Received 28 July 2023; revised 29 December 2023; accepted 13 February 2024.
Date of publication 20 February 2024; date of current version 28 February 2024.

The associate editor coordinating the review of this article and approving it for publication was H. Ko.

Digital Object Identifier 10.1109/TMLCN.2024.3368367

On Learning Generalized Wireless MAC Communication Protocols via a Feasible Multi-Agent Reinforcement Learning Framework

LUCIANO MIUCCIO[1] (Member, IEEE), SALVATORE RIOLO[1], (Member, IEEE), SUMUDU SAMARAKOON[2] (Member, IEEE), MEHDI BENNIS[2] (Fellow, IEEE), AND DANIELA PANNO[1] (Member, IEEE)

[1]Department of Electrical, Electronics and Computer Engineering (DIEEI), University of Catania, 95123 Catania, Italy
[2]Centre for Wireless Communications, University of Oulu, 90014 Oulu, Finland

CORRESPONDING AUTHOR: S. RIOLO (salvatore.riolo@unict.it)

The work of Luciano Miuccio and Daniela Panno was supported in part by the European Union through the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" under Grant PE00000001-program "RESTART." The work of Salvatore Riolo was supported by the Italian Ministero dell'Università e della Ricerca (MUR), Azioni IV.4 "Dottorati e contratti di ricerca su tematiche dell'innovazione," under Project PON R&I 2014-2020. The work of Sumudu Samarakoon and Mehdi Bennis was supported in part by the Research Council of Finland (former Academy of Finland) 6G Flagship Programme under Grant 346208 and in part by the European Union through the projects CENTRIC under Grant 101096379 and 6G-INTENSE under Grant 101139266.

ABSTRACT Automatically learning medium access control (MAC) communication protocols via multi-agent reinforcement learning (MARL) has received huge attention to cater to the extremely diverse real-world scenarios expected in 6G wireless networks. Several state-of-the-art solutions adopt the centralized training with decentralized execution (CTDE) learning method, where agents learn optimal MAC protocols by exploiting the information exchanged with a central unit. Despite the promising results achieved in these works, two notable challenges are neglected. First, these works were designed to be trained in computer simulations assuming an omniscient environment and neglecting communication overhead issues, thus making the implementation impractical in real-world scenarios. Second, the learned protocols fail to generalize outside of the scenario they were trained on. In this paper, we propose a new feasible learning framework that enables practical implementations of training procedures, thus allowing learned MAC protocols to be tailor-made for the scenario where they will be executed. Moreover, to address the second challenge, we leverage the concept of state abstraction and imbue it into the MARL framework for better generalization. As a result, the policies are learned in an abstracted observation space that contains only useful information extracted from the original high-dimensional and redundant observation space. Simulation results show that our feasible learning framework exhibits performance comparable to that of the infeasible solutions. In addition, the learning frameworks adopting observation abstraction offer better generalization capabilities, in terms of the number of UEs, number of data packets to transmit, and channel conditions.

INDEX TERMS 6G, multi-agent reinforcement learning, abstraction, generalization, feasibility, protocol learning.

I. INTRODUCTION

While 5G is rolled out globally and the standardization discussions for its future evolution are taking place, researchers in academia and industry have already been reflecting on visions, use cases, and disruptive key technologies for 6G systems [1]. Apart from new spectrum technologies, the support of simultaneous communications and sensing, and extreme connectivity requirements, it is expected that machine learning (ML) and artificial intelligence (AI) will play a defining role in the development of 6G networks end-to-end across its design, deployment, and operational phases [2], [3]. ML/AI-based network automation will play a vital role in simplifying network management and optimization, such that multiple elements like virtualized network function placement, slicing, mobility management, radio resource management, and spectrum sharing will natively depend on ML/AI.

However, despite these efforts, the integration of AI/ML into several areas of the radio access network (RAN) has not yet been considered. One of these areas regards the medium access control (MAC) protocols, where the heterogeneous services expected by the 6G networks impose significant challenges on meeting the diversified quality of service (QoS) requirements [4]. The standardization bodies started to face this problem, and in [5] several improvements to the conventional radio access procedures are provided without taking into account AI/ML. However, this process involves high costs in terms of time and effort [6]. Although standardization has not given any directives on how to address the protocol learning problem from an AI/ML perspective, it is clear that AI/ML can be used to design efficient communication protocols tailored for specific QoS requirements [7], [8].

Among the ML techniques, reinforcement learning (RL) can be instrumental in developing solutions for sequential decision-making problems, especially in robotics, games, distributed control, and economics [9]. The growing importance of RL arises from the fact that it does not require a pre-existing dataset, differently from the supervised and unsupervised learning. Conventional single-agent RL problems are usually modeled as Markov decision processes (MDPs), where the agent learns through trial-and-error interactions with the environment to learn an optimal behavior (i.e., policy) to reach a given goal. However, communication networks, such as cellular networks or wireless industrial networks [10], consist of multiple decision-makers that do not align with the conventional RL setting. Thus, analyzing them requires multi-agent reinforcement learning (MARL) frameworks.

Given the nature of communication protocols, they can be cast as cooperative MARL problems [11], [12], where all agents learn to coordinate in a communication efficient manner. Dealing with cooperative MARL is far more onerous than the single-agent case, since it introduces several additional issues, the most common of which are non-stationarity [13], scalability, and partial observability [14], [15]. To deal with these problems, recently several learning methods have been proposed. Among them, the most important are fully centralized learning (FLC), independent learning (IL), and centralized training with decentralized execution (CTDE), which differ in terms of the vision of the global state [16]. The first two learning methods totally resolve one of above issues at the expense of increasing the others, while the CTDE method provides a partial resolution of all three issues. For this reason, the latter is considered one of the most promising approaches for dealing with cooperative MARL settings [17].

In the CTDE method, during the centralized training phase, each agent computes its own policy exploiting not only its partial observation of the global state but also some global information (e.g., a centralized reward and/or the global state value) provided by the environment and/or a centralized entity (e.g., the critic). During execution, the system works in a decentralized fashion, i.e., each policy provides the agent with the optimal probability distribution among all actions given the partial observation.

A. RELATED WORK

Recent advances in deep RL and learning-to-communicate techniques [18] have led to the emergence of data-driven MAC protocols for the physical (PHY) [19], [20] and MAC layers [6], [21], [22], [23], [24], [25]. These techniques represent a cost-effective and flexible approach to boosting the performance of actual wireless networks. In this paper, we focus on the MAC layer. To implement a proper MAC protocol for a wireless access, the main building blocks are the data plane and the control plane. The latter defines what control information is available to the radio nodes and when it is available. As a result, the control messages lead the actions that can be taken in the data plane.

However, most current works treating the problem of emerging new MAC protocols with ML focus mainly on the data plane [23], [24]. To the best of our knowledge, the only ones that assess the problem of MAC protocol learning considering both the control and the data planes are [6], [21], [22], [25]. Therein, the authors examined the MAC protocol learning in an uplink radio network consisting of a single base station (BS) and several user equipments (UEs). The objective is to deliver MAC protocol data units (PDUs) to the BS through a shared data radio channel. One strong assumption is that the network provides for each UE a dedicated control channel, instead of a more realistic common channel control. The authors explore the MARL paradigm, where UEs are cast as learning agents trained to obtain optimal policies representing the MAC protocols.

In [6] the UEs adopt the Tabular Q-learning [26] as a learning algorithm, where UEs share the same copy of the Q-table and each UE updates its local copy exploiting its experience. As a result, UEs obtain the same shared policy. As regards the reward design, the authors define a very simple global reward function that can be calculated by any entity based only on the length of the episode, without exploiting any other information available.

In [21] each UE learns the MAC protocol by exploiting the actor-critic multi-agent deep deterministic policy gradient (MADDPG) [27]. Specifically, each agent has a local policy function, namely actor, together with a common centralized critic that estimates the value of the global state. The authors exploit the parameter sharing technique to update a unique central copy of the actor network for all the UEs. The global reward function is more sophisticated than [6], and can be computed knowing the buffer status of all UEs and the observation of the BS.

In our prior works [22], [25], we adopt an approach similar to [21], but the update of the policy function is carried out by means of the actor-critic multi-agent proximal policy optimization (MAPPO) algorithm [28]. Similar to [21], the computation of the reward function requires information from all the network entities.

B. MOTIVATION AND CONTRIBUTION

All the works described in the previous subsection adopt CTDE as learning scheme and consider a typical implementation where centralized training is carried out via simulation. The environment is an omniscient entity, without any placement in the network, that provides the learning agents (i.e., UEs) with their observations and global rewards. However, from the perspective of using RL-based MAC protocols in real scenarios (e.g., industrial environments), a learning framework where training can be performed only via simulation is not the best solution. Indeed, obtaining the best performance in the execution phase would require the use of an accurate simulation environment containing both a proper traffic model for the UEs and a channel model tailor-made for the considered scenario.

To overcome these severe limitations, the introduction of a new learning system that can be trained directly in a real-world scenario is of great value. Then, the implementation of the above state-of-the-art works on the field would require finding which physical entity collects the observations and computes the global reward, how it receives the information needed to get these parameters, and how it provides them to the learning agents. However, this design might require an intensive and continuous exchange of information, making the implementation not practical. Hereinafter, we term this issue as ‘‘feasibility’’, and define a learning system as ‘‘feasible’’ if it enables practical implementation of the training procedures in real scenarios. To the best of our knowledge, no other state-of-the-art works address this problem.

In addition, a further issue related to the adoption of RL-based MAC protocols is the robustness to changes in scenarios in terms of the number of UEs, traffic patterns, channel conditions, and so on. Indeed, the resulting MAC protocols would typically work properly as long as the network scenario maintains the same conditions in which they were trained on and fail to generalize outside of their training distributions. As a consequence, this aspect prevents the applicability of RL-based approaches to a dynamic wireless environment that would require re-training the learning framework every time it undergoes a change. The lack of generalization of RL-based MAC protocols had already been highlighted in [6], without providing any solution. We already addressed this problem in [22], where we provided a preliminary approach to improve the generalization performance in RL-based MAC protocols, introducing the concept of observation abstraction (OA).

In this paper, in view of applying the MARL paradigm for designing new wireless MAC protocols in real-world scenarios, we focus on both of the above issues. As regards feasibility, the straightforward solution would be to propose a feasible learning framework that removes any exchange of information among all network entities, i.e., UEs are isolated. This means that each learning agent (i.e., UE) exploits the local view of the network to compute its own reward and train its policy, accordingly. Although this seems like an optimal solution from the practical implementation point of view, a reward computed by every single agent would not lead to convergence in a cooperative problem [29]. Hence, the objective of the proposed feasible MARL is to reduce the exchange of information among the network entities, while still allowing the agents to learn the right behavior.

As regards the second issue, our idea is that the lack of generalization stems from the fact that during the training phase the agents learn their policies in the original observational space, which is specified for the scenario, instead of learning observation representations, which are invariant over multiple scenarios. The ability to learn from abstracted observations enables better generalization and robustness to previously unseen configurations. The notion of abstraction is based on learning invariance across configurations while filtering out irrelevant information [30]. Through this concept, the agents may reason about only the salient features of their scenario while ignoring what is irrelevant. Consequently, agents are able to solve considerably more complex problems than they would be able to without the use of abstraction.

Driven by the above analysis, in this paper, we propose a novel feasible wireless MAC communication learning framework that leverages abstraction to learn in a real-scenario new wireless MAC communication protocols with good generalization capabilities compared to infeasible state-of-the-art solutions. To the best of our knowledge, no other works in literature deal with the definition of a wireless MAC communication learning framework that considers both feasibility and generalization issues. The main contribution of this work can be summarized as follows.

1) We investigate the problem of learning a wireless MAC protocol that takes into account both the data and control planes to maximize the number of transmitted PDUs, while minimizing the time spent to do so, and reducing the number of collisions in the data plane.
2) Starting from the communication scenarios presented in [6], [21], [22], [25], we propose a new system model that overcomes the strong limitations of assuming UE-dedicated uplink control channels, by introducing a more realistic common control channel, where all UEs attempt their access procedure in a contention mode.
3) We cast the problem of learning the wireless MAC protocol as a feasible MARL problem. Toward this, we set local observations that each agent is able to compute without any additional information, define which entity calculates the reward and how it provides the rewards to the agents, and finally design a reward function that is based only on the information available at this entity.
4) For solving the feasible MARL problem, we present our multi-agent implementation of the PPO algorithm.

Therein, instead of adopting a conventional centralized critic, our implementation provides a decentralized critic network for each agent.

5) We introduce the concept of OA and leverage the concept of parameter sharing to boost the generalization performances of the learned MAC protocols with respect to the number of dPDUs and the number of learning agents, respectively. We present a new autoencoder (AE) architecture to derive the optimal OA function that maps the original observation space to the abstracted observation space. Through a rigorous analysis, we derive the cardinality of the original observation space.

We evaluate the performance of our feasible learning approach against the infeasible solutions [6], [22]. We integrate the OA to both the proposed feasible approach and the infeasible solution [22]. Simulation results show that learning schemes with OA exhibit slightly higher convergence times compared to those without OA, but better generalization performances in terms of both the number of UEs and dPDUs. The comparison between the infeasible solution [22] and the feasible solution both implementing the OA shows that the performance of the feasible solution is in line with the infeasible one.

The rest of the article is organized as follows. Background on RL and MARL is reported in the next section. The system model and the UEs-BS interaction as a feasible MARL are described in Section III and IV, respectively. The proposed learning approach integrating the implementation of the MAPPO is described in Section V. The learning of generalized policies is reported in Section VI. The performance evaluation is drawn in Section VIII. Finally, the conclusions are drawn in Section IX.

II. BACKGROUND ON RL

A. SINGLE-AGENT RL

Single-agent reinforcement learning (RL) is an area of machine learning concerned with the problem of one agent that learns the best behavior from experience, adopting a trial and error process. The thing it interacts with is called environment. The agent and the environment interact at each of a sequence of discrete time steps, denoted as $t \in \{0, 1, \ldots, t_{\text{end}} - 1\}$. At each time step t, the agent receives from the environment a representation of the environment's state, denoted as $s^t \in \mathcal{S}$, where $\mathcal{S}$ is the state space.

