

A Decentralized Pilot Assignment Algorithm for Scalable O-RAN Cell-Free Massive MIMO

Myeung Suk Oh, *Student Member, IEEE*, Anindya Bijoy Das, *Member, IEEE*,
Seyyedali Hosseinalipour, *Member, IEEE*, Taejoon Kim, *Senior Member, IEEE*,
David J. Love, *Fellow, IEEE*, and Christopher G. Brinton, *Senior Member, IEEE*

Abstract—Radio access networks (RANs) in monolithic architectures have limited adaptability to support different network scenarios. Recently, open-RAN (O-RAN) techniques have begun adding enormous flexibility to implementations. O-RAN is a natural architecture for cell-free massive multiple-input multiple-output (MIMO) systems, where many geographically-distributed access points (APs) are employed to achieve ubiquitous coverage and enhanced user performance. In this paper, we address the decentralized pilot assignment (PA) problem in scalable O-RAN-based CFmMIMO systems. We propose a low-complexity PA scheme using a multi-agent reinforcement learning (MA-DRL) framework in which multiple learning agents perform distributed learning in the O-RAN communication architecture to suppress contamination. Our approach does not require prior channel knowledge but instead relies on real-time interactions with the environment during the learning procedure. In addition, we design a codebook search (CS) scheme that exploits the decentralization of our O-RAN CFmMIMO architecture, where different codebook sets can be utilized to further improve PA performance without any significant additional complexities. Numerical evaluations verify that our proposed scheme provides substantial computational scalability advantages and improvements in channel estimation performance compared to the state-of-the-art.

Index Terms—Open-RAN (O-RAN), cell-free massive MIMO, deep reinforcement learning, pilot assignment.

I. INTRODUCTION

A. Open Radio Access Network (O-RAN)

Next generation wireless technologies will likely employ many dispersed radio access networks (RANs) for

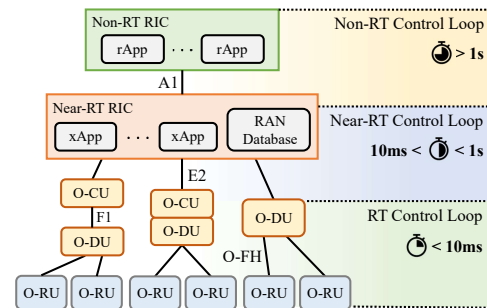


Fig. 1: O-RAN architecture with different types of control loops.

ubiquitous coverage and enhanced user performance [1], [2]. However, interconnecting different RANs to create one seamless network requires well-defined network functions and interfaces which are flexible in their integration capability. Recently, the evolution of software-defined open RAN (O-RAN) solutions have added enormous flexibility to the implementation of current 5G networks [3]–[5] and development of emerging 6G networks. O-RAN offers software-defined disaggregation on virtual network functions (VNFs) and necessary interfaces to support their coordination, allowing system implementations adaptive to various architectural settings. With this openness and flexibility, O-RAN promotes interoperability across different RAN vendors and allows network operators to adapt to different wireless environments.

O-RAN adopts the functional split defined in 3GPP [6] and defines three distinct units [7]: the open central unit (O-CU), open distributed unit (O-DU), and open radio unit (O-RU). Moreover, O-RAN operation is divided into three different control loops [7]: the real-time (RT), near-RT, and non-RT loops executing at different time-scales. The resulting O-RAN architecture and standard names of interfaces therein, which enable practical implementation of many RAN operations, are depicted in Fig. 1.

O-RAN offers two types of RAN intelligent controllers (RICs) [7] as shown in Fig. 1: near-RT RICs and non-RT RICs. Each of these RICs handles tasks manageable in different time-scales. O-RAN offers virtualization of both RICs, which promotes flexibility in implementing data-driven intelligence tasks that will be key components of emerging wireless networks. Various operations can

Manuscript received 7 January 2023; revised 9 June 2023; revised 28 August 2023; accepted 6 September 2023. This work was supported in part by the National Science Foundation (NSF) under Grant CNS2146171, CNS2212565, CNS2225577, EEC1941529, the Office of Naval Research (ONR) under Grant N000142112472, and the NSF and Office of the Under Secretary of Defense – Research and Engineering, ITE2326898, as part of the NSF Convergence Accelerator Track G: Securely Operating Through 5G Infrastructure Program.

M. S. Oh, A. B. Das, D. J. Love, and C. G. Brinton are with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907 USA (e-mail: {oh223, das207, djlove, cgb}@purdue.edu).

S. Hosseinalipour is with the Department of Electrical Engineering, University at Buffalo–SUNY, Buffalo, NY, 14260 USA (email: alipour@buffalo.edu).

T. Kim is with the Department of Electrical Engineering and Computer Science, the University of Kansas, Lawrence, KS, 66045 USA (email: taejoonkim@ku.edu).

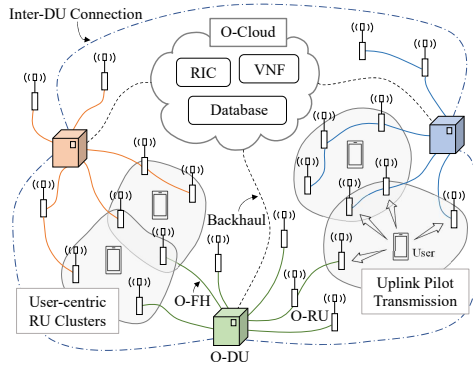


Fig. 2: A decentralized CFmMIMO system realized in O-RAN.

be implemented via custom third-party applications called *xApps/rApps* [7], allowing RICs to be much more accessible to the public. In this work, we will consider the implementation of machine learning (ML) algorithms over these RICs to optimize pilot signal assignments.

Due to these aforementioned advantages offered by O-RAN, a number of opportunities to utilize O-RAN on future wireless technologies seem promising:

- **Massive multiple-input multiple-output (MIMO) beamforming (BF):** To implement ML-based BF strategies that handle both latency-sensitive (e.g., RT beam selection with quick updates) and data-intensive (e.g., policy update using a large dataset) tasks is challenging, and O-RAN provides a platform for these approaches [8]–[10]. ML tasks are implemented in RICs, and BF operation can be split over O-RU and O-DU (e.g., option 7.2x [11]) to maximize efficiency.
- **Unmanned aerial vehicle (UAV) networking:** UAVs are typically deployed in dynamic environments (e.g., emergency rescue and aerial surveillance [12]), where the network infrastructure is required to be extremely flexible and adaptive. Flexibility and interoperability offered by O-RAN can be exploited to meet this architectural need [13], [14].
- **Localization via channel charting:** Channel charting is a data-driven localization technique [15] that maps a user to radio geometry using channel information. For the practical implementation of channel charting, O-RAN can offer a balanced distribution of heavy computational load coming from the data that is consistently collected and updated for each user.

B. Cell-free Massive MIMO

One innovative idea to address the shortcomings of 5G cellular networks is to remove cell boundaries using many dispersed transmission/reception points. This idea falls within the academic definition of cell-free massive MIMO (CFmMIMO) [16]–[18]. By deploying many geo-distributed access points (APs), CFmMIMO system alleviates the existing cell-edge problems by substantially improving both the reliability [19] and

energy efficiency [20] compared to cellular massive MIMO. These enhancements are due to the user-centric paradigm offered by CFmMIMO, where a group of APs are dynamically selected to serve each user.

In the early CFmMIMO literature, a system with APs connected to a single processing unit (PU) was considered for centralized operation. However, in a scalable system where the number of users and APs grow large, the resulting complexity becomes prohibitive [21]. Thus, CFmMIMO with multiple decentralized PUs (Fig. 2), each of which is connected to a disjoint subset of APs, has been introduced to consider scalability [21]–[24]. The decentralization allows the system to scale while still being practical by reducing the computational and fronthaul load on each PU [18]. Nevertheless, implementing centralized CFmMIMO techniques (e.g., signal adaptation and resource allocation) into a decentralized architecture is a challenging task.

C. CFmMIMO Pilot Assignment Problem

In CFmMIMO, reliable channel estimation at both transmitter and receiver is critical to facilitate advanced diversity and signal processing techniques. For channel estimation, a set of orthogonal pilots are used. However, when the number of users grows beyond the number of available pilots, some users must share their pilots with others, leading to pilot contamination (PC) that can significantly degrade the channel estimation performance [25]. To cope with PC, various pilot assignment (PA) methods have been studied in the CFmMIMO literature [26]–[32].

In [26], a greedy PA scheme with iterative pilot updates was proposed to mitigate PC. A pilot reuse scheme to acquire dynamic sets of users for pilot sharing was proposed in [27]. In [28], a PA strategy, in which the same pilot is assigned to users with minimum overlapping APs, was proposed. Other methods to solve the PA problem include k-means clustering [29], graph coloring [30], tabu-search [31], and Hungarian [32] algorithms.

