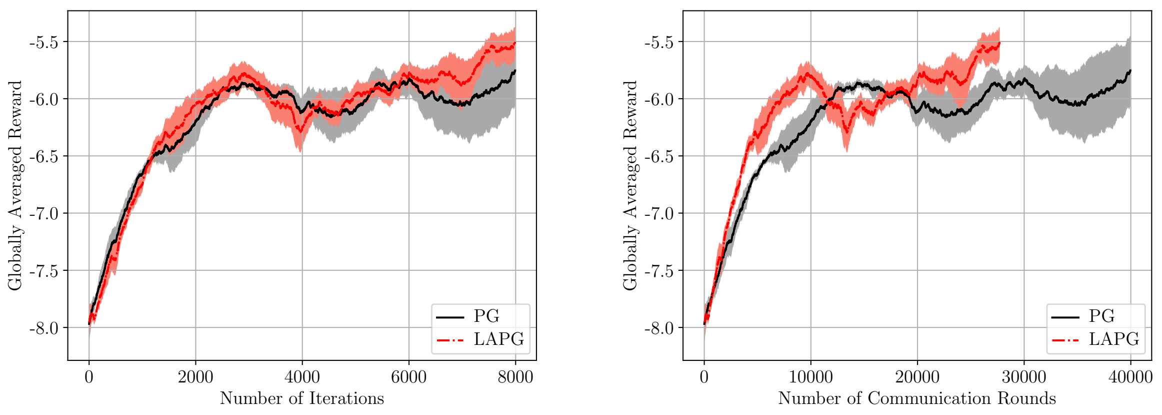

Figure 7: Iteration and communication complexity in a five-agent environment (Softplus activation).

which is a smooth approximation of the RELU activation function. Clearly, the comparison in Figure 7 confirms that LAPG still converges within much fewer number of communication rounds than PG with the smooth activation functions. These observations shed the light on the potential applicability of our LAPG algorithm to large-scale DRL problems, when the communication cost of exchanging policy gradients is high especially using the over-parameterized neural networks as policy approximators.

6. Concluding Remarks

This paper studies the distributed reinforcement learning (DRL) problem involving a central controller and a group of heterogeneous learners, which includes the popular *multi-agent collaborative* RL and the *parallel* RL settings. Targeting DRL applications in communication-constrained environments, our goal was to learn a DRL policy minimizing the loss aggregated over all learners, using as few communication rounds as possible. We developed a promising communication-cognizant method for DRL that we term Lazily Aggregated Policy Gradient (LAPG) approach. LAPG can achieve the same convergence rates as PG, and requires fewer communication rounds given that the learners in DRL are heterogeneous. Promising empirical performance on the multi-agent cooperative navigation task corroborated our theoretical findings.

While LAPG enjoys reduced communication overhead, it requires sufficiently many mini-batch trajectories (high sample complexity) to obtain low-variance policy gradients. Along with other limitations of LAPG, our future work will aim at i) reducing the sample complexity by properly reusing outdated trajectories; and, ii) incorporating smoothing techniques to handle nonsmooth loss functions induced by the nonsmooth policy parameterization.

Appendix A. Preliminary assumptions and lemmas

In this section, we introduce several supporting lemmas that will lead to the subsequent convergence and communication complexity analysis of LAPG.

Define the finite-horizon approximation of the policy gradient (8) as

$$\nabla_T \mathcal{L}_m(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\mathcal{T}|\boldsymbol{\theta})}\left[\sum_{t=0}^{T}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta})\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right] \tag{28}$$

and its single-trajectory stochastic estimate as

$$\hat{\nabla}_T \mathcal{L}_m(\boldsymbol{\theta}) = \sum_{t=0}^{T} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau | \mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t). \tag{29}$$

We have the following lemma that bounds the discrepancy between them.

Lemma 5 (bounded PG deviation) *For the finite-horizon approximation of the policy gradient* (28) *and its corresponding version* (29)*, at any $\boldsymbol{\theta}$ and any learner m, their discrepancy is bounded by*

$$\left\| \hat{\nabla}_T \mathcal{L}_m(\boldsymbol{\theta}) - \nabla_T \mathcal{L}_m(\boldsymbol{\theta}) \right\| \leq V_m \tag{30}$$

where V_m is a constant depending on $G, \gamma, \bar{\ell}_m$.

Proof: Using the definition of the G(PO)MDP gradient, we have that

$$\begin{aligned}
&\left\| \hat{\nabla}_T \mathcal{L}_m(\boldsymbol{\theta}) - \nabla_T \mathcal{L}_m(\boldsymbol{\theta}) \right\| \\
=&\left\| \sum_{t=0}^{T} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau | \mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) - \mathbb{E}_{\mathcal{T} \sim \mathbb{P}(\cdot|\boldsymbol{\theta})} \left[\sum_{t=0}^{T} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau | \mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right] \right\| \\
\leq& 2 \sup_{\mathcal{T} \sim \mathbb{P}(\cdot|\boldsymbol{\theta})} \sum_{t=0}^{T} \left\| \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau | \mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right\| \\
\overset{(a)}{\leq}& 2 \sum_{t=0}^{T} t G \gamma^t \bar{\ell}_m \leq 2 G \bar{\ell}_m \sum_{t=0}^{\infty} t \gamma^t \\
=& \frac{2 G \bar{\ell}_m \gamma}{(1-\gamma)^2} := V_m
\end{aligned} \tag{31}$$

where (a) follows from the upper bounds in Assumptions 1 and 2, and V_m is the uniform upper bound of the G(PO)MDP stochastic policy gradient. ■