The agent interacts with the environment by performing an action a^t inside its own action set $\mathcal{A}$ according to its policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, where Δ is a probability space. In detail, $\pi(a^t \mid s^t)$ is the probability of selecting the action a^t given that the state s^t, and $\pi(\cdot \mid s^t)$ denotes the probability distribution among all actions given that the state is s^t. One time step later, as a consequence of the action a^t, the state s^t will transit to $s^{t+1} \in \mathcal{S}$ and the agent obtains from the environment a numerical reward $R^{t+1}(s^t, a^t, s^{t+1})$. The reward should quantify the effect (benefit or drawback) of the action performed by the agent. We define a trajectory τ of length t_{end} as a sequence of states and actions, i.e.:

$$\tau = (s^0, a^0, s^1, a^1, \ldots, s^{t_{\text{end}}-1}, a^{t_{\text{end}}-1}). \tag{1}$$

The discounted return following a given τ from time instant t is

$$G^t(\tau) = \sum_{k=t}^{t_{\text{end}}-1} \gamma^{\wedge(k-t)} R^{k+1}(s^k, a^k, s^{k+1}), \tag{2}$$

where $^{\wedge(\cdot)}$ is the exponentiation operator and $\gamma \in (0, 1)$ is the discount factor, which determines the impact of future rewards on the current decision. The objective is to find the optimal policy π^* so that

$$\pi^* = \arg\max_{\forall \pi} \left\{\mathbb{E}_\pi[G^t(\tau)]\right\}, \forall t \in \{0, 1, \ldots, t_{\text{end}}\}, \tag{3}$$

where $\mathbb{E}_\pi[G^t(\tau)]$ denotes the expected discounted return following a given policy π.

Finally, we introduce some additional definitions. For each state $s \in \mathcal{S}$, given a policy π, the state value function is:

$$V^\pi(s) = \mathbb{E}_\pi\left[G^t(\tau)\big|_{s^t=s}\right]. \tag{4}$$

Given the couple (s, a) and the policy π, the action-value function is

$$Q^\pi(s, a) = \mathbb{E}_\pi\left[G^t(\tau)\big|_{s^t=s, a^t=a}\right]. \tag{5}$$

For a given couple (s, a) and a policy π, the advantage function is

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s). \tag{6}$$

To obtain π^* several different approaches are available in the literature [26], [31], and can be classified into two main classes, namely, value-based and policy gradient. Value-based methods, such as Q-learning or state-action-reward-state-action (SARSA) [26], traditionally involve a deterministic policy, whereas policy gradient methods, such as REINFORCE or proximal policy optimization (PPO) [31], naturally generate a stochastic policy that can also converge towards a deterministic policy [26]. Given that the objective of this paper is to leverage RL augmented with communication to allow a novel MAC protocol to emerge, we aim to explore a large range of possible MAC protocols, considering the potential of both deterministic and stochastic protocols. This means that the most appropriate choice for the RL algorithm falls into the policy gradient category.

In policy gradient methods, the policy is modeled through a parameterized function with respect to θ, denoted as π_θ and is updated explicitly to maximize the expected discounted return by exploiting the policy gradient theorem [26]. Nevertheless, this method may introduce variability in estimating the policy gradient, potentially causing instability during training or slowing convergence. Several policy gradient methods are available in the literature, with PPO standing out as an algorithm explicitly designed to mitigate this issue [31].

Compared to REINFORCE, the PPO introduces two main features to reduce the gradient variance. First, the objective function for the policy introduces an estimate of the advantage function (6) denoted as $\hat{A}^t$. The advantage function estimates how much better or worse it is to take a specific action a^t in state s^t than following the policy's default behavior. Second, the PPO algorithm aims to reduce the high variance in the gradient estimate between the old policy and the new policy, by limiting the size of the policy update. To do so, it introduces the probability ratio at a generic time step t as

$$r^t(\theta) = \frac{\pi_\theta(a^t|s^t)}{\pi_{\theta_{\text{old}}}(a^t|s^t)}, \tag{7}$$

and defines the following objective function for the policy gradient:

$$L^t_{\text{PG,CLIP}}(\theta) = \min\left(r^t(\theta)\hat{A}^t, \text{clip}\left(r^t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}^t\right), \tag{8}$$

with ϵ representing a hyperparameter usually set to 0.2. The first term inside the minimum is the unclipped objective function that includes $\hat{A}^t$, while the second term clips the probability ratio $r^t(\theta)$ outside the interval $[1-\epsilon, 1+\epsilon]$. In addition to this, PPO combines the objective function (8) with an additional entropy regularization term, which balances exploration and exploitation. This term prevents excessive exploration strategies that may lead to high variance in gradient estimates, especially in environments with high variability or stochastic elements [32]. Therefore, the overall objective function of the policy is

$$L_{\text{PPO}}(\theta) = L^t_{\text{PG,CLIP}}(\theta) + cS\left[\pi_\theta\left(\cdot \mid s^t\right)\right], \tag{9}$$

where c is the entropy coefficient and $S\left[\pi_\theta\left(\cdot \mid s^t\right)\right]$ is the policy entropy.

However, to compute the objective function of the policy, we need to estimate the advantage function (6) because it depends on $V(s)$ and $Q(s, a)$, which both are unknowns. To solve this, a variety of different methods can be used [33]. Typically, the estimated advantage function is calculated as:

$$\hat{A}^t = G^t(\tau) - \hat{V}(s^t), \tag{10}$$

where $\hat{V}(s^t)$ is an estimate of the value of state s^t. As a consequence, the agent needs to learn $\hat{V}(s^t)$, for each state $s^t \in \mathcal{S}$. To do so, the estimated value state is modeled through a parameterized function with respect to ϕ, denoted as $\hat{V}_\phi$. The simplest method to estimate $\hat{V}(s^t)$ is called temporal difference (TD) 1-step approach [26], and the state value function is estimated by solving the following regression problem, whose objective function is:

$$L_{\text{VF}}(\phi) = \left[\hat{V}_\phi(s^t) - G^t(\tau)\right]^{\wedge(2)}. \tag{11}$$

This function is trained jointly with the policy, but unlike the policy function, it is used only during the training.

To complete the description of the PPO algorithm, we describe the training steps.

1) The policy parameter θ and the state value parameter ϕ are initialized at random.
2) The policy function π_θ is used to collect the trajectory τ (1).
3) For each time step t of τ, the following operations are carried out. First, the state value function $\hat{V}_\phi$ is used to provide $\hat{V}_\phi(s^t)$. Then, $G^t(\tau)$ in (2) and the advantage value $\hat{A}^t$ in (10) are calculated.
4) The policy function π_θ is updated by maximizing (9) and the state value function $\hat{V}_\phi$ is updated by minimizing (11).
5) Steps 2-4 are repeated until the convergence is reached.

B. MULTI-AGENT RL

Multi-agent RL (MARL) is an extension of RL for multi-agent systems, where N agents interact with the environment. Typical MARL problems are modeled as a multi-agent partially observable MDP (MPOMDP), defined by $\langle \mathcal{N}, \mathcal{A}_i, \mathcal{S}, T, \mathcal{O}_i, \pi_i, R, \gamma \rangle$, described in the following.

- $\mathcal{N}$ is the set of all agents.
- $\mathcal{A}_i$ is the action space for agent i. At each time step t, each agent $i \in \mathcal{N}$ performs an action $a_i^t \in \mathcal{A}_i$. We denote with $\mathbf{a}^t$ the joint action tuple $(a_1^t, a_2^t, \ldots, a_N^t)$.
- $\mathcal{S}$ is the global state space of the environment.
- $T(s^{t+1} \mid s^t, \mathbf{a}^t)$ denotes the transition probability from s^t to s^{t+1} given the joint action tuple $\mathbf{a}^t$.
- $\mathcal{O}_i$: observation space of the agent i. At time t, the environment provides to agent i a partial observation of the global state $s^t \in \mathcal{S}$, defined as $o_i^t \in \mathcal{O}_i$. Specifically, $o_i^t = h_i(s^t)$, where $h_i \colon \mathcal{S} \to \mathcal{O}_i$ is the mapping function for agent i.
- $\pi_i \colon \mathcal{O}_i \to \Delta(\mathcal{A}_i)$ is the policy of agent i. In detail, we denote with $\pi_i(a_i^t \mid o_i^t)$ the probability to take a_i^t when observing o_i^t, and with $\pi_i(\cdot \mid o_i^t)$ the probability distribution among all possible actions in $\mathcal{A}_i$ given observation o_i. We also define $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_N]$ as the policy vector.
- $R^{t+1}(s^t, \mathbf{a}^t, s^{t+1})$ is the reward received after transitioning from state s^t to state s^{t+1}, due to action $\mathbf{a}^t$. In general, in MARL problems, the reward can be divided into two categories: global and local. The global reward approach assigns the same global reward to all agents without distinguishing their contributions. The local reward approach provides different rewards to each agent based solely on its individual behavior.

We redefine a trajectory τ of length t_{end} as the following sequence of states and joint actions:

$$\tau = (s^0, \boldsymbol{a}^0, s^1, \boldsymbol{a}^1, \ldots, s^{t_{\text{end}}-1}, \boldsymbol{a}^{t_{\text{end}}-1}), \tag{12}$$

and the discounted return following a given τ from time instant t as:

$$G^t(\tau) = \sum_{k=t}^{t_{\text{end}}-1} \gamma^{\wedge(k-t)} R^{k+1}(s^k, \boldsymbol{a}^k, s^{k+1}). \tag{13}$$

The expected discounted return for a given policy vector $\boldsymbol{\pi}$ is denoted as:

$$\mathbb{E}_{\boldsymbol{\pi}}[G^t(\tau)]. \tag{14}$$

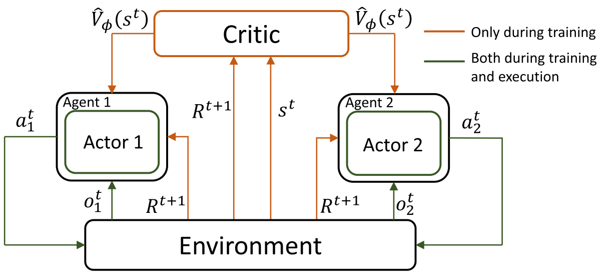


FIGURE 1. Conventional CTDE implementation with actor-critic structure.

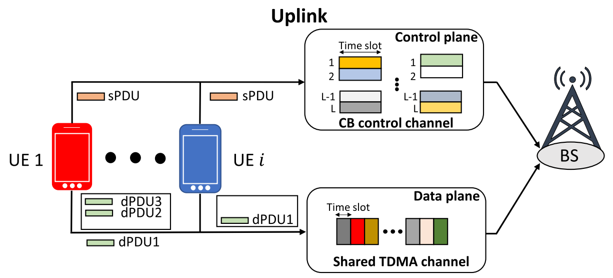


FIGURE 2. High-level depiction of the uplink system model.

The objective of a multi-agent RL algorithm is to find the optimal policy vector $\boldsymbol{\pi}^*$ so that

$$\boldsymbol{\pi}^* = \underset{\forall \boldsymbol{\pi}}{\arg\max} \left\{ \mathbb{E}_{\boldsymbol{\pi}}[G^t(\tau)] \right\}, \forall t \in \{0, 1, \ldots, t_{\text{end}}\}. \quad (15)$$

Due to the additional issues introduced by multi-agent settings, to obtain $\boldsymbol{\pi}^*$ we need to select both a proper learning method and an RL algorithm among the ones available in the literature. On the basis of the learning method chosen, the single-agent RL algorithm must be suitably adapted for supporting multi-agent settings.

In this paper, we choose as a learning method the CTDE, whose typical implementation adopts the actor-critic structure shown in Fig. 1 in the case of $N = 2$ agents. It consists of a single centralized critic network for all agents, and a decentralized actor network i for each agent $i \in \mathcal{N}$. Specifically, during the training procedure, for each time step t, the environment sends the global state s^t to the critic, and the partial observation $o_i^t = h_i(s^t)$ to each agent i. The critic network provides all agents with an estimation of the global state value $\hat{V}_\phi(s^t)$, whereas each actor network i provides the environment with an action $a_i^t \sim \pi_i\left(\cdot \mid o_i^t\right)$. As a result of the joint action $\boldsymbol{a}^t$, the state transits to s^{t+1} with probability $T(s^{t+1} \mid s^t, \boldsymbol{a}^t)$. Then, the environment computes the reward $R^{t+1}(s^t, \boldsymbol{a}^t, s^{t+1})$, and sends it to the agents and the critic. Each actor network utilizes both the reward and the estimated state value for updates, while the critic network is updated solely using the reward. During the execution, the system works in a decentralized fashion, and the critic stops its activity.

III. SYSTEM MODEL

Consider a radio network composed of a single BS serving a N homogeneous UEs needing to deliver P MAC PDUs to the BS.

A. UPLINK TRANSMISSION

In the uplink transmission, we consider both the data plane and the control plane, as shown in Fig. 2.

Data Plane: Each UE $i \in \mathcal{N}_{\text{UE}}$ has a data PDUs (dPDUs) storage capability, modeled as a buffer with first-in first-out (FIFO) policy, which contains at most Q dPDUs. We consider the following dPDU traffic model. The UL Tx buffer of each UE in $\mathcal{N}_{\text{UE}}$ is filled with $P \leq Q$ dPDUs at $t = 0$. UEs can send dPDUs using the same physical uplink shared channel (PUSCH) operating according to a time division multiple access (TDMA) scheme, with time slots of equal length. We suppose that each time slot t has a length such that each UE can transmit exactly one dPDU inside the time slot, including the propagation delay within the medium and other additional delays.

Since the uplink data plane acts as a shared TDMA channel, simultaneous transmissions result in collisions. As a consequence, the BS fails to correctly decode dPDUs. if more than one UE transmits its dPDU in the same time slot, then a collision occurs and the BS cannot correctly decode any received dPDU. Moreover, we assume that the data plane transmissions are modeled as a packet erasure channel, i.e., the dPDU sent by only one UE in a given time slot is correctly decoded with a fixed probability equal to 1-TBLER, where TBLER is the transport block error rate. Finally, in time slot t, UE i is able to carry out only one of the following actions in the data plane.

1) Transmit the first dPDU in the buffer.
2) Delete the first dPDU in its buffer.
3) Do nothing.