These prior works [26]–[32], however, conduct PA via centralized processing. Thus, their computational complexities become prohibitive as the number of users grows large (e.g., Fig. 13a). While one can always consider conducting a set of uncoordinated local PAs, such an architecture without global orchestration can degrade the overall performance. Successful decentralization requires a carefully designed coordination strategy to achieve performance comparable to the centralized case. To the best of our knowledge, no work has yet developed and analyzed such a well-engineered distributed PA for CFmMIMO systems. In addition, these works [26]–[32] use closed-form expressions derived from Bayesian estimation, requiring any relevant information (e.g., pathloss) to be known a priori. The required information is in general obtained via estimation (e.g., pathloss can be

estimated after collecting power measurements); however, for large-scale systems, especially under a dynamic environment, accurately estimated prior information is often not available due to the large overhead imposed, underscoring the need to develop a PA scheme that does not require prior knowledge. As a viable approach to address these issues, in this work, we adopt a learning-based optimization technique called multi-agent deep reinforcement learning (MA-DRL) to conduct decentralized PA in a CFmMIMO system.

D. Overview of Methodology and Contributions

Motivated by the aforementioned challenges, we focus on PA in scalable CFmMIMO systems. As CFmMIMO deploys a large number of APs for ubiquitous coverage, it is crucial to maintain a great level of implementation flexibility and interoperability across different RANs for scalability. Hence, we propose to design our CFmMIMO system in O-RAN architecture. As O-RAN balances operational complexities and computational loads via a functional split along the network (i.e., O-RU/DUs and RICs), O-RAN becomes a natural solution for scalable CFmMIMO systems.

Based on the O-RAN CFmMIMO system, we formulate a decentralized PA problem and develop a learning-based PA scheme to solve it. We resort to a MA-DRL framework, where a group of agents individually perform their learning to provide a low-complexity solution without an explicit training stage [33]–[35]. Our scheme is designed to operate in the near-RT RIC of O-RAN. We summarize the key contributions of our work below.

- We design our CFmMIMO system based on the O-RAN architecture (Sec. II). We specifically focus on channel estimation and PA models considering practical aspects (e.g., fronthaul overhead and operational complexity by each functional unit), which can be adopted to the O-RAN CFmMIMO systems.
- We design a Markov game model (Sec. III-C) for our MA-DRL which leads to an efficient solution for our decentralized PA problem. In particular, we formulate our reward based on observations that are directly measurable at the O-RUs. Thus, our scheme does not require prior knowledge of channel statistics, which is different from previous PA algorithms [26]–[32].
- We propose a novel learning-based PA scheme (Sec. III-D) aiming to minimize the total mean squared error (MSE) across the users. By adopting the MA-DRL framework, our scheme provides a low-complexity PA solution, the computation complexity of which increases at a much lower rate compared to the previous PA algorithms and therefore offers a scalability advantage to support large-scale systems.
- Utilizing the decentralization of our system, we consider two effective ways to improve the PA performance: (i) inter-DU message passing for observation

sharing and (ii) low-complexity codebook search (CS) algorithm (Sec. III-E) that jointly operates with our PA scheme. Numerical results verify that these approaches can further improve the PA performance.

- We show that our PA scheme can maintain its performance over a mobile environment, which is possible due to (i) the DRL framework that naturally performs adaptive learning and (ii) the CS algorithm with iterative greedy search. Previous PA methods only consider a static environment and do not address the user mobility.
- We numerically evaluate (Sec. IV) the performance of our PA scheme against the state-of-the-art [31], [32] in both channel estimation performance and computational complexity. The results show that our scheme outperforms the benchmarks in terms of sum-MSE and scalability.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first describe the architecture of the considered O-RAN-based CFmMIMO system (Sec. II-A) to establish a foundation for CFmMIMO decentralization. Then, after describing the channel model (Sec. II-B), we provide details of codebook-based channel estimation (Sec. II-C) and uplink/downlink data transmission (Sec. II-D). Finally, we formulate our decentralized PA problem (Sec. II-E) and explain the relationship between the PA task and channel estimation performance.

A. CFmMIMO Configuration in O-RAN Architecture

Our decentralized O-RAN CFmMIMO system is illustrated in Fig. 2. We consider M single-antenna O-RUs and U O-DUs collected in sets $\mathcal{M} = \{1, 2, \dots, M\}$ and $\mathcal{U} = \{1, 2, \dots, U\}$, respectively. The O-RUs are randomly placed using uniform distribution across the area that is divided into U disjoint regions for system decentralization, and each O-DU is deployed to one of the regions. Each O-RU is connected to one of the O-DUs in $\mathcal{U}$ via an open fronthaul (O-FH) connection such that O-RUs within each subdivided region are connected to the same O-DU. We define $\mathcal{M}_u^{\text{DU}} \subseteq \mathcal{M}$ as the set of O-RUs connected to O-DU $u \in \mathcal{U}$. We assume inter-DU connections [36] to form fully user-centric RU clusters since the users can be served by RUs from different sets of $\mathcal{M}_u^{\text{DU}}$. We assume ideal O-FH and inter-DU connections [30], [31].

Here, we have our O-DUs connected to O-Cloud [7] via backhaul network (Fig. 2). O-Cloud is the cloud computing platform that supports the virtualized network functions (VNFs) within O-RAN, which include RICs. In designing our PA scheme, we specifically focus on the near-RT RIC that communicates with O-DUs via E2 interface (Fig. 1). Within the near-RT RIC, we assume U independent learning agents, each of which has a one-to-one correspondence to one of the O-DUs in the system. Note that we assume multiple agents to fully impose

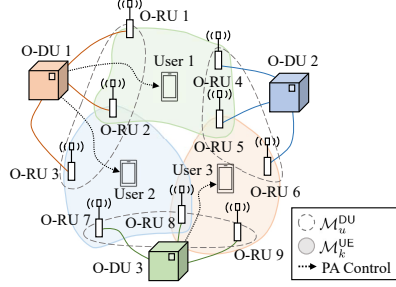


Fig. 3: A list of our defined sets and their visual examples for the given decentralized cell-free O-RAN layout.

decentralization on our system. Thus, each agent in the near-RT RIC conducts local learning through the O-DU and O-RUs connected. We also consider a single non-RT RIC interacting with the near-RT RIC via an A1 interface (Fig. 1), which is responsible for learning model updates of the near-RT RIC.

Next, we consider K single-antenna users in a set $\mathcal{K} = \{1, 2, \dots, K\}$. For each user k , a user-centric RU cluster is formed such that only $M_k^{\text{UE}} \ll M$ O-RUs are engaged to serve the user, where we define $\mathcal{M}_k^{\text{UE}} \subset \mathcal{M}$ to be the set of O-RUs serving user $k \in \mathcal{K}$ (i.e., $M_k^{\text{UE}} = |\mathcal{M}_k^{\text{UE}}|$ where $|\cdot|$ denotes the set cardinality). Each $\mathcal{M}_k^{\text{UE}}$ is assumed to be selected and updated using a procedure independent from our PA (e.g., radio resource control (RRC) setup procedure [37]). We also define $\mathcal{K}_m^{\text{RU}} \subset \mathcal{K}$ to be the set of users served by O-RU $m \in \mathcal{M}$.

Since we have U multiple agents performing PA, each user $k \in \mathcal{K}$ must belong to one of these agents. To develop user-to-agent pairings, we consider two different types of users: (i) user k whose $\mathcal{M}_k^{\text{UE}}$ is connected to a single O-DU u , i.e., $\mathcal{M}_k^{\text{UE}} \subseteq \mathcal{M}_u^{\text{DU}}$, which we simply pair that user k to the corresponding agent u , and (ii) user k whose $\mathcal{M}_k^{\text{UE}}$ consists of O-RUs from different O-DUs. For the second type, a serving O-DU [36], which can be defined by any reasonable criterion (e.g., the O-DU with the most number of O-RUs serving the user), is determined and paired with the user. We define $\mathcal{K}_u^{\text{DU}}$ to be the set of users whose PA is managed by O-DU u .

Example 1. We consider a scenario with $U = 3$, $M = 9$, and $K = 3$, which is illustrated in Fig. 3. For decentralization, each O-DU is connected to three O-RUs that are closest (e.g., $\mathcal{M}_1^{\text{DU}} = \{1, 2, 3\}$), and user-centric RU clusters with $M_k^{\text{UE}} = 4$ are formed for each user (e.g., $\mathcal{M}_1^{\text{UE}} = \{1, 2, 4, 5\}$). Note that an O-RU can serve multiple users (e.g., $\mathcal{K}_2^{\text{RU}} = \{1, 2\}$). Since each user needs an agent for PA, the user is paired to one of the three O-DUs (e.g., $\mathcal{K}_1^{\text{DU}} = \{1, 2\}$).

