Communication-Efficient and Federated Multi-Agent Reinforcement Learning

Mounssif Krouka, Anis Elgabli, *Member, IEEE*, Chaouki Ben Issaid, *Member, IEEE*, and Mehdi Bennis, *Fellow, IEEE*

***Abstract*—In this paper, we consider a distributed reinforcement learning setting where agents are communicating with a central entity in a shared environment to maximize a global reward. A main challenge in this setting is that the randomness of the wireless channel perturbs each agent's model update while multiple agents' updates may cause interference when communicating under limited bandwidth. To address this issue, we propose a novel distributed reinforcement learning algorithm based on the alternating direction method of multipliers (ADMM) and "*over air aggregation*" using analog transmission scheme, referred to as A-RLADMM. Our algorithm incorporates the wireless channel into the formulation of the ADMM method, which enables agents to transmit each element of their updated models over the same channel using analog communication. Numerical experiments on a multi-agent collaborative navigation task show that our proposed algorithm significantly outperforms the digital communication baseline of A-RLADMM (D-RLADMM), the lazily aggregated policy gradient (RL-LAPG), as well as the analog and the digital communication versions of the vanilla FL, (A-FRL) and (D-FRL) respectively.**

***Index Terms*—Reinforcement learning, ADMM, analog communications, distributed optimization, policy gradient.**

I. Introduction

OWING to the strict and stringent requirements for 5G and beyond applications such as industry 4.0, network edge intelligence is of paramount importance [1]. One key challenge in these applications is how to optimize distributed systems where different entities (agents) communicate wirelessly in the same environment and share limited communication resources (e.g., limited bandwidth). The unique characteristics of multi-agent systems is that agents do not evolve alone in the environment and must take into account other agents who also perceive and act in the environment. Generally speaking, we can distinguish three scenarios: (i) fully cooperative where agents work together to maximize a common long-term return, (ii) fully competitive where the return of agents typically amounts to zero, and (iii) a combination of both where the environment includes both cooperative and competitive agents. In this work, we focus on the fully cooperative setting, which represents a great portion of the Multi-agent reinforcement learning (MARL) settings where multiple agents interact in a shared environment and collaborate towards maximizing their rewards. MARL entails sequential decision making procedures, where agents take different actions over sequences of time in a stochastic environment. In contrast to supervised learning where data distribution is stationary, the distribution used to sample data in the RL setting depends on time-varying policy parameters, which introduces non-stationarity and makes the problem more challenging. Many MARL algorithms were proposed to solve real-world problems such as spectrum sharing [2], 360 degree video streaming [3], multi-player gaming [4], and robot navigation [5].

Manuscript received December 1, 2020; revised June 24, 2021 and October 7, 2021; accepted November 17, 2021. Date of publication November 26, 2021; date of current version March 8, 2022. This work is supported by Academy of Finland 6G Flagship (grant no. 318927) and project SMARTER, projects EU-ICT IntellIoT and EUCHISTERA LearningEdge, and CONNECT, Infotech-NOOR, and NEGEIN. The associate editor coordinating the review of this article and approving it for publication was L.-C. Wang. *(Corresponding author: Mounssif Krouka.)*

The authors are with the Centre of Wireless Communications, University of Oulu, 90014 Oulu, Finland (e-mail: mounssif.krouka@oulu.fi; anis.elgabli@oulu.fi; chaouki.benissaid@oulu.fi; mehdi.bennis@oulu.fi).

Digital Object Identifier 10.1109/TCCN.2021.3130993

A. Related Works

1) Distributed Reinforcement Learning (DRL): The existence of a central parameter server (PS) capable of collecting information such as joint actions, rewards, and observations, as well as designing policies for all agents has been assumed in several works [6]–[9] when studying DRL. In DRL, one potential solution is centralized learning where at every step, each agent sends its raw data to a PS [10]. Upon receiving the data from all the agents, the PS calculates a global action and broadcasts it to the agents. In the context of cooperative learning, a centralized learning scheme significantly simplifies the analysis, enabling the use of tools developed for the single-agent RL case. However, when agents send the raw data to the PS, privacy is violated. Such information could be used by a malicious server/agent to infer some of the system's properties. Moreover, computing the global action at the PS introduces a high computation complexity. To ensure privacy, the work in [11] presents a new learning setting, namely federated learning (FL), whose goal is to allow learning of a global model using a distributed learning approach. At every training iteration, each agent performs one or a few local computations and uploads its model to the PS that aggregates the outputs of all the agents, performs a global computation, and produces a global model that is downloaded by all the agents. By alternating between local operation and global operation, eventually all the agents converge to a unique global model. FL preserves privacy since the raw data never leaves its origin. Moreover, the PS only performs model aggregation and the high complexity of computations is offloaded to the agents. However,

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

the frequent communication exchanges of information between the PS and the agents lead to a considerable overhead that turns to be the major bottleneck of the system performance in many RL scenarios. In [12], the authors propose a fully decentralized federated RL framework algorithm to solve a sequential clinical treatment problem. The work provides guarantees over the privacy of electronic medical records (EMRs) with the help of additively homomorphic encryption. However, the encrypted information is exchanged between the nodes through private authenticated channels which requires huge communication resources as the number of nodes increases. Hence, there is a need for communication-efficient RL algorithms.

2) Communication-Efficient MARL: Recently, several works investigated the communication-efficiency aspect in the context of MARL, aiming to reduce the number of communication rounds between the PS and the agents. The work in [13] proposes a federated reinforcement learning (FRL) architecture that aims to decrease the personalization time of multiple agents to their environment. The authors in [6] propose a communication-efficient algorithm that lazily aggregates the policy gradients by skipping some agents in some of the communication rounds while achieving the same convergence rate as policy gradient. In [14], the authors propose a randomized communication-efficient multi-agent actor-critic algorithm for DRL problem when a network of agents cooperate, using only communication with their local neighbors, to maximize a global average reward under the assumption of strongly connected graphs. The authors in [15] present a hierarchical distributed algorithm that reduces communication cost by differentiating the roles of the agents during the evaluation process. By doing so, they were able to choose various mixing matrices different from the often used doubly stochastic one in order to save communication by allowing unidirectional information exchange among agents. The authors show that their proposed algorithm achieves the same order of convergence rate as the state-of-the-art methods.

3) Communication Design of DRL: However, similar to any distributed system, DRL system suffers from interference caused by simultaneous transmission of different agents. To mitigate this problem, orthogonal channel allocation through digital transmission is used. Nonetheless, this approach is not communication-efficient and is not scalable since the number of required communication channels grows linearly with respect to the number of agents. In addition to that, when the agents upload their models, there is a chance for eavesdroppers or even the honest but curious PS to reconstruct the models if their trajectory is distinguishable, thus privacy can be violated by inferring training data. We note that the aggregation operation at the PS side only requires the sum of the local models without the need for separate individual models, which motivates the use of analog over-the-air schemes.

As opposed to digital transmission where streams of bits are sent, analog signals represent different elements by their amplitudes. The i^{th} element of the update is transmitted over a shared channel among all the agents. This drastically reduces the amount of required bandwidth, thereby improving communication efficiency. At the PS side, the received signal is perturbed by the channel, and the models are hidden behind the channel perturbations, thus lowering the chances of privacy infringement. However, the convergence towards the global model depends on the channels perturbations and fading. To overcome this, most of the work in analog schemes suggest to solve the problem by inverting the channel before transmitting the signal [16]–[20]. This limits the transmissions only for the channels that are greater than a certain threshold. However, this leads to another problem due to the power limitations at the agents' side and the effect of the heuristic choice of the threshold υ on the convergence. Moreover, if one agent has a good channel gain, its model updates will be unfolded to the PS, which leads to data exposure. Therefore, there is a need for a non channel-inversion algorithm to protect the privacy of the agents, to account for the power limitations at the agents' side, and to take into account the channel perturbations and their effect on the convergence.

B. Our Contributions

The major contributions of this work are summarized as follows.