Control Plane: Each UE i can send a signaling PDU (sPDU) in the physical uplink control channel (PUCCH). The sPDU received by the BS is interpreted as a scheduling request (SR), i.e., a reservation request for the transmission of the dPDU in the next time slot. In [6], [21], [22], the control channels are assumed to be error-free and dedicated to each UE. In contrast, in this paper, we introduce a more realistic system model, where the UEs also compete for the transmission of SRs, i.e., the PUCCH is modeled as a contention-based (CB) TDMA channel. In detail, the time domain is divided into time slots, in the same way as the data plane channel, but each time slot provides L different signatures[1] [34]. When a UE transmits one sPDU in the CB control channel, the UE first selects randomly one of the available L signatures and then transmits the sPDU with the selected signature. If multiple UEs send their sPDU with identical signatures, then a collision in the control plane occurs. To summarize, for each time slot t, in the control

[1]The signature represents a specific pattern to be adopted inside the same time slot. It can be provided by exploiting random sequences (e.g., preamble sequences), by exploiting the frequency domain, or by dividing the time slot into mini-slots.

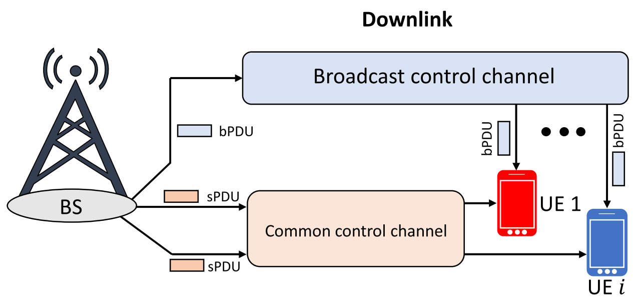


FIGURE 3. **High-level depiction of the downlink system model.**

plane, UE i is able to randomly select one signature and transmit the SR.

B. DOWNLINK TRANSMISSION

Since data are only transmitted by UEs in our scenario, the downlink shown in Fig. 3 has only the control plane.

Control Plane: The control channels are divided into two categories, namely, broadcast control channel (BCCH) and common control channel (CCCH). The BCCH is used by the BS to broadcast information such as cell and network identity. The broadcast PDUs (bPDUs) are transmitted in each time slot and allow the UEs to synchronize their frequencies to that of the transmitting base site, and to synchronize to the time slots of the TDMA frame. Other information useful for all the UEs can be encapsulated in the bPDUs. The CCCH is responsible for transferring control information from the BS to the specific UE i, such as a scheduling grant (SG) or an acknowledgement (ACK) message.

Similar to [6], [22], we assume that the BS is a MAC expert, i.e., it adopts a given MAC protocol for sending the sPDU to UEs. At each time slot t, the adopted MAC protocol acts as follows.

- The BS checks if the dPDU has been correctly received and decoded in the previous time step $t-1$. If so, then it sends an ACK message to the related UE that has made a successful data transmission.
- The BS checks the SRs received from all UEs in time slot $t-1$. If the BS received one or more SRs, then it chooses randomly one of the requesting UEs and responds with a scheduling grant (SG). We note that if the UE has made a successful data transmission concurrently with a SR, then the BS will ignore this SR and send only an ACK message to it.

IV. LEARNING A WIRELESS MAC PROTOCOL AS A FEASIBLE MARL PROBLEM

The goal of this paper is to find an optimal MAC protocol adopted by the UEs that maximizes the number of unique dPDUs successfully received by the BS, while minimizing the time spent to do so, and the number of collisions to improve energy consumption. In order to effectively reach this goal, we cast the UEs-BS interaction as an MPOMDP, as described in Section II-B.

The aim of this section is to define a feasible MARL problem that goes beyond the concept of an omniscient environment that gives the observations and the rewards to the agents. Toward this, we set local observations that each agent is able to compute without any additional information from the environment. Moreover, to implement a feasible reward assignment, we define i) which entity calculates the reward, ii) how it provides the rewards to the agents, and iii) a reward function that is based only on the information available at this entity. To map our system model to the feasible MPOMDP, we use the following notations and assumptions.

- The discrete time steps of the MARL correspond to the time slots of the communication channel.
- $\mathcal{N} = \mathcal{N}_{\text{UE}} \cup \mathcal{N}_{\text{BS}}$, where $\mathcal{N}_{\text{UE}}$ is the set containing N UEs (i.e., learning agents) and $\mathcal{N}_{\text{BS}}$ is the set containing the BS (i.e., expert agent).
- The UEs interact with the BS for the time of one episode that lasts t_{end} time slots. An episode ends when one of the following conditions is met. First, the P dPDUs of each and all of the N MAC learners have successfully reached the BS and the learners have removed them from their buffers. Second, the maximum length of one episode, denoted as $t_{\max}$, is reached.

The observation and action spaces for the BS and the UEs are described in the next subsections, respectively. Then, we define the global state and treat the reward assignment problem.

A. BASE STATION

The BS is a MAC expert agent that operates in the DL control plane following the protocol described in Section III. The observation space of BS is denoted as $\mathcal{O}_{\text{BS}}$. At time t, the BS observes $o_{\text{BS}}^t = (b_{\text{BS}}^t, b_{\text{BS}}^{t-1}, \ldots, b_{\text{BS}}^{t-M}) \in \mathcal{O}_{\text{BS}}$, where M is the considered memory length. We endow the agents with an internal memory that stores the history. The motivation for this memory is to address the problem of partial observability [35]. b_{BS}^t is a discrete variable that assumes $N+2$ possible values. In detail, $b_{\text{BS}}^t = 0$ if the data channel is idle, $b_{\text{BS}}^t = i$ if the data channel is detected busy with a single dPDU from UE i that is correctly decoded, and $b_{\text{BS}}^t = N + 1$ if there is non-decodable energy in the data channel (i.e., a collision occurred).

The action space for the BS is denoted as $\mathcal{A}_{\text{BS}}$. For each time slot t, the BS sends a control message m_i^t to each UE i, where $m_i^t \in \mathcal{M}_{\text{BS}} = \{0, 1, 2\}$. Here, $m_i^t = 2$ represents an ACK message that confirms a correct reception in the previous time slot $t-1$, $m_i^t = 1$ refers to a SG message, and $m_i^t = 0$ indicates that no signaling message has been sent to UE i.

Being the BS a MAC expert agent, it adopts the deterministic policy described in Section III that, on the basis of o_{BS}^t, provides into the CCCH the message vector $\boldsymbol{m}^t = [m_1^t, m_2^t, \ldots, m_N^t]$.

B. USER EQUIPMENTS

Due to the homogeneous nature of the UEs, each agent $i \in \mathcal{N}_{\text{UE}}$ has the same action space $\mathcal{A}_{\text{UE}}$. Agent i performs an action $a_i = (a_{i,\text{u}}, a_{i,\text{s}}) \in \mathcal{A}_{\text{UE}}$, where the data plane action $a_{i,\text{u}} \in \mathcal{A}_{\text{UE,u}} = \{0, 1, 2\}$ and the control plane action $a_{i,\text{s}} \in \mathcal{A}_{\text{UE,s}} = \{0, 1\}$. Specifically, $a_{i,\text{u}} = 1$ means that the agent transmits the first dPDU in its buffer (if any), $a_{i,\text{u}} = 2$ means it deletes the first dPDU in the buffer, and $a_{i,\text{u}} = 0$ to do nothing. For the control plane, $a_{i,\text{s}} = 1$ means to randomly select a signature and send an SG message, while $a_{i,\text{s}} = 0$ means do not transmit any signaling message.

Due to the homogeneous nature of the UEs, each agent also has the same observation space $\mathcal{O}_{\text{UE}}$. We define an observation by $o_i^t = (b_i^t, b_i^{t-1}, a_i^{t-1}, m_i^{t-1}, \dots, b_i^{t-M}, a_i^{t-M}, m_i^{t-M})$, where $b_i^t \in \mathcal{B} = \{0, 1, \dots, Q\}$ is the buffer status at time t. We note that the messages m_i^{t-k}, with $k = 1, \dots, M$, have been received from the BS inside the CCCH during the previous time slots. The other information, i.e., the previous actions a_i^{t-k} and the buffer statuses b_i^{t-k} are available at the UE side by default. As a consequence, at each time step t, the local observation o_i^t can be easily computed by each agent i without any additional information from the environment. Being the UEs learning agents, each UE $i \in \mathcal{N}_{\text{UE}}$ should learn a policy $\pi_i \colon \mathcal{O}_{\text{UE}} \rightarrow \Delta(\mathcal{A}_{\text{UE}})$.

C. GLOBAL STATE AND REWARD ASSIGNMENT

$\mathcal{S}$ is the global state space of the environment. At time t, the global state s^t is described by $s^t = (\mathbf{b}^t, \mathbf{b}^{t-1}, \mathbf{a}^{t-1}, \mathbf{m}^{t-1}, \dots, \mathbf{b}^{t-M}, \mathbf{a}^{t-M}, \mathbf{m}^{t-M}) \in \mathcal{S}$, where $\mathbf{b}^t = (b_1^t, b_2^t, \dots, b_N^t, b_{\text{BS}}^t)$ is a tuple containing all the buffer states for each agent i and the observation of the BS, and $\mathbf{a}^t = (a_1^t, a_2^t, \dots, a_N^t)$ is the joint action tuple.

$R^{t+1}(s^t, \mathbf{a}^t, s^{t+1})$ is the reward received after transitioning from state s^t to state s^{t+1}, due to the joint action $\mathbf{a}^t$. In generic MARL problems, at time step $t+1$, each agent receives either a global or a local reward value from the environment. In this paper, due to the cooperative nature of the task, we adopt a global reward, i.e., the environment provides to each agent the same reward $R^{t+1}(s^t, \mathbf{a}^t, s^{t+1})$, which quantifies the benefit of the joint actions performed by all the N agents. This design decision reflects the objective of optimizing the performance of the whole network, rather than that of individual agents. Here, we assume that the BS is the physical entity that broadcasts the reward to UEs by exploiting the BCCH.

The reward function should be properly designed to inform the agents the quality of their decisions/actions in terms of achieving the cooperative goal. The reward function is typically hand-designed, and so, it requires a high expertise and domain knowledge of the designer.

In [6], the authors define a very simple reward function generating -1 for each time step t until the episode ends. This reward design motivates MAC learners to finish the episode as quickly as possible. Although it is a simple reward that can be computed and sent to the UEs by the BS, it does not take into account the additional information that may help agents to obtain $\boldsymbol{\pi}^*$ in a shorter time. Indeed, in [6], the convergence time for a simple problem consisting of $N = 2$ learning agents and $P = 2$ is in the order of million episodes.

Conversely, in [21], [22], the reward function is more sophisticated, i.e., $R^{t+1} \in \{\rho, -1, -\rho\}$, where ρ is an integer > 1. Therein, the agents are rewarded with ρ if there exists a UE that has successfully transmitted a new dPDU, and with $-\rho$ if a UE deletes a dPDU that has not been received by the BS. In [22], the agents are also rewarded with ρ when a UE deletes its dPDU having previously transmitted it successfully. For all other situations, the conventional reward of $R^{t+1} = -1$ is granted for all agents to minimize the number of communication time slots. Despite the significant reduction in the convergence time [21], on the other hand, the reward computation requires an omniscient environment with access to the information available at both the BS and the buffer status of each UE. For this reason, we term this reward function as "infeasible reward".

In contrast, in this paper, we propose a new reward function that falls in between the two approaches. Specifically, the BS should be able to compute and send the reward to the UEs, as [6], but the reward should also contain more information to help learning agents to achieve convergence quickly, same as [21], [22]. Toward this, the proposed reward function is defined as

$$R^{t+1} = \begin{cases} +\rho & \text{if } b_{\text{BS}}^t \neq \{0, N+1\}, \\ -\rho & \text{if } b_{\text{BS}}^t = N+1, \\ -1 & \text{otherwise.} \end{cases} \tag{16}$$

Here, the agents are rewarded with ρ if a new dPDU has been received by the BS, and penalized with $-\rho$ if a collision occurred. The latter condition aims to minimize the number of collisions to improve the energy efficiency at the UEs during their packet transmissions. We note that this objective is not treated by the "infeasible reward" designed in [22]. In addition, differently from [21], [22], the reward does not contain any information from the buffer status of each UE and so, the BS can generate the reward without additional information. For this reason, we term this reward function as "feasible reward".

V. TRAINING ALGORITHM FOR SOLVING THE FEASIBLE MARL PROBLEM

In this section, we present our implementation of the training algorithm adopted for solving the MPOMDP, whose components have been described in the previous section. Then, we detail the training procedure for the entire framework.

A. MULTI-AGENT IMPLEMENTATION OF THE PPO ALGORITHM

As reported in Section II-A, we focus on the PPO algorithm and, in this subsection, we present our multi-agent implementation, termed as multi-agent PPO (MAPPO). We adopt the CTDE as the learning method and make appropriate implementation choices that differ from the conventional ones shown in Fig. 1.

The MAPPO algorithm leverages the single-agent PPO where both a policy function with parameter θ and a state value function with parameter ϕ are learned. Specifically, these functions are represented as two separated deep neural networks (DNNs): an actor network with parameter θ and a critic network with parameter ϕ. Each agent $i \in \mathcal{N}_{\text{UE}}$ has its actor (policy) network with parameters θ_i. Policy network i uses the partial observation o_i^t of the global state s^t as the input and results a categorical distribution over actions in the discrete action space $\mathcal{A}_{\text{UE}}$ as the output.

Compared to the conventional single-agent implementation described in Section II-A, our multi-agent implementation collects samples from N_{tr} trajectories instead of a single trajectory τ, to further mitigate the high variance issue in gradient estimates. As a consequence, the ith actor network aims to maximize the following objective function:

$$L_{\text{PPO},i}(\theta_i) = \frac{1}{N_{\text{tr}}}\sum_{j=1}^{N_{\text{tr}}}\mathbb{E}_{\tau_j}\left[L^t_{\text{PG,CLIP},i}(\theta_i) + cS\left[\pi_{\theta_i}(\cdot \mid o_i^t)\right]\right], \tag{17}$$

where $\mathbb{E}_{\tau_j}[\dots]$ indicates the average over the experience collected in the trajectory τ_j,

$$L^t_{\text{PG,CLIP},i}(\theta_i) = \min\left(r_i^t(\theta_i)\hat{A}_i^t,\, \text{clip}\left(r_i^t(\theta_i),\, 1-\epsilon,\, 1+\epsilon\right)\hat{A}_i^t\right), \tag{18}$$

$$r_i^t(\theta_i) = \frac{\pi_{\theta_i}(a_i^t|o_i^t)}{\pi_{\theta_{i,\text{old}}}(a_i^t|o_i^t)}, \tag{19}$$

and $\hat{A}_i^t$ is the estimated advantage function related to agent i.