Lemma 6 (finite horizon approximation) *For the infinite-horizon problem* (2) *and its finite-horizon approximation, for any $\boldsymbol{\theta}$, the corresponding policy gradients are bounded by*

$$\| \nabla \mathcal{L}(\boldsymbol{\theta}) - \nabla_T \mathcal{L}(\boldsymbol{\theta}) \| \leq \sum_{m \in \mathcal{M}} G \bar{\ell}_m \left(T + \frac{\gamma}{1-\gamma} \right) \gamma^T := \sigma_T. \tag{32}$$

Proof: For any $\boldsymbol{\theta} \in \mathbb{R}^d$, it follows that

$$\begin{aligned}
\| \nabla \mathcal{L}(\boldsymbol{\theta}) - \nabla_T \mathcal{L}(\boldsymbol{\theta}) \| &= \left\| \mathbb{E}_{\mathcal{T} \sim \mathbb{P}(\cdot|\boldsymbol{\theta})} \left[\sum_{m \in \mathcal{M}} \sum_{t=T}^{\infty} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau | \mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right] \right\| \\
&\overset{(a)}{\leq} \mathbb{E}_{\mathcal{T} \sim \mathbb{P}(\cdot|\boldsymbol{\theta})} \left[\left\| \sum_{m \in \mathcal{M}} \sum_{t=T}^{\infty} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau | \mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right\| \right] \\
&\overset{(b)}{\leq} \mathbb{E}_{\mathcal{T} \sim \mathbb{P}(\cdot|\boldsymbol{\theta})} \left[\sum_{m \in \mathcal{M}} \sum_{t=T}^{\infty} \left\| \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau | \mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right\| \right] \\
&\overset{(c)}{\leq} \mathbb{E}_{\mathcal{T} \sim \mathbb{P}(\cdot|\boldsymbol{\theta})} \left[\sum_{m \in \mathcal{M}} \sum_{t=T}^{\infty} t G \gamma^t \bar{\ell}_m \right] = \mathbb{E}_{\mathcal{T} \sim \mathbb{P}(\cdot|\boldsymbol{\theta})} \left[\sum_{m \in \mathcal{M}} G \bar{\ell}_m \sum_{t=T}^{\infty} t \gamma^t \right]
\end{aligned} \tag{33}$$

where (a) uses the Jensen's inequality, (b) follows from the triangular inequality, and (c) uses the bounds on the loss and the score functions in Assumptions 1 and 2. We can calculate the summation as

$$\sum_{t=T}^{\infty} t\gamma^t = \left(\frac{T}{1-\gamma} + \frac{\gamma}{(1-\gamma)^2}\right)\gamma^T. \tag{34}$$

Plugging (34) into (33) leads to

$$\|\nabla\mathcal{L}(\boldsymbol{\theta}) - \nabla_T\mathcal{L}(\boldsymbol{\theta})\| \le \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\sum_{m\in\mathcal{M}} G\bar{\ell}_m \sum_{t=T}^{\infty} t\gamma^t\right] = \sum_{m\in\mathcal{M}} G\bar{\ell}_m\left(T + \frac{\gamma}{1-\gamma}\right)\frac{\gamma^T}{1-\gamma} \tag{35}$$

from which the proof is complete. ■

Appendix B. Proof of Lemma 1

Using the smoothness of $\mathcal{L}_m$, and thus of $\mathcal{L}$ in Lemma 3, we have that

$$\mathcal{L}(\boldsymbol{\theta}^{k+1}) - \mathcal{L}(\boldsymbol{\theta}^k) \le \left\langle \nabla\mathcal{L}(\boldsymbol{\theta}^k), \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\rangle + \frac{L}{2}\left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2. \tag{36}$$

Note that (13) can be also written as (cf. $\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) := \sum_{m\in\mathcal{M}}\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$)

$$\begin{aligned}\boldsymbol{\theta}^{k+1} &= \boldsymbol{\theta}^k - \alpha\sum_{m\in\mathcal{M}}\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k) - \alpha\sum_{m\in\mathcal{M}_c^k}\left(\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)\right) &(37)\\ &= \boldsymbol{\theta}^k - \alpha\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) + \alpha\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k &(38)\end{aligned}$$

where $\mathcal{M}_c^k$ is the set of agents that *do not* communicate with the controller at iteration k.