B. Time-varying Channel Model

We assume a periodic channel estimation with time interval T_e and indicate each estimation instance using index $i = 0, 1, \dots, N$. The channel between user $k \in \mathcal{K}$ and O-RU $m \in \mathcal{M}$ during channel estimation instance i

is formally expressed as

$$g_{km}^{(i)} = \sqrt{\beta_{km}^{(i)}} h_{km}^{(i)}, \quad (1)$$

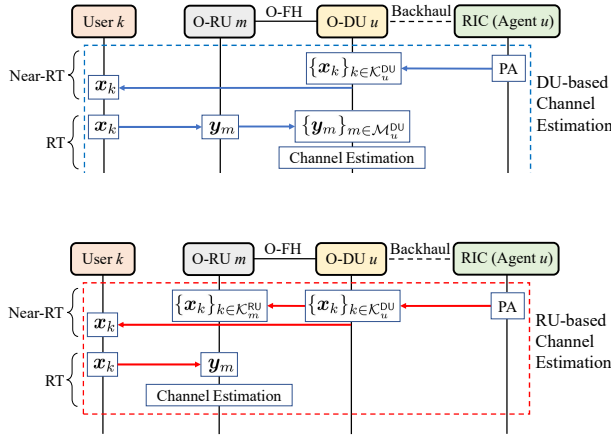
where $h_{km}^{(i)} = \mu_k h_{km}^{(i-1)} + \sqrt{(1 - \mu_k^2)} n_{km}^{(i)}$ is the small-scale fading factor following a first-order time-varying Gauss-Markov process for $i = 1, 2, \dots, N$. The perturbation terms $\{n_{km}^{(i)}\}$ are zero-mean, unit-variance complex Gaussian random variables that are independent and identically distributed (i.i.d.) over k, m , and i , i.e., $n_{km}^{(i)} \sim \mathcal{CN}(0, 1)$. At $i = 0$, we assume $h_{km}^{(0)} \sim \mathcal{CN}(0, 1)$ to be mutually independent from $n_{km}^{(1)}$. The correlation coefficient μ_k for user k is defined as $\mu_k = J_0(2\pi \frac{v_k}{c} f_c T_e)$ [38], where $J_0(\cdot)$ is the Bessel function of the first kind of order zero, v_k is the velocity of user k , f_c is the carrier frequency, and $c = 3 \times 10^8$ m/s is the speed of light. The magnitude of $h_{km}^{(i)}$ is designed to follow a Rayleigh distribution, which is effective in modeling a dense scattering wireless environment [39]. The term $\beta_{km}^{(i)}$ in (1) is the large-scale fading factor that is inversely proportional to the distance between user k and O-RU m at the channel estimation instance i . There exist multiple realistic large-scale fading models, including the 3GPP urban-micro line-of-sight pathloss model [40].

C. Codebook-based Channel Estimation

We consider uplink channel estimation with T_p channel uses for each estimation instance, allowing T_p orthogonal pilots to be available. For each instance i , user $k \in \mathcal{K}_u^{\text{DU}}$ is assigned with one of the T_p pilots in a mutually orthogonal codebook $\mathcal{T}_u^{(i)} = \{\phi_{u,1}^{(i)}, \phi_{u,2}^{(i)}, \dots, \phi_{u,T_p}^{(i)}\}$, where each $\phi_{u,t}^{(i)}$ for $t = 1, 2, \dots, T_p$ is a unit-norm complex vector of length T_p . Thus, for $t, t' = 1, 2, \dots, T_p$, $(\phi_{u,t}^{(i)})^H \phi_{u,t'}^{(i)} = 1$ if $t = t'$, and zero otherwise, where $(\cdot)^H$ denotes the conjugate transpose. We denote the pilot assigned to user k as $\mathbf{x}_k^{(i)}$.

For channel estimation, each user $k \in \mathcal{K}$ transmits the assigned pilot $\mathbf{x}_k^{(i)}$. The signal vector (of length T_p) received by O-RU $m \in \mathcal{M}$ is then expressed as $\mathbf{y}_m^{(i)} = \mathbf{X}^{(i)} \mathbf{g}_m^{(i)} + \mathbf{w}_m^{(i)} = \sum_{k \in \mathcal{K}} g_{km}^{(i)} \mathbf{x}_k^{(i)} + \mathbf{w}_m^{(i)}$, where $\mathbf{X}^{(i)} = [\mathbf{x}_1^{(i)} \mathbf{x}_2^{(i)} \dots \mathbf{x}_K^{(i)}]$ is the $T_p \times K$ pilot matrix and $\mathbf{g}_m^{(i)} = [g_{1m}^{(i)} g_{2m}^{(i)} \dots g_{Km}^{(i)}]^T$ is the channel vector (of length K) for O-RU m . Here, $\mathbf{w}_m^{(i)} \sim \mathcal{CN}(\mathbf{0}, \sigma^2 \mathbf{I}_{T_p})$ is the zero-mean complex Gaussian noise vector of length T_p with covariance $\sigma^2 \mathbf{I}_{T_p}$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix.

We discuss two different channel estimation structures within O-RAN architecture, which we illustrate in Fig. 4, and compare their communication overhead by computing the number of bits exchanged during a single near-RT control loop. One structure (Fig. 4a) performs channel estimation at O-DU whereas the estimation occurs at O-RU in the other structure (Fig. 4b). We assume that N_n RT loops occur for each near-RT loop.



(b) RU-based channel estimation.

Fig. 4: A block diagram of two different channel estimation structures.

Suppose representing the PA information $\mathbf{x}_k^{(i)}$ and received signal $\mathbf{y}_m^{(i)}$ requires $b_d = \log_2 T_p$ and $b_u = 2BT_p$ bits, respectively, where $2B$ is the number of bits used to represent a complex number. For DU-based channel estimation, the RIC first sends out the PA information of the controlled users $\{\mathbf{x}_k^{(i)}\}_{k \in \mathcal{K}_u^{\text{DU}}}$ over the backhaul using $b_d |\mathcal{K}_u^{\text{DU}}|$ bits. O-DU u then passes this pilot information to the master O-RU that serves user k over the O-FH using another $b_d |\mathcal{K}_u^{\text{DU}}|$ bits. For each of the N_n RT loops, $\mathbf{y}_m^{(i)}$ from each O-RU $m \in \mathcal{M}_u^{\text{DU}}$ must be collected by the O-DU, which results in $b_u |\mathcal{M}_u^{\text{DU}}| N_n$ bits exchanged over the O-FH. Hence, the total amount of overhead for DU-based channel estimation is $\sum_{u=1}^U (2b_d |\mathcal{K}_u^{\text{DU}}| + b_u |\mathcal{M}_u^{\text{DU}}| N_n)$ bits. Note that $2b_d |\mathcal{K}_u^{\text{DU}}|$ accounts for the data transferred in both backhaul and O-FH links.

For the RU-based channel estimation, each of the O-RUs serving user k must be informed with the pilot information $\mathbf{x}_k^{(i)}$ to conduct channel estimation. Therefore, in addition to the $b_d |\mathcal{K}_u^{\text{DU}}|$ bits exchanged between the RIC and O-DU over the backhaul, $\sum_{m \in \mathcal{M}_u^{\text{DU}}} b_d |\mathcal{K}_m^{\text{RU}}|$ bits must be transferred via O-FH to deliver the pilot information to the O-RUs. Note that, for RU-based channel estimation, no RT data transfer is required since the estimation occurs at each O-RU. Hence, the total amount of overhead for RU-based channel estimation is given by $b_d \sum_{u=1}^U (|\mathcal{K}_u^{\text{DU}}| + \sum_{m \in \mathcal{M}_u^{\text{DU}}} |\mathcal{K}_m^{\text{RU}}|)$. Table I shows the amount of overhead in bits per near-RT loop required for channel estimations when $M = 96$, $U = 4$, $M_k^{\text{UE}} = 8$, and $B = 8$. From the result, we confirm that RU-based estimation imposes less overhead than DU-based one does. Note that it also benefits from latency advantage as no data exchange is needed for RT loops. Hence, similar to the work in [26], we assume our channel estimation to take place at O-RUs.

