Distributed Policy Gradient with Heterogeneous Computations for Federated Reinforcement Learning

Ye Zhu, Xiaowen Gong
Department of Electrical and Computer Engineering
Auburn University
Auburn, AL 36849, USA
{yzz0211,xgong}@auburn.edu

2023 57th Annual Conference on Information Sciences and Systems (CISS) | 978-1-6654-5181-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CISS56502.2023.10089771

***Abstract*—The rapid advances in federated learning (FL) in the past few years have recently inspired federated reinforcement learning (FRL), where multiple reinforcement learning (RL) agents collaboratively learn a common decision-making policy without exchanging their raw interaction data with their environments. In this paper, we consider a general FRL framework where agents interact with different environments with identical state and action spaces but different rewards and dynamics. Motivated by the fact that agents often have heterogeneous computation capabilities, we propose a Federated Heterogeneous Policy Gradient (FedHPG) algorithm for FRL, where agents can use different numbers of data trajectories (i.e., batch sizes) and different numbers of local computation iterations for their respective PG algorithms. We characterize performance bounds for the learning accuracy of FedHPG, which shows that it achieves a learning accuracy ϵ with sample complexity of $O(1/\epsilon^2)$, which matches the performance of existing RL algorithms. The results also show the impacts of local iteration numbers and batch sizes for iteration on the learning accuracy. We also extend FedHPG to heterogeneous policy gradient variance reduction (FedHPGVR) algorithm based on the variance reduction method, and analyze the convergence of this algorithm. The theoretical results are verified empirically for benchmark RL tasks.**

I. Introduction

As an emerging ML paradigm, *federated learning* (FL) carries out model training in a distributed manner [1]: Instead of collecting data from a possibly large number of devices to a central server in the cloud for training, FL trains a global ML model by aggregating local ML models computed distributedly across edge devices based on their local data. One significant advantage of FL is to preserve the privacy of individual devices' data. Moreover, since only local ML models rather than local data are sent to the server, the communication costs can be greatly reduced. Furthermore, FL can exploit substantial computation capabilities of ubiquitous smart devices.

Along a different avenue, as a powerful paradigm of ML, reinforcement learning (RL) has been increasingly studied in the past few years. Generally, the objective of RL training is to acquire the optimal policy to maximize the long-term reward. Recently, RL has recently been applied to enormous real-world sequential decision-making problems. However, despite its extraordinary performance in simulated environments, RL always suffers from many challenges in practical scenarios [2] such as limited samples when learning, especially in large state space or action space. For example, when RL is applied to train a walking robot, its performance is limited by the number of samples generated by itself.

To exploit the benefits of FL for RL, federated reinforcement learning [3] has been proposed recently as a promising approach, where multiple RL agents exchange their knowledge to collectively learn a better decision-making policy. Compared to conventional distributed RL where agents can communicate collected samples directly to learn the policy, FRL can greatly reduce communication costs while protecting agents' privacy.

In order to fully realize the potential of FRL, several challenges need to be addressed due to salient features of FL. First, FRL implies heterogeneous data across agents, which mean that agents can interact with diverse environments with different reward and state transition functions. Second, in contrast to conventional distributed ML where nodes communicate after every iteration of local computations, an agent may perform multiple local iterations of computation before exchanging its local model with other agents. While this feature can reduce communication costs of FRL, it may also slow down the convergence of the global model due to local drifts.

Moreover, in FRL, we also need to take into account some unique features of RL that distinguish it from supervised or unsupervised learning. First, RL is an online learning process where data samples are collected while a policy is learned via computations from the data samples. The policy used to collect data samples can be different from the policy currently evaluated in search for the optimal policy. Second, when the state and/or action space is large, the amount of information required to be exchanged could be enormous. In this case, the policy gradient method is efficient in finding the optimal policy.

In this paper, we explore FRL where agents perform

The work of Ye Zhu and Xiaowen Gong was supported by NSF CCSS grant No. 2121215.

978-1-6654-5181-9/23/$31.00 ©2023 IEEE

distributed PG with heterogeneous computation configurations. In FRL, besides heterogeneous environments faced by agents, agents' devices also have heterogeneous computation capabilities. Indeed, the computation capabilities (including computation costs) of agents' devices can be highly diverse in practice. For example, the computing resources (e.g., CPU, memory) available on a low-end smartphone would be much less powerful than those on a high-profile GPU-based desktop, and it can vary greatly over time depending on the demand of other computation tasks on the device. However, existing studies on FRL only considered homogeneous computation configurations of devices, including local iteration numbers and mini-batch sizes. Therefore, this scheme is not flexible and adaptable to heterogeneous and time-varying computation capabilities of devices.