- We propose A-RLADMM, a distributed and communication-efficient RL approach based on over the air aggregation and alternating direction method of multipliers (ADMM) [21], [22] that provides fast convergence and efficient use of the communication resources.
- A-RLADMM is an extension of the work in [23] to the MARL collaborative setting. However, A-RLADMM accounts for more challenges in terms of deployment in real-world decentralized RL setting.
- A-RLADMM allows the agents to transmit their updates without channel inversion and prevents the channel perturbations from affecting the convergence by integrating the effects of the channel in the algorithm update process.
- Simulations results show that our proposed algorithm significantly outperforms the digital communication version of A-RLADMM (D-RLADMM), the lazily aggregated policy gradient (RL-LAPG), the digital communication version of vanilla FL (D-FRL) as well as the analog version of FL (A-FRL) since it significantly reduces the number of communication uploads.

The remainder of this paper is organized as follows: we start by describing the DRL setting in Section II. We then provide in Section III a brief description of the idea of the Analog Federated ADMM (A-FADMM). In Section IV, we present our proposed framework, Analog Reinforcement Learning ADMM (A-RLADMM). In Section IV, we describe our simulation setting and numerical results for a navigation task. Finally, the paper ends with a summary of the main results.

II. Distributed Reinforcement Learning

In this section we briefly describe the key aspects of the proposed DRL problem and policy gradient methods.

A. Problem Statement

Consider a set $\mathcal{N} = \{1, \ldots, N\}$ of N distributed agents communicating with a PS. To model the sequential decision-making process that characterizes the multi-agent collaborative RL, Markov decision process (MDP) [24] defines a suitable mathematical structure where the actions are performed by the agents. Similar to the single-agent RL setting, the sextuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \nu, \{\ell_n\}_{n\in\mathcal{N}})$ describes the DRL problem under the MDP umbrella [25] such that:

- $\mathcal{S}$ is the state space that is shared by all agents.
- $\mathcal{A} = \prod_{i=1}^{N} \mathcal{A}_i$ is the joint action space where $\{\mathcal{A}_i\}_{i=1}^{N}$ are the local actions spaces.
- $\mathcal{P}$ denotes the transition kernels space with the mapping $\mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ denotes the distribution over space $\mathcal{S}$.
- $\gamma \in (0, 1)$ is the discounting factor.
- ν is the initial state distribution at time $t = 0$.
- $\ell_n : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the loss associated with agent n.

In addition to the sextuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \nu, \{\ell_n\}_{n\in\mathcal{N}})$, we have the joint stochastic policy given by $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N)$ where $\boldsymbol{\pi}_n : \mathcal{S} \to \Delta(\mathcal{A}_n)$ is the local policy for agent n that describes the conditional distribution of all the actions given the current state $\mathbf{s}$ for agent n. The policy $\boldsymbol{\pi}_n$ encodes the behavior of the agent n: in a state $\mathbf{s}_t \in \mathcal{S}$, the agent will choose the action $\boldsymbol{\pi}_n(\mathbf{s}_t) \in \mathcal{A}_n$, which causes the system to transition to a state $\mathbf{s}_{t+1}$. Taking into account the case in which the time is discrete in an infinite horizon, a policy $\boldsymbol{\pi}$ generates a trajectory $\mathcal{T} := \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots\}$ with $\mathbf{s}_t \in \mathcal{S}$ and $\mathbf{a}_t \in \mathcal{A}$. In the multi-agent collaborative RL setting, we aim to minimize the infinite-horizon discounted long-term loss aggregated over all agents, formally expressed as:

$$\min_{\boldsymbol{\pi}} \sum_{n=1}^{N} f_n(\boldsymbol{\pi}) \text{ with } f_n(\boldsymbol{\pi}) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(.|\boldsymbol{\pi})}\left[\sum_{t=0}^{\infty} \gamma^t \ell_n(\mathbf{s}_t, \mathbf{a}_t)\right], \tag{1}$$

Note that the expectation is taken over a random trajectory $\mathcal{T}$ generated given the policy $\boldsymbol{\pi}$. A trajectory $\mathcal{T}$ is generated according to the following transition probability

$$\begin{aligned}\mathbb{P}(\mathcal{T}|\boldsymbol{\pi}) &= \mathbb{P}(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots, |\boldsymbol{\pi}) \\ &= \nu(\mathbf{s}_0) \prod_{t=0}^{\infty} \boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t)\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t),\end{aligned} \tag{2}$$

where $\nu(\mathbf{s}_0)$ is the initial state probability and $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is the transition probability of $\mathbf{s}_{t+1}$ given the current state $\mathbf{s}_t$ when taking the action $\mathbf{a}_t$.

B. Policy Parametrization

In continuous state and action spaces, tabular RL approaches are no longer tractable, and the intended solver typically involves function approximation to alleviate the complexity. Hence, to overcome the difficulty of learning a function, policy gradient (PG) method [26], [27] was proposed. PG finds the policy that maximizes the system reward by searching in a restricted class of parametrized policies, i.e., policies parametrized by a parameter $\boldsymbol{\theta}$. A commonly used policy is the Gaussian policy which can be described as follows [28]

$$\boldsymbol{\pi}(\cdot|\mathbf{s}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{s}; \boldsymbol{\theta}), \boldsymbol{\Sigma}(\mathbf{s}; \boldsymbol{\theta})), \tag{3}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and the covariance matrix, respectively.

A policy $\boldsymbol{\pi}$ parametrized by $\boldsymbol{\theta} \in \mathbb{R}^d$ is denoted as $\boldsymbol{\pi}(\cdot|\mathbf{s}; \boldsymbol{\theta})$, or simply $\boldsymbol{\pi}(\boldsymbol{\theta})$. Hence, the problem in (1) can be rewritten in the parametrized form as follows

$$\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} f_n(\boldsymbol{\theta}) \text{ with } f_n(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\sum_{t=0}^{\infty} \gamma^t \ell_n(\mathbf{s}_t, \mathbf{a}_t)\right], \tag{4}$$

where we assume each function $f_n(\cdot)$ to be L_n-smooth. The smoothness assumption is standard when studying many non-convex optimization algorithms. In the remainder of the paper, we assume that the local function f_n is differentiable. Note that even if f_n is nondifferentiable but continuous, we could still replace the gradient ∇f_n by one of the subgradients ∂f_n, and the derivations still hold. To search for the best policy, the problem in (4) can be solved in an iterative manner. However, in practical distributed systems, each f_n needs to be handled by a single agent. This motivates the introduction of a global consensus formulation where the objective function becomes separable across the agents.

Before we describe our distributed consensus algorithm for multi-agent collaborative reinforcement learning over analog transmission which is extended from A-FADMM [23], we will briefly highlight the key steps of A-FADMM which was proposed for the supervised learning setting.

III. Analog Federated ADMM (A-FADMM)

In this section, we briefly highlight the key aspects of A-FADMM. Unlike in digital transmission, analog schemes adopt the notion that all the agents transmit the i^{th} element of their updates using the i^{th} sub-carrier. Therefore, the aggregated updates from the agents are assigned the same channel.

A. Static and Noise Free Channel

Under static and noise free communication channel, A-FADMM problem is formulated as

$$\min_{\boldsymbol{\Theta}, \{\boldsymbol{\theta}_n\}_{n=1}^{N}} \sum_{n=1}^{N} f_n(\boldsymbol{\theta}_n) \tag{5}$$

$$\text{s.t. } h_{n,i}\boldsymbol{\theta}_{n,i} = h_{n,i}\boldsymbol{\Theta}_i, \quad \forall n, i \tag{6}$$

where $\boldsymbol{\Theta}_i$ and $\boldsymbol{\theta}_{n,i}$ are the i^{th} elements of the global model $\boldsymbol{\Theta}$ at the PS and the local model $\boldsymbol{\theta}_n$ at agent n side, respectively, and $h_{n,i}$ is the wireless channel coefficient of agent n at subcarrier i. Note that, the only difference compared to the standard ADMM formulation [29]–[31] is introducing the channel in the constraint (6). Introducing the channel does not change the optimal solution of the problem. However, it integrates the channel into the updating steps of the primal and dual variables. Therefore, it allows aggregating the channel perturbed signals of all agents/workers and avoid

channel inversion which leads to several issues [16]–[19]. The Augmented Lagrangian of (5)-(6) is formulated as follows