Next, in case of user-centric RU clustering, each RU $m \in \mathcal{M}$ only needs to estimate $|\mathcal{K}_m^{\text{RU}}|$ different channels (i.e., $\{g_{km}^{(i)}\}_{k \in \mathcal{K}_m^{\text{RU}}}$) associated with users in

TABLE I

The amount of overhead in bits per near-RT loop to perform channel estimation

Estimation	$T_p = 4$		$T_p = 8$	
	$K = 24$	$K = 72$	$K = 24$	$K = 72$
DU-based	61,536	61,728	123,024	123,312
RU-based	432	1,296	648	1,944

$\mathcal{K}_m^{\text{RU}}$. For estimating the channel, we consider two different techniques called *pilot-matching* [19] and *least-square* [41] estimations. If we set $\hat{\mathbf{g}}_m^{(i)} = [\hat{g}_{km}^{(i)}]_{k \in \mathcal{K}_m^{\text{RU}}}^T$ as the $|\mathcal{K}_m^{\text{RU}}|$ -length estimated channel vector from O-RU m during the channel estimation instance i , pilot-matching and least-square estimations are expressed as

$$\hat{\mathbf{g}}_m^{(i)} = (\bar{\mathbf{X}}_m^{(i)})^H \mathbf{y}_m^{(i)} = (\mathbf{Z}_m^{(i)})^H (\mathbf{X}^{(i)})^H \mathbf{y}_m^{(i)} \quad (2)$$

$$\text{and } \hat{\mathbf{g}}_m^{(i)} = (\bar{\mathbf{X}}_m^{(i)})^H (\mathbf{X}^{(i)} (\mathbf{X}^{(i)})^H)^{-1} \mathbf{y}_m^{(i)} \\ = (\mathbf{Z}_m^{(i)})^H (\mathbf{X}^{(i)})^H (\mathbf{X}^{(i)} (\mathbf{X}^{(i)})^H)^{-1} \mathbf{y}_m^{(i)}, \quad (3)$$

respectively, where $\bar{\mathbf{X}}_m^{(i)} = \mathbf{X}^{(i)} \mathbf{Z}_m^{(i)} = [\mathbf{x}_k^{(i)}]_{k \in \mathcal{K}_m^{\text{RU}}}$ is the $T_p \times |\mathcal{K}_m^{\text{RU}}|$ pilot matrix of the users served by O-RU m . We define a $K \times |\mathcal{K}_m^{\text{RU}}|$ selection matrix $\mathbf{Z}_m^{(i)} = [\mathbf{z}_k^{(i)}]_{k \in \mathcal{K}_m^{\text{RU}}}$ where $\mathbf{z}_k^{(i)}$ is the K -length unit-vector with its k -th element being *one*. Now, when some of K users share the pilot, $\mathbf{X}^{(i)}$ is not unitary (i.e., $(\mathbf{X}^{(i)})^H \mathbf{X}^{(i)} \neq \mathbf{I}_K$), so the least-square estimation in (3), which utilizes the pseudo-inverse term $(\mathbf{X}^{(i)})^H (\mathbf{X}^{(i)} (\mathbf{X}^{(i)})^H)^{-1}$ to negate the PC, yields better estimation performance. However, in the least-square approach, since $\mathbf{X}^{(i)}$ needs to be known to every O-RU and the size of $\mathbf{X}^{(i)}$ increases linearly with K , the resulting overhead causes significant delay as the number of users grows. Note that, for the case of pilot-matching, each O-RU m only needs to know $\{\mathbf{x}_k^{(i)}\}_{k \in \mathcal{K}_m^{\text{RU}}}$ to obtain $\bar{\mathbf{X}}_m^{(i)}$. This motivates the pilot-matching channel estimation scheme in (2) for scalability [19]. The estimated channel $\hat{g}_{km}^{(i)}$ is then expressed as

$$\hat{g}_{km}^{(i)} = (\mathbf{x}_k^{(i)})^H \mathbf{y}_m^{(i)} = \sum_{k' \in \mathcal{K}} g_{k'm}^{(i)} (\mathbf{x}_k^{(i)})^H \mathbf{x}_{k'}^{(i)} + (\mathbf{x}_k^{(i)})^H \mathbf{w}_m^{(i)} \\ = g_{km}^{(i)} + \sum_{\substack{k' \in \mathcal{K} \\ k' \neq k}} g_{k'm}^{(i)} (\mathbf{x}_k^{(i)})^H \mathbf{x}_{k'}^{(i)} + (\mathbf{x}_k^{(i)})^H \mathbf{w}_m^{(i)}. \quad (4)$$

Note that the summation term in the last equality captures the effect of PC.

D. Data Transmission Model

For uplink (downlink) data transmission, the estimated channel in (4) is used as a combiner (a precoder), the details of which are given as follows. For uplink transmission, each user k transmits a data signal x_k^u . Then, the received signal y_m^u at O-RU m is given by $y_m^u = \sum_{k \in \mathcal{K}} g_{km}^{(i)} \sqrt{\rho_k} x_k^u + w_m^u$, where ρ_k and w_m^u are the transmit power of user k and uplink additive Gaussian noise on O-RU m with variance σ_u^2 , respectively. For each user $k \in \mathcal{K}_m^{\text{RU}}$, O-RU m computes $(\hat{g}_{km}^{(i)})^* y_m^u$ and

transfers it to the user's serving O-DU. After collecting the conjugate-multiplied signals from the O-RUs in $\mathcal{M}_k^{\text{UE}}$, the serving O-DU combines them to obtain the data signal $\bar{x}_k^u$ expressed as

$$\bar{x}_k^u = \sum_{m \in \mathcal{M}_k^{\text{UE}}} \sum_{k' \in \mathcal{K}} (\hat{g}_{km}^{(i)})^* g_{k'm}^{(i)} \sqrt{\rho_{k'}} x_{k'}^u + \sum_{m \in \mathcal{M}_k^{\text{UE}}} (\hat{g}_{km}^{(i)})^* w_m^u. \quad (5)$$

Based on (5) and the formulation in [18], the effective uplink signal to interference plus noise ratio (SINR) of user k is given by (7), where the expectation is over the random variables.

For downlink transmission, the data signal x_k^d is transmitted by the O-RUs serving user k (i.e., O-RU $m \in \mathcal{M}_k^{\text{UE}}$) after applying the conjugate beamforming expressed as $\bar{x}_{km}^d = (\hat{g}_{km}^{(i)})^* x_k^d / |\hat{g}_{km}^{(i)}|$. The received signal $\bar{y}_k^d$ for user k is then given by

$$\begin{aligned} \bar{y}_k^d &= \sum_{k' \in \mathcal{K}} \sum_{m \in \mathcal{M}_{k'}^{\text{UE}}} g_{km}^{(i)} \bar{x}_{k'm}^d + w_k^d \\ &= \sum_{k' \in \mathcal{K}} \sum_{m \in \mathcal{M}_{k'}^{\text{UE}}} g_{km}^{(i)} (\hat{g}_{k'm}^{(i)})^* x_{k'}^d + w_k^d, \end{aligned} \quad (6)$$

where w_k^d is the downlink additive noise on user k with variance σ_d^2 . Based on (6) and the approach in [18], the effective downlink SINR is given by (8), where the expectation is over the random variables.

Based on (7) and (8), the achievable uplink and downlink spectral efficiencies (SEs) for user k are computed as $R_k^u = \log_2(1 + \text{SINR}_k^u)$ and $R_k^d = \log_2(1 + \text{SINR}_k^d)$, respectively. Note that these SE metrics can be used to quantify the uplink/downlink data transmission performance [18], [26]. Since the SINR expressions contain the estimated channel term $\hat{g}_{km}^{(i)}$, the SEs are directly impacted by the performance of channel estimation.

E. Problem Formulation

We use MSE of the channel estimation described in Sec. II-C for our PA performance metric. For user k served by the O-RUs in $\mathcal{M}_k^{\text{UE}}$, we define the MSE of the channel estimate in (4) as

$$\begin{aligned} \text{MSE}_k^{(i)} &= \mathbb{E} \left[\sum_{m \in \mathcal{M}_k^{\text{UE}}} \left| \hat{g}_{km}^{(i)} - g_{km}^{(i)} \right|^2 \right] \\ &= \sum_{m \in \mathcal{M}_k^{\text{UE}}} \mathbb{E} \left[\left| \hat{g}_{km}^{(i)} - g_{km}^{(i)} \right|^2 \right] \\ &= \sum_{m \in \mathcal{M}_k^{\text{UE}}} \mathbb{E} \left[\left| \sum_{\substack{k' \in \mathcal{K} \\ k' \neq k}} g_{k'm}^{(i)} (\mathbf{x}_k^{(i)})^H \mathbf{x}_{k'}^{(i)} + (\mathbf{x}_k^{(i)})^H \mathbf{w}_m^{(i)} \right|^2 \right] \\ &= \sum_{m \in \mathcal{M}_k^{\text{UE}}} \sum_{\substack{k' \in \mathcal{K} \\ k' \neq k}} \beta_{k'm}^{(i)} \left| (\mathbf{x}_k^{(i)})^H \mathbf{x}_{k'}^{(i)} \right|^2 + \sigma^2, \end{aligned} \quad (9)$$

where the expectation is taken over the channel and noise. The third equality holds as we substitute $\hat{g}_{km}^{(i)}$ with (4).

Next, the last equality holds since (i) $g_{km}^{(i)}$ and $\mathbf{w}_m^{(i)}$ are i.i.d. across k and m and (ii) $\mathbb{E}[|g_{km}^{(i)}|^2] = \beta_{km}^{(i)}$ and $\mathbb{E}[\|\mathbf{w}_m^{(i)}\|_2^2] = \sigma^2$. From (9), we see that the MSE is directly proportional to the interference caused by PC, and thus can be used as an effective metric to quantify the PA performance.