The main contributions of this paper are summarized as follows:

- We propose a Federated Heterogeneous Policy Gradient (FedHPG) algorithm for FRL, where agents can use heterogeneous computation configurations (i.e., batch size, local iteration number) to perform the policy gradient method in a distributed manner. The proposed FedHPG algorithm allows distributed agents with heterogeneous computation capabilities to perform FRL efficiently.
- We conduct convergence analysis for the FedHPG algorithm by characterizing performance bounds on the learning accuracy as a function of agents' heterogeneous computation configurations. Our results show that the sampling complexity for achieving a certain learning accuracy ϵ is $O(\frac{1}{\epsilon^2})$, which matches the performance of existing RL algorithms. The results also characterize the impacts of agents' computation configurations on the learning accuracy, based on which we can find appropriate design of these parameters.
- We extend the FedHPG algorithm to the heterogeneous policy gradient variance reduction (FedHPGVR) algorithm based on the variance reduction method, which utilizes past policy gradients to reduce variances of current policy gradients. We analyze the convergence of this algorithm, which shows that its sampling complexity is also as good as existing RL algorithms.
- We evaluate the performance of the FedHPG algorithms by conducting numerical experiments for a benchmark application of RL. The experimental results demonstrate the effectiveness of the proposed algorithms.

The remainder of this paper is organized as follows. Section II reviews related work. In Section III, we propose Federated Heterogeneous Policy Gradient (FedHPG) for FRL. In Section IV, we analyze the convergence of the FedHPG algorithm. In Section V, we extend FedHPG to FedHPGVR. Numerical results based on experiments are provided in Section VI.

II. Related Work

Distributed and Multi-Task Reinforcement Learning. Distributed reinforcement learning (DRL) involves multiple agents operating in a distributed fashion. As a major class of DRL, *parallel RL* [4], [5] uses multiple learners to solve a large-scale single-agent RL task [6], [7], where the learners aim to learn a common policy for different instances of the *same environment*. Another major class of DRL is multi-agent reinforcement learning (MARL), where a group of agents operate in a common environment where all agents' action influence the global state transition and aim to find a joint policy combining all local policies in a collaborative manner [8]–[10], or find their respective policies in a non-collaborative manner [11]. These prior work of distributed RL are different from FRL which is studied in this paper, since 1) agents in FRL can interact with diverse environments and collaboratively aim to learn a common policy for their different environments; 2) FRL involves some unique features of FL, including multiple local computation iterations before communications, and heterogeneous computation capabilities of agents which are considered in this paper.

Multi-task reinforcement learning (MTRL) considers a number of different but possibly similar RL tasks [12], and aims to find a good policy for each task. Typically, each task is drawn from a finite set of Markov decision processes with identical state and action spaces, but different reward and transition model parameters [13]. Therefore, MTRL is different from FRL, as the goal of FRL is to find a single common policy that performs well for multiple different tasks on average.

Federated Learning. FL has emerged as a disruptive computing paradigm for ML by democratizing the learning process to potentially many individual users using their end devices. The past few years have seen tremendous research on FL. However, prior work on FL predominantly focused on federated supervised learning (and a few of them on federated unsupervised learning). The settings of FRL have significant differences compared to those of federated supervised learning, due to the salient features of RL, such as the online learning nature of RL. There have been a few recent work on FRL [4], [14]–[16]. However, these work have not considered distributed FL algorithms with heterogeneous computation configurations due to heterogeneous computation capabilities of agents, which is the focus of this paper.