As reported in Section II-A, for calculating $\hat{A}_i^t$, we also need to learn the estimated state value function. As shown in Fig. 1, the conventional approach provides a centralized critic network that takes as input the global state s^t. Adopting this centralized critic gives the impression to provide two main benefits: a centralized critic fosters "cooperative behavior", and also stabilizes (or speeds up) the training. However, as reported in [36], these intuitions have never been formally proven or empirically tested and, in general, the adoption of a centralized critic does not necessarily improve cooperation compared to the adoption of decentralized critics. Note that the global state s^t is not available at the UE. For this reason, the proposed MAPPO algorithm adopts a decentralized critic network for each agent i. Specifically, the input of the critic network for agent i is the partial observation o_i^t of the global state s^t and the output is denoted as $\hat{V}_{\phi_i}(o_i^t)$, which is a real value representing an estimated value of the local observation o_i^t. The critic network for agent i is trained with the aim of minimizing the following objective function:

$$L_{\text{VF},i}(\phi_i) = \frac{1}{N_{\text{tr}}}\sum_{j=1}^{N_{\text{tr}}}\mathbb{E}_{\tau_j}\left[\left(\hat{V}_{\phi_i}(o_i^t) - G^t(\tau_j)\right)^{\wedge(2)}\right], \tag{20}$$

where $G^t(\tau_j)$ is the same for all the learning agents $i \in \mathcal{N}_{\text{UE}}$, since it depends on the global states s^t and s^{t+1}, and on the joint action $\boldsymbol{a}^t$. The main advantage of this approach is that each agent needs to know only the cumulative rewards $G^t(\tau_j)$ as centralized information to train its own networks, rather than the global state s^t. Finally, the advantage value in (18) is estimated as

$$\hat{A}_i^t = G^t(\tau_j) - \hat{V}(o_i^t). \tag{21}$$

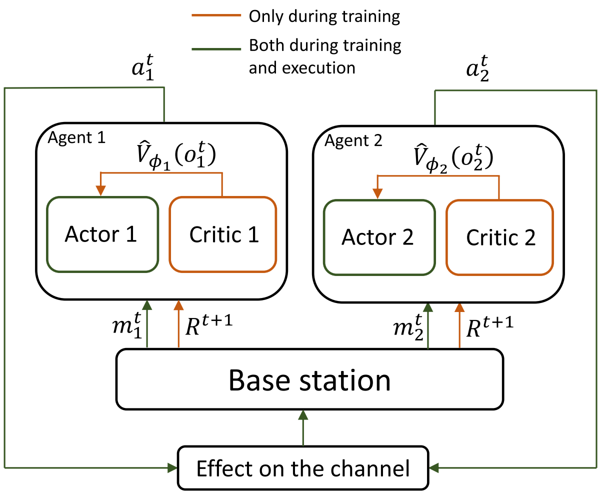


FIGURE 4. **Proposed CTDE-based learning framework.**

As a result of the choices made, the proposed learning framework is shown in Fig. 4, in the case of $N = 2$ agents. Each agent $i \in \{1, 2\}$ has its own pair of actor and critic networks with parameters θ_i and ϕ_i, respectively. During the training, at each time step t, the agent receives from the BS the control message m_i^t inside the CCCH. By collecting the messages received in the previous time slots jointly with the information available at the UE side by default, the agent easily computes its local observation o_i^t. This observation is used by the critic network i to provide the actor network i with an estimation of the local observation o_i^t value, whereas actor network i outputs $\pi_{\theta_i}(\cdot \mid o_i^t)$ and agent i performs action $a_i^t \sim \pi_{\theta_i}\left(\cdot \mid o_i^t\right)$. The BS observes the effect of joint action $\boldsymbol{a}^t$ on the radio channel, computes the reward R^{t+1}, and sends it to the agent by exploiting the BCCH.

Consequently, during the training, the proposed feasible implementation provides a reduced information exchange among the network entities. In fact, it is up to the BS to send information using only common control channels, without establishing dedicated control channels for the UEs. In turn, UEs need only to listen to this control information, without transmitting additional control data. During the execution, this exchange of information stops, and the system works in a decentralized fashion.

To better understand the advantages compared to the conventional CTDE implementation shown in Fig. 1, we evaluate the information exchange provided by that scheme. Fig. 1 shows a centralized critic that receives the global state s^t and the reward R^{t+1} from the omniscient environment, and provides each agent with an estimation of the global state value $\hat{V}_\phi(s^t)$. To implement that scheme in a real-world scenario, we need to identify the physical entity where the critic is located, how it receives the information needed to get both the global state and the reward, and how it provides $\hat{V}_\phi(s^t)$ to the learning agents.

The most convenient solution in terms of information exchange is to insert the critic inside the BS. As reported in Section IV-C, s^t is composed of the buffer states of all agents, the observation of the BS, the joint action tuple $\boldsymbol{a}^t$, and the message vector $\boldsymbol{m}^t$. Therefore, the BS should receive, at each time step t and from each agent i, the action a_i^t (involving both the data and control plane) and its buffer status. The cost in terms of information exchange is twofold. Firstly, it is necessary to reserve valuable radio resources in the form of control channels dedicated to each UE throughout the entire training phase. Secondly, for each time step, UEs should transmit their information on the dedicated control channel, even in cases of inactivity both on the control plane (i.e., $a_{i,\mathrm{s}} \neq 1$) and on the data plane (i.e., $a_{i,\mathrm{u}} \neq 1$). As regards how to send the output, $\hat{V}_\phi(s^t)$ must be broadcast to all agents, and BCCH could be used for this purpose.

In Section VIII, we compare the performance obtained by adopting the learning framework in Fig. 4 against the infeasible solution [22], which adopts the CTDE implementation in Fig. 1.

B. TRAINING DETAILS

The proposed training algorithm is presented in Algorithm 1 and depicted in Fig. 5. As shown, the updates of the agents' networks are carried out with a periodicity corresponding to N_{tr} episodes. The algorithm works as follows. The policy network and the estimated state value networks are initialized with random weights. The agents exploit their own policy network to generate an experience consisting of N_{tr} episodes as shown in Lines 5-17. In each episode j, the agents interact with the environment until either $t = t_{\max}$ or the cooperative task have been successfully completed. As a consequence, a complete trajectory lasts t_{end} time steps, where $t_{\mathrm{end}} \leq t_{\max}$ is in general not the same for each episode. At each time step t of the generic trajectory τ_j, the experience of each agent i is stored inside the time step tuple

$$T_i^t = \left(o_i^t, a_i^t, \pi_{\theta_{i,\mathrm{old}}}\left(a_i^t \mid o_i^t\right), R^{t+1}\right). \tag{22}$$

At the end of one experience (i.e., N_{tr} episodes), all the time step tuples T_i^t related to agent i have been inserted inside the related rollout list T_i. This list is used by agent i to compute the discounted returns $G^t(\tau_j)$ (see Lines 18-22). Next, to stabilize learning and improve convergence, z-score normalization is applied to the discounted returns (refer to Line 23). After that, the advantage values $\hat{A}_i^t$ are also computed (see Lines 24-30). Then, normalized rewards and advantage values are used to compute the losses (17) and (20). Finally, to further enhance training stability, adaptive learning rates based on the Adam optimizer are utilized for updating the networks (refer to Lines 31-33). After the update, a new experience is generated by exploiting the updated networks. The procedure is iteratively carried out until the convergence.

Algorithm 1 The proposed MAPPO algorithm

```
Inputs: Learning rate $\alpha$ and discount factor $\gamma$.
1:  Initialize the policy networks $\{\pi_{\theta_i}\}_{i=1}^N$ and the estimated state
    value networks $\{\hat{V}_{\phi_i}\}_{i=1}^N$ with random weights.
2:  repeat
3:      Set $\{\pi_{\theta_{i,\mathrm{old}}} \leftarrow \pi_{\theta_i}\}_{i=1}^N$
4:      Initialize the experience rollouts list $\{T_i\}_{i=1}^N$ as empty.
5:      for $j \in \{1, \ldots, N_{\mathrm{tr}}\}$ do
6:          Initialize the time step counter $t = 0$.
7:          repeat
8:              for each agent $i \in \mathcal{N}_{\mathrm{UE}}$ do
9:                  Observe $o_i^t$.
10:                 Extract $a_i^t \sim \pi_{\theta_i}(\cdot \mid o_i^t)$.
11:                 Execute the action $a_i^t$.
12:                 Compute $\pi_{\theta_{i,\mathrm{old}}}\left(a_i^j \mid o_i^j\right)$.
13:             end for
14:             Each agent $i$ receives $R^{t+1}$ and appends the time step
                tuple $T_i^t$ (22) inside $T_i$.
15:             Increase the time step counter.
16:         until the trajectory $\tau_j$ has been completed.
17:     end for
18:     for each trajectory $\tau_j$ with $j \in \{1, \ldots, N_{\mathrm{tr}}\}$ do
19:         for $t \in \{0, \ldots, t_{\mathrm{end}} - 1\}$ do
20:             Exploit $T_i$ to compute $G^t(\tau_j)$ in (13).
21:         end for
22:     end for
23:     Apply z-score normalization individually to the discounted
        returns $G^t(\tau_j)$ for all time steps $t$ within each trajectory $\tau_j$.
24:     for each trajectory $\tau_j$ with $j \in \{1, \ldots, N_{\mathrm{tr}}\}$ do
25:         for $t \in \{0, \ldots, t_{\mathrm{end}} - 1\}$ do
26:             for each $i \in \mathcal{N}_{\mathrm{UE}}$ do
27:                 Compute the advantage value $\hat{A}_i^t$ in (21).
28:             end for
29:         end for
30:     end for
31:     for each $i \in \mathcal{N}_{\mathrm{UE}}$ do
32:         Update the policy network $\pi_{\theta_i}$ and the estimated state
            value network $\hat{V}_{\phi_i}$ using the Adam optimizer with $\alpha$
            by maximizing $L_{\mathrm{PPO},i}$ in (17) and by minimizing $L_{\mathrm{VF},i}$
            in (20), respectively.
33:     end for
34: until convergence
```

In addition, we improve learning by running K gradient descent steps on the same experience. The benefit is to allow the networks to learn more from the same experience. This approach is typically avoided in vanilla policy gradient algorithms due to the risk of taking large steps on local samples. However, it is commonly adopted in PPO since the adopted clipped policy gradient objective function avoids the risk of destructively large policy updates.

VI. LEARNING GENERALIZED MAC PROTOCOLS

So far, we focused on the feasibility of the proposed MAC protocol learning approach. Another relevant issue is the generalization capability, which is addressed in this section. In the context of RL in general, the policies are typically overfitted to the scenario in which they have been trained,

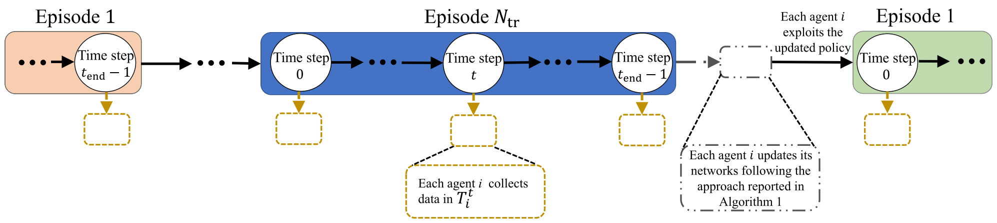

Episode 1
Time step $t_{\text{end}}-1$
Episode N_{tr}
Time step 0
Time step t
Time step $t_{\text{end}}-1$
Each agent i exploits the updated policy
Episode 1
Time step 0
Each agent i collects data in $\mathcal{T}_i^t$
Each agent i updates its networks following the approach reported in Algorithm 1

FIGURE 5. Temporal dynamics of the training algorithm.

in terms of the observation/state space. Moreover, in the case of MARL, the policies can be also overfitted to the number of learning agents.

Concretely, the problem of overfitting to the observation/state space can be explained as follows. Consider that the policy networks of the agents have been trained in the observation space $\mathcal{O}_{\text{UE}}$. As a consequence, the agents are able to make their sub-optimal decision as long as their local observations belong to $\mathcal{O}_{\text{UE}}$. For instance, if the training has been carried out with $P = 2$, the best test performance can be observed with a system with $P = 2$. The testing scenarios with a larger number of dPDUs (e.g., $P = 10$) are likely to fail due to unseen observations. In other words, the policies fail to generalize (i.e., there is no guarantee that their policies will be able to make the most appropriate decisions) because they have not been trained to handle these inputs.

As regards the overfitting to the number of learning agents, it is an issue specific to MARL. Consider that the training has been carried out with $N = 2$ agents, which corresponds to two policies π_1 and π_2. If we want to test the functioning of the protocol with a larger number of agents (e.g., $N = 4$ agents), the first issue is to choose which of the learned policies should be adopted by the other agents. One straightforward solution to overcome this is to force agents 3 and 4 to adopt one of the learned policies chosen at random. Nevertheless, there is no guarantee that the policy customized for being used by one single agent will work also if another agent adopts at the same time the same policy. For instance, π_1 may force agent 1 to transmit only on even slots and π_2 forces agent 2 on odd slots. This type of policy is clearly overfitted on two agents. Indeed, if agent 3 and 4 adopt at the same time π_1 and π_2, respectively, then the MAC protocol fails to work.

The common solution to work with a new scenario (characterized by a different number of dPDUs to be transmitted and/or a different number of UEs) is to perform a new training phase (known as retraining) in the new scenario. However, performing a new training phase involves an exchange of information, energy consumption, and time, which is of large impact in real deployment scenarios (not a simulated interaction). Moreover, if we consider retraining the policies by adopting a larger number of dPDUs to be transmitted, then the training time will be typically larger since the optimal policies will be learned in a larger observation space $\mathcal{O}_{\text{UE}}$. To understand the impact of the buffer dimension on the cardinality of $\mathcal{O}_{\text{UE}}$, we refer readers to Appendix A containing the analytical computation of the cardinality of $\mathcal{O}_{\text{UE}}$. The same consideration is valid when the number of learning agents increases.

In this paper, we aim to boost the generalization performances of the learned policies with respect to both the number of dPDUs and the number of learning agents. For doing so, we introduce below the concept of observation abstraction and the concept of parameter sharing.

A. GENERALIZATION TO THE NUMBER OF dPDUs

In this subsection, we introduce the concept of observation abstraction that facilitates generalization to the number of dPDUs. The conventional policies are defined in the original observation space, which typically contains redundant and noisy observations. This affects the ability to generalize respect to different tasks (e.g., the number of dPDUs). The observation abstraction aims to overcome this problem. Each learning agent (i.e., UE) will learn a policy in a reduced-size set containing abstracted observations, instead of in the original observation space $\mathcal{O}_{\text{UE}}$.