Since our system involves U agents, each of which handles the PA of user $k \in \mathcal{K}_u^{\text{DU}}$, we can formulate the PA optimization problem for agent u as

$$(\mathcal{P}_u) : \min_{\{\mathbf{x}_k^{(i)}\}_{k \in \mathcal{K}_u^{\text{DU}}}} \sum_{k \in \mathcal{K}} \text{MSE}_k^{(i)} \quad (10)$$

$$\text{s.t. } \mathbf{x}_k^{(i)} \in \mathcal{T}_u^{(i)}, \quad \forall k \in \mathcal{K}_u^{\text{DU}}, \quad (11)$$

$$\begin{aligned} \|\phi_{u,t}^{(i)}\|_2^2 &= 1, \quad \left(\phi_{u,t}^{(i)} \right)^H \phi_{u,t'}^{(i)} = 0 \text{ if } t \neq t', \\ \forall t, t' &= 1, 2, \dots, T_p. \end{aligned} \quad (12)$$

If $\beta_{km}^{(i)}$, $\forall k, m$ is known, one can directly evaluate $\sum_{k \in \mathcal{K}} \text{MSE}_k^{(i)}$ using (9) and solve $\mathcal{P}_u$ using an existing PA algorithm (e.g., the previous works [26]–[32]). However, in large-scale systems, such prior knowledge is often not available, and one can no longer evaluate the objective function in a straightforward manner. Suppose the knowledge is somehow available for the MSE to be evaluated, but some of these algorithms (e.g., PAs using the Tabu-search [31] and Hungarian algorithm [32]) still cannot be considered as the complexity becomes prohibitive for a large number of users. To address both issues, we solve $\mathcal{P}_u$ via a distributed learning framework, details of which are given in Sec. III. The decentralization imposed in this work allows our PA scheme to be much more scalable.

III. SCALABLE PA SCHEME FOR O-RAN CFMMIMO

In this section, we first describe how our proposed PA scheme is framed in O-RAN (Sec. III-A). Next, after providing preliminaries on MA-DRL (Sec. III-B), we design a Markov game model perceiving our PA problem (Sec. III-C), and show that the action selection in our learning framework corresponds to minimizing the PC (Theorem 1). Finally, we provide implementation details for our DRL-based PA scheme (Sec. III-D) and iterative CS algorithm (Sec. III-E).

A. Pilot Assignment Framework in O-RAN Architecture

Our PA scheme for CFmMIMO is designed based on the O-RAN architecture defined in Sec. II-A. Its conceptual block diagram is illustrated in Fig. 5. Here the PA is conducted under three different O-RAN control loops which have been described earlier in Fig. 1.

1) *RT loop*: We assume that a single round of channel estimation steps described in Sec. II-C takes place in each RT loop. Hence, we denote the index of each RT loop using the same notation used for indexing the channel estimation instance. In each RT loop i , users transmit

$$\text{SINR}_k^u = \frac{\rho_k \left| \mathbb{E} \left[\sum_{m \in \mathcal{M}_k^{\text{UE}}} (\hat{g}_{km}^{(i)})^* g_{km}^{(i)} \right] \right|^2}{\sum_{k' \in \mathcal{K}} \rho_{k'} \mathbb{E} \left[\sum_{m \in \mathcal{M}_k^{\text{UE}}} (\hat{g}_{km}^{(i)})^* g_{k'm}^{(i)} \right]^2 - \rho_k \left| \mathbb{E} \left[\sum_{m \in \mathcal{M}_k^{\text{UE}}} (\hat{g}_{km}^{(i)})^* g_{km}^{(i)} \right] \right|^2 + \sigma_u^2 \sum_{m \in \mathcal{M}_k^{\text{UE}}} \mathbb{E} \left[|\hat{g}_{km}^{(i)}|^2 \right]} \quad (7)$$

$$\text{SINR}_k^d = \frac{\left| \mathbb{E} \left[\sum_{m \in \mathcal{M}_k^{\text{UE}}} g_{km}^{(i)} (\hat{g}_{km}^{(i)})^* \right] \right|^2}{\left| \mathbb{E} \left[\sum_{m \in \mathcal{M}_k^{\text{UE}}} g_{km}^{(i)} (\hat{g}_{km}^{(i)})^* \right] \right|^2 - \left| \mathbb{E} \left[\sum_{m \in \mathcal{M}_k^{\text{UE}}} g_{km}^{(i)} (\hat{g}_{km}^{(i)})^* \right] \right|^2 + \sigma_d^2} \quad (8)$$

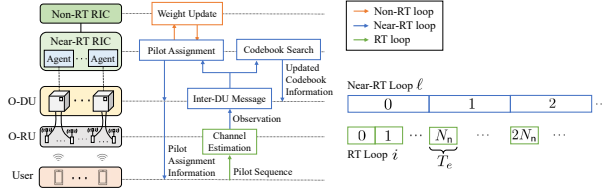


Fig. 5: A block diagram of the proposed PA scheme.

their assigned pilots, and the O-RU m completes the channel estimation to obtain $\hat{g}_{km}^{(i)}$ for $k \in \mathcal{K}_m^{\text{RU}}$.

2) *Near-RT loop*: Near-RT loop occurs once in every N_n RT loops. During each near-RT loop, O-DU u collects observation data, which we describe later in Sec. III-C, from the O-RUs in $\mathcal{M}_u^{\text{DU}}$ and transfers it to the near-RT RIC for learning. At the same time, each agent u in the near-RT RIC conducts PA on the users in $\mathcal{K}_u^{\text{DU}}$. We use $\ell = 0, 1, \dots, \lfloor \frac{N}{N_n} \rfloor$ to denote the index of near-RT loop, thus, ℓ -th near-RT loop occurs during the $N_n \ell$ -th RT loop (or the $N_n \ell$ -th channel estimation instance). The relationship between i and ℓ is visualized in Fig. 5.

To further improve our PA performance, two acceleration techniques are introduced:

- *Inter-DU message passing*: We consider inter-DU message passing which occurs at each near-RT loop. The inter-DU connection is essential for fully realizing user-centric RU clusters in decentralized CFmMIMO [36], and we exploit this feature to improve our PA performance. With inter-DU messages, we aim to reinforce the data observed by the local group of O-RUs (i.e., O-RUs of $\mathcal{M}_u^{\text{DU}}$). The details on inter-DU message passing are provided in Sec. III-D.
- *Codebook searching*: We leverage the decentralization of our system and develop a CS algorithm that operates jointly with our PA scheme. We adopt the idea of quasi-orthogonal codebooks [42], [43] to be used across the agents. In multi-cell systems, where each cell conducts its own PA to the serving users, using non-identical orthogonal codebooks across the cells has shown improved system performance [42], [43]. Inspired by this, we rotate the codebook of each agent in an iterative manner to find the codebook orientation with the minimum MSE of channel estimation. The detailed steps of our CS scheme are provided in Sec. III-E.

3) *Non-RT loop*: The non-RT loop is utilized to handle time-insensitive tasks. In our PA scheme, the update of the learning parameters for near-RT RIC occurs over this loop. Here, the non-RT loop occurs once in every

N_{non} RT loops, and we denote $q = 0, 1, \dots, \lfloor \frac{N}{N_{\text{non}}} \rfloor$ as the non-RT loop index. As described in Fig. 1, a near-RT loop duration can be as short as 10 ms while the shortest duration for non-RT loop is a second [7]. Hence, we assume $N_{\text{non}} \gg N_n$.

B. Preliminaries on MA-DRL

MA-DRL addresses scenarios where multiple agents perform simultaneous decision-making based on a Markov game model [44]. For our decentralized PA problem, we define MA-DRL using a tuple $(\{\mathbf{S}_u^{(\ell)}\}_{u \in \mathcal{U}}, \{\mathbf{a}_u^{(\ell)}\}_{u \in \mathcal{U}}, \{r_u^{(\ell)}\}_{u \in \mathcal{U}})$, where $\mathbf{S}_u^{(\ell)}$, $\mathbf{a}_u^{(\ell)}$, and $r_u^{(\ell)}$ are respectively the *state*, *action*, and *reward* of the agent u during the ℓ -th near-RT loop. For each loop ℓ , agent u with a state $\mathbf{S}_u^{(\ell)}$ makes an action $\mathbf{a}_u^{(\ell)}$ to interact with the environment. Subsequently, the agent makes an observation and computes a reward $r_u^{(\ell)}$ which helps to find the next state $\mathbf{S}_u^{(\ell+1)}$.