III. Federated Reinforcement Learning with Heterogeneous Policy Gradient

Similar to FL where multiple agents collectively train a shared model, FRL is proposed to find a single and common policy that is simultaneously effective for a number of tasks executing on corresponding agents. It is worth noting that each agent independently samples and calculates gradients by interacting with its own environment. The Markov decision process (MDP) at agent k can be given by: $\mathcal{M}_k \triangleq \{\mathcal{S}, \mathcal{A}, \mathcal{P}_k, \mathcal{R}_k, \gamma_k\}$ where the state space and action space have to be common across tasks. $\mathcal{P}_k(s'|s,a)$ denotes the transition probabilities from state s to state s' by taking action a in the environment of agent k. $\mathcal{R}_k(s,a)$ and γ_k are the reward function and discount factor for agent k respectively. In our setting, all MDPs are episodic with trajectory horizon H. Multiple

agents aim to learn a common policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The trajectory sampled by any stationary policy can showed as $l_k = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}_k$ where the return along the trajectory is $\mathcal{R}(\tau_k) \triangleq \sum_{t=0}^{H-1} \gamma_k^t \mathcal{R}_k(s_t, a_t)$. Generally, the performance of a policy π can be written as

$$J_k(\pi) = \mathbb{E}_{\tau_k \sim p_k(\cdot|\pi)}[\mathcal{R}(l_k)|\mathcal{M}_k] \tag{1}$$

where $p_k(\cdot|\pi)$ denotes the density distribution induced by policy π of all possible trajectories in the environment of agent k. The goal of agents in RL is to find $\pi^* \in \arg\max_\pi \{J_k(\pi)\}$.

To parameterize the policy, we use π_θ to denote the policy parameterized by $\theta \in \mathbb{R}^d$. For convenience, we use θ to denote the policy π_θ. The goal of agents in FRL is to cooperatively find the common policy θ^* which can maximize the total cumulative discounted rewards

$$\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \sum_{k=1}^{K} J_k(\theta), \tag{2}$$

where $J_k(\theta)$ is a non-concave objective.

Policy gradient methods have been widely deployed in model-free RL. Taking the gradient of $J_k(\theta)$ with respect of θ, we can get [17]:

$$\nabla J_k(\theta) = \mathbb{E}_{l_k \sim p_k(\cdot|\theta)}[\nabla \log p_\theta(l_k)|\mathcal{R}(l_k)]. \tag{3}$$

The policy can be optimized by running gradient ascent algorithms. However, it is impossible to calculate the full gradient in (3). Hence stochastic gradient ascent is typically used to update the policy. In particular, agent k samples a batch of trajectories $\{l_{k,s}\}_{s=0}^{B_k}$ using policy θ_m in each iteration m and update the policy by $\theta_{m+1} = \theta_m + \gamma \hat{\nabla}_{B_k} J_k(\theta_m)$ where γ is the step size and $\hat{\nabla}_{B_k} J_k(\theta)$ is the policy estimate of (3) based on sampled trajectories $\{l_{k,s}\}_{s=0}^{B_k}$. B_k is the number of sampled trajectories in each local policy iteration at agent k which can be different among agents since agents have heterogeneous capability of sample collection. $\hat{\nabla}_{B_k} J_k(\theta)$ can be defines as

$$\hat{\nabla}_{B_k} J_k(\theta) = \frac{1}{B_k} \sum_{s=1}^{B_k} g(l_{k,s}|\theta) \tag{4}$$

where $g(l_{k,s}|\theta)$ denote the unbiased estimate of the true gradient $\nabla \log p_\theta(l_k)|\mathcal{R}(l_k)$, i.e., $\nabla J_k(\theta)$. The most common gradient estimators for policy gradient such as REINFORCE and GPOMDP. The REINFORCE [18] is:

$$g(l\,|\theta) = \left[\sum_{h=0}^{H-1} \nabla_\theta \log \pi_\theta(a_h\,|s_h)\right] \left[\sum_{h=0}^{H-1} \gamma^h R(s_h, a_h) - C_b\right]. \tag{5}$$

The GPOMDP [19] is:

$$g(l\,|\theta) = \sum_{h=0}^{H-1} \left[\sum_{t=0}^{h} \nabla_\theta \log \pi_\theta(a_h\,|s_h)\right] \cdot (\gamma^h R(s_h, a_h) - C_{bh}). \tag{6}$$

Note that C_b and C_{bh} are the corresponding baselines.

A. Algorithm Description

The pseudocode for the proposed federated heterogeneous policy gradient (FedHPG) in reinforcement learning is shown in algorithm 1. FedHPG starts with a randomly initialized parameter $\tilde{\theta}_0$ at the server. At the beginning of the t-th round, the server keeps a snapshot of its parameter from the previous round and broadcasts this parameter to all agents.