Plugging (37) into $\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k), \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\rangle$ leads to (cf. $\hat{\boldsymbol{\theta}}_m^k = \hat{\boldsymbol{\theta}}_m^{k-1}$, $\forall m\in\mathcal{M}_c^k$)

$$\begin{aligned}\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k), \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\rangle &= -\alpha\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k), \hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\rangle\\ &= -\alpha\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k), \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) + \hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\rangle\\ &= -\alpha\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 - \alpha\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k), \hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\rangle. &(39)\end{aligned}$$

Using $2\mathbf{a}^\top\mathbf{b} = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - \|\mathbf{a}-\mathbf{b}\|^2$, we can re-write the inner product in (39) as

$$\begin{aligned}&\left\langle -\nabla\mathcal{L}(\boldsymbol{\theta}^k), \hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\rangle\\ =&\frac{1}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{1}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2 - \frac{1}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2\\ \overset{(a)}{=}&\frac{1}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{1}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2 - \frac{1}{2\alpha^2}\left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2 &(40)\end{aligned}$$

where (a) follows from the LAPG update (37).

Define the policy gradient for the finite-horizon discounted reward as

$$\nabla_T \mathcal{L}(\boldsymbol{\theta}) := \sum_{m\in\mathcal{M}} \nabla_T \mathcal{L}_m(\boldsymbol{\theta}) \;\text{ with }\; \nabla_T \mathcal{L}_m(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\nabla \log \mathbb{P}(\mathcal{T}|\boldsymbol{\theta})\left(\sum_{t=0}^{T}\gamma^t \ell_m(\mathbf{s}_t,\mathbf{a}_t)\right)\right] \tag{41}$$

and decompose the second term in (40) as

$$\begin{aligned}
&\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2\\
=&\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) + \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2\\
\overset{(b)}{=}&3\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2
\end{aligned} \tag{42}$$

where (b) follows from the inequality $\|\mathbf{a}+\mathbf{b}+\mathbf{c}\|^2 \le 3\|\mathbf{a}\|^2 + 3\|\mathbf{b}\|^2 + 3\|\mathbf{c}\|^2$. Combining (39), (40) and (42), and plugging into (36), the claim of Lemma 1 follows.

Appendix C. Proof of Lemma 2

The policy gradient concentration result in Lemma 2 builds on the following concentration inequality.

Lemma 7 (concentration inequality (Pinelis, 1994)) *If $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N \in \mathbb{R}^d$ denote a vector-valued martingale difference sequence satisfying $\mathbb{E}[\mathbf{X}_n|\mathbf{X}_1,\cdots,\mathbf{X}_{n-1}] = \mathbf{0}$, and $\|\mathbf{X}_n\| \le V$, $\forall n$, then for any scalar $\delta \in (0,1]$, we have*

$$\mathbb{P}\left(\left\|\sum_{n=1}^{N}\mathbf{X}_n\right\|^2 > 2\log(2/\delta)V^2N\right) \le \delta. \tag{43}$$

Therefore, viewing $\mathbf{X}_n := \hat{\nabla}_T\mathcal{L}_m(\boldsymbol{\theta}) - \nabla_T\mathcal{L}_m(\boldsymbol{\theta})$, and using the bounded PG deviation in Lemma 5, we can readily arrive at Lemma 2.

Appendix D. Proof of Lemma 3

For any $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$, it follows that

$$\begin{aligned}
\|\nabla\mathcal{L}_m(\boldsymbol{\theta}_1) - \nabla\mathcal{L}_m(\boldsymbol{\theta}_2)\| = \Bigg\|&\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_1)}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]\\
&-\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_2)}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]\\
&+\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_2)}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]\\
&-\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_2)}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_2)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]\Bigg\|.
\end{aligned} \tag{44}$$

We can bound the first difference term in (44) as (cf. use $\mathcal{T} \sim \boldsymbol{\theta}_1$ for $\mathcal{T} \sim \mathbb{P}(\cdot|\boldsymbol{\theta}_1)$)

$$\begin{aligned}
&\Bigg\|\mathbb{E}_{\mathcal{T}\sim\boldsymbol{\theta}_1}\Bigg[\sum_{t=0}^{\infty}\Bigg(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\Bigg)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Bigg]-\mathbb{E}_{\mathcal{T}\sim\boldsymbol{\theta}_2}\Bigg[\sum_{t=0}^{\infty}\Bigg(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\Bigg)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Bigg]\Bigg\|\\
=&\Bigg\|\int\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_1)\sum_{t=0}^{\infty}\Bigg(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\Bigg)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)-\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_2)\sum_{t=0}^{\infty}\Bigg(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\Bigg)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\mathbf{d}\mathcal{T}\Bigg\|\\
=&\Bigg\|\sum_{t=0}^{\infty}\int\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)\Bigg(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\Bigg)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\Bigg(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\Bigg)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\mathbf{d}\mathcal{T}_t\Bigg\|\\
\leq&\sum_{t=0}^{\infty}\int\Bigg\|\big(\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\big)\Bigg(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\Bigg)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Bigg\|\mathbf{d}\mathcal{T}_t\\
\leq&\sum_{t=0}^{\infty}\int\Big|\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\Big|\Bigg\|\Bigg(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\Bigg)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Bigg\|\mathbf{d}\mathcal{T}_t \qquad (45)
\end{aligned}$$

where we use $\mathcal{T}_t$ for a t-slot trajectory $\{\mathbf{s}_0,\mathbf{a}_0,\mathbf{s}_1,\mathbf{a}_1,\cdots,\mathbf{s}_{t-1},\mathbf{a}_{t-1}\}$.