$$\begin{aligned}\mathcal{L}_\rho\Big(\boldsymbol{\Theta}, \{\boldsymbol{\theta}_n\}_{n=1}^N, \boldsymbol{\lambda}\Big) &= \sum_{n=1}^{N} f_n(\boldsymbol{\theta}_n) + \sum_{i=1}^{d}\sum_{n=1}^{N} \boldsymbol{\lambda}_{n,i}^* h_{n,i}(\boldsymbol{\theta}_{n,i} - \boldsymbol{\Theta}_i) \\ &+ \frac{\rho}{2}\sum_{i=1}^{d}\sum_{n=1}^{N} \big|h_{n,i}\big|^2 \big(\boldsymbol{\theta}_{n,i} - \boldsymbol{\Theta}_i\big)^2, \end{aligned} \tag{7}$$

where $\boldsymbol{\lambda}_{n,i}^*$ is the complex conjugate of the dual variable $\boldsymbol{\lambda}_{n,i}$, d is the size of the model, and ρ is the constant that controls the mismatch between the i^{th} element of the local model $\boldsymbol{\theta}_{n,i}$ and the global model $\boldsymbol{\Theta}_i$ for agent n. At iteration $k+1$, every agent n updates its primal variable $\boldsymbol{\theta}_{n,i}^{k+1}$ by minimizing $\mathcal{L}_\rho(\boldsymbol{\Theta}^k, \boldsymbol{\theta}_n, \boldsymbol{\lambda}^k)$. Thus, $\boldsymbol{\theta}_{n,i}^{k+1}$ should satisfy

$$\nabla_i f_n\Big(\boldsymbol{\theta}_n^{k+1}\Big) + \boldsymbol{\lambda}_{n,i}^{k}{}^* h_{n,i} + \rho\big|h_{n,i}\big|^2\Big(\boldsymbol{\theta}_{n,i}^{k+1} - \boldsymbol{\Theta}_i^k\Big) = 0, \tag{8}$$

where $\nabla_i f_n(\cdot)$ is the i^{th} element in the gradient of $f_n(\cdot)$. Next, the PS collects the primal variables of the agents and updates the i^{th} element of global model $\boldsymbol{\Theta}_i^{k+1}$ by minimizing $\mathcal{L}_\rho(\boldsymbol{\Theta}, \boldsymbol{\theta}_n^{k+1}, \boldsymbol{\lambda}^k)$. Setting the derivative of $\mathcal{L}_\rho(\boldsymbol{\Theta}, \boldsymbol{\theta}_n^{k+1}, \boldsymbol{\lambda}^k)$ equal to zero results in the following update rule

$$\boldsymbol{\Theta}_i^{k+1} = \frac{1}{\sum_{n=1}^N \big|h_{n,i}\big|^2}\sum_{n=1}^{N}\Big(\big|h_{n,i}\big|^2\boldsymbol{\theta}_{n,i}^{k+1} + \boldsymbol{\lambda}_{n,i}^{k}{}^* h_{n,i}/\rho\Big). \tag{9}$$

For the PS to solve (9), in the uplink, every agent n transmits $h_{n,i}{}^*\boldsymbol{\theta}_{n,i}^{k+1} + \boldsymbol{\lambda}_{n,i}^{k}{}^*/\rho$ for the i^{th} element of its update to the PS for a duration of T seconds where $h_{n,i}{}^*$ the complex conjugate of $h_{n,i}$. Finally, every agent n locally updates the i^{th} element of its dual variable $\boldsymbol{\lambda}_{n,i}^{k+1}$ using

$$\boldsymbol{\lambda}_{n,i}^{k+1} = \boldsymbol{\lambda}_{n,i}^{k} + \rho h_{n,i}\Big(\boldsymbol{\theta}_{n,i}^{k+1} - \boldsymbol{\Theta}_i^{k+1}\Big). \tag{10}$$

B. Time-Varying and Noisy Channels

In the uplink, every agent n transmits $h_{n,i}^{k+1^*}\boldsymbol{\theta}_{n,i}^{k+1} + \boldsymbol{\lambda}_{n,i}^{k}{}^*/\rho$ for the i^{th} element of its update to the PS for a duration of $\bar{T}$ seconds. At the receiving side, the PS receives $\sum_{n=1}^N(|h_{n,i}^{k+1}|^2\boldsymbol{\theta}_{n,i}^{k+1} + \boldsymbol{\lambda}_{n,i}^{k}{}^* h_{n,i}^{k+1}/\rho) + z_i^{k+1}(t)$ for every second $t \in [0, \bar{T}]$, where $z_i^{k+1}(t) \sim \mathcal{CN}(0, N_0)$. Applying the matched filter, the received signals are integrated during $\bar{T}$ seconds, divided by $\bar{T}$, and sampled at $t = \bar{T}$. This results in $\sum_{n=1}^N(|h_{n,i}^{k+1}|^2\boldsymbol{\theta}_{n,i}^{k+1} + \boldsymbol{\lambda}_{n,i}^{k}{}^* h_{n,i}^{k+1}/\rho) + \hat{z}_i^{k+1}$ with the reduced noise $\hat{z}_i^{k+1} \sim \mathcal{CN}(0, N_0/\bar{T})$.

Correspondingly, the global model $\boldsymbol{\Theta}_i^{k+1}$ is updated as follows

$$\boldsymbol{\Theta}_i^{k+1} = \frac{\sum_{n=1}^N\Big(\Big|h_{n,i}^{k+1}\Big|^2\boldsymbol{\theta}_{n,i}^{k+1} + \boldsymbol{\lambda}_{n,i}^{k}{}^* h_{n,i}^{k+1}/\rho + \mathbf{Re}\{\hat{z}_i^{k+1}\}\Big)}{\sum_{n=1}^N\Big|h_{n,i}^{k+1}\Big|^2}, \tag{11}$$

where $\mathbf{Re}\{\cdot\}$ denotes the real part of a complex number.

In the downlink, every agent n receives the update from the PS as $h_{n,i}^{k+1}\boldsymbol{\Theta}_i^{k+1} + \hat{z}_{n,i}^{k+1}$. The received signal is multiplied by $h_{n,i}^{k+1^*}$ and $|h_{n,i}^{k+1}|^2\boldsymbol{\Theta}_i^{k+1} + h_{n,i}^{k+1^*}\hat{z}_{n,i}^{k+1}$ is used to fit into the primal update

$$\begin{aligned}\nabla_i f_n\Big(\boldsymbol{\theta}_n^{k+1}\Big) + \boldsymbol{\lambda}_{n,i}^{k}{}^* h_{n,i}^{k+1} + \rho\Big|h_{n,i}^{k+1}\Big|^2\Big(\boldsymbol{\theta}_{n,i}^{k+1} - \boldsymbol{\Theta}_i^k\Big) \\ -\rho\,\mathbf{Re}\Big\{h_{n,i}^{k+1^*}\hat{z}_{n,i}^{k+1}\Big\} = 0. \end{aligned} \tag{12}$$

The dual variable is updated follows as

$$\boldsymbol{\lambda}_{n,i}^{k+1} = \boldsymbol{\lambda}_{n,i}^{k} + \rho h_{n,i}^{k+1}(\boldsymbol{\theta}_{n,i}^{k+1} - \boldsymbol{\Theta}_i^{k+1}) - \rho\hat{z}_{n,i}^{k+1}. \tag{13}$$

IV. A-RLADMM Framework

In this section, we describe the extension of A-FADMM to solve the distributed collaborative multi-agent RL problem. We recall the problem in (1) and we recast it in a global consensus formulation.

$$\min_{\boldsymbol{\Pi}, \{\boldsymbol{\pi}_n\}_{n=1}^N} \quad \sum_{n=1}^{N} f_n(\boldsymbol{\pi}_n) \tag{14}$$

$$\text{s.t.} \quad \boldsymbol{\pi}_n = \boldsymbol{\Pi}, \quad \forall n \tag{15}$$

where $\boldsymbol{\Pi}$ is the global policy. Restricting our search over a class of parametrized policies, and considering over air aggregation and analog communication, the formulation in (14)-(15) can be rewritten in parametrized from as

$$\min_{\boldsymbol{\Theta}, \{\boldsymbol{\theta}_n\}_{n=1}^N} \quad \sum_{n=1}^{N} f_n(\boldsymbol{\theta}_n) \tag{16}$$

$$\text{s.t.} \quad h_{n,i}\boldsymbol{\theta}_{n,i} = h_{n,i}\boldsymbol{\Theta}_i, \quad \forall n, i. \tag{17}$$

The augmented Lagrangian of problem (16)-(17) is the same as (7).