At the $(\tau_k + 1)$-th local iteration in the t-th round, agent k sampled a batch $\{l_{t,s}^{k,\tau}\}_{s=1}^{B_k}$ using current policy $\theta_t^{k,\tau}$, then the local policy can be updated by the gradient $\hat{\nabla}_{B_k} J_k(\theta)$ which is an unbiased estimate of (3) using sampled trajectories $\{l_{t,s}^{k,\tau}\}_{s=1}^{B_k}$. After executing τ_k local update, agent k sends $\theta_t^{k,\tau}$ to the server. Then the server collects K parameters of local policy returned from the agents. Then the server aggregates the parameter and broadcasts to all agents to start a new round of federation.

Algorithm 1 Federated Heterogeneous Policy Gradient (FedHPG)

Input: number of rounds T, batch size $\{B_k\}$, stepsize η, initial model $\tilde{\theta}_0$
for $t = 1$ to T **do**
  $\theta_0^t \leftarrow \tilde{\theta}_{t-1}$
  all agents $k = 1, 2, ..., K$ do concurrently:
  $\theta_t^{k,0} = \tilde{\theta}_t$
  for $\tau = 0$ to $\tau_k - 1$ **do**
    Sample B_k trajectories $\{l_{t,s}^{k,\tau}\}_{s=1}^{B_k}$ from $p_k(\cdot|\theta_t^{k,\tau})$
    update: $\theta_t^{k,\tau+1} = \theta_t^{k,\tau} + \frac{\gamma}{B_k} \sum_{s=1}^{B_k} g(\theta_t^{k,\tau})$
  $\tilde{\theta}_{t+1} = \frac{1}{K} \sum_{k=1}^{K} \theta_t^{k,\tau_k - 1}$
Output: $\{\tilde{\theta}_t\}_{t=1}^{T}$

IV. Convergence Analysis of FedHPG

In this section, we present the performance of FedHPG. Before that, we introduce a few essential assumptions required for our theoretical analysis throughout the rest of this paper, all of which are common in the literature. Due to space limitation, the complete proofs of results in this paper can be found in our online technical report [20].

Assumption 1. (**Bounded gradient variance**) For each agent k, there is a constant σ_k such that $\mathbb{E}\|g(l_k\,|\theta) - \nabla J_k(\theta)\|^2 \leq \sigma_k^2$ for all policy π_θ.

Assumption 1 is a standard assumption commonly used in stochastic non-convex optimization.

Assumption 2. (**Bounded policy gradient**) For each state-action pair (s, a), $\pi_\theta(a\,|s)$ is the policy of an agent at state s where $\theta \in \mathbb{R}^d$. There exist constants G and F such that

$$|\nabla_\theta \log \pi_\theta(a\,|s)| \leq G, \quad \left\|\nabla_\theta^2 \log \pi_\theta(a\,|s)\right\| \leq F.$$

Assumption 2 guarantees the smoothness assumption on the objective function $J(\theta)$ commonly used in non-convex optimization.

Assumption 3. (**Bounded gradient**) For all agent k, there is a boundedness assumption on the gradient norm $\mathbb{E}\|g(l_k|\theta)\|^2 \leq M$ for all policy π_θ.

Assumption 3 can be easily satisfied by clipping the gradients during training. Actually, it can be deduced from Assumption 2 [17]. For brevity, we put it as Assumption 3.

Assumption 4. (**Function smoothness**) For all agents k, the objective function is L_k-smoothness.

$$\|\nabla J_k(\theta_1) - \nabla J_k(\theta_2)\| \leq L_k \|\theta_1 - \theta_2\|.$$

Assumption 4 can also be deduced from Assumption 2 [17]. The smoothness of the objective function is critical in the convergence analysis of optimization algorithms, especially in nonconvex optimization. Based on Assumption 2 and the proof of smoothness in cumulative rewards, we can show that the aggregated cumulative rewards $J(\theta)$ is L-smooth with $\sum_{k=1}^{K} L_k$ [21]. FedHPG can guarantee the following convergence result which provides some useful insights.