In the non-RT loop, once an agent has completed multiple interactions with the environment, its policy on action selection for a given state is optimized by updating the weights of its respective deep neural network (DNN). The action is selected based on the Q-value [45] denoted by $Q(\mathbf{S}_u^{(\ell)}, \mathbf{a}_u^{(\ell)})$. The Q-value quantifies the quality of an agent's action for a given state. Thus, it is important for the agent to obtain accurate Q-values to make correct decisions. In DRL, these Q-values are computed via a DNN, the weights of which are trained with experiences so that a correct (i.e., Q-value-maximizing) action can be selected upon each decision-making.

In perceiving our PA task as a multi-agent learning problem, there are two conditions to consider [46]. First, multiple agents making independent decisions simultaneously implies the environment is never seen as stationary to an action of a single agent. Second, due to the decentralized architecture, each agent only obtains a part of the observation available from the entire environment. Due to these conditions, in multi-agent learning, careful design of the Markov game model is crucial for achieving performance comparable to centralized learning.

C. Markov Game Model for Decentralized PA

In our O-RAN CFmMIMO setting, channel estimation is repeated for every RT loop i , forming a periodic interaction with the environment. The near-RT PA corresponds to action selection that affects the environment and resulting observation. Based on this, we formally

define each component of the tuple presented in Sec. III-B to perceive our PA task as a Markov game model.

1) *States*: To represent the PA status of agent u on users in $\mathcal{K}_u^{\text{DU}}$ at the start of near-RT loop ℓ , we define the state as $\mathbf{S}_u^{(\ell)} = \Phi_u^{(\ell)}$ which is a $|\mathcal{K}_u^{\text{DU}}| \times T_p$ sized matrix where

$$[\Phi_u^{(\ell)}]_{k,t} = \begin{cases} 1 & \text{if } \mathbf{x}_k^{(N_n\ell)} = \phi_{u,t}^{(N_n\ell)}, \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

PC occurs when users share a pilot, and this can be indicated by the *ones* in each column of $\Phi_u^{(\ell)}$. Hence, $\Phi_u^{(\ell)}$ can become an effective means to represent the condition of PA for each agent, and we aim to have the agents accurately perceive the relationship between their PA (i.e., their actions) and the resulting PC.

2) *Actions*: We consider sequential updates on the pilots, where the pilot of only a single user is changed with every action. If we consider actions that assign pilots to all $|\mathcal{K}_u^{\text{DU}}|$ users at once, this would lead our action space to take $T_p^{|\mathcal{K}_u^{\text{DU}}|}$ possible combinations and suffer from the ‘‘curse of dimensionality’’. We hence define actions as an ordered pair indicating the user of interest and the pilot to be assigned, respectively. The action of agent u at near-RT PA ℓ is formally defined as $\mathbf{a}_u^{(\ell)} = (k, t)$, where $k \in \mathcal{K}_u^{\text{DU}}$ and $t \in \{1, 2, \dots, T_p\}$. With this setting, there are total $|\mathcal{K}_u^{\text{DU}}|T_p$ possible actions for agent u to take, resulting in a more computationally scalable action space.

3) *Rewards*: We propose to compute the reward of each agent u on the ℓ -th near-RT PA based on the average sum-power of the channel estimates obtained by the O-RUs. Note that, for each action (i.e., near-RT PA) taken by an agent, N_n channel estimations are conducted by O-RU m to acquire a set of $\hat{\mathbf{g}}_m^{(i)}$ for $N_n\ell \leq i < N_n(\ell+1)$. Using this information, the O-RU m computes

$$p_{km}^{(\ell)} = \frac{1}{N_n} \sum_{n=0}^{N_n-1} \left| \hat{\mathbf{g}}_{km}^{(N_n\ell+n)} \right|^2 \quad (14)$$

on user $k \in \mathcal{K}_m^{\text{RU}}$ during the near-RT loop ℓ and sends it to the corresponding O-DU. At the end of this transfer, O-DU u collects different sets of $p_{km}^{(\ell)}$ from each O-RU $m \in \mathcal{M}_u^{\text{DU}}$ (i.e., $\{p_{km}^{(\ell)}\}_{k \in \mathcal{K}_m^{\text{RU}}\}_{m \in \mathcal{M}_u^{\text{DU}}}$). In decentralized PA, each agent $u \in \mathcal{U}$ is responsible for a disjoint subset of K users, and it is desirable for the agent to have access to $p_{km}^{(\ell)}$ from all O-RUs associated with the users (i.e., $\{p_{km}^{(\ell)}\}_{m \in \mathcal{M}_k^{\text{UE}}\}_{k \in \mathcal{K}_u^{\text{DU}}}$). However, as each O-DU u is only connected to O-RUs of $\mathcal{M}_u^{\text{DU}}$, $\{p_{km}^{(\ell)}\}_{m \in \mathcal{M}_k^{\text{UE}} \cap \mathcal{M}_u^{\text{DU}}\}_{k \in \mathcal{K}_u^{\text{DU}}}$ only gets collected by the agent. Hence, O-DU u ends up computing the observation data to be transferred to the agent u as $\bar{p}_u^{(\ell)} = \sum_{k \in \mathcal{K}_u^{\text{DU}}} \sum_{m \in \mathcal{M}_k^{\text{UE}} \cap \mathcal{M}_u^{\text{DU}}} p_{km}^{(\ell)}$.

Note that the rest of information required by agent u (i.e., $\{p_{km}^{(\ell)}\}_{m \in \mathcal{M}_k^{\text{UE}} \setminus \mathcal{M}_u^{\text{DU}}\}_{k \in \mathcal{K}_u^{\text{DU}}}$) has been collected by

other O-DUs. As mentioned earlier in Sec. III-A, since we consider inter-DU messages, this information can be transferred to each corresponding O-DU. Then, each O-DU u can now compute the reinforced observation data which is expressed as

$$\tilde{p}_u^{(\ell)} = \bar{p}_u^{(\ell)} + \sum_{k \in \mathcal{K}_u^{\text{DU}}} \sum_{m \in \mathcal{M}_k^{\text{UE}} \setminus \mathcal{M}_u^{\text{DU}}} p_{km}^{(\ell)} = \sum_{k \in \mathcal{K}_u^{\text{DU}}} \sum_{m \in \mathcal{M}_k^{\text{UE}}} p_{km}^{(\ell)}. \quad (15)$$

The observation data computed by O-DU u in (15) is transferred to agent u via a backhaul, and the reward for agent u at near-RT loop ℓ is subsequently computed using the mapping function

$$r_u^{(\ell)}(p) = (p_{\max} - p)/(p_{\max} - p_{\min}), \quad (16)$$

where $p = \tilde{p}_u^{(\ell)}$ by the availability of inter-DU message. The function (16) converts the observation data into a reward range such that lower values of p are rewarded higher. Here $[p_{\min}, p_{\max}]$ is the range of observation data, which we assume is set by the non-RT RIC.

We now show that the learning via our Markov model leads to taking an action that minimizes the degree of PC. The basic mechanism of learning we utilize is that, for each given state $\mathbf{S}_u$, we want the agent u to select the action that maximizes its Q-value [45], i.e.,

$$\mathbf{a}_u^* = \arg \max_{\mathbf{a}_u \in \mathcal{A}_u} Q(\mathbf{S}_u, \mathbf{a}_u), \quad (17)$$

where $\mathcal{A}_u$ is the set of all possible actions for agent u . The training in DRL is done by updating the network weights via regression toward the experiences obtained. The Q-value, which is the numerical output of the trained network, is then expected to follow the average of these experiences, i.e., the Q-value is updated through training to yield $Q(\mathbf{S}_u, \mathbf{a}_u) = \mathbb{E}[r_u(p)|(\mathbf{S}_u, \mathbf{a}_u)]$.

For each near-RT loop ℓ , the following theorem shows that, with inter-DU message passing, the action selected via (17) is the best action in terms of minimizing the degree of local PC.