However, in contrast to supervised learning, policy gradient method is used to solve the local problem at each agent. We first describe the algorithm under the assumption that the $h_{n,i}^{k+1} = h_{n,i}^k\ \forall k$, and later in the section we relax this assumption. The detailed steps are as follows.

At the outer iteration $k+1$, given the global model parameter $\boldsymbol{\Theta}^k$, agent n initializes a local variable $\boldsymbol{\phi}_{n,i}$ to $\boldsymbol{\theta}_n^k$, i.e., $\boldsymbol{\phi}_n^0 = \boldsymbol{\theta}_n^k$, then at the local iteration $j+1$, one PG iteration is performed

$$\begin{aligned}\boldsymbol{\phi}_{n,i}^{j+1} = \boldsymbol{\phi}_{n,i}^{j} - \alpha\Big[\nabla_i f_n\Big(\boldsymbol{\phi}_n^j\Big) + \boldsymbol{\lambda}_{n,i}^{k}{}^* h_{n,i}^{k+1} + \rho\Big|h_{n,i}^{k+1}\Big|^2 \\ \Big(\boldsymbol{\phi}_{n,i}^j - \boldsymbol{\Theta}_i^k\Big) - \rho\,\mathbf{Re}\Big\{h_{n,i}^{k+1^*}\hat{z}_{n,i}^{k+1}\Big\}\Big], \end{aligned} \tag{18}$$

where $\boldsymbol{\phi}_{n,i}^{j+1}$ is the i^{th} element of the primal variable of agent n computed after k global iterations and $j+1$ local iterations, and α is the learning rate. After J iterations, we set $\boldsymbol{\theta}_n^{k+1} = \boldsymbol{\phi}_n^J$.

Given the objective function for agent n

$$f_n(\boldsymbol{\phi}_n^j) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(.|\boldsymbol{\phi}_n^j)}\left[\sum_{t=0}^{\infty}\gamma^t\ell_n(\mathbf{s}_t, \mathbf{a}_{n,t})\right], \tag{19}$$

the gradient is computed as follows

$$\nabla_i f_n(\boldsymbol{\phi}_n^j) = \nabla_i\left(\int_\tau \mathbb{P}(\mathcal{T}|\boldsymbol{\phi}_n^j)\left[\sum_{t=0}^{\infty}\gamma^t\ell_n(\mathbf{s}_t, \mathbf{a}_{n,t})\right]d\tau\right)$$

$$= \left(\int_{\tau} \nabla_i \, \mathbb{P}(\mathcal{T}|\boldsymbol{\phi}_n^j) \left[\sum_{t=0}^{\infty} \gamma^t \ell_n(\mathbf{s}_t, \mathbf{a}_{n,t}) \right] d\tau \right). \quad (20)$$

Using the log-trick, we can write

$$\nabla_i \, \mathbb{P}\Big(\mathcal{T}|\boldsymbol{\phi}_n^j\Big) = \mathbb{P}\Big(\mathcal{T}|\boldsymbol{\phi}_n^j\Big) \, \nabla_i \log \mathbb{P}\Big(\mathcal{T}|\boldsymbol{\phi}_n^j\Big). \quad (21)$$

Substituting (21) in (20) we get

$$\begin{aligned} \nabla_i f_n(\boldsymbol{\phi}_n^j) &= \left(\int_{\tau} \mathbb{P}(\tau|\boldsymbol{\phi}_n^j) \, \nabla_i \log\big(\mathbb{P}(\mathcal{T}|\boldsymbol{\phi}_n^j)\big) \left[\sum_{t=0}^{\infty} \gamma^t \ell_n(\mathbf{s}_t, \mathbf{a}_{n,t}) \right] d\tau \right) \\ &= \left(\int_{\tau} \mathbb{P}(\mathcal{T}|\boldsymbol{\phi}_n^j) \left[\sum_{t=0}^{\infty} \nabla_i \log\big(\mathbb{P}(\mathcal{T}|\boldsymbol{\phi}_n^j)\big) \gamma^t \ell_n(\mathbf{s}_t, \mathbf{a}_{n,t}) \right] d\tau \right) \\ &= \mathbb{E}_{\mathcal{T} \sim \mathbb{P}\left(\cdot|\boldsymbol{\phi}_n^j\right)} \left[\sum_{t=0}^{\infty} \nabla_i \log\big(\mathbb{P}(\mathcal{T}|\boldsymbol{\phi}_n^j)\big) \gamma^t \ell_n(\mathbf{s}_t, \mathbf{a}_{n,t}) \right]. \end{aligned} \quad (22)$$

We recall the probability of generating a random trajectory $\mathcal{T}$ as

$$\begin{aligned} \mathbb{P}\Big(\mathcal{T}|\boldsymbol{\phi}_n^j\Big) &= \mathbb{P}\Big(\mathbf{s}_0, \mathbf{a}_{n,0}, \mathbf{s}_1, \mathbf{a}_{n,1}, \mathbf{s}_2, \mathbf{a}_{n,2}, \ldots, |\boldsymbol{\phi}_n^j\Big) \\ &= \nu(\mathbf{s}_0) \prod_{t=0}^{\infty} \boldsymbol{\pi}\left(\mathbf{a}_{n,t}|\mathbf{s}_t, \boldsymbol{\phi}_n^j\right) \mathbb{P}\big(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_{n,t}\big). \end{aligned} \quad (23)$$

Taking the logarithm and differentiating with respect to the i^{th} element, we get

$$\nabla_i \log\Big(\mathbb{P}\Big(\mathcal{T}|\boldsymbol{\phi}_n^j\Big)\Big) = \sum_{t=0}^{\infty} \nabla_i \log\Big(\boldsymbol{\pi}\left(\mathbf{a}_{n,t}|\mathbf{s}_t, \boldsymbol{\phi}_n^j\right)\Big), \quad (24)$$

where we have used the fact that $\nabla_i \log(\nu(\mathbf{s}_0)) = 0$ and $\nabla_i \log(\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_{n,t})) = 0$. Substituting (24) in (22), we get (25), shown at the bottom of the page.

Since the true dynamics of the MDP model may not be known, or the expectation in (25) is difficult to compute, the average is used as an unbiased estimate of the expected value. This estimate is computed using M batch trajectories that run over T time slots each. Hence, we get (26), shown at the bottom of the page, where $\mathcal{T} := \{\mathbf{s}_{n,0}^m, \mathbf{a}_{n,0}^m, \mathbf{s}_{n,1}^m, \mathbf{a}_{n,1}^m, \ldots, \mathbf{s}_{n,T}^m, \mathbf{a}_{n,T}^m, \ldots\}$ is the m^{th} T-slot episode (trajectory) generated at agent n.

When $h_{n,i}^{k+1} \neq h_{n,i}^k \ \forall k$, we perform a preliminary phase in order to cope with the channel changes and guarantee convergence of both the primal and dual variables. First, we obtain the dual variable $\bar{\boldsymbol{\lambda}}_{n,i}^k$ using (12) given the previous update $\boldsymbol{\theta}_{n,i}^k$. Next, we update the global variable $\bar{\boldsymbol{\Theta}}_i^k$ using both $\boldsymbol{\theta}_{n,i}^k$ and $\bar{\boldsymbol{\lambda}}_{n,i}^k$. After the preliminary phase is complete, every agent solves the local problem and updates $\boldsymbol{\theta}_{n,i}^{k+1}$ using (18). Equivalent to the A-FADMM update steps presented above, the global model $\boldsymbol{\Theta}_i^{k+1}$ is updated as follows:

$$\boldsymbol{\Theta}_i^{k+1} = \frac{\sum_{n=1}^{N} \left(\left| h_{n,i}^{k+1} \right|^2 \boldsymbol{\theta}_{n,i}^{k+1} + \left(\bar{\boldsymbol{\lambda}}_{n,i}^k \right)^* h_{n,i}^{k+1} / \rho + \mathbf{Re}\left\{ \hat{z}_i^{k+1} \right\} \right)}{\sum_{n=1}^{N} \left| h_{n,i}^{k+1} \right|^2}. \quad (27)$$