Theorem 1. Suppose the initial state distribution across agents is uniform and the gradient estimator is the REINFORCE or GPOMDP estimator. If the stepsize $\gamma < \min\left\{\frac{1}{L}, \frac{2(K+\bar{\tau})}{LK\alpha_B}\right\}$, the average gradient norm of the objective is bounded by:

$$\begin{aligned}&\min_{t=1,\ldots T} \mathbb{E}\|\nabla J(\theta_t)\|^2 \\ &\leq \frac{2\mathbb{E}J(\theta^*)-J(\theta_0)}{\gamma\bar{\tau}T} + \frac{\gamma^2L^2V}{6K\bar{\tau}}\sum_{k=1}^{K}\frac{\sigma_k^2\tau_k(\tau_k-1)(2\tau_k-1)}{B_k} \\ &+\frac{\gamma LV}{K\bar{\tau}}\sum_{k=1}^{K}\frac{{\tau_k}^2\sigma_k^2}{B_k} + \frac{\gamma LM}{\bar{\tau}}\left(\gamma L\alpha_A + \left(1-\frac{\gamma L}{2}\right)\alpha_B\right)\end{aligned} \tag{7}$$

where α_A and α_B are defined as following:

$$\alpha_A = \frac{1}{6K}\sum_{k=1}^{K}\tau_k(\tau_k-1)(2\tau_k-1)$$
$$\alpha_B = \frac{1}{K}\sum_{k=1}^{K}\tau_k(\tau_k-1)$$

Remark 1. Theorem 1 characterizes an upper bound on the learning accuracy of FedHPG, which consists of four terms. As $\gamma > 0$, the 1st term converges to zero as T increases. Besides, it gives an $O(1/T)$ convergence rate which matches that of [4] and the results in non-convex optimization. The 2nd and 3rd terms are caused by the variances of stochastic policy gradient descent, and thus these two terms decrease as the batch sizes $\{B_k\}$ increase (and diminish to 0 as $\{B_k\}$ go to infinity).

Remark 2. We observe that the last three terms all depend on agents' local iteration numbers $\{\tau_k\}$, and they increase as $\{\tau_k\}$ increase. Intuitively, due to agents' heterogeneous data (as a result of their heterogeneous environments), more local computation iterations drives each agent's local policy more towards its local optimal policy and possibly away from the global optimal policy (also known as "local drifts" in existing works on federated supervised learning [22], [23]). As a result, the learning accuracy bound increases as the local iteration numbers go up.

We also observe that the impacts of agents' batch sizes on the learning accuracy can vary for different agents, which depend on their local iteration numbers: an agent with a larger local iteration number τ_k has a larger weight of its batch size B_k in the bound. Therefore, it is beneficial to choose agents' batch sizes based on their weights on the learning accuracy.

We note that the local iteration numbers minimize the bound when they are all equal to 1 (i.e., $\tau_k = 1, \forall i$). However, it has been widely shown that multiple local iterations achieve good performance empirically in practice while reducing communication costs (compared to communicating after every local iteration). Therefore, we should set local iteration numbers based on their empirical performance as well as agents' computation capabilities, rather than the learning accuracy bound (which captures the worst-case scenario).

Remark 3. We observe that the 4th term in the bound does not depend on the number of communication rounds T and the batch sizes $\{B_k\}$. Note that such a term does not exist when the local iteration numbers are all equal to 1 (which is captured in some prior work [22]). Therefore, the 4th term cannot be controlled by T and $\{B_k\}$. However, we can make this term arbitrarily small by choosing a sufficiently small stepsize γ. To this end, the stepsize γ should satisfy the condition below (see our technical report [20] for details):

$$\gamma < \min\left\{\frac{1}{L}, \frac{\epsilon\bar{\tau}}{(\alpha_A+\alpha_B)LM}, \frac{2(K+\bar{\tau})}{LK\alpha_B}\right\}. \tag{8}$$

Sample complexity is an important performance metric for RL, as it quantifies how much experience need to be acquired in order to find a good policy. Note that data samples for RL come from interactions with the environment and thus data collection is a costly and time-consuming process. Formally, sample complexity measures how large is the training set required in order to learn a near optimal policy [24] (typically ϵ-approximate optimal policy [4], [17], [25]).

Corollary 1. The policy found by Algorithm 1 is ϵ-approximate, i.e., $\mathbb{E}\|\nabla J(\theta_t)\|^2 < \epsilon$, if the number of rounds T and agents' batch sizes $\{B_k\}$ satisfy the following conditions:

$$T = O\left(\tfrac{1}{\epsilon^2}\right)$$
$$B_k = O(1)$$

which implies that the sample complexity of each agent is given by:

$$\mathbb{E}\left[Traj(\epsilon)\right] \leq T\tau_k B_k \leq O\left(\frac{1}{\epsilon^2}\right).$$

which agrees with the results of [18], [25], [26].