For the remaining difference term in (45), we can bound it as

$$\begin{aligned}
\Big|\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\Big|=&\Bigg|\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_1)\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)-\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_2)\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\Bigg|\\
=&\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\Bigg|\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_1)-\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_2)\Bigg|\\
=&\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\Bigg|\prod_{t=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_1)-\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_2)\Bigg|\\
=&\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\Bigg|(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)^\top\nabla\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\Bigg|. \qquad (46)
\end{aligned}$$

Note that we have

$$\begin{aligned}
\Bigg|(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)^\top\nabla\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\Bigg| =& \Bigg|(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)^\top\nabla\log\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\Bigg|\\
=&\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\Bigg|(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)^\top\nabla\log\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\Bigg|\\
\leq&\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\Bigg\|\sum_{\nu=0}^{t-1}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\Bigg\|\Big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\Big\|\\
\leq& tG\Big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\Big\|\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}}) \qquad (47)
\end{aligned}$$

and if plugging (46) and (47) into (45), it follows that

$$\begin{aligned}
&\sum_{t=0}^{\infty}\int\Big|\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\Big|\Big\|\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Big\|\mathbf{d}\mathcal{T}_t\\
\leq&\sum_{t=0}^{\infty}\int\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})tG\Big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\Big\|\Big\|\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Big\|\mathbf{d}\mathcal{T}_t\\
\leq&\sum_{t=0}^{\infty}\int\mathbb{P}(\mathcal{T}_t|\tilde{\boldsymbol{\theta}})tG\Big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\Big\|tG\gamma^t\bar{\ell}_m\mathbf{d}\mathcal{T}_t=\sum_{t=0}^{\infty}t^2G^2\gamma^t\bar{\ell}_m\Big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\Big\|\\
=&\left(\frac{\gamma}{(1-\gamma)^2}+\frac{2\gamma^2}{(1-\gamma)^3}\right)G^2\bar{\ell}_m\Big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\Big\| \qquad (48)
\end{aligned}$$

where we use the equation that $\sum_{t=0}^{\infty}t^2\gamma^t=\frac{\gamma}{(1-\gamma)^2}+\frac{2\gamma^2}{(1-\gamma)^3}$.

We can separately bound the second difference term in (44) as

$$\begin{aligned}
&\Bigg\|\mathbb{E}_{\mathcal{T}\sim\boldsymbol{\theta}_2}\Bigg[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Bigg]-\mathbb{E}_{\mathcal{T}\sim\boldsymbol{\theta}_2}\Bigg[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_2)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Bigg]\Bigg\|\\
\leq&\int\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_2)\Bigg\|\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)-\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_2)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\Bigg\|\mathbf{d}\mathcal{T}\\
\leq&\int\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_2)\sum_{t=0}^{\infty}\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\sum_{\tau=0}^{t}\Big\|\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_1)-\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_2)\Big\|\mathbf{d}\mathcal{T}\\
\leq&\int\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_2)\sum_{t=0}^{\infty}\gamma^t\bar{\ell}_m\sum_{\tau=0}^{t}F\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\|\mathbf{d}\mathcal{T}\leq\sum_{t=0}^{\infty}\gamma^t\bar{\ell}_mtF\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\|=\frac{F\bar{\ell}_m\gamma}{(1-\gamma)^2}\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\|. \qquad (49)
\end{aligned}$$

Combining (48) and (49), we have that

$$\begin{aligned}
\|\nabla\mathcal{L}_m(\boldsymbol{\theta}_1)-\nabla\mathcal{L}_m(\boldsymbol{\theta}_2)\|\leq&\left(\frac{F}{(1-\gamma)^2}+\left(\frac{1}{(1-\gamma)^2}+\frac{2\gamma}{(1-\gamma)^3}\right)G^2\right)\gamma\bar{\ell}_m\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\|\\
:=&L_m\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\|. \qquad (50)
\end{aligned}$$

Similarly, we can bound the Lipschitz constant of $\nabla\mathcal{L}(\boldsymbol{\theta})$, $\nabla_T\mathcal{L}(\boldsymbol{\theta})$, $\nabla_T\mathcal{L}_m(\boldsymbol{\theta})$, and the proof is complete.

Appendix E. Proof of Theorem 1

Using the definition of $\mathbb{V}^k$ in (20), it follows that (with the short-hand notation $\beta_d:=\frac{3}{2\alpha}\sum_{\tau=d}^{D}\xi_\tau$)

$$\begin{aligned}
\mathbb{V}^{k+1}-\mathbb{V}^k=&\mathcal{L}(\boldsymbol{\theta}^{k+1})-\mathcal{L}(\boldsymbol{\theta}^k)+\sum_{d=1}^{D}\beta_d\Big\|\boldsymbol{\theta}^{k+2-d}-\boldsymbol{\theta}^{k+1-d}\Big\|^2-\sum_{d=1}^{D}\beta_d\Big\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\Big\|^2\\
\overset{(a)}{\leq}&-\frac{\alpha}{2}\Big\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\Big\|^2+\frac{3\alpha}{2}\Bigg\|\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\Bigg\|^2+\sum_{d=2}^{D}\beta_d\Big\|\boldsymbol{\theta}^{k+2-d}-\boldsymbol{\theta}^{k+1-d}\Big\|^2+\frac{3\alpha}{2}\Big\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k)-\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\Big\|^2\\
&+\left(\frac{L}{2}-\frac{1}{2\alpha}+\beta_1\right)\Big\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^k\Big\|^2-\sum_{d=1}^{D}\beta_d\Big\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\Big\|^2+\frac{3\alpha}{2}\Big\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)-\nabla\mathcal{L}(\boldsymbol{\theta}^k)\Big\|^2 \qquad (51)
\end{aligned}$$

where (a) uses (14) in Lemma 1.