The dual variable are then updated as

$$\boldsymbol{\lambda}_{n,i}^{k+1} = \bar{\boldsymbol{\lambda}}_{n,i}^k + \rho h_{n,i}^{k+1} \left(\boldsymbol{\theta}_{n,i}^{k+1} - \boldsymbol{\Theta}_i^{k+1} \right) - \rho \hat{z}_{n,i}^{k+1}. \quad (28)$$

Finally, in order to account for the maximum power budget P, every agent n needs to calculates its power factor ω_n such that $(\omega_n^{k+1})^2 \sum_{i=1}^{d} |h_{n,i}^{k+1^*} \boldsymbol{\theta}_{n,i}^{k+1} + (\bar{\boldsymbol{\lambda}}_{n,i}^k)^*/\rho|^2 = P$. After sending ω_n to the PS, the latter decides $\omega^{k+1} = \min\{\omega_1^{k+1}, \omega_2^{k+1}, \ldots, \omega_N^{k+1}\}$ that is broadcasted to all the agents. After that, every agent transmits $\omega^{k+1}(h_{n,i}^{k+1^*} \boldsymbol{\theta}_{n,i}^{k+1} + (\bar{\boldsymbol{\lambda}}_{n,i}^k)^*/\rho)$ and the obtained updates at the PS are $\sum_{n=1}^{N}(|h_{n,i}^{k+1}|^2 \boldsymbol{\theta}_{n,i}^{k+1} + (\bar{\boldsymbol{\lambda}}_{n,i}^k)^* h_{n,i}^{k+1}/\rho) + \hat{z}_i^{k+1}/\omega^{k+1}$ after the division by ω^{k+1}. We note that the exchange of power factors ω_n^{k+1} and ω^{k+1} can be performed across separate control channels due to its negligible communication overhead [32]. The algorithm pseudocode is detailed in Algorithm 1.

V. Numerical Evaluation

To validate the performance of our algorithm presented in Section IV, we consider a multi-agent navigation task which we will describe in details. All the experiments were conducted using Python 3.6 on an Intel i5 CPU, 8 GB, 2.20 GHz, DDR3 RAM. In our simulation, we consider the local policy of every agent $\boldsymbol{\pi}_n(\boldsymbol{\theta}_n)$ to be parametrized by a deep neural network (DNN) that consists of three-fully connected layers with 30, 10, and 9 neurons, respectively. Rectified linear unit (ReLu) activation function is used in the first two layers, while the output layer is a softmax operator with the 9 neurons corresponding to the agents' movement actions, as described later in this section. As a consequence, the number of model elements is $d = 750$.

A. Problem Set-Up

As illustrated in Fig. 1, we consider a multi-agent collaborative navigation task comprising of a number of moving agents. We consider a discrete system where a set $\mathcal{N} = \{1, \ldots, N\}$ of N agents (robots, cars) are moving in a two-dimensional grid plane and aim to reach their target landmarks (destinations).

At every time step $t \in \{0, 1, 2, \ldots\}$, the position of the agent n is defined as $\mathbf{s}_n(t)$ from the location space $\mathcal{S}$ and can be written as the pair $\mathbf{s}_n(t) = \langle s_n^x(t), s_n^y(t) \rangle$ with

$$\nabla_i f_n\Big(\boldsymbol{\phi}_n^j\Big) = \mathbb{E}_{\mathcal{T} \sim \mathbb{P}\left(\cdot|\boldsymbol{\phi}_n^j\right)} \left[\sum_{t=0}^{\infty} \left(\sum_{\tau=0}^{t} \nabla_i \log\Big(\boldsymbol{\pi}\left(\mathbf{a}_{n,\tau}|\mathbf{s}_\tau, \boldsymbol{\phi}_n^j\right)\Big) \right) \gamma^t \ell_n\big(\mathbf{s}_t, \mathbf{a}_{n,t}\big) \right] \quad (25)$$

$$\hat{\nabla}_i f_n\Big(\boldsymbol{\phi}_n^j\Big) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{T} \sum_{\tau=0}^{t} \nabla_i \log \boldsymbol{\pi}\left(\mathbf{a}_{n,\tau}^m|\mathbf{s}_\tau^m, \boldsymbol{\phi}_n^j\right) \gamma^t f_n\big(\mathbf{s}_t^m, \mathbf{a}_{n,t}^m\big) \quad (26)$$

Algorithm 1 Analog Reinforcement Learning ADMM (A-RLADMM)

Input: $N, f_n(\boldsymbol{\theta}_n), \rho, K$, **Output**: $\boldsymbol{\theta}_n \quad \forall n$.
Initialization: $\boldsymbol{\theta}_n^{(0)}, \boldsymbol{\Theta}^{(0)}, \boldsymbol{\lambda}_n^{(0)}, \quad \forall n$.
while $k \leq K$ **do**
  PRELIMINARY PART:
  PS sends $h_{n,i}^{k+1}$ to all agents across all subcarriers.
  All agents in parallel:
  for $i = 1, \cdots, d$ **do**
    $\bar{\boldsymbol{\theta}}_{n,i}^{k} = \boldsymbol{\theta}_{n,i}^{k}$
    Find $(\bar{\boldsymbol{\lambda}}_{n,i}^{k})^{*}$ from (12) given $\bar{\boldsymbol{\theta}}_{n,i}^{k}$.
  end for
  Send $(h_{n,i}^{k+1})^{*}\bar{\boldsymbol{\theta}}_{n,i}^{k} + (\bar{\boldsymbol{\lambda}}_{n,i}^{k})^{*}/\rho$, $\forall i = 1, \cdots, d$ to PS.
  Parameter Server:
  Find $\bar{\boldsymbol{\Theta}}_{i}^{k+1}$ using (27) given $\bar{\boldsymbol{\theta}}_{n,i}^{k}$, $(\bar{\boldsymbol{\lambda}}_{n,i}^{k})^{*}$.
  Broadcast $\bar{\boldsymbol{\Theta}}_{i}^{k+1}$, $\forall i = 1, \cdots, d$ to all agents.
  MAIN PART:
  All agents in parallel:
  for $i = 1, \cdots, d$ **do**
    for $j = 1, \cdots, J$ **do**
      Find $\boldsymbol{\phi}_{n,i}^{j}$ using (18).
    end for
    $\boldsymbol{\theta}_{n,i}^{k+1} = \boldsymbol{\phi}_{n,i}^{J}$.
  end for
  Send $(h_{n,i}^{k+1})^{*}\boldsymbol{\theta}_{n,i}^{k+1} + (\bar{\boldsymbol{\lambda}}_{n,i}^{k})^{*}/\rho, \forall i = 1, \cdots, d$ to the PS.
  Parameter Server:
  Find $\boldsymbol{\Theta}_{i}^{k+1}$ using (27) given $\boldsymbol{\theta}_{n,i}^{k+1}$, $(\bar{\boldsymbol{\lambda}}_{n,i}^{k})^{*}$.
  Broadcast $\boldsymbol{\Theta}_{i}^{k+1}, \forall i = 1, \cdots, d$ to all agents.
  All agents in parallel:
  Update $\boldsymbol{\lambda}_{n,i}^{k+1}$ locally via (28), $\forall i = 1, \ldots, d$.
  $k \leftarrow k + 1$
end while

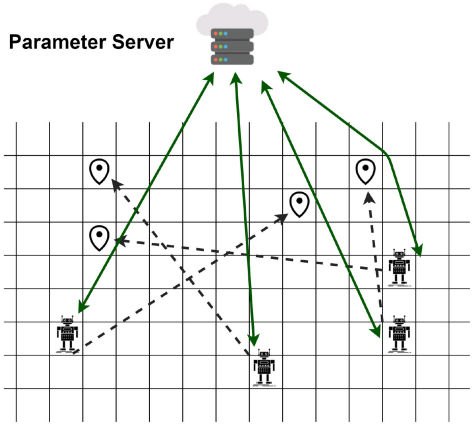


Fig. 1. Schematic illustration of agents moving in a grid-like plane.