Theorem 1. With $\tilde{p}_u^{(\ell)}$ available, for a given state $\mathbf{S}_u^{(\ell)}$, taking the action $\mathbf{a}_u^{(\ell)}$ which satisfies (17) is equivalent to finding the action that minimizes the degree of pilot contamination occurring on local users in $\mathcal{K}_u^{\text{DU}}$ during the near-RT loop ℓ , which is expressed as

$$\sum_{k \in \mathcal{K}_u^{\text{DU}}} \sum_{m \in \mathcal{M}_k^{\text{UE}}} \sum_{n=0}^{N_n-1} \sum_{\substack{k' \in \mathcal{K} \\ k' \neq k}} \beta_{k'm}^{(N_n\ell+n)} \left| (\mathbf{x}_k^{(N_n\ell)})^H \mathbf{x}_{k'}^{(N_n\ell)} \right|^2. \quad (18)$$

Proof. First, in terms of the parameters defined in our model, we find the expected reward at near-RT loop ℓ for a given state-action pair $(\mathbf{S}_u^{(\ell)}, \mathbf{a}_u^{(\ell)})$, which is given as

$$\mathbb{E}[r_u^{(\ell)}(\tilde{p}_u^{(\ell)})|(\mathbf{S}_u^{(\ell)}, \mathbf{a}_u^{(\ell)})] = \frac{p_{\max} - \mathbb{E}[\tilde{p}_u^{(\ell)}]}{p_{\max} - p_{\min}}, \quad (19)$$

where the equality holds from (16). Recalling (17), the learning conducted at each agent u aims to find the action

achieving the maximum Q-value $Q(\mathbf{S}_u, \mathbf{a}_u)$, which we discussed to yield $\mathbb{E}[r_u(p)](\mathbf{S}_u, \mathbf{a}_u)$. Thus, the action selection mechanism of agent u can be expressed as

$$\mathbf{a}_u^{(\ell)} = \arg \max_{\mathbf{a}_u \in \mathcal{A}_u} \mathbb{E}[r_u^{(\ell)}(\tilde{p}_u^{(\ell)}) | (\mathbf{S}_u^{(\ell)}, \mathbf{a}_u)]. \quad (20)$$

Now combining (19) and (20), we can say that

$$\begin{aligned} \mathbf{a}_u^{(\ell)} &= \arg \min_{\mathbf{a}_u \in \mathcal{A}_u} \sum_{k \in \mathcal{K}_u^{\text{DU}}} \sum_{m \in \mathcal{M}_k^{\text{UE}}} \mathbb{E}[p_{km}^{(\ell)}] \\ &= \arg \min_{\mathbf{a}_u \in \mathcal{A}_u} \frac{1}{N_n} \sum_{k \in \mathcal{K}_u^{\text{DU}}} \sum_{m \in \mathcal{M}_k^{\text{UE}}} \sum_{n=0}^{N_n-1} \mathbb{E} \left[\left| \hat{g}_{km}^{(N_n \ell + n)} \right|^2 \right], \end{aligned} \quad (21)$$

where the first and second equalities are obtained using (15) and (14), respectively. Now, for $n = 0, 1, \dots, N_n - 1$, using (4) we have

$$\begin{aligned} \mathbb{E} \left[\left| \hat{g}_{km}^{(N_n \ell + n)} \right|^2 \right] &= \mathbb{E} \left[\left| g_{km}^{(N_n \ell + n)} \right|^2 \right] + \sum_{\substack{k' \in \mathcal{K} \\ k' \neq k}} \mathbb{E} \left[\left| g_{k'm}^{(N_n \ell + n)} \right|^2 \right] \\ &\quad + \mathbb{E} \left[\left| (\mathbf{x}_k^{(N_n \ell + n)})^H \mathbf{x}_{k'}^{(N_n \ell + n)} \right|^2 \right] + \mathbb{E} \left[\left| (\mathbf{x}_k^{(N_n \ell + n)})^H \mathbf{w}_m^{(N_n \ell + n)} \right|^2 \right] \\ &= \beta_{km}^{(N_n \ell + n)} + \xi_{km}^{(\ell, n)} + \sigma^2, \end{aligned} \quad (22)$$

where $\xi_{km}^{(\ell, n)} = \sum_{\substack{k' \in \mathcal{K} \\ k' \neq k}} \beta_{k'm}^{(N_n \ell + n)} |(\mathbf{x}_k^{(N_n \ell + n)})^H \mathbf{x}_{k'}^{(N_n \ell + n)}|^2$ reflects the PC discussed in Sec. II-E. By the definition of $\hat{g}_{km}^{(i)}$ in (4), taking the expectation of $|\hat{g}_{km}^{(i)}|^2$ leaves only the autocorrelation terms for $\hat{g}_{km}^{(i)}$ and $\mathbf{w}_m^{(i)}$, corresponding to $\beta_{km}^{(N_n \ell + n)} = \mathbb{E}[|g_{km}^{(N_n \ell + n)}|^2]$ and $\sigma^2 = \mathbb{E}[|(\mathbf{x}_k^{(N_n \ell + n)})^H \mathbf{w}_m^{(N_n \ell + n)}|^2]$ in (22), since the channel and noise are assumed uncorrelated across k and m .

Now, since (i) $\xi_{km}^{(\ell, n)}$ is the only term that is impacted by action $\mathbf{a}_u$, i.e., $\beta_{km}^{(N_n \ell + n)}$ and σ^2 in (22) are independent from PA and (ii) $\mathbf{x}_k^{(i)}$ only changes once every N_n RT loops, i.e., $\mathbf{x}_k^{(N_n \ell + n)}$ is fixed for $n = 0, 1, \dots, N_n - 1$, by ignoring $\frac{1}{N_n}$ as a scaling factor, (21) is equivalent to

$$\mathbf{a}_u^{(\ell)} = \arg \min_{\mathbf{a}_u \in \mathcal{A}_u} \sum_{k \in \mathcal{K}_u^{\text{DU}}} \sum_{m \in \mathcal{M}_k^{\text{UE}}} \sum_{n=0}^{N_n-1} \sum_{k' \in \mathcal{K}, k' \neq k} \beta_{k'm}^{(N_n \ell + n)} \left| (\mathbf{x}_k^{(N_n \ell)})^H \mathbf{x}_{k'}^{(N_n \ell)} \right|^2, \quad (23)$$

which represents the degree of PC at near-RT loop ℓ over the users in $\mathcal{K}_u^{\text{DU}}$. ■

From Theorem 1, we conclude that learning based on our Markov games model is equivalent to performing the pilot update which minimizes the interference due to PC at each near-RT PA. According to (18), the PA made at each near-RT loop ℓ couples with the pathloss occurring over the corresponding N_n RT loops. Since we do not assume prior knowledge on the pathloss $\beta_{km}^{(i)}$, we cannot evaluate the exact MSE. However, through the reward we define and the learning mechanism of DRL, we can still design our PA scheme such that the MSE performance is

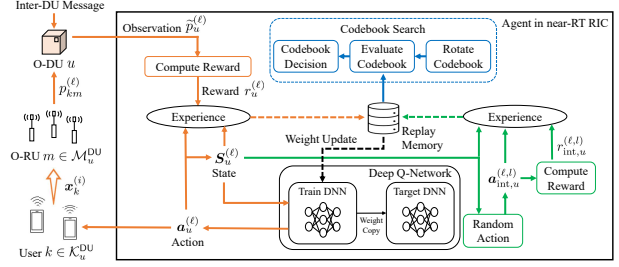


Fig. 6: A block diagram overview of our PA scheme, consisting of non-RT DNN training and near-RT PA updates.

improved over time. For static scenarios, where $\beta_{km}^{(i)}$ is constant over i , all the actions taken over near-RT loops (i.e., the entire series of successive pilot updates) are contributing to look for a single optimal PA solution that minimizes the sum-MSE. On the other hand, for mobile scenarios, each action is led to focus on minimizing the sum-MSE resulted from the current channel statistics by leveraging the past information. Our PA scheme is designed to cope with time varying small-scale and large-scale fading factors upon continuous training.

D. MA-DRL-based Pilot Assignment Scheme

Given the setting in the previous subsections, we describe our PA scheme in detail using the MA-DRL framework to find the solution to our decentralized PA problem. Our PA scheme trains learning algorithms to perform sequential pilot updates that aim to reduce the sum average power of channel estimates across the users. As shown by Theorem 1, minimizing the sum average power of channel estimates is equivalent to minimizing the overall MSE of channel estimation since both expressions share the PC term that is directly impacted by PA. Through sequential updates, our algorithm follows a greedy search framework that significantly reduces the dimension of the action space for a more practical MSE minimization. We incorporate MA-DRL via the deep Q-network (DQN) that utilizes neural network layers for approximating Q-values. An individual DQN is implemented at each agent in the near-RT RIC for distributed learning. Fig. 6 provides an overview of our methodology, which is also outlined in Alg. 1. We detail each of the steps in the following:

Near-RT PA: At $\ell = 0$, each agent u randomly assigns one of the T_p sequences in $\mathcal{T}_u^{(0)}$ to its associated users in $\mathcal{K}_u^{\text{DU}}$, from which the state $\mathbf{S}_u^{(0)}$ is generated. For each subsequent near-RT loop ℓ , the agent u takes an action $\mathbf{a}_u^{(\ell)}$ via an ϵ -greedy method [45] to update one user's pilot sequence and obtain a new state $\mathbf{S}_u^{(\ell+1)}$. If the agent decides to take its action based on Q-values, the state $\mathbf{S}_u^{(\ell)}$ is used as a $|\mathcal{K}_u^{\text{DU}}| \times T_p$ input to the DQN, which outputs the Q-value vector of size $|\mathcal{K}_u^{\text{DU}}| T_p$. The action with the highest Q-value is then selected. Since N_n RT channel estimations occur during a single loop of near-RT PA, each O-DU u collects $\{p_{km}^{(\ell)}\}_{k \in \mathcal{K}_u^{\text{DU}}}$ from the O-RUs