V. Extension to FedHPGVR with Variance Reduction

Stochastic variance reduction policy gradient (SVRPG) is a typical algorithm to decrease the variance which reuses past gradient computations to reduce the variance of the current gradient estimate [25]. In federated reinforcement learning architecture, we propose FedHPGVR algorithm based on

SVRPG algorithm, shown as Algorithm 2. At the beginning of the t-th epoch, SVRPG will first regard the current policy θ_0^t as a reference point and each agent use it to sample B_k trajectories and compute a gradient estimator $\mu_t^k = \frac{1}{B_k}\sum_{i=1}^{B_k} g(l_{t,s}^k \,|\theta_0^t)$, where g is the REINFORCE or GPOMDP estimator computed in k-th agent (a.k.a. k-th environment). Then at the n-th iteration within the t-th epoch, the agent k samples b_k trajectories using the current policy $\theta_{t,n}^k$ and then update the policy based on the following semi-stochastic gradient:

$$v_{t,n}^k = \frac{1}{b_k}\sum_{j=1}^{b_k}\left[g(l_{t,n}^{k,j}\,|\theta_{t,n}^k - w\cdot g(l_{t,n}^{k,j}\,|\theta_t^0)\right] + \mu_t^k, \qquad (9)$$

where $w := w(l_{t,n}^{k,j}\,|\theta_{t,n}^k, \theta_t^0) = p(l_{t,n}^{k,j}\,|\theta_t^0)\Big/p(l_{t,n}^{k,j}\,|\theta_{t,n}^k)$ which is known as the importance weight from $p(l_{t,n}^{k,j}\,|\theta_t^0)$. Importance weight can preserve the unbiasedness of the gradient estimate [25].

At the beginning of analyzing the performance of algorithm 2, we put some additional lemmas and assumptions which will be used in following analysis.

Algorithm 2 Federated Heterogeneous Policy Gradient with Variance Reduction (FedHPGVR)

Input: number of rounds T, batch sizes B_k and b_k, stepsize η, initial model $\tilde{\theta}_0$
for $t = 1$ to T **do**
  $\theta_0^t \leftarrow \tilde{\theta}_{t-1}$
  for $k = 1$ to K **do**
    Sample B_k trajectories $\{l_{t,s}^k\}_{s=1}^{B_k}$ from $p_k(\cdot|\theta_t^0)$
    $\mu_t^k = \frac{1}{B_k}\sum_{i=1}^{B_k} g(l_{t,s}^k\,|\theta_0^t)$
    for $n = 0$ to $N-1$ **do**
      Sample b_k trajectories $\{l_{t,n}^{k,j}\}_{j=1}^{b_k}$ from $p_k(\cdot\,|\theta_{t,n}^k)$
      $v_{t,n}^k = \frac{1}{b_k}\sum_{j=1}^{b_k}\left[g(l_{t,n}^{k,j}\,|\theta_{t,n}^k - w\cdot g(l_{t,n}^{k,j}\,|\theta_t^0)\right] + \mu_t^k$
      (For ease of notation, $w := w(l_{t,n}^{k,j}\,|\theta_{t,n}^k, \theta_t^0)$)
      $\theta_{t,n+1}^k = \theta_{t,n}^k + \gamma v_{t,n}^k$
  $\tilde{\theta}_t \leftarrow \frac{1}{K}\sum_{k=1}^{K}\theta_{t,N}^k$
Output: $\{\tilde{\theta}_t\}_{t=1}^T$

Assumption 5. Let $W < \infty$ be a constant. For each pair of policies (target policy and evaluation policy) shown in Algorithm 2 and for each trajectory, we have

$$Var[w(l\,|\theta_1, \theta_2)] \le W \quad \forall\ \theta_1, \theta_2 \in \mathbb{R}^d, l \sim p_k(\cdot\,|\theta_2)$$

Theorem 2. Suppose the initial state distribution across agents is uniform and the gradient estimator is the REINFORCE or GPOMDP estimator. If the stepsize $\gamma < \frac{1}{L}$, then the average gradient norm of the objective is bounded by:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla J(\theta_t)\|^2 \le \frac{2K\mathbb{E}[J(\theta^*)-J(\theta_0)]}{T\gamma(K-1-\beta)} + \frac{(\beta\gamma L+1)(K-1-\beta)\zeta}{K\beta}\sum_{k=1}^{K}\frac{N}{b_k} + \frac{(\beta\gamma L+1)(K-1-\beta)\xi}{K\beta}\sum_{k=1}^{K}\frac{N}{B_k} \qquad (10)$$

where ζ and ξ are some positive constants depending on W, M, G and F.