$s_n^x(t)$ and $s_n^y(t)$ being the vertical and horizontal coordinates, respectively. The landmark of every agent is set at the position $\mathbf{q}_n(t) = \langle q_n^x(t), q_n^y(t) \rangle$ with horizontal coordinate $q_n^x(t)$ and vertical coordinate $q_n^y(t)$. In order to reach their landmarks, every agent n at time step t makes a transition from $\mathbf{s}_n(t)$ to $\mathbf{s}_n(t+1)$ by applying a movement decision $\mathbf{a}_n(t) = \langle a_n^x(t), a_n^y(t) \rangle$ where $a_n^x(t)$ and $a_n^y(t)$ represent the horizontal and vertical move of the agent, respectively. The action $\mathbf{a}_n(t)$ is chosen from the set of different movements which is defined as

$$\mathcal{A} = \{\langle 1,0\rangle, \langle 1,1\rangle, \langle 0,1\rangle, \langle -1,1\rangle, \langle -1,0\rangle, \langle -1,-1\rangle, \langle 0,-1\rangle, \langle 1,-1\rangle, \langle 0,0\rangle\}. \tag{29}$$

Consequently, we can write

$$\mathbf{s}_n(t+1) = \mathbf{s}_n(t) + \mathbf{a}_n(t), \tag{30}$$

After transitioning to position $\mathbf{p}_n(t+1)$, agent n receives a reward $R_n(t)$ that is collected at every time step t until the end of the training trajectory $\mathcal{T}$ (episode), i.e., $t = T$. In this MARL setting, we assume that the state is globally observable, i.e., the locations of other agents are available for every agent. Accordingly, the goal of every agent n is to arrive at its destination. This can be done by maximizing the difference between the current and the next distance to the landmark positions while avoiding collision with other agents. The state, action, and reward function for agent n are defined as follows.

- *Reward function:*

$$R_n(t) = \|\mathbf{q}_n - \mathbf{s}_n(t)\|_2 - \|\mathbf{q}_n - \mathbf{s}_n(t+1)\|_2 + \eta \mathbf{1}_{(\|\mathbf{q}_n - \mathbf{s}_n(t+1)\|_2 = 0)} - \beta \mathbf{1}_{(\sum_{m \neq n}^{N} \|\mathbf{s}_n(t+1) - \mathbf{s}_m(t+1)\|_2 = 0)}, \tag{31}$$

  where η is set as the reward when the agent reaches its landmark position, $\mathbf{1}_C$ is the indicator function for the collision, i.e., $\mathbf{1}_C = 1$ when agents collide, otherwise $\mathbf{1}_C = 0$. The parameter β is the collision penalty and $\|\cdot\|_2$ denotes the ℓ_2-norm. Note $\ell_n(\mathbf{s}_t, \mathbf{a}_{n,t}) = -R_n(t)$.
- *State:* The state is assumed to be globally observable, i.e., the agent n observes the positions of other agents $\mathbf{s}_m(t) \ \forall\, m \in N$.
- *Action:* A movement action $\mathbf{a}_n(t) \in \mathcal{A}$ is a probability vector of length A^n where every element is a global movement action for all the agents at time step t.

B. Network and Communication Environment

We use $N = 6$ agents with SNR $= 20$dB. We consider that the agents are allocated a total of 270 subcarriers. According to LTE standards [32], at every 1ms, every subcarriers provides $W_i = 15$ KHz of bandwidth. The channel realization is randomly generated by a Rayleigh fading distribution with zero mean and unit variance and is coherent during one global iteration. The rest of the simulation parameters are listed in Table I.

For A-RLADMM, every agent sends the i^{th} element of its variables using the i^{th} subcarrier. The number of uploads required to upload the variables depends solely on the number of assigned subcarriers, i.e., in our case we have $d = 750$, thus A-RLADMM requires $\lceil \frac{750}{270} \rceil = 3$ time slots to upload all the agents' variables per global iteration.

Under digital communication, D-RLADMM can alleviate the effects of the channel fading and noise using adaptive modulation and coding as addressed next. The PS receives the signal $y(t) = h(t)x(t) + z(t)$, where $x(t)$ is the transmitted signal $x(t)$, $h(t)$ is the channel gain, and $z(t)$ is the

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Batch trajectories M	5 batches
Trajectory duration T	60 seconds
Number of local iterations J	30 iterations
Mismatch parameter ρ	0.6
Learning rate α	0.01
Destination reward η	3
Collision penalty β	1
Discount factor γ	0.99
Channel inversion threshold υ	10^{-6}
AWGN power spectral density N_0	10^{-9} W/Hz

noise term with $z(t) \sim \mathcal{CN}(0, N_0 W)$, W is the bandwidth. The received signal $y(t)$ can be decoded without error only if the code rate $r(t)$ is smaller than or equal to the channel capacity $C(t)$, i.e., $r(t) \leq C(t)$. Thereafter, following the Shannon formula, the channel capacity is expressed as $C(t) = Wlog_2(1+SNR(t))$ where $SNR(t) = \frac{P|h(t)|^2}{N_0 W}$. Since the SNR is known at the transmitter, we can always adjust the code rate such that $r(t) = C(t)$ and guarantee error-free decoding of the received update. In D-RLADMM, the required number of uploading time slots depends on both the available bandwidth and the channel gain. Thus, considering that every element of the update variables consumes 32 bits, we can define $\hat{\mathscr{T}}_n$ as the minimum uploading time $\mathscr{T}_n$ for the agent n that satisfies the condition $\int_{t=1}^{\mathscr{T}_n} \sum_{i=1}^{i=270/N} r_{n,i}(t)dt \geq 32d$, where $r_{n,i}(t) = W_i log_2(1 + SNR_{n,i}(t))$. Consequently, the required time for uploading all the agent's updates is $\hat{\mathscr{T}} = \max\{\hat{\mathscr{T}}_1, \hat{\mathscr{T}}_2, \ldots, \hat{\mathscr{T}}_N\}$.

C. Baselines

We compare our proposed algorithm with the following benchmarks.

- *D-FRL:* The vanilla PG algorithm as illustrated in Fig. 3(a). The aim is to minimize $\frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{\Theta})$ by locally minimizing $f_n(\boldsymbol{\theta})$ at every agent and globally averaging the models at the PS. To solve this problem, every agent interacts with the environment and locally computes the gradient presented in (26) by running M batch episodes. After that, the model gradients are sent to the PS through orthogonal channels. Finally, the model gradients are aggregated at the PS side and the model is updated. This process continues until reaching the maximum channel uses. We notice that this baseline is not communication-efficient, since it requires huge bandwidth resources and it is vulnerable to model reconstruction attacks.
- *A-FRL:* As shown in Fig. 3(b), the analog communication version of D-FRL considers the over-the-air aggregation scheme. This is motivated by the fact that the PS only requires the sum of the models rather than the individual models. Henceforth, every agent n transmits an analog signal over a shared channel across all agents. After computing the gradients locally at the agents' side, every i^{th} element of the gradients is transmitted using the i^{th} subcarrier. We note that the obtained signal at the PS is perturbed by the channel fading. To mitigate this effect ,we consider the channel inversion technique where every agent multiplies its signal by the inverse of the channel gain before transmitting to the PS. Transmission is allowed only when $|h_{n,i}| > \upsilon$.
- *D-RLADMM:* D-RLADMM is the digital communication version of our proposed algorithm A-RLADMM which is illustrated in Fig. 3(c). The aim is to minimize $\frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{\Theta})$ by making use of the ADMM algorithm. Using digital transmission, agents are allocated separate channels in order to upload their updates to the PS. Every agent n solves a minimization problem in order to update its primal variable $\boldsymbol{\theta}_n$, then it sends it to the PS. The global variable $\boldsymbol{\Theta}$ is updated at the PS followed by the dual variable $\boldsymbol{\lambda}_n$ which is updated locally. We assume orthogonal bandwidth allocation, i.e., the total bandwidth is divided and allocated equally for every agent. We assume that every element in the model is transmitted using 32 bits.
- *RL-LAPG:* RL-LAPG is a PG-based method proposed in [6] named Lazily Aggregated Policy Gradient. It adaptively skips the policy gradient communication during iterations thus reducing the communication overhead without degradation of the learning performance. It allows the agents to transmit when the innovation between the evaluations of the current gradient and the old gradient is greater than a certain threshold.
- *ϵ-Greedy:* The epsilon-greedy policy tackles the exploration-exploitation tradeoff with RL algorithms. It takes a greedy action with a probability of $(1-\epsilon)$ and a an exploratory action with a probability of ϵ. This approach ensures the exploration of the action space. In our simulations, we consider $\epsilon = 0.1$.
- *x10:* Refers to the algorithm x when we use 10 times more subcarriers (10 times higher bandwidth than the default choice).