Using $(\sum_{n=1}^{N} a_n)^2 \leq N \sum_{n=1}^{N} a_n^2$, it follows that

$$\Big\| \sum_{m\in\mathcal{M}_c^k} \delta\hat{\nabla}_m^k \Big\|^2 = \Big\| \sum_{m\in\mathcal{M}_c^k} \nabla\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^k\big) - \nabla\mathcal{L}_m\big(\boldsymbol{\theta}^k\big) \Big\|^2 \tag{52a}$$

$$\leq \big|\mathcal{M}_c^k\big| \sum_{m\in\mathcal{M}_c^k} \Big\| \nabla\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^k\big) - \nabla\mathcal{L}_m\big(\boldsymbol{\theta}^k\big) \Big\|^2 \tag{52b}$$

$$\overset{(b)}{\leq} \frac{|\mathcal{M}_c^k|^2}{\alpha^2 M^2} \sum_{d=1}^{D} \xi_d \Big\| \boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d} \Big\|^2 + 6\sigma_{N,\delta/K}^2 \tag{52c}$$

where (b) uses the communication trigger condition (16), and the fact that $\sigma_{N,\delta/K}^2 = M \sum_{m\in\mathcal{M}} \sigma_{m,N,\delta/K}^2$.

Plugging (52) into (51), we have (for convenience, define $\beta_{D+1} = 0$ in the analysis)

$$\begin{aligned}
&\mathbb{V}^{k+1} - \mathbb{V}^k \\
&\leq -\frac{\alpha}{2} \Big\| \nabla\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 + \sum_{d=1}^{D} \left(\frac{3\xi_d \left|\mathcal{M}_c^k\right|^2}{2\alpha M^2} - \beta_d + \beta_{d+1} \right) \Big\| \boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d} \Big\|^2 + 9\alpha\sigma_{N,\delta/K}^2 \\
&+ \left(\frac{L}{2} - \frac{1}{2\alpha} + \beta_1 \right) \Big\| \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k \Big\|^2 + \frac{3\alpha}{2} \Big\| \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 + \frac{3\alpha}{2} \Big\| \hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2.
\end{aligned} \tag{53}$$

After defining some constants to simplify the notation, the proof is then complete.

Furthermore, using $\beta_d := \frac{3}{2\alpha} \sum_{\tau=d}^{D} \xi_\tau$, if the stepsize α, and the trigger constants $\{\xi_d\}$ satisfy

$$\alpha \leq \frac{\big(1 - 3\sum_{d=1}^{D} \xi_d\big)}{L} \tag{54}$$

then it is easy to verify that expressions in the parentheses in (53) are all nonpositive. Hence, we have that the descent in the Lyapunov function is bounded as

$$\begin{aligned}
&\mathbb{V}^{k+1} - \mathbb{V}^k \\
&\leq -\frac{\alpha}{2} \Big\| \nabla\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 + \frac{3\alpha}{2} \Big\| \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 + \frac{3\alpha}{2} \Big\| \hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 + 9\alpha\sigma_{N,\delta/K}^2.
\end{aligned} \tag{55}$$

Rearranging terms in (55), and summing up over $k = 1, \cdots, K$, we have

$$\begin{aligned}
\frac{1}{K} \sum_{k=1}^{K} \Big\| \nabla\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 &\leq \frac{2}{\alpha K} \mathbb{V}^1 + \frac{3}{K} \sum_{k=1}^{K} \Big\| \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 + \frac{3}{K} \sum_{k=1}^{K} \Big\| \hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 + 18\sigma_{N,\delta/K}^2 \\
&\overset{(c)}{\leq} \frac{2}{\alpha K} \mathbb{V}^1 + 3\sigma_T^2 + 21\sigma_{N,\delta/K}^2, \quad \text{w.p. } 1-\delta
\end{aligned} \tag{56}$$

where (c) follows from the finite-horizon truncation error in Lemma 6, and the gradient concentration result in Lemma 2 together with the union bound.

Therefore, using Lemmas 2 and 6, it readily follows that there exist $T = \mathcal{O}(\log(1/\epsilon))$, $K = \mathcal{O}(1/\epsilon)$, and $N = \mathcal{O}\left(\log(K/\delta)/\epsilon\right)$ such that

$$\frac{1}{K} \sum_{k=1}^{K} \Big\| \nabla\mathcal{L}(\boldsymbol{\theta}^k) \Big\|^2 \leq \frac{2}{\alpha K} \mathbb{V}^1 + 3\sigma_T^2 + 21\sigma_{N,\delta/K}^2 \leq \epsilon, \quad \text{w.p. } 1-\delta \tag{57}$$

from which the proof is complete.