Remark 4. We observe that the three terms in the bound depend on the number of rounds T, batch sizes $\{B_k\}$, and $\{b_k\}$, respectively, and these terms are decreasing with respect to these parameters. Therefore, to achieve a vanishing convergence error, we should choose sufficiently large T, $\{B_k\}$, and $\{b_k\}$, such that each term can be made arbitrarily small. In this case, the sample complexity is given by $O\left(\frac{K}{\epsilon^2}\right)$.

VI. Numerical Experiments

In this section, we evaluate the performance of our proposed FedHPG algorithm on GridWorld, which is a common benchmark platform for RL. In this problem, the agent is placed in a grid of cells, where each cell can be defined by the desired goal, an obstacle, or empty. For solving this multi-task GridWorld, the agents implement Algorithm 1 where the local gradients are estimated using a Monte-Carlo approach, and the states are their locations in the grid. After 100 training episodes, the agents agree on a unified policy, whose performance is tested in parallel in all environments. The agent selects an action from $(up, down, left, right)$ to move to the next cell. It then receives a reward of +1 if it reaches the desired goal, -1 if it gets into an obstacle, and 0 otherwise. The goal of the agent is to reach a desired position in a minimum number of steps (or maximize its cumulative rewards).

1) Impact of batch sizes

We compare globally averaged reward among 4 agents while using heterogeneous batch size to update the local policy network over rounds showed in Fig.1. We observe that for the case of heterogeneous batch size for agents to updating the policy, the algorithm executed on that agents have larger batch for updating will have better performance, which conforms to Theorem 1.

2) Impact of local iteration numbers

We compare globally averaged reward among 4 agents while using different local iteration number to update the local policy over rounds. The first case which all agents use larger local iteration number, $\tau_1 = 5, \tau_2 = 5, \tau_3 = 6, \tau_4 = 6$ and the second case which agents use $\tau_1 = 2, \tau_2 = 2, \tau_3 = 3, \tau_4 = 3$ respectively are compared in Fig.2. We observe that when agents use more local iteration number will achieve good performance.

3) Impact of FedHPGVR

We typically evaluate the effect of the number of inner loop N on the performance of FedHPGVR algorithm, showed in Fig.3. We observe when N is bigger, the algorithm will show good performance since more local sampling give better quality of gradient estimates.

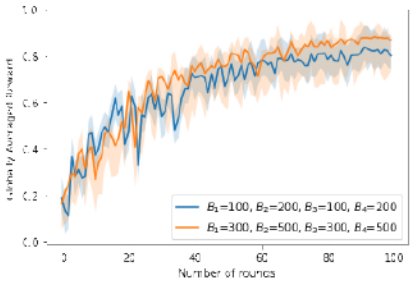

Fig. 1. Heterogeneous batch size for updating

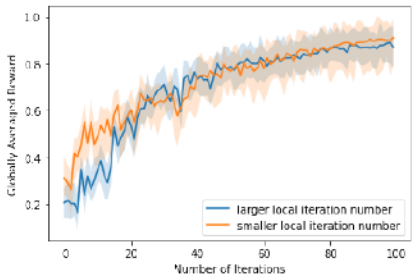

Fig. 2. Heterogeneous local iteration number

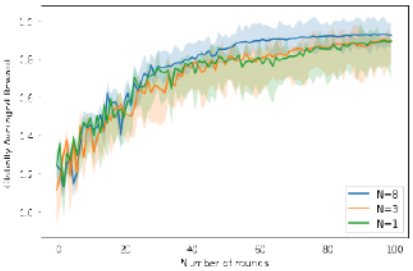

Fig. 3. Different N in FedHPGVR

VII. Conclusion and Future Work

In this paper, we propose a federated heterogeneous policy gradient (FedHPG) algorithm for FRL, where agents can use heterogeneous computation configurations (i.e., batch size and local iteration number) to perform the policy gradient method in a distributed manner. We characterized its performance bounds on the learning accuracy as a function of heterogeneous computation configurations, which showed that the sampling complexity matches the performance of existing RL algorithms. We extended the FedHPG algorithm to FedHPGVR which utilizes past policy gradients to reduce variances of current policy gradients, and analyzed the convergence of this algorithm. We demonstrated performance of the proposed algorithms via experimental results.