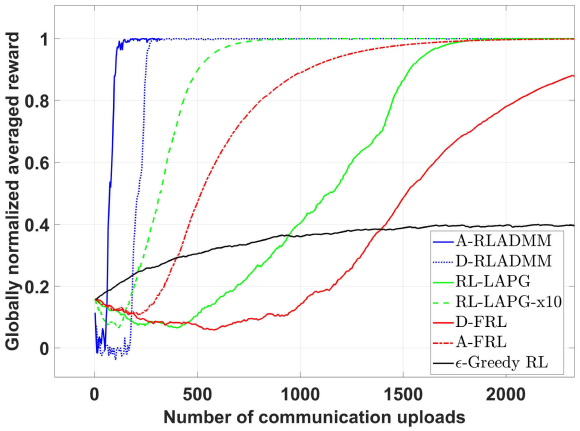


Fig. 2. Communication efficiency (Globally normalized averaged reward w.r.t number of communication uploads).

D. Results and Discussion

From Fig. 2, we see that A-RLADMM needs fewer number of communication uploads compared to the other baselines. RL-LAPG performs better than D-FRL since it requires less number of communication rounds to achieve a desirable

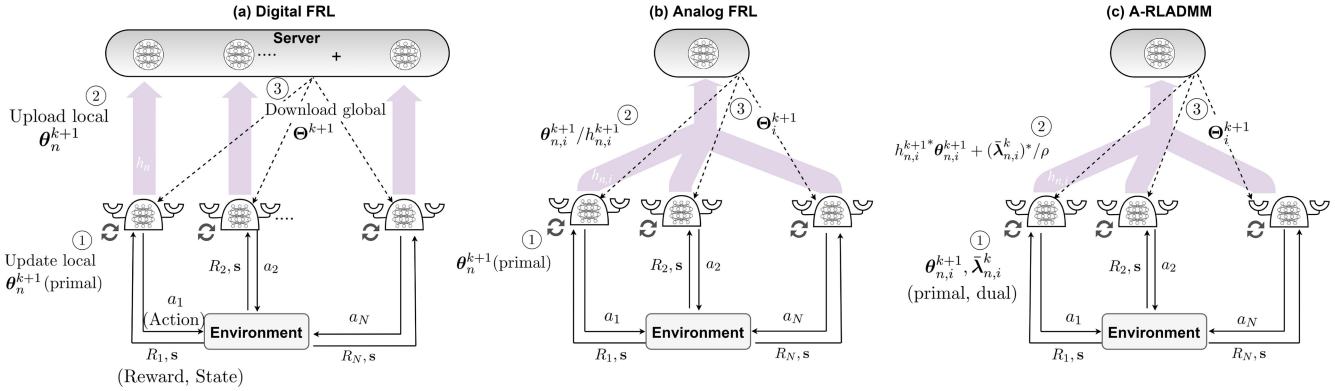


Fig. 3. Schematic illustrations of: (a) digital federated reinforcement learning (FRL), (b) analog FRL with channel inversion, (c) *analog-reinforcement learning ADMM (A-RLADMM)* without channel inversion.

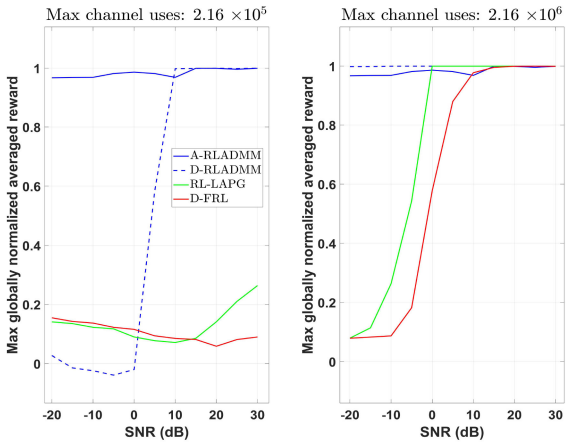


Fig. 4. Energy efficiency (maximum reward w.r.t SNR).

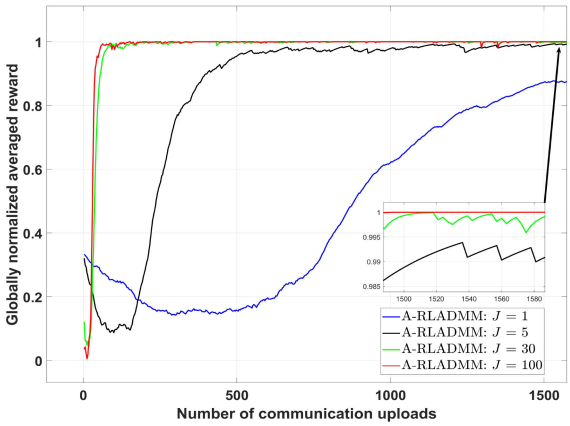


Fig. 5. Effect of the number of local iterations on the algorithm performance (Globally normalized averaged reward w.r.t number of communication uploads).

learning reward. On the other hand, we notice that A-FRL outperforms D-FRL. This is due to the fact that, for D-FRL, the PS requires a bandwidth scheduling algorithm to decide which agent to schedule on each subcarrier. Hence, splitting the available bandwidth between the agents results in higher number of communication uploads. The digital version of RLADMM (D-RLADMM) performs better than its digital counterparts, as well as the analog FRL (A-FRL). This observation shows the slow convergence of the first-order methods as both FRL and RL-LAPG require much higher number of communication uploads to reach their highest reward. We notice that ϵ-Greedy baseline performs the worst among all the baselines.

Fig. 4 illustrates the maximum attainable globally normalized averaged reward for both analog and digital versions of RLADMM, as well as D-FRL and RL-LAPG algorithms for different values of SNR under a constraint imposed on the maximum number of channel uses. The number of channel uses at time slot I is $CU = \sum_{i=1}^{I} D_i$, where D_i is the number of subcarriers at time slot i. We compare both algorithms under $CU_1 = 2.16 \times 10^5$ and $CU_2 = 2.16 \times 10^6$ channel uses. We see that A-RLADMM maintains a robust performance across the SNR values for both numbers of channel uses. At the low SNR regime, both D-FRL and RL-LAPG algorithms struggle to converge to the optimal policy and reach the maximum reward under both choices of the maximum number of channel uses. This is due to the high requirements in terms of communication resources. For CU_1, D-RLADMM achieves significantly higher reward compared to D-FRL and RL-LAPG at large SNR values. The reason is that D-RLADMM requires less number of time slots to converge. For A-RLADMM, the performance for both CU_1 and CU_2 cases is comparable at low SNR regime due to the fact that the received updates are very noisy, and increasing the number transmission time slots failed to improve the averaged reward thus the algorithm converges to a suboptimal policy.

While D-RLADMM suffers from scarce bandwidth resources for CU_1 case, the performance of D-RLADMM drastically improves as the maximum number channel uses is increased. For the high SNR regime, we notice that both A-RLADMM and D-RLADMM algorithms overcome the effect of the noise and achieve the maximum globally averaged reward for the CU_2 case. In addition to that, for CU_2, all the baselines achieve the maximum reward and converge to the optimal policy.

To study the sensitivity of the algorithm to the hyperparameters, we consider the effects of the number of local iterations and the number of batch episodes on the obtained globally averaged reward for our proposed algorithm, which

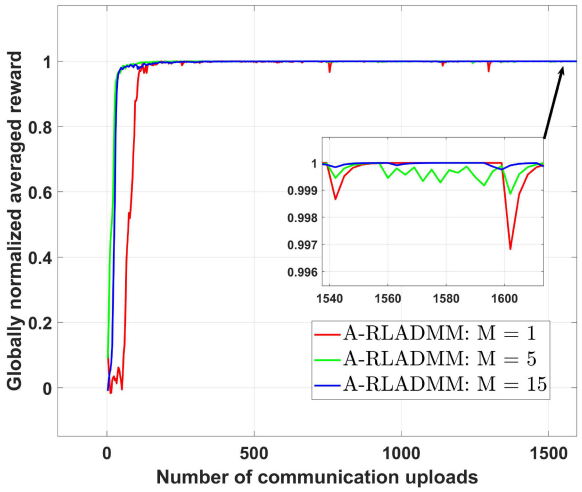


Fig. 6. Effect of the number of batch episodes on the algorithm performance (Globally normalized averaged reward w.r.t number of communication uploads).

are illustrated in Figs. 5 and 6, respectively. In Fig. 5, the number of local iterations affects the solution of (18). We notice that running for low number of iterations ($J \in \{1, 5\}$) degrades the performance of the algorithm as it requires very high number of communication uploads to converge. This is due to the fact that more GD steps leads to closer point to the local optima. As we increase the number of the local iterations, we notice an improvement in the performance. However, this comes at the expense of more computational time.