Appendix F. Proof of Lemma 4

The idea is essentially to show that if (24) holds, then the learner m will not violate the LAPG conditions in (16) so that it does not upload, if it has uploaded at least once during the last d iterations.

To prove this argument, for the difference of two policy gradient evaluations, we have that

$$\begin{aligned}
&\left\|\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2\\
=&\left\|\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) + \nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k) + \nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k) - \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2\\
\overset{(a)}{\leq}&3\left\|\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})\right\|^2 + 3\left\|\nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k) - \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2\\
\overset{(b)}{\leq}&6\sigma_{m,N,\delta}^2 + 3\left\|\nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2, \quad \text{w.p. } 1-2\delta/K\\
\overset{(c)}{\leq}&6\sigma_{m,N,\delta}^2 + 3L_m^2\left\|\hat{\boldsymbol{\theta}}_m^{k-1} - \boldsymbol{\theta}^k\right\|^2, \quad \text{w.p. } 1-2\delta/K
\end{aligned} \tag{58}$$

where (a) uses $\|\mathbf{a}+\mathbf{b}+\mathbf{c}\|^2 \leq 3\|\mathbf{a}\|^2 + 3\|\mathbf{b}\|^2 + 3\|\mathbf{c}\|^2$; (b) uses Lemma 7 twice; and (c) follows from the smoothness property in Lemma 3.

Furthermore, suppose that at iteration k, the most recent iteration that the learner m did communicate with the controller is iteration $k-d'$ with $1 \leq d' \leq d$. Thus, we have $\hat{\boldsymbol{\theta}}_m^{k-1} = \boldsymbol{\theta}^{k-d'}$, which implies that

$$\begin{aligned}
6\sigma_{m,N,\delta}^2 + 3L_m^2\left\|\hat{\boldsymbol{\theta}}_m^{k-1} - \boldsymbol{\theta}^k\right\|^2 &= 6\sigma_{m,N,\delta}^2 + 3L_m^2\left\|\boldsymbol{\theta}^{k-d'} - \boldsymbol{\theta}^k\right\|^2\\
&= 6\sigma_{m,N,\delta}^2 + 3d'L^2\mathbb{H}(m)\sum_{b=1}^{d'}\left\|\boldsymbol{\theta}^{k+1-b} - \boldsymbol{\theta}^{k-b}\right\|^2\\
&\overset{(d)}{\leq} 6\sigma_{m,N,\delta}^2 + \frac{\xi_d}{\alpha^2M^2}\sum_{b=1}^{d'}\left\|\boldsymbol{\theta}^{k+1-b} - \boldsymbol{\theta}^{k-b}\right\|^2\\
&\overset{(e)}{\leq} 6\sigma_{m,N,\delta}^2 + \frac{\sum_{b=1}^{D}\xi_b\left\|\boldsymbol{\theta}^{k+1-b} - \boldsymbol{\theta}^{k-b}\right\|^2}{\alpha^2M^2}
\end{aligned} \tag{59}$$

where (d) follows since the condition (24) is satisfied, so that

$$\mathbb{H}(m) \leq \frac{\xi_d}{3d\alpha^2L^2M^2} \leq \frac{\xi_d}{3d'\alpha^2L^2M^2} \tag{60}$$

and (e) follows from our choice of $\{\xi_d\}$ such that for $1 \leq d' \leq d$, we have $\xi_d \leq \xi_{d'} \leq \ldots \leq \xi_1$ and $\|\boldsymbol{\theta}^{k+1-b} - \boldsymbol{\theta}^{k-b}\|^2 \geq 0$. Since (59) is exactly the RHS of (16), the trigger condition (16) will not be activated, and the learner m does not communicate with the controller at iteration k.

Note that the above argument holds for any $1 \leq d' \leq d$, and thus if (24) holds, the learner m communicates with the controller at most every other d iterations. Since (58) holds with probability $1-2\delta/K$, by using union bound, this argument holds with probability $1-2\delta$ for all $k \in \{1, \cdots, K\}$.

Appendix G. Proof of Theorem 2

Recalling the Lyapunov function (20), we have

$$\mathbb{V}^k := \mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*) + \sum_{d=1}^{D}\frac{3\sum_{j=d}^{D}\xi_j}{2\alpha}\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \tag{61}$$

Using (55) in the proof of Theorem 1, and choosing the stepsize as $\alpha = \frac{1}{L}\big(1 - 3\sum_{d=1}^{D}\xi_d\big)$, we have

$$\mathbb{V}^{k+1} - \mathbb{V}^k \leq -\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3\alpha}{2}\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3\alpha}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 9\alpha\sigma^2_{N,\delta/K}. \tag{62}$$

Summing up both sides from $k = 1, \ldots, K$, and initializing $\boldsymbol{\theta}^{1-D} = \cdots = \boldsymbol{\theta}^0 = \boldsymbol{\theta}^1$, we have

$$\begin{aligned}\frac{1}{K}\sum_{k=1}^{K}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 &\leq \frac{2L\left[\mathcal{L}(\boldsymbol{\theta}^1) - \mathcal{L}(\boldsymbol{\theta}^*)\right]}{(1 - 3\sum_{d=1}^{D}\xi_d)K} + 3\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 18\sigma^2_{N,\delta/K}\\ &\leq \frac{2L\left[\mathcal{L}(\boldsymbol{\theta}^1) - \mathcal{L}(\boldsymbol{\theta}^*)\right]}{(1 - 3\sum_{d=1}^{D}\xi_d)K} + 3\sigma_T^2 + 21\sigma^2_{N,\delta/K}, \quad \text{w.p. } 1 - \delta\end{aligned} \tag{63}$$