The new abstracted observation set can be defined by means of an observation abstraction (OA) function that clusters and aggregates similar observations from the original $\mathcal{O}_{\text{UE}}$. The policy defined in the abstracted observation space is termed abstracted policy. When we test these learned abstracted policies with a number of dPDUs that differs from the one the policies have been trained in, the OA function maps them into the same abstracted observations handled during the training. As a consequence, the abstracted policies will be able to make appropriate decisions. In the following, we summarize how OA is adopted within the proposed learning protocol.

To develop efficient abstracted policies, we first define the abstracted MPOMDP (AMPOMDP) by four components.

i) The original MPOMDP presented in Section IV.
ii) An abstraction of $\mathcal{O}_{\text{UE}}$ denoted as $\mathcal{O}^{(\psi)}$.
iii) The OA function $\psi : \mathcal{O}_{\text{UE}} \rightarrow \mathcal{O}^{(\psi)}$ that maps each observation $o_i^t \in \mathcal{O}_{\text{UE}}$ into an abstracted observation $o_i^{t,(\psi)} \in \mathcal{O}^{(\psi)}$.
iv) The abstracted policies operating on $\mathcal{O}^{(\psi)}$ denoted as $\pi_i^{(\psi)} : \mathcal{O}^{(\psi)} \rightarrow \Delta(\mathcal{A}_{\text{UE}}), \forall i \in \mathcal{N}_{\text{UE}}$.

Differently from [22], where the omniscient environment provides abstracted observations to the agents, in this paper, we present a feasible AMPOMDP that works as follows. We consider that each agent $i \in \mathcal{N}_{\text{UE}}$ knows the OA function ψ, because it is to be broadcast by the BS. At each time step

t, each agent i computes its local observation $o_i^t \in \mathcal{O}_{\text{UE}}$. This original observation is passed through ψ yielding the abstracted observation $\psi(o_i^t) = o_i^{t,(\psi)}$. Then, agent i uses $o_i^{t,(\psi)}$ to take the action a_i^t according to its abstracted policy $\pi_i^{(\psi)}$. At time step $t+1$, the BS computes R^{t+1} on the basis of its local observation and provides it to the UEs by means of the BCCH.

The objective of the AMPOMDP is to learn the optimal abstracted policy vector $\boldsymbol{\pi}^{(\psi)} = [\pi_1^{(\psi)}, \dots, \pi_N^{(\psi)}]$ so that

$$\boldsymbol{\pi}^{*(\psi)} = \underset{\forall \boldsymbol{\pi}^{(\psi)}}{\arg\max} \left\{ \mathbb{E}_{\boldsymbol{\pi}^{(\psi)}}[G^t(\tau)] \right\}, \forall t \in \{0, 1, \dots, t_{\text{end}}\}. \tag{23}$$

To solve the AMPOMDP, we propose to adopt Algorithm 1 with the following three changes. First, Line 9 is substituted with

9: Observe o_i^t and calculate $o_i^{t,(\psi)} = \psi(o_i^t)$.

Second, $\pi_{\theta_i}^{(\psi)}$ substitutes π_{θ_i}. Third, $o_i^{t,(\psi)}$ substitutes o_i^t, except for Line 9.

In summary, given the OA function ψ, the AMPOMDP represents a slight variation of the original MPOMDP. The challenge of finding the appropriate OA function for the proposed AMPOMDP is discussed in the next section.

B. GENERALIZATION TO THE NUMBER OF UEs

In this subsection, we aim to generalize to the number of UEs. Since the learning agents are homogeneous (i.e., they share the same action and observation space), we adopt the parameter sharing approach [18] for both the policy and estimated observation value function networks. This approach permits the agents to train one common shared policy network and one common shared estimated observation value network. By exploiting the parameter sharing technique, even if the training has been carried out with $N = 2$ agents, we can test the functioning of the protocol with a larger number of agents (e.g., $N = 4$ agents) that will adopt the same shared policy, which has been trained to be used by more agents at the same time.

Despite the benefits of generalization to the number of learning agents, the parameter sharing is a generic approach, and there are no guidelines on how to implement it in practice [18]. To overcome this gap, we provide a possible implementation to exchange and update the weights of the shared networks, consisting of the parameters θ and ϕ. In Fig. 5 we show that the update of the agents' networks is performed at the end of N_{tr} episodes, and also by adopting the parameter sharing approach, the update periodicity remains the same.

To carry out this update among all UEs, we describe a simple distributed mechanism that follows a round process. A high-level depiction in the case of two agents is shown in Fig. 6. It consists of four steps, as follows.

1) At the beginning of the procedure, both Agent 1 and Agent 2 have the same weights from the previous update. Then, by following Line 32 of Algorithm 1, on the basis of the information stored in its experience rollout list T_1, Agent 1 updates the initial weights into temporary parameters (θ_1 and ϕ_1).
2) Agent 1 transmits the temporary parameters to Agent 2.
3) Agent 2 considers weights θ_1 and ϕ_1 as the weights to be updated and performs the update by following the Algorithm 1 Line 32 and by exploiting T_2. The updated weights become (θ_2 and ϕ_2), and the agent passes them to the next agent, and so on.
4) The agent who last updated the weights will send them to all agents so that these final weights are applied to their networks to be used for the subsequent training episodes. In this specific case, Agent 2 sends the final weights to Agent 1.

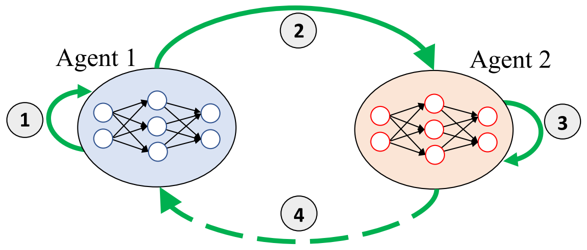


FIGURE 6. High-level representation of the distributed approach for sharing the network parameters among two learning agents during the training.

Doing so, at the end of this update routine, all the agents obtain the same weights, thus enabling the parameter sharing.[2] The transmission of the parameters among agents can be made through direct control-plane links, which can be established with several technologies, ranging from wireless personal area network (WPAN) standards, e.g., Bluetooth or Zigbee, to transmission at FR1 frequencies. Hence, this approach permits to end the training phase having the same final policy for each agent.

Another challenge in making feasible the generalization to the number of UEs involves how new agents acquire the shared policy during the operational phase. To address this issue, we propose an implementable solution wherein the BS periodically broadcasts both the policy and the OA function ψ to all new agents. These pieces of information can be mapped within a predefined system information block (SIB) sent into the BCCH. Therefore, the new agents can receive the policy and ψ by listening to the broadcast channel.

VII. OA FUNCTION DERIVATION

Several approaches can be adopted to derive the appropriate OA function and, in this paper, we resort to the concept

[2]The same target could be achieved through the implementation of a centralized federated learning approach [37], but our mechanism offers the advantage of not needing a central parameter server and an aggregation function. On the other hand, unlike federated learning methods that leverage parallel training among agents, our solution requires sequential training, resulting expensive as the number of agents rises. However, due to the expected generalization feature, the proposed framework can be trained with a limited number of agents (e.g., two), making our mechanism suitable.

of apprenticeship learning [38]. Therein, the authors present a single-agent RL problem, where the aim is to find the proper state abstraction function $\psi : \mathcal{S} \rightarrow \mathcal{S}^{(\psi)}$, where $\mathcal{S}^{(\psi)}$ denotes a new representation of the state space, termed abstracted state space, with $|\mathcal{S}^{(\psi)}| \ll |\mathcal{S}|$ that contains the most useful information yielding an efficient decision-making for the agent. In [30], learning the abstracted policy $\pi^{(\psi)}$ is carried out by observing one expert demonstrator following the policy $\pi^{(\mathrm{E})}$ in the original state domain. The goal of the state abstraction is tantamount to compressing $\mathcal{S}$ into $\mathcal{S}^{(\psi)}$, so that $\mathcal{S}^{(\psi)}$ provides the learning agents with an effective understanding of the environment to allow the agent to follow the expert policy in the abstracted space.

This gives rise to an interesting trade-off between *state space compression* and the ability of the agent to follow the expert policy, expressed as a *divergence* between the expert policy $\pi^{(\mathrm{E})}$ and the abstracted policy $\pi^{(\psi)}$ in the compressed space $\mathcal{S}^{(\psi)}$. To quantify this divergence, the authors adopt the average Kullback-Leibler (KL) divergence:

$$d\left\{\pi^{(\mathrm{E})}, \pi^{(\psi)}\right\} = \mathop{\mathbb{E}}_{s\in\mathcal{S}}\left\{D_{\mathrm{KL}}\left(\pi^{(\mathrm{E})}(\cdot \mid s) \parallel \pi^{(\psi)}(\cdot \mid \psi(s))\right)\right\}, \tag{24}$$

where

$$D_{\mathrm{KL}}\left(\pi^{(\mathrm{E})}(\cdot \mid s) \parallel \pi^{(\psi)}(\cdot \mid \psi(s))\right) = \sum_{a\in\mathcal{A}} \pi^{(\mathrm{E})}(a \mid s) \log\left(\frac{\pi^{(\mathrm{E})}(a \mid s)}{\pi^{(\psi)}(a \mid \psi(s))}\right). \tag{25}$$

Departing from the apprenticeship learning for the single-agent MDPs that rely on a unique optimal expert policy [30], we extend this concept for the MPOMDP scenario presented in this paper. The aim is to find a new representation $\mathcal{O}^{(\psi)}$ with $|\mathcal{O}^{(\psi)}| \ll |\mathcal{O}_{\mathrm{UE}}|$ that contains the most useful information yielding an efficient decision-making for each agent $i \in \mathcal{N}_{\mathrm{UE}}$. Differently from [30], in order to improve the robustness and retain an appropriate amount of information allowing the agents to learn diversified policies, we allow agents to adopt and exploit the information gathered from a set $\mathcal{P}^{(\mathrm{E})} = \left\{\pi_1^{(\mathrm{E})}, \pi_2^{(\mathrm{E})}, \ldots, \pi_G^{(\mathrm{E})}\right\}$ containing G expert policies defined in the original observation space $\mathcal{O}_{\mathrm{UE}}$. Specifically, we consider G scenarios, where in the gth scenario each of all the N learning agents adopts the same expert policy $\pi_g^{(\mathrm{E})}$. In this view, we also introduce a set $\mathcal{P}^{(\psi)} = \left\{\pi_1^{(\psi)}, \pi_2^{(\psi)}, \ldots, \pi_G^{(\psi)}\right\}$ containing the corresponding abstracted policies defined in the abstracted observation space $\mathcal{O}^{(\psi)}$.

The objective is re-defined as finding the optimal OA function ψ with $|\mathcal{O}^{(\psi)}| \ll |\mathcal{O}_{\mathrm{UE}}|$, which minimizes the following divergence loss function:

$$\begin{aligned} L_{\mathrm{div}} &= \sum_{g=1}^{G} d\left\{\pi_g^{(\mathrm{E})}, \pi_g^{(\psi)}\right\} \\ &= \sum_{g=1}^{G} \sum_{o\in\mathcal{O}_{\mathrm{UE}}} \sum_{a\in\mathcal{A}_{\mathrm{UE}}} \left[\pi_g^{(\mathrm{E})}(a \mid o) \log\left(\frac{\pi_g^{(\mathrm{E})}(a \mid o)}{\pi_g^{(\psi)}(a \mid \psi(o))}\right)\right]. \end{aligned} \tag{26}$$

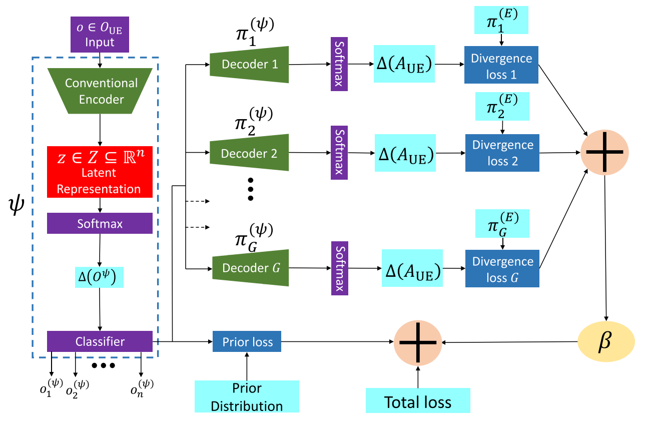


FIGURE 7. The proposed AE-based abstraction architecture.

Since it is not straightforward to solve this equation, we propose a new autoencoder (AE) architecture, which is composed of two deep neural networks (DNNs), namely encoder and decoder. The conventional AE architecture works as follows. The encoder maps the input space $\mathcal{X} \subseteq \mathbb{R}^m$ into a latent space $\mathcal{Z} \subseteq \mathbb{R}^n$, with $m, n \in \mathbb{N}$ and $n < m$. The decoder decompresses the latent space $\mathcal{Z}$ into the original input space $\mathcal{X}$. In detail, for any $x \in \mathcal{X}$, the encoder produces the latent representation $z \in \mathcal{Z}$. Then, the decoder receives as input z and outputs a representation $\hat{x} \in \mathcal{X}$ as close as possible to the original input x. Both the encoder and decoder are trained jointly to minimize the mean square error between each input x and the related output $\hat{x}$.

Starting from the conventional AE architecture, we propose a new architecture represented in Fig. 7. Therein, the proposed AE receives the observations $o \in \mathcal{O}_{\mathrm{UE}}$ as the input. Instead of reducing the input dimension, the aim of the new encoder is to reduce the cardinality of the input set $\mathcal{O}_{\mathrm{UE}}$ to obtain $|\mathcal{O}^{(\psi)}| \ll |\mathcal{O}_{\mathrm{UE}}|$. This objective is realized by enforcing the proposed encoder model to act as a multi-class classifier, in which, each sample from $\mathcal{O}_{\mathrm{UE}}$ is assigned to one and only one of n different abstracted observations, with $n \ll |\mathcal{O}_{\mathrm{UE}}|$. Specifically, the sample $o \in \mathcal{O}_{\mathrm{UE}}$ is assigned to the abstracted observation $o_k^{(\psi)} \in \mathcal{O}^{(\psi)}$ with $k \in \{1, 2, \ldots, n\}$.