Equivalently, Fig. 6 reflects the approximation accuracy of the gradient expectation in (25) when estimated by the gradient estimate in (26) using different number of batch episodes. Using the values $M \in \{1, 5, 15\}$, we notice that we get a greater and more stable reward as we run for more episodes. These results validate the law of large numbers, as we can get closer to the expected value as we run for more trials. However, practically, the choice of M is restricted by the computation time and the nature of the problem in hand.

VI. Conclusion

This paper studies the DRL setting where several agents are communicating with a parameter server in a shared environment while also collaborating to maximize a global reward. To address the issue of wireless channel randomness under communication-constrained environments, we proposed an algorithm that embodies the wireless channel into the formulation of the ADMM method using analog transmission scheme, referred to as A-RLADMM. Experimental results show that our proposed algorithm significantly outperforms the digital communication version of A-RLADMM (D-RLADMM) as well as the analog and the digital communication version of the vanilla FL, (D-FRL) and (A-FRL) respectively. For future work, the impact of synchronization among the agents could be investigated. Moreover, other communication-efficient methods could be applied such as quantization and sparsification, especially for scenarios where huge model sizes are deployed.

References

[1] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[2] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Oct. 2019.

[3] M. Krouka, A. Elgabli, M. S. Elbamby, C. Perfecto, M. Bennis, and V. Aggarwal, "Cross layer optimization and distributed reinforcement learning approach for tile-based 360 degree wireless video streaming," 2020, *arXiv:2011.06356*.

[4] O. Vinyals *et al.*, "Grandmaster level in starcraft II using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350–354, Oct. 2019.

[5] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Auton. Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[6] T. Chen, K. Zhang, G. B. Giannakis, and T. Başar, "Communication-efficient distributed reinforcement learning," 2018, *arXiv:1812.03239*.

[7] J. Dibangoye and O. Buffet, "Learning to act in decentralized partially observable MDPs," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1233–1242.

[8] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2017, pp. 66–83.

[9] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[10] A. Khan, C. Zhang, D. D. Lee, V. Kumar, and A. Ribeiro, "Scalable centralized deep multi-agent reinforcement learning via policy gradients," 2018, *arXiv:1805.08776*.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[12] Z. Xue *et al.*, "A resource-constrained and privacy-preserving edge-computing-enabled clinical decision system: A federated reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9122–9138, Jul. 2021.

[13] C. Nadiger, A. Kumar, and S. Abdelhak, "Federated reinforcement learning for fast personalization," in *Proc. IEEE 2nd Int. Conf. Artif. Intell. Knowl. Eng. (AIKE)*, 2019, pp. 123–127.

[14] Y. Lin *et al.*, "A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, 2019, pp. 5562–5567.

[15] J. Ren and J. Haupt, "A communication efficient hierarchical distributed optimization algorithm for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. Real-World Sequential Decis. Making Workshop*, 2019, pp. 1–14.

[16] M. M. Amiri and D. Gündüz, "Over-the-air machine learning at the wireless edge," in *Proc. IEEE 20th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2019, pp. 1–5.

[17] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, Jan. 2020.

[18] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient radio resource allocation for federated edge learning," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2020, pp. 1–6.

[19] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020.

[20] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Communication-efficient split learning based on analog communication and over the air aggregation," 2021, *arXiv:2106.00999*.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," in *Foundations and Trends in Machine Learning*, vol. 3. Hanover, MA, USA: Now Publ., Jan. 2011, pp. 1–122. [Online]. Available: https://doi.org/10.1561/2200000016

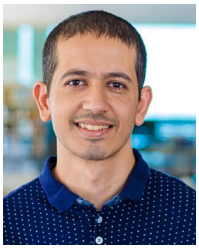
[22] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, "Parallel multi-block ADMM with o(1/k) convergence," *J. Sci. Comput.*, vol. 71, no. 2, pp. 712–736, 2017.

[23] A. Elgabli, J. Park, C. B. Issaid, and M. Bennis, "Harnessing wireless channels for scalable and privacy-preserving federated learning," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5194–5208, Aug. 2021.

[24] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: Wiley, 2014.
[25] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
[26] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2000, pp. 1057–1063.
[27] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *J. Artif. Intell. Res.*, vol. 15, no. 1, pp. 319–350, 2001.
[28] M. P. Deisenroth, *Efficient Reinforcement Learning Using Gaussian Processes*, vol. 9. Karlsruhe, Germany: KIT Sci. Publ., 2010.
[29] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM J. Optim.*, vol. 26, no. 1, pp. 337–364, 2016.
[30] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Trans. Signal Process.*, vol. 63, no. 2, pp. 482–497, Jan. 2015.
[31] M. Hong and Z.-Q. Luo, "On the linear convergence of the alternating direction method of multipliers," *Math. Program.*, vol. 162, nos. 1–2, pp. 165–199, 2017.
[32] *V15.2.0 Release 15TR 38.802 V14.1.0*, 3GPP Standard TS 38.211, Jun. 2017.

Mounssif Krouka received the B.Sc. degree from the Institute of Electrical and Electronics Engineering (IGEE ex-INELEC), Boumerdes University, Boumerdes, Algeria, in 2015, and the M.Sc. degree in wireless communications engineering from the University of Oulu, Oulu, Finland, in 2018, where he is a Doctoral Researcher with the Centre for Wireless Communications. His research interests include ultra-reliable low-latency communications, distributed optimization, resource scheduling, and machine learning.

Anis Elgabli (Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the University of Tripoli, Libya, in 2004, the M.Eng. degree from UKM, Malaysia, in 2007, and the M.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2015 and 2018, respectively. He is currently a Postdoctoral Researcher with the Centre for Wireless Communications, University of Oulu. His main research interests include heterogeneous networks, radio resource management, vehicular communications, video streaming, and distributed machine learning. He was a recipient of the Best Paper Award in HotSpot Workshop, 2018 (Infocom 2018) and the most JUFO points in 2020 at the Center of Wireless Communication, University of Oulu.



Chaouki Ben Issaid (Member, IEEE) received the Diplôme d'Ingénieur degree in economics and financial engineering from the Ecole Polytechnique de Tunisie in 2013, and the master's degree in applied mathematics and computational science and the Ph.D. degree in statistics from the King Abdullah University of Science and Technology in 2015 and 2019, respectively. He is currently a Postdoctoral Fellow with the Centre for Wireless Communications, University of Oulu. His current research interests include communication-efficient distributed learning and machine learning applications for wireless communication.

Mehdi Bennis (Fellow, IEEE) is a Professor with the Centre for Wireless Communications, University of Oulu, Finland, an Academy of Finland Research Fellow, and the Head of the Intelligent Connectivity and Networks/Systems Group (ICON). He has published more than 200 research papers in international conferences, journals, and book chapters. His main research interests are in radio resource management, heterogeneous networks, game theory, and distributed machine learning in 5G networks and beyond. He was a recipient of several prestigious awards, including the 2015 Fred W. Ellersick Prize from the IEEE Communications Society, the 2016 Best Tutorial Prize from the IEEE Communications Society, the 2017 EURASIP Best paper Award for the *Journal of Wireless Communications and Networks*, the all-University of Oulu Award for research, the 2019 IEEE ComSoc Radio Communications Committee Early Achievement Award, and the 2020 Clarviate Highly Cited Researcher by the Web of Science. He is an Editor of IEEE TRANSACTIONS ON COMMUNICATIONS and a Specialty Chief Editor for Data Science for Communications in the *Frontiers in Communications and Networks*.