With regard to PG, following the standard analysis, it can guarantee that

$$\begin{aligned}\frac{1}{K}\sum_{k=1}^{K}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 &\leq \frac{2L}{K}\left[\mathcal{L}(\boldsymbol{\theta}^1) - \mathcal{L}(\boldsymbol{\theta}^*)\right] + 3\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2\\ &\leq \frac{2L}{K}\left[\mathcal{L}(\boldsymbol{\theta}^1) - \mathcal{L}(\boldsymbol{\theta}^*)\right] + 3\sigma_T^2 + 3\sigma^2_{N,\delta/K}, \quad \text{w.p. } 1 - \delta\end{aligned} \tag{64}$$

If the parameters T and N are chosen large enough (cf. (22)), so the first terms in the RHS of (63) and (64) each dominates the corresponding remaining two error terms. Therefore, to achieve the same ϵ-gradient error, with probability $1 - 2\delta$, the number of needed iterations under LAPG is $(1 - 3\sum_{d=1}^{D}\xi_d)^{-1}$ times that of PG.

Regarding the number of needed communication rounds, similar to the derivations in (Chen et al., 2018b, Proposition 1), we can use Lemma 4 to show that the LAPG's average communication rounds per iteration is $(1 - \Delta\bar{\mathbb{C}}(h; \{\gamma_d\}))$ times that of PG with probability $1 - 2\delta$. Together with the number of needed iterations discussed above, we arrive at (27) with probability $1 - 4\delta$.

As $h(\cdot)$ is non-decreasing, for a given γ', if $\gamma_D \geq \gamma'$, it readily follows that $h(\gamma_D) \geq h(\gamma')$. Together with the definition of $\Delta\bar{\mathbb{C}}(h; \{\gamma_d\})$ in (25), we arrive at

$$\Delta\bar{\mathbb{C}}(h; \{\gamma_d\}) = \sum_{d=1}^{D}\left(\frac{1}{d} - \frac{1}{d+1}\right)h(\gamma_d) \geq \sum_{d=1}^{D}\left(\frac{1}{d} - \frac{1}{d+1}\right)h(\gamma_D) \geq \frac{D}{D+1}h(\gamma'). \tag{65}$$

Therefore, if we choose the parameters as

$$\xi_1 = \xi_2 = \ldots = \xi_D = \xi \qquad \text{and} \qquad \alpha = \frac{1 - 3D\xi}{L} \qquad \text{and} \qquad \gamma_d = \frac{\xi/d}{3\alpha^2L^2M^2},\ d = 1, \ldots, D \tag{66}$$

the total communication is reduced if the following relation is satisfied (cf. (27))

$$\frac{\mathbb{C}_{\text{LAPG}}(\epsilon)}{\mathbb{C}_{\text{PG}}(\epsilon)} = \left(1 - \frac{D}{D+1}h(\gamma')\right)\cdot\frac{1}{1 - 3D\xi} < 1. \tag{67}$$

Clearly, (67) holds if we have $h(\gamma') > 3(D+1)\xi$. On the other hand, the condition $\gamma_D \geq \gamma'$ requires

$$\xi/D \geq \gamma'(1 - D\xi)^2M^2. \tag{68}$$

Clearly, if $\xi > \gamma' DM^2$, then (68) holds. In all, if we have

$$\gamma' < \frac{\xi}{DM^2} < \frac{h(\gamma')}{3(D+1)DM^2} \tag{69}$$

then $\mathbb{C}_{\text{LAPG}}(\epsilon) \leq \mathbb{C}_{\text{PG}}(\epsilon)$ in Theorem 2 holds with probability $1 - 4\delta$.

References

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Proc. Advances in Neural Info. Process. Syst.*, pages 1709–1720, Long Beach, CA, December 2017.

Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *J. Artificial Intelligence Res.*, 15:319–350, 2001.

Justin A Boyan and Michael L Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Proc. Advances in Neural Info. Process. Syst.*, pages 671–678, Denver, CO, November 1994.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540*, 2016. URL `https://github.com/openai/gym`.

Tianyi Chen, Sergio Barbarossa, Xin Wang, Georgios B. Giannakis, and Zhi-Li Zhang. Learning and management for Internet-of-Things: Accounting for adaptivity and scalability. *Proc. of the IEEE*, November 2018a.

Tianyi Chen, Georgios B Giannakis, Tao Sun, and Wotao Yin. LAG: Lazily aggregated gradient for communication-efficient distributed learning. In *Proc. Advances in Neural Info. Process. Syst.*, Montreal, Canada, December 2018b. URL `arxiv.org/abs/1805.09965`.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. of the Assoc. for the Advanc. of Artificial Intell.*, pages 746–752, Orlando, FL, October 1998.