Note that $|\mathcal{O}^{(\psi)}| \in \{1, 2, \ldots, n\}$ is held in general, since some abstracted observations $o_k^{(\psi)}$ may not be assigned to any input o. Therefore, this new encoder represents the observation abstraction function ψ. Instead of reconstructing the inputs as in conventional AE, the proposed decoder serves as an abstract policy network that maps each abstracted observation $o_k^{(\psi)}$ from $\mathcal{O}^{(\psi)}$ to a distribution over the action space $\mathcal{A}_{\mathrm{UE}}$. Since we consider a set $\mathcal{P}^{(\mathrm{E})}$ of G expert policies, we adopt G decoders, where each decoder is trained to produce the gth abstracted policy $\pi_g^{(\psi)}$. Each gth network aims to minimize the KL divergence with respect to $\pi_g^{(\mathrm{E})}$ as per (26).

Similar to the conventional AE, both encoder and decoders are jointly trained.

Finally, the proposed loss function consists of the sum of two parts.

The first one, named divergence loss, aims to achieve the goal (26), while the second one, named prior loss, acts as a regularization term on the latent representation to make the distributions returned by the encoder close to a prior distribution $\mathbf{p}$.

We propose to regularize the training with a prior distribution to avoid overfitting in the latent representation of the data so that the decoder networks can provide proper abstracted policies. For this, the regularization term is expressed as the KL divergence between the distribution at the output of the encoder and the prior $\mathbf{p}$ as a uniform distribution among all the possible labels:

$$L_{\text{prior}} = \mathop{\mathbb{E}}_{o \in \mathcal{O}_{\text{UE}}} \left\{ D_{\text{KL}}(\Delta(\mathcal{O}^{(\psi)}), \mathbf{p} \right\}. \tag{27}$$

The trade-off between the divergence loss and the regularization term is expressed by means of the hyperparameter $\beta \in \mathbb{R}_{\geq 0}$. The total loss is expressed as:

$$L_{\text{tot}} = L_{\text{prior}} + \beta L_{\text{div}}. \tag{28}$$

As $\beta \to 0$, the prior becomes more important, whereas as $\beta \to \infty$, minimizing divergence is prioritized. Given the latent space dimension n, the weights and biases of both the encoder and the decoder models are randomly initialized and updated via (28) by using the gradient descent (GD) method for N_{abs} episodes with the Adam optimizer and a learning rate l_{abs}.

VIII. PERFORMANCE EVALUATION

In this section, we focus on the assessment of the AE-based abstraction scheme to obtain an appropriate OA function, then we examine the performance of the proposed MAC protocol learning approach in terms of convergence time of the training phase, feasibility limits, and generalization capabilities.

A. ABSTRACTION EVALUATION AND PARAMETER SETTINGS

In Fig. 7, the encoder is a DNN composed of 3 hidden layers, each one with 512 neurons and the rectified linear unit (ReLU) as activation function. We adopt 2 decoders (i.e, $G = 2$ expert policies), in which each decoder is a DNN with one hidden layer of 100 neurons and the ReLU as activation function. Moreover, we set $l_{\text{abs}} = 2.5 \cdot 10^{-4}$ and $N_{\text{abs}} = 10\,000$, $\beta = 1000$. All the possible elements contained in the input set $\mathcal{O}_{\text{UE}}$ and its cardinality are rigorously derived in Appendix A. We set $M = 2$ and $Q = 10$ and, as a consequence, from (38), we obtain $|\mathcal{O}_{\text{UE}}| = 13104$.

Regarding the expert policies, we adopt $\pi_1^{(\text{E})}$ as the conventional grant-based transmission, where the UE only transmits the dPDU following the reception of a scheduling grant, and deletes a dPDU following the reception of the ACK. As $\pi_2^{(\text{E})}$, we adopt a grant-free transmission, where the UE transmits the dDPU immediately after it is available in the buffer, and deletes it after the transmission without waiting for the ACK message.

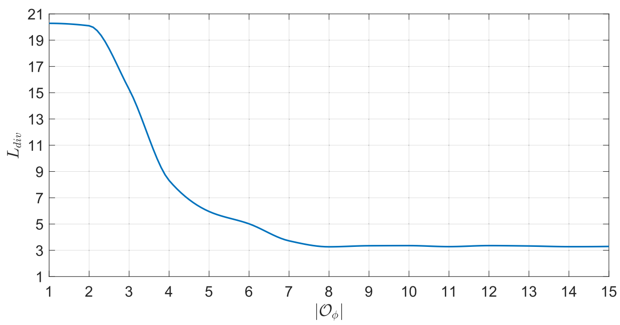


FIGURE 8. Evaluation of the OA performances by varying the abstracted observation space cardinality. Here, $|\mathcal{O}_{\text{UE}}| = 13104$.

For the setting of the latent space dimension and the related OA function, we performed the following simulation campaign. Starting from $n = 1$ (i.e., $|\mathcal{O}^{(\psi)}| = 1$), we evaluated the loss (26) at the end of the training. Then, we increased the value of n (i.e., the dimension of the latent space) with 1 unit until the loss (26) at the end of the training reached a plateau. As shown in Fig. 8, the optimal cardinality of $\mathcal{O}^{(\psi)}$ results $\left|\mathcal{O}^{(\psi)}\right| = 8 \ll |\mathcal{O}_{\text{UE}}|$, and the related OA function ψ is periodically transmitted by the BS.

B. BENCHMARK SCHEMES AND TRAINING SETTINGS

We evaluate the performance of the feasible learning approach, named **FM**, consisting of the proposed MPOMDP adopting the feasible reward described in Section IV-C, with our implementation of the MAPPO algorithm described in Section V. We also consider the solution [22], termed as **UM**, which adopts the infeasible reward described in Section IV-C, with a conventional CTDE implementation of the MAPPO algorithm shown in Fig. 1. More precisely, we compare these two schemes, both when they adopt the integration of the observation abstraction and when they do not. Therefore, the notation here adopted is: $\mathbf{FM}_{\text{O}_\psi}$ and $\mathbf{FM}_{\text{O}}$ for the scheme **FM** with and without OA, respectively; $\mathbf{UM}_{\text{O}_\psi}$ and $\mathbf{UM}_{\text{O}}$ for the scheme **UM** with and without OA, respectively.

Moreover, we compare these schemes with the approach [6], adapted to our system model, and denote it as $\mathbf{Q}_{\text{O}}$. Specifically, $\mathbf{Q}_{\text{O}}$ applies in our MPOMDP the tabular Q-learning as a training algorithm with the same hyperparameters as the original paper [6] without OA, and adopts the infeasible reward described in Section IV-C. We note that, due to the tabular nature of the algorithm adopted by $\mathbf{Q}_{\text{O}}$, the agents act randomly when they encounter an observation that has not been seen during the training.

For training all schemes, the number of UEs $N = 2$, the number of dPDUs to transmit $P = 2$, and the TBLER equal to 10^{-4} are used. All the training parameters together with the hyperparameters are reported in Table 1.

TABLE 1. **Training parameters.**

Parameter	Symbol	Value
Number of different signatures per time slot	L	64
Discount factor	γ	0.99
Max. duration of episode	$t_{\max}$	32
Reward function parameter	ρ	3
Number of learning agents	N	2
number of dPDUs per agent	P	2
Transport block error rate	TBLER	10^{-4}
Num. of neurons per hidden layer, critic		64
Num. of neurons per hidden layer, actor		64
Memory length	M	2
Learning rate	α	10^{-3}
Number of training episodes	N_{tr}	100k
Number of gradient descent steps	K	80
Act. function per layer, critic		{t, t, i}[a]
Act. function per layer, actor		{t, t, s}[b]
Clipping value	ϵ	0.2
Entropy coefficient	c	0.01

[a] t = tanh function, i = identity function.
[b] t = tanh function, s = softmax function.

C. TRAINING PERFORMANCE

For each scheme, we carried out 6 independent training sessions with different seeds. For each session and scheme, we determined the cumulative reward value per episode.

In Fig. 9, we show the average cumulative reward values over the training sessions (continuous line) and their range of variation (cloud around the average value). We note that the upper bound of the cumulative reward of $\mathbf{FM_O}/\mathbf{FM_{O_\psi}}$ and $\mathbf{UM_O}/\mathbf{UM_{O_\psi}}$ are different because they do not adopt the same reward function. As a consequence, the aim of these curves is twofold. Firstly, they compare the maximum value of the average cumulative reward between schemes adopting the same reward function (i.e., $\mathbf{FM_O}$ vs $\mathbf{FM_{O_\psi}}$ and $\mathbf{UM_O}$ vs $\mathbf{UM_{O_\psi}}$). Secondly, they compare all learning approaches in terms of the episodes required to achieve convergence.

As regards the first aim, it can be seen that the schemes without OA achieve a slightly higher average cumulative reward value than the relative systems with OA. The reason for this gap is that schemes without OA have access to the original observation space, which is specific for the training scenario, whereas schemes with OA to a reduced number of abstracted observations, which are invariant over multiple scenarios. This implies that schemes without OA reach a near-optimal solution for the specific configuration in which they were trained (i.e., the maximum cumulative reward) but with a scarce generalization to unseen configurations. On the other hand, schemes with OA reach a sub-optimal policy for the specific training scenario but with the advantage of enabling good generalization performance across different scenarios, as shown in the next subsection.

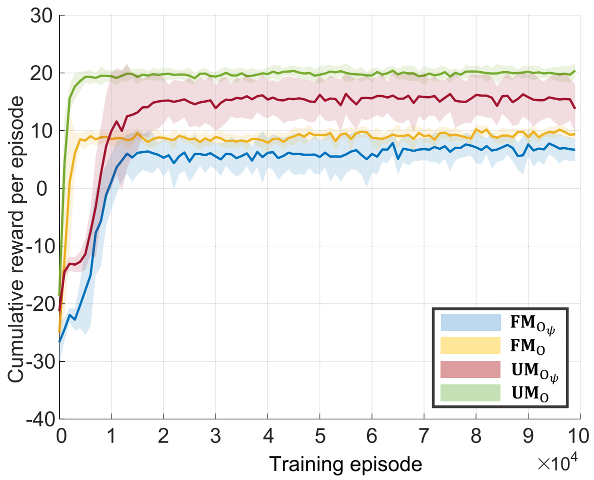


FIGURE 9. **Convergence time in the training phase. Number of agents $N = 2$, number of dPDUs per agent $P = 2$, and TBLER $= 10^{-4}$.**

As regards the convergence time, schemes without OA exhibit lower convergence times compared to those with OA. We believe that this difference arises from the reduced number of abstracted observations in schemes employing OA. Indeed, this reduction can lead to more diverse options for the same abstracted observation, potentially causing loops in the decision-making process and thereby delaying convergence. However, despite the longer training time, the significant advantage of schemes adopting OA is to prevent the need for a costly re-training procedure every time the scenario configuration changes.

D. RESULTS IN THE OPERATIONAL PHASE AND DISCUSSION

In this subsection, we report the performances obtained in the operational phase. After the training phase, for each of the 6 training sessions, we selected the last trained instance, and then we tested each instance for 1000 episodes with different seeds.

1) FEASIBLE VS INFEASIBLE LEARNING APPROACH

In this first simulation campaign, we have chosen not to consider the integration of the abstraction to focus on the performance obtained by adopting $\mathbf{FM_O}$ and $\mathbf{UM_O}$ under the same conditions for which they have been trained. The results of this simulation campaign are reported through the boxplots shown in Fig. 10. In the boxplots, the red line is the median, the blue box represents the interquartile range (IQR) from the first quartile (Q1) to the third quartile (Q3), the whiskers represent the minimum (Q1-1.5 IQR) and the maximum (Q3+1.5 IQR) values, and the outliers are plotted individually using the '+' marker symbol.

Fig. 10a shows the total number of successfully delivered dPDUs, Fig. 10b the number of time slots experiencing a collision, Fig. 10c the number of useless retransmissions per dPDU (i.e., the number of times that a dPDU already successfully received by the BS has been retransmitted by

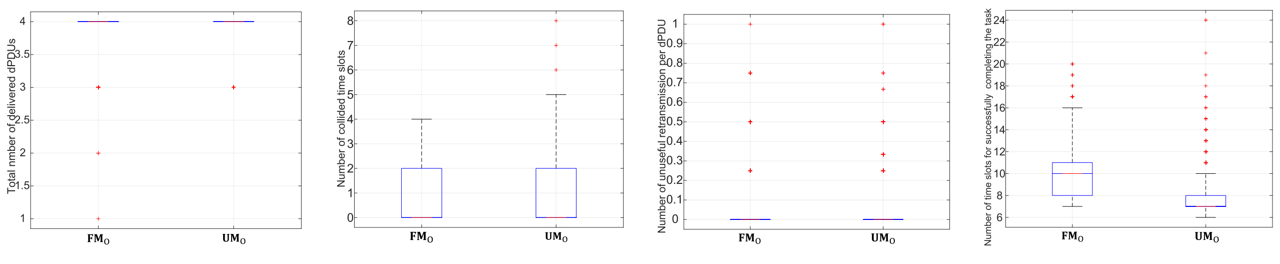


FIGURE 10. **Performance evaluation in the testing phase. Number of agents $N = 2$, number of dPDUs per agent $P = 2$, and TBLER $= 10^{-4}$.**

the UE), and Fig. 10d the number of time slots needed for successfully completing the task, considering only the cases in which the task has been completed.

As regards Fig. 10a, we note that both protocols perform very well in sending all 4 dPDUs, with very few outlier cases. The same holds for Fig. 10c, where the number of useless retransmissions per dPDU is almost 0 across the simulation results. Slight differences arise when considering Figs. 10b and 10d.

In detail, in Fig. 10b, we observe that for both the feasible and infeasible schemes, both the median number of collided time slots and the minimum are 0, whereas Q3 is 2. This indicates that 50% of the simulation results correspond to the optimal value (i.e., they do not involve any collisions), while 75% of the simulation data deviate slightly from the optimal value, with at most two time slots being collided. As regards the remaining 25% of the results, $\mathbf{UM_O}$ exhibits a higher whisker than the feasible scheme and other outliers extend up to 8 time slots. The slightly better performance of the $\mathbf{FM_O}$ can be explained by the fact that the feasible reward explicitly penalizes collision occurrences, unlike the infeasible one.

This reduction in collisions comes at the expense of the time needed to complete the task, as illustrated in Fig. 10d. Indeed, therein we show that the median value of $\mathbf{FM_O}$ is 2 time slots higher than $\mathbf{UM_O}$, and the IQR is 3 vs. 1. This outcome can be attributed to $\mathbf{UM_O}$'s goal of minimizing task completion time without considering collision risks, whereas the feasible reward aims to minimize task completion time while also reducing collisions. Consequently, $\mathbf{FM_O}$ may restrict UEs from transmitting in certain time slots where collision risks exist, thereby prolonging the time required to complete the task.