For future work, we will explore other RL settings for FRL with heterogeneous computations, including temporal difference (TD) based learning such as Q-learning and the Actor-Critic (AC) methods. These cases will be more challenging to study due to the complex structure of TD-based learning.

References

[1] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[2] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[3] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: Techniques, applications, and open challenges," *arXiv preprint arXiv:2108.11887*, 2021.

[4] X. Fan, Y. Ma, Z. Dai, W. Jing, C. Tan, and B. K. H. Low, "Fault-tolerant federated reinforcement learning with theoretical guarantee," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1007–1021, 2021.

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[6] Y. Li and D. Schuurmans, "Mapreduce for parallel reinforcement learning," in *European Workshop on Reinforcement Learning*, pp. 309–320, Springer, 2011.

[7] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, *et al.*, "Massively parallel methods for deep reinforcement learning," *arXiv preprint arXiv:1507.04296*, 2015.

[8] S. Zeng, T. Chen, A. Garcia, and M. Hong, "Learning to coordinate in multi-agent systems: A coordinated actor-critic algorithm and finite-time guarantees," in *Learning for Dynamics and Control Conference*, pp. 278–290, PMLR, 2022.

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

[10] F. Hairi, J. Liu, and S. Lu, "Finite-time convergence and sample complexity of multi-agent actor-critic reinforcement learning with average reward," in *International Conference on Learning Representations*, 2021.

[11] K. Zhang, Z. Yang, and T. Başar, "Decentralized multi-agent reinforcement learning with networked agents: Recent advances," *Frontiers of Information Technology & Electronic Engineering*, vol. 22, no. 6, pp. 802–814, 2021.

[12] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[13] E. Brunskill and L. Li, "Sample complexity of multi-task reinforcement learning," *arXiv preprint arXiv:1309.6821*, 2013.

[14] H. Jin, Y. Peng, W. Yang, S. Wang, and Z. Zhang, "Federated reinforcement learning with environment heterogeneity," in *International Conference on Artificial Intelligence and Statistics*, pp. 18–37, PMLR, 2022.

[15] S. Zeng, M. A. Anwar, T. T. Doan, A. Raychowdhury, and J. Romberg, "A decentralized policy gradient approach to multi-task reinforcement learning," in *Uncertainty in Artificial Intelligence*, pp. 1002–1012, PMLR, 2021.

[16] X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang, "Federated transfer reinforcement learning for autonomous driving," in *Federated and Transfer Learning*, pp. 357–371, Springer, 2023.

[17] P. Xu, F. Gao, and Q. Gu, "An improved convergence analysis of stochastic variance-reduced policy gradient," in *Uncertainty in Artificial Intelligence*, pp. 541–551, PMLR, 2020.

[18] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[19] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

[20] Technical Report. https://www.dropbox.com/s/c47cbw69przvdel/HC.pdf?dl=0.

[21] T. Chen, K. Zhang, G. B. Giannakis, and T. Başar, "Communication-efficient policy gradient methods for distributed reinforcement learning," *IEEE Transactions on Control of Network Systems*, vol. 9, no. 2, pp. 917–929, 2021.

[22] S. Khodadadian, P. Sharma, G. Joshi, and S. T. Maguluri, "Federated reinforcement learning: Linear speedup under markovian sampling," in *International Conference on Machine Learning*, pp. 10997–11057, PMLR, 2022.

[23] A. Khaled, K. Mishchenko, and P. Richtárik, "Tighter theory for local sgd on identical and heterogeneous data," in *International Conference on Artificial Intelligence and Statistics*, pp. 4519–4529, PMLR, 2020.

[24] S. M. Kakade, *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

[25] M. Papini, D. Binaghi, G. Canonaco, M. Pirotta, and M. Restelli, "Stochastic variance-reduced policy gradient," in *International conference on machine learning*, pp. 4026–4035, PMLR, 2018.

[26] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," *Advances in neural information processing systems*, vol. 28, 2015.