Marc Peter Deisenroth. *Efficient Reinforcement Learning Using Gaussian Processes*, volume 9. KIT Scientific Publishing, Karlsruhe, Germany, 2010.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Intl. Conf. Auto. Agents and Multi-agent Systems*, pages 66–83, 2017.

Michael I. Jordan, Jason D. Lee, and Yun Yang. Communication-efficient distributed statistical inference. *J. American Statistical Association*, to appear, 2018.

Sham M Kakade. A natural policy gradient. In *Proc. Advances in Neural Info. Process. Syst.*, pages 1531–1538, Vancouver, Canada, December 2002.

Soummya Kar, José MF Moura, and H Vincent Poor. QD-learning: A collaborative distributed strategy for multi-agent reinforcement learning through Consensus + Innovations. *IEEE Trans. Sig. Proc.*, 61(7):1848–1862, July 2013.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Proc. Advances in Neural Info. Process. Syst.*, pages 1008–1014, Denver, CO, December 2000.

Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proc. Intl. Conf. Machine Learn.*, Stanford, CA, June 2000.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proc. USENIX Symp. Operating Syst. Design and Implement.*, volume 14, pages 583–598, Broomfield, CO, October 2014.

Yuxi Li and Dale Schuurmans. Mapreduce for parallel reinforcement learning. In *European Workshop on Reinforcement Learning*, pages 309–320. Springer, 2011.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proc. Intl. Conf. Learn. Representations*, San Juan, Puerto Rico, May 2016.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proc. Advances in Neural Info. Process. Syst.*, Long beach, CA, December 2017.

Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, April 2017. URL `https://research.googleblog.com/2017/04/federated-learning-collaborative.html`.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. Intl. Conf. Artificial Intell. and Stat.*, pages 1273–1282, Fort Lauderdale, FL, April 2017.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proc. Intl. Conf. Machine Learn.*, pages 1928–1937, New York City, NY, June 2016.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. In *Proc. Intl. Conf. Machine Learn. on Deep Learn. Workshop*, Lille, France, July 2015a.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint:1507.04296*, 2015b.

Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proc. Intl. Conf. Machine Learn.*, pages 2681–2690, Sydney, Australia, June 2017.

Matteo Papini, Matteo Pirotta, and Marcello Restelli. Adaptive batch size for safe policy gradients. In *Proc. Advances in Neural Info. Process. Syst.*, pages 3591–3600, Long beach, CA, December 2017.

Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. In *Proc. Intl. Conf. Machine Learn.*, pages 4026–4035, Stockholm, Sweden, July 2018.

Iosif Pinelis. Optimum bounds for the distributions of martingales in banach spaces. *The Annals of Probability*, 22(4):1679–1706, October 1994.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Proc. Advances in Neural Info. Process. Syst.*, pages 693–701, Granada, Spain, December 2011.

Jeff Schneider, Weng-Keen Wong, Andrew Moore, and Martin Riedmiller. Distributed value functions. In *Proc. Intl. Conf. Machine Learn.*, pages 371–378, Bled, Slovenia, June 1999.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proc. Intl. Conf. Machine Learn.*, pages 1889–1897, Lille, France, July 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint:1707.06347*, July 2017.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint:1610.03295*, 2016.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proc. Intl. Conf. Machine Learn.*, Beijing, China, June 2014.

Sebastian U Stich. Local SGD converges fast and communicates little. *arXiv preprint:1805.09767*, May 2018.

Ion Stoica, Dawn Song, Raluca Ada Popa, David Patterson, Michael W. Mahoney, Randy Katz, Anthony D. Joseph, Michael Jordan, Joseph M Hellerstein, Joseph E Gonzalez, et al. A Berkeley view of systems challenges for AI. *arXiv preprint:1712.05855*, December 2017.

Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, June 2000.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proc. Advances in Neural Info. Process. Syst.*, pages 1057–1063, Denver, CO, December 2000.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *arXiv preprint:1507.04296*, July 2015.

Hoi-To Wai, Zhuoran Yang, Zhaoran Wang, and Mingyi Hong. Multi-agent reinforcement learning via double averaging primal-dual optimization. In *Proc. Advances in Neural Info. Process. Syst.*, Montreal, Canada, December 2018.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learn.*, 8(3-4):279–292, May 1992.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992.

David H Wolpert, Kevin R Wheeler, and Kagan Tumer. General principles of learning-based multi-agent systems. In *Proc. of the Annual Conf. on Autonomous Agents*, pages 77–83, Seattle, WA, May 1999.

Kaiqing Zhang, Wei Shi, Hao Zhu, Emiliano Dall'Anese, and Tamer Başar. Dynamic power distribution system management with a locally connected communication network. *IEEE J. of Selected Topics in Sig. Proc.*, 12(4):673–687, 2018a.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Networked multi-agent reinforcement learning in continuous spaces. In *Proc. IEEE Conf. Decision and Control*, pages 5872–5881, Miami, FL, December 2018b.

Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Başar. Fully decentralized multi-agent reinforcement learning with networked agents. In *Proc. Intl. Conf. Machine Learn.*, pages 5872–5881, Stockholm, Sweden, July 2018c.

Yuchen Zhang and Xiao Lin. DiSCO: Distributed optimization for self-concordant empirical loss. In *Proc. Intl. Conf. Machine Learn.*, pages 362–370, Lille, France, June 2015.