We note that the increase in time to complete the task does not conform to how much the number of collisions is reduced. This discrepancy arises from the fact that the feasible reward accesses only the information available to the BS and consequently, at the aim of minimizing collisions, $\mathbf{FM_O}$ may prevent transmissions even in cases where an omniscient environment's knowledge would have allowed transmission (for example, when only one of the two UEs has data to transmit).

In summary, $\mathbf{FM_O}$ shows performance in line with $\mathbf{UM_O}$, by successfully achieving the task of transmitting all dPDUs and avoiding useless retransmissions. Moreover, it obtains a lower risk of collision, at the cost of a slightly worse time taken to successfully complete the task. We remark that the results of $\mathbf{FM_O}$ have been obtained through a training procedure that involves a reduced information exchange among the network entities compared to $\mathbf{UM_O}$, as described in Section V-A.

2) GENERALIZATION ASSESSMENT

Now, we compare the solutions in terms of generalization to the number of dPDUs, to the TBLER, and to the number of UEs, and show the results in Figs. 11, 12, and 13, respectively. Similar to Fig. 10, the figures consists of four sub-figures, each one reporting the average values for the same metrics defined above. Regarding sub-figures (d), we note that, in the case a learning scheme fails always in completing the task, the related curve is not shown.

Generalization With Respect to the Number of dPDUs: Fig. 11 shows the KPI values obtained in the operational phase when the evaluation is carried out using the training parameters listed in Table 1, except for $t_{\max} = 300$ and $P \in \{1, \ldots, 10\}$.

Regarding Fig. 11a, the results show that $\mathbf{Q_O}$ performs well for the P value it was trained on (i.e., $P = 2$), while its performance degrades for higher P value. On the contrary, the solutions adopting the MAPPO algorithm without abstraction, i.e., $\mathbf{FM_O}$, and $\mathbf{UM_O}$, seem to show an initial intrinsic capacity of generalization in terms of total number of successfully transmitted packets within a certain range. However, this generalization ability does not persist when additional metrics are considered.

In particular, when $P = 4$, MAPPO without abstraction still shows a good number of successfully transmitted packets thanks to a higher average number of useless retransmissions per dPDU, as shown in Fig. 11c. Instead, as expected, the performance of generalized systems with abstraction clearly outperforms previous benchmark solutions for all metrics, see Fig. 11.

The results also show that on average $\mathbf{UM_{O_\psi}}$ performs slightly better than $\mathbf{FM_{O_\psi}}$ for 3 out of 4 considered metrics. This result can be justified since the infeasible reward is based on a more complete knowledge of the state. However, both exploit the intrinsic generalization ability of the on-policy algorithm and jointly reduce the uncertainties related to the different observation spaces through OA, obtaining

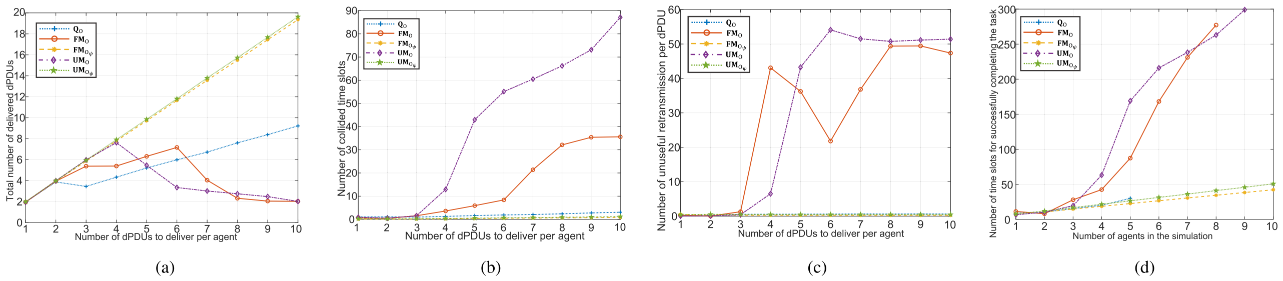


FIGURE 11. Performance evaluation in the testing phase. Number of agents $N = 2$, number of dPDUs per agent $P \in \{1, \ldots, 10\}$, and TBLER $= 10^{-4}$.

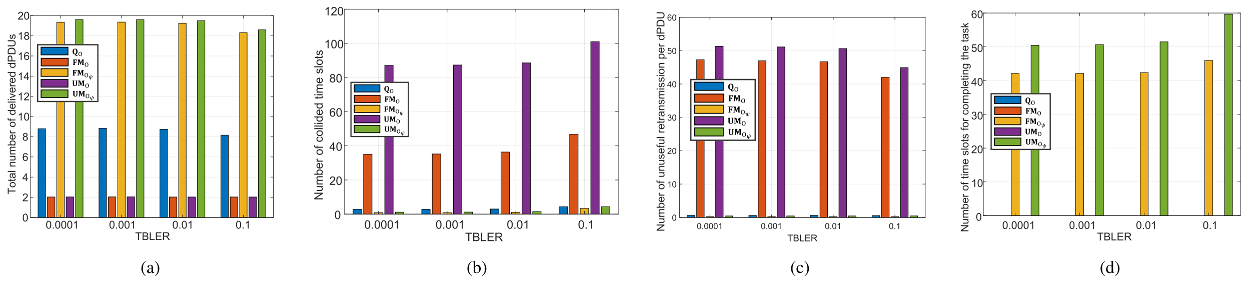


FIGURE 12. Performance evaluation in the testing phase. Number of agents $N = 2$, number of dPDUs per agent $P = 10$, and TBLER$\in \{10^{-4}, \ldots, 10^{-1}\}$.

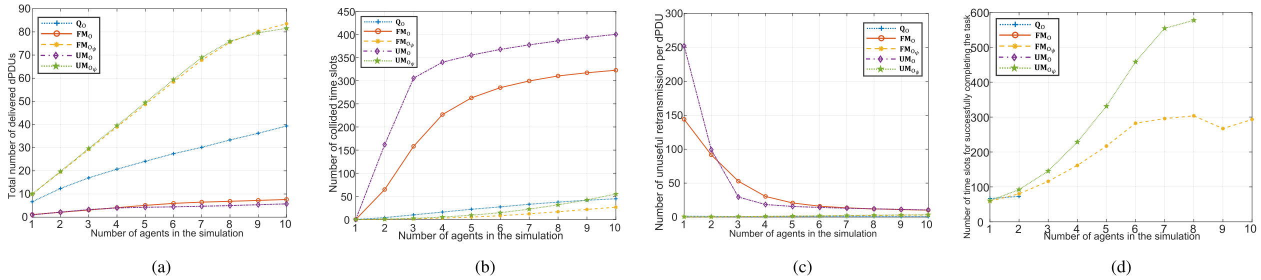


FIGURE 13. Performance evaluation in the testing phase. Number of agents $N \in \{1, \ldots, 10\}$, mean arrival time $\lambda = 1$, number of dPDUs per agent $P = 10$, TBLER $= 10^{-4}$.

almost perfect performance for all considered intervals of P, reaching in the most difficult configuration, i.e., at $P = 10$, a performance boost of about two- and ten-fold compared to $\mathbf{Q}_{\mathrm{O}}$ and $\mathbf{UM}_{\mathrm{O}}$.

Generalization With Respect to the TBLER Values: Fig. 12 depicts the metric values obtained using the training parameters listed in Table 1, except for $t_{\max} = 300$ and TBLER $\in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. We note that by increasing the TBLER value, the expected performance degradation is small for each scheme. We underline that only the schemes adopting the abstracted observation space are able to complete the task, and for this reason, Fig. 12d exhibits only the performance achieved by those schemes.

Generalization With Respect to the UE Number: Finally, in Fig. 13 the maximum number of UEs active in the whole simulation interval is variable. In particular, we have adopted an arrival process of active users, with $P = 10$ packets to be transmitted, described by a Poisson distribution with mean arrival time λ. The simulations were carried out by varying both $\lambda = \{1, 20\}$, and $t_{\max} = 600$, but for space reasons, we report only the results obtained with $\lambda = 1$ (heavy traffic load), because performances obtained were also similar for the other cases tested.

Fig. 13 shows that schemes that adopt abstraction significantly outperform other solutions for any metric. In particular, in Fig. 13d, we note that the feasible scheme with OA performs better even than the related infeasible scheme for high loads. In fact, the latter scheme fails to complete the task. The reason could be that the proposed feasible reward function aims also to minimize number of collisions, unlike the infeasible one.

This aspect is less pronounced when the number of agents is 2 (see Fig. 10b) since the probability of collision is decidedly lower than in the case of $N \geq 9$ UEs. As can be seen in Fig. 13b, with a high UE number, the number of collisions for the infeasible scheme is 100% higher than the feasible one, thus contributing to task failure.

Summary: The proposed $\mathbf{FM}_{\mathrm{O}_\psi}$ scheme, which adopts jointly the feasible learning approach and the OA, achieves good generalization performance as the number of UEs, TBLER, and the number of dPDUs varies. The advantage over $\mathbf{UM}_{\mathrm{O}_\psi}$ is more evident when the number of UEs is high, thanks to the feasible reward that penalizes the occurrence of collision. This generalization capability prevents the need for re-training procedures every time the scenario configuration changes. This result is obtained at the cost of slightly increased training time, which however requires a reduced exchange of information compared to the other infeasible solution.

IX. CONCLUSION

In this work, we studied the problem of learning MAC protocols in an uplink TDMA transmission scenario via a CTDE-based MARL framework considering both the user and the control planes. The goal is to maximize the number of successfully transmitted packets while minimizing the time spent doing so and the number of collisions. Compared to state-of-the-art works, our framework considers the challenges of the feasibility of the solution for implementation in real-world scenarios and the generalization performance outside of the training distribution.

To do so, we presented a feasible MPOMDP that overcomes the assumption of an omniscient environment, and designed a new reward function that does not require information exchange among the network entities to be computed, while still allowing the learning agents to learn the optimal policies. To solve the MPOMDP, we also presented our implementation of the MAPPO algorithm that adopts local critics as a replacement of the centralized critic.

Then, we leveraged the concept of state abstraction and defined an AMPOMDP that facilitates the generalization to the number of dPDUs. The AMPOMDP introduces the concepts of OA function, which maps the original observation space into an abstracted observation space, and abstracted policies, which operate on the abstracted observation space. Finally, we applied the parameter sharing approach for the actor and policy networks to generalize to the number of UEs.

The simulation results showed that MAC protocols learned by adopting either infeasible solutions that can be trained only via simulation or our feasible learning framework exhibit performance comparable. with that of the MAC protocols obtained by using infeasible solutions that can be trained only via simulation. In addition, the proposed feasible framework adopting both the OA and the parameter sharing approach learns MAC protocols that reach the established goal, while generalizing in terms of number of dPDUs to transmit, TBLER, and number of UEs. The generalization feature is obtained at the cost of a slightly higher training time compared to solutions without OA.

APPENDIX A
COMPUTATION OF THE CARDINALITY OF THE OBSERVATION SPACE FOR THE UEs

In this section, we calculate the cardinality of the set $\mathcal{O}_{\text{UE}}$. We remind that the learning agents are homogeneous, and so, they share the same observation space $\mathcal{O}_{\text{UE}}$. Given a generic learning agent $i \in \mathcal{N}_{\text{UE}}$, the elements belonging to the set $\mathcal{O}_{\text{UE}}$ are all possible partial observations of the agent i at each time t, denoted as o_i^t. We remind that $o_i^t = (b_i^t, b_i^{t-1}, a_i^{t-1}, m_i^{t-1}, \dots, b_i^{t-M}, a_i^{t-M}, m_i^{t-M})$, where $b_i^{t-k} \in \mathcal{B}$, $a_i^{t-k} \in \mathcal{A}_{\text{UE}}$, and $m_i^{t-k} \in \mathcal{M}_{\text{BS}}$, with $k = 0, 1, \dots, M$. As a consequence, the set $\mathcal{O}_{\text{UE}}$ can be easily seen as all possible arrangements between the elements in $\mathcal{B}$ and the M-arrangements with repetition (also called M-permutations with repetition) of the elements in $\mathcal{B}$, $\mathcal{A}_{\text{UE}}$, and $\mathcal{M}_{\text{BS}}$. Thus, the dimension of $\mathcal{O}$ is $4M+1$ and the cardinality is:

$$|\mathcal{O}_{\text{UE}}| = |\mathcal{B}| \cdot (|\mathcal{B}| \cdot |\mathcal{A}_{\text{UE}}| \cdot |\mathcal{M}_{\text{BS}}|)^M. \tag{29}$$

Consequently, $|\mathcal{O}_{\text{UE}}|$ increases significantly as both M and the buffer size Q increase. Nonetheless, this definition of $\mathcal{O}_{\text{UE}}$ does not take into account that some combinations of elements (i.e., some partial observations) cannot occur.

For this reason, we derive the set $\mathcal{O}_{\text{UE}}$ containing only all possible partial observations. Hence, we take into account that, given $b_i^t \in \mathcal{B}$, b_i^{t-1} cannot assume each value of $\mathcal{B}$. For deriving the range of possible values, we consider two simple dPDUs arrival models [6]:

1) **Full buffer start**: The UL Tx buffer of each agent is filled with $P = Q$ dPDUs at $t = 0$.
2) **Empty buffer start**: The UL Tx buffer is empty at $t = 0$. Then, it is filled with a given probability with one new dPDU each time step until a maximum of Q dPDUs have been generated.

Given these two arrival models, the following considerations are valid. Each agent i can delete at most one packet at a time (via action $a_{i,\text{u}} = 2$) and can add at most one packet at a time to its buffer following the second packet arrival rate. This means that the range of possible values assumed by b_i^{t-1} depends both on b_i^t and on the data action $a_{i,u}^{t-1}$. The same statement is valid for b_i^{t-2}, which depends on the value of b_i^{t-1} and $a_{i,u}^{t-2}$, and so on. In general, b_i^{t-k}, with $k = 1, 2, \dots, M$, can only assume the values in $\mathcal{B}^{(k)}$, where:

$$\mathcal{B}^{(k)} = \begin{cases} \left\{b_i^{t-k+1}, \max\left(b_i^{t-k+1} - 1, 0\right)\right\}, & \text{if } a_{i,u}^{t-1} \in \{0, 1\} \\ \left\{\min\left(b_i^{t-k+1} + 1, Q\right), b_i^{t-k+1}\right\} & \text{if } a_{i,u}^{t-1} = 2. \end{cases} \tag{30}$$

As shown, $\left|\mathcal{B}^{(k)}\right| \le 2, \forall k = 1, 2, \dots, M$ and thus

$$\left|\mathcal{O}_{\text{UE}}^{(M)}\right| \le |\mathcal{B}| \cdot (2 \cdot |\mathcal{A}_{\text{UE}}| \cdot |\mathcal{M}_{\text{BS}}|)^N. \tag{31}$$

As shown in (30), $\left|\mathcal{B}^{(k)}\right| \le 2, \forall k = 1, 2, \dots, M$. Let define $D_l^{(M)}$, with $l \in \mathcal{B}$, as the number of all possible arrangements we can obtain when $b_i^t = l$ with memory M. Let us start analyzing the case $M = 1$. We obtain:

$$\left|\mathcal{B}^{(1)}\right| = \begin{cases} 1, & \text{if } (b_i^t = 0 \text{ and } a_{i,u}^{t-1} \in \{0, 1\}) \\ & \text{or } (b_i^t = Q \text{ and } a_{i,u}^{t-1} = 2) \\ 2, & \text{otherwise.} \end{cases} \tag{32}$$

Starting from which, it can be easily derived that:

$$D_l^{(1)} = \begin{cases} \frac{2}{3} \cdot 2 \cdot |\mathcal{A}_{\text{UE}}| \cdot |\mathcal{M}_{\text{BS}}|, & \text{if } l = 0 \\ \frac{5}{6} \cdot 2 \cdot |\mathcal{A}_{\text{UE}}| \cdot |\mathcal{M}_{\text{BS}}|, & \text{if } l = Q \\ 2 \cdot |\mathcal{A}_{\text{UE}}| \cdot |\mathcal{M}_{\text{BS}}| & \text{otherwise.} \end{cases} \quad (33)$$

Starting from $D_l^{(1)}$,it follows:

$$\left|\mathcal{O}_{\text{UE}}^{(1)}\right| = \sum_{l \in \mathcal{B}} D_l^{(1)} = |\mathcal{B}| \cdot 2 \cdot |\mathcal{A}_{\text{UE}}| \cdot |\mathcal{M}_{\text{BS}}| - |\mathcal{A}_{\text{UE}}| \cdot |\mathcal{M}_{\text{BS}}| \quad (34)$$

Now, let us consider the case $M = 2$. In this case we have $\mathcal{B}^{(1)}$ and $\mathcal{B}^{(2)}$. As regards the cardinality of the last one, it can be derived as follows:

$$\left|\mathcal{B}^{(2)}\right| = \begin{cases} 1, & \text{if } (b_i^{t-1} = 0 \text{ and } a_{i,u}^{t-2} \in \{0, 1\}) \\ & \text{or } (b_i^{t-1} = Q \text{ and } a_{i,u}^{t-2} = 2) \\ 2, & \text{otherwise,} \end{cases} \quad (35)$$

where the value of b_i^{t-1} depends on b_i^t, and so on $D_l^{(1)}$. Starting from (35), $D_l^{(2)}$ can be derived as a function of (33) as follows:

$$D_l^{(2)} = \begin{cases} D_0^{(1)} \left[\frac{3}{4} D_0^{(1)} + \frac{1}{4} D_1^{(1)}\right], & \text{if } l = 0 \\ D_1^{(1)} \left[\frac{2}{3} D_1^{(1)} + \frac{1}{3} D_0^{(1)}\right], & \text{if } l = 1 \\ D_{Q-1}^{(1)} \left[\frac{5}{6} D_{Q-1}^{(1)} + \frac{1}{6} D_Q^{(1)}\right], & \text{if } l = Q-1 \\ D_Q^{(1)} \left[\frac{3}{5} D_Q^{(1)} + \frac{2}{5} D_{Q-1}^{(1)}\right], & \text{if } l = Q \\ (2 \cdot |\mathcal{A}_{\text{UE}}| \cdot |\mathcal{M}_{\text{BS}}|)^2 & \text{otherwise.} \end{cases} \quad (36)$$

It is straightforward to derive that, for a general case $M > 1$, the related $D_l^{(M)}$ can be recursively derived as a function of $D_l^{(1)}$ and $D_l^{(M-1)}$ as:

$$D_l^{(M)} = \begin{cases} D_0^{(1)} \left[\frac{3}{4} D_0^{(M-1)} + \frac{1}{4} D_1^{(M-1)}\right], & \text{if } l = 0 \\ D_Q^{(1)} \left[\frac{3}{5} D_Q^{(M-1)} + \frac{2}{5} D_{Q-1}^{(M-1)}\right], & \text{if } l = Q \\ D_l^{(1)} \Big[\frac{1}{2} D_l^{(M-1)} + \frac{1}{3} D_{l-1}^{(M-1)} & \\ \qquad + \frac{1}{6} D_{l+1}^{(M-1)}\Big], & \text{otherwise} \end{cases} \quad (37)$$

and so

$$\left|\mathcal{O}_{\text{UE}}^{(M)}\right| = \sum_{l \in \mathcal{B}} D_l^{(M)}. \quad (38)$$

REFERENCES

[1] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-native air interface," *IEEE Commun. Mag.*, vol. 59, no. 5, pp. 76–81, May 2021.

[2] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, Aug. 2019.

[3] L. Miuccio, D. Panno, and S. Riolo, "A wasserstein GAN autoencoder for SCMA networks," *IEEE Wireless Commun. Lett.*, vol. 11, no. 6, pp. 1298–1302, Jun. 2022.

[4] P. Kamble and A. N. Shaikh, "6G wireless networks: Vision, requirements, applications and challenges," in *Proc. 5th Int. Conf. Adv. Sci. Technol. (ICAST)*, Mumbai, India, Dec. 2022, pp. 577–581.

[5] *Medium Access Control (MAC) Protocol Specification*, document TS 38.321, 3GPP, 2023.

[6] A. Valcarce and J. Hoydis, "Toward joint learning of optimal MAC signaling and wireless channel access," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 4, pp. 1233–1243, Dec. 2021.

[7] E. Testi, E. Favarelli, and A. Giorgetti, "Reinforcement learning for connected autonomous vehicle localization via UAVs," in *Proc. IEEE Int. Workshop Metrol. Agricult. Forestry (MetroAgriFor)*, Nov. 2020, pp. 13–17.

[8] A. V. Rial et al., "The role of AI in 6G MAC," *TechRxiv*, Jul. 2023. [Online]. Available: https://www.techrxiv.org/doi/full/10.36227/techrxiv.23708310.v2

[9] L. Busoniu, R. Babuska, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst. Man, Cybern., C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[10] G. Cuozzo et al., "Enabling URLLC in 5G NR IIoT networks: A full-stack end-to-end analysis," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit*, Grenoble, France, Jun. 2022, pp. 333–338.

[11] A. A. Khan and R. S. Adve, "Centralized and distributed deep reinforcement learning methods for downlink sum-rate optimization," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 8410–8426, Dec. 2020.

[12] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in IoT," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9763–9773, Jun. 2021.

[13] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," 2017, *arXiv:1707.09183*.

[14] D. Lee, N. He, P. Kamalaruban, and V. Cevher, "Optimization for reinforcement learning: From a single agent to cooperative agents," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 123–135, May 2020.

[15] T. Li et al., "Applications of multi-agent reinforcement learning in future internet: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 2, pp. 1240–1279, 2nd Quart., 2022.

[16] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," 2017, *arXiv:1705.08926*.

[17] T. Wang, J. Wang, C. Zheng, and C. Zhang, "Learning nearly decomposable value functions via communication minimization," 2019, *arXiv:1910.05366*.

[18] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," 2016, *arXiv:1605.06676*.

[19] L. Xiao, X. Lu, T. Xu, W. Zhuang, and H. Dai, "Reinforcement learning-based physical-layer authentication for controller area networks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2535–2547, 2021.

[20] Z. Zhang, J. Hou, X. Chu, H. Zhou, G. Wei, and J. Zhang, "Multi-agent deep reinforcement learning based downlink beamforming in heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 22, no. 6, pp. 4247–4263, Jun. 2023.

[21] M. P. Mota, A. Valcarce, J.-M. Gorce, and J. Hoydis, "The emergence of wireless MAC protocols with multi-agent reinforcement learning," in *Proc. IEEE Globecom Workshops*, Madrid, Spain, Dec. 2021, pp. 1–6.

[22] L. Miuccio, S. Riolo, S. Samarakoon, D. Panno, and M. Bennis, "Learning generalized wireless MAC communication protocols via abstraction," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2022, pp. 2322–2327.

[23] H. B. Pasandi and T. Nadeem, "Towards a learning-based framework for self-driving design of networking protocols," *IEEE Access*, vol. 9, pp. 34829–34844, 2021.

[24] F. Al-Tam, N. Correia, and J. Rodriguez, "Learn to schedule (LEASCH): A deep reinforcement learning approach for radio resource scheduling in the 5G MAC layer," *IEEE Access*, vol. 8, pp. 108088–108101, 2020.

[25] L. Miuccio, S. Riolo, M. Bennis, and D. Panno, "On learning intrinsic rewards for faster multi-agent reinforcement learning based MAC protocol design in 6G wireless networks," in *Proc. IEEE Int. Conf. Commun.*, Rome, Italy, May 2023, pp. 466–471.

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[27] 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 Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[28] C. Yu et al., "The surprising effectiveness of PPO in cooperative, multi-agent games," 2021, *arXiv:2103.01955*.

[29] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *Knowl. Eng. Rev.*, vol. 27, no. 1, pp. 1–31, Feb. 2012.

[30] D. Abel, D. Arumugam, L. Lehnert, and M. Littman, "State abstractions for lifelong reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 10–19.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[32] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[33] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*. [Online]. Available: https://arxiv.org/abs/1506.02438

[34] M. U. Khan, E. Testi, E. Paolini, and M. Chiani, "Preamble detection in asynchronous random access using deep learning," *IEEE Wireless Commun. Lett.*, vol. 13, no. 2, pp. 279–283, Feb. 2024.

[35] J. Foerster et al., "Stabilising experience replay for deep multi-agent reinforcement learning," 2017, *arXiv:1702.08887*.

[36] X. Lyu, Y. Xiao, B. Daley, and C. Amato, "Contrasting centralized and decentralized critics in multi-agent reinforcement learning," in *Proc. 20th Int. Conf. Auton. Agents MultiAgent Syst. (AAMAS)*, 2021, pp. 844–852.

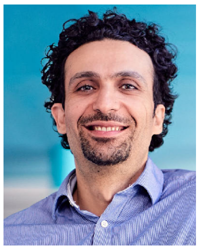
[37] M. Skocaj, P. E. I. Rivera, R. Verdone, and M. Erol-Kantarci, "Uplink scheduling in federated learning: An importance-aware approach via graph representation learning," in *Proc. IEEE Int. Conf. Commun. Workshops*, Rome, Italy, May 2023, pp. 1014–1019.

[38] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auto. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

LUCIANO MIUCCIO (Member, IEEE) received the bachelor's degree in electronics engineering, and the master's degree (cum laude) in telecommunications engineering from the Department of Electrical, Electronics and Computer Engineering (DIEEI), University of Catania, Catania, Italy, in 2015 and 2018, respectively, and the Ph.D. degree in systems, energy, computer and telecommunications engineering (with the additional label of "Doctor Europaeus") from the University of Catania in November 2022. From May 2019 to September 2019, he was an early-stage researcher in "Study of methodologies for the multi-objective optimization of parametric systems" with the University of Catania. He was a Visiting Ph.D. Student with the Centre for Wireless Communications (CWC), Oulu, Finland, for five months, in 2022. Since March 2023, he has been an Assistant Professor of telecommunications with the DIEEI, University of Catania. His research interests include green networking, NOMA techniques, and AI for B5G/6G radio resource management.

SALVATORE RIOLO (Member, IEEE) received the bachelor's degree in electronics engineering, the master's degree (cum laude) in telecommunications engineering, and the Ph.D. degree in systems, energy, computer and telecommunications engineering from the Department of Electrical, Electronics and Computer Engineering (DIEEI), University of Catania, Catania, Italy, in 2012, 2017, and 2021, respectively. From August 2021 to December 2021, he was a Post-Doctoral Researcher with the DIEEI, University of Catania, where he has been an Assistant Professor of telecommunications since January 2022. In 2022, he was a Visiting Researcher with the Centre for Wireless Communications, University of Oulu, Oulu, Finland, for six months. His research interests include radio resource management for B5G/6G networks, green networking, massive machine-type communications, and the convergence of artificial intelligence with radio access networks.

SUMUDU SAMARAKOON (Member, IEEE) received the B.Sc. degree (Hons.) in electronic and telecommunication engineering from the University of Moratuwa, Sri Lanka, in 2009, the M.Eng. degree from Asian Institute of Technology, Thailand, in 2011, and the Ph.D. degree in communication engineering from the University of Oulu, Finland, in 2017. He is currently an Assistant Professor with the Centre for Wireless Communications, University of Oulu, and a member of the Intelligent Connectivity and Networks/Systems (ICON) Group. His main research interests include heterogeneous networks, small cells, radio resource management, reinforcement learning, and game theory. He received the Best Paper Award from the European Wireless Conference and Excellence Awards for innovators and the Outstanding Doctoral Student in the Radio Technology Unit, CWC, University of Oulu, in 2016.



MEHDI BENNIS (Fellow, IEEE) is currently a Professor with the Centre for Wireless Communications, University of Oulu, Finland, and the Head of the Intelligent Connectivity and Networks/Systems Group (ICON). He has published more than 300 research papers in international conferences, journals, and book chapters. His main research interests include radio resource management, game theory, and distributed AI in 5G/6G networks. 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 2020–2021 Clarviate Highly Cited Researcher by the Web of Science.

DANIELA PANNO (Member, IEEE) received the Laurea degree (Hons.) in electrical engineering from the University of Catania, Catania, Italy, in 1989, and the Ph.D. degree in electronic engineering and computer science engineering from the University of Palermo, Palermo, Italy, in 1993. Since 1998, she has been an Associate Professor of telecommunications with the University of Catania. Her current research interests include massive IoT scenarios, green networking, and AI-based radio resource management for B5G/6G networks. She has attended several international workshops and symposia as an invited speaker. She has served as a Guest Editor for *Computer Communications*.

Open Access funding provided by 'Università degli Studi di Catania' within the CRUI CARE Agreement