Algorithm 1: Proposed Pilot Assignment Scheme

```

1 Input: Pilot length  $T_p$ , number of RT loops  $N$ , number
  of RT loops per near-RT loop  $N_n$ , number of internal
  loops  $L$ , set of users managed by O-DU  $u$   $\mathcal{K}_u^{\text{DU}}$ , set
  of O-RUs managed by O-DU  $u$   $\mathcal{M}_u^{\text{DU}}$ , set of users
  served by O-RU  $m$   $\mathcal{K}_m^{\text{RU}}$ , set of O-RUs serving the
  user  $k$   $\mathcal{M}_k^{\text{UE}}$ , training period, update period
2 Initialize near-RT loop index  $\ell = 0$ ; randomize the
  parameter vectors  $\theta_u^{\text{tr}}$  and  $\theta_u^{\text{ta}}$ 
3 Generate  $\mathcal{T}_u^{(N_n\ell)}$ ; randomly assign  $\{\phi_k^{(N_n\ell)}\}_{k \in \mathcal{K}_u^{\text{DU}}}$ 
4 for  $\ell = 0$  to  $N$  do
5   Compute  $\mathbf{S}_u^{(\ell)}$  using (13)
6   if  $\ell > 0$  then
7     Compute  $r_u^{(\ell-1)}(\tilde{p}_u^{(\ell-1)})$  using (16); store
       $(\mathbf{S}_u^{(\ell-1)}, \mathbf{a}_u^{(\ell-1)}, r_u^{(\ell-1)}(\tilde{p}_u^{(\ell-1)}), \mathbf{S}_u^{(\ell)})$  in the
      memory
8     for  $l = 0$  to  $L - 1$  do
9       Select  $\mathbf{a}_{\text{int},u}^{(\ell,l)}$  randomly; compute  $\mathbf{S}_{\text{int},u}^{(\ell,l)}$ 
        using (13); compute  $r_{\text{int},u}^{(\ell,l)}$  using (25)
10      Store  $(\mathbf{S}_u^{(\ell)}, \mathbf{a}_{\text{int},u}^{(\ell,l)}, r_{\text{int},u}^{(\ell,l)}, \mathbf{S}_{\text{int},u}^{(\ell,l)})$  in the
        memory
11    if  $\epsilon$ -greedy then select  $\mathbf{a}_u^{(\ell)}$  randomly else
       $\mathbf{a}_u^{(\ell)} = \arg \max_{\mathbf{a}_u} Q_{\theta_u^{\text{tr}}}(\mathbf{S}_u^{(\ell)}, \mathbf{a}_u)$ 
12    Update the PA according to  $\mathbf{a}_u^{(\ell)}$ 
13    for  $i = 0$  to  $N_n - 1$  do
14      User  $k \in \mathcal{K}_u^{\text{DU}}$  transmits  $\phi_k^{(N_n\ell+i)}$ ; O-RU
         $m \in \mathcal{M}_u^{\text{DU}}$  estimates  $\{\hat{g}_{km}^{(i)}\}_{k \in \mathcal{K}_m^{\text{RU}}}$  using (4)
15    if  $\text{mod}(\ell, \text{training period}) = 0$  then generate a
      batch from the memory and train  $\theta_u^{\text{tr}}$  via SGD
      on (24)
16    if  $\text{mod}(\ell, \text{update period}) = 0$  then set  $\theta_u^{\text{ta}} = \theta_u^{\text{tr}}$ 
17 Output: Updated pilot sequences  $\{\phi_k^{(N)}\}_{k \in \mathcal{K}_u^{\text{DU}}}$ 

```

in $\mathcal{M}_u^{\text{DU}}$ and computes $\tilde{p}_u^{(\ell)}$ with the aid of inter-DU message passing. The O-DU transfers $\tilde{p}_u^{(\ell)}$ to its agent in the near-RT RIC, which computes $r_u^{(\ell)}(p)$ and stores an experience tuple $(\mathbf{S}_u^{(\ell)}, \mathbf{a}_u^{(\ell)}, r_u^{(\ell)}(p), \mathbf{S}_u^{(\ell+1)})$ in a *replay memory* of size D_m .

Non-RT DNN Training: The learning of each agent u is carried out by two DNNs called the *train* and *target* networks [33], [47], where their network parameter vectors are denoted by θ_u^{tr} and θ_u^{ta} , respectively. Once enough experiences have been collected in the memory, a *mini-batch* of size D_b is randomly selected from the memory and used to update θ_u^{tr} minimizing the loss:

$$L(\theta_u^{\text{tr}}) = \mathbb{E} \left[y_\ell - Q_{\theta_u^{\text{tr}}}(\mathbf{S}_u^{(\ell)}, \mathbf{a}_u^{(\ell)}) \right], \quad (24)$$

where $y_\ell = r_u^{(\ell)} + \gamma \max_{\mathbf{a}} Q_{\theta_u^{\text{ta}}}(\mathbf{S}_u^{(\ell+1)}, \mathbf{a})$ with γ being the discount factor. Here $Q_{\theta}(\mathbf{S}, \mathbf{a})$ represents the Q-value for a given pair of state $\mathbf{S}$ and action $\mathbf{a}$ computed via a DNN of weight vector θ . The update is done using stochastic gradient descent (SGD). This step is equivalent to the training phase of supervised learning in the sense

that each experience becomes a training datapoint and the label is replaced by the reward. Here, the weights of θ_u^{tr} are periodically copied to target network θ_u^{ta} , with the length of this period as a design parameter.

Experience generation: By the O-RAN capability, the value of N_n can vary and impact the rate of experiences being collected to each agent, i.e., the number of experiences collected for a given amount of time varies by N_n . If N_n is too large, a sufficient size of data required to perform effective training may not be collected within a desired time period. To resolve the issue and utilize time more efficiently, we exploit the architecture of O-RAN and introduce an internal experience-generating loop inside the near-RT RIC. This internal loop is executed L times to take additional L hypothetical actions during a single near-RT loop. In particular, once a *real* experience is obtained via the ℓ -th near-RT loop, we generate L extra *virtual* experiences by taking a random action and evaluating the corresponding reward for each internal loop. We define the reward by the l -th internal loop of the ℓ -th near-RT PA as

$$r_{\text{int},u}^{(\ell,l)}(p) = \left(1 - \kappa_u^{(\ell,l)} / \kappa_{\max}\right) r_u^{(\ell)}(p), \quad (25)$$

where $\kappa_u^{(\ell,l)} = \left| \sum_{t=1}^{T_p} \left(\sum_{k \in \mathcal{K}_u^{\text{DU}}} (\phi_k^{(N_n\ell)})^H \mathbf{x}_k^{(\ell,l)} - \left\lfloor \frac{|\mathcal{K}_u^{\text{DU}}|}{T_p} \right\rfloor \right) \right|$ is the penalty for having more than necessary number of users sharing the same pilot sequence and $\kappa_{\max} = 2|\mathcal{K}_u^{\text{DU}}|(T_p - 1)/T_p$ is the maximum penalty obtainable. Integrating this internal loop alongside near-RT PA, we can generate L more experiences to accelerate the convergence of our scheme and train our DNNs to favor sequence combinations that have more evenly spread number of users across T_p sequences.

E. Iterative Codebook Search (CS) Algorithm

We describe our CS algorithm that is designed to work with the PA scheme in Sec. III-D. As each agent assigns pilots to its local users using the codebook $\mathcal{T}_u^{(i)}$, CS is iteratively conducted so that the final set of U codebook sets, when combined with our PA solution, suppresses the PC to the minimum degree. We detail each of the steps in the following.

First, we assign each agent $u \in \mathcal{U}$ with an identical codebook, i.e., $\mathcal{T}_1^{(0)} = \mathcal{T}_2^{(0)} = \dots = \mathcal{T}_U^{(0)}$, and initiate our PA scheme without CS to ensure that the agents first learn and improve their PA only based on the interference resulted from pilot sharing. We design our algorithm to begin its iterative CS only after the learning on PA is stabilized so that the PA and CS do not impair each other from converging. We determine the PA of agent u to be stable when the state $\mathbf{S}_u^{(\ell)}$ remains unchanged over N_{cs} near-RT loops. Once the agent u has given the same PA for N_{cs} consecutive times at the end of near-RT loop ℓ_u^* , the agent is perceived as stable and becomes subject for

CS. Note that ℓ_u^* is likely to vary for each agent due to our decentralized PA framework.

If we design our agents to conduct CS in parallel, it becomes difficult to accurately evaluate a codebook as multiple actions simultaneously affect the environment. Hence, we propose to have each agent take a turn and conduct CS while the rest of agents is paused from the search. To implement a such design, we define an operation called the CS run in which an isolated CS is conducted for each agent $u \in \mathcal{U}_{cs}^{(v)}$, where $\mathcal{U}_{cs}^{(v)}$ is the set of agents subject for CS during the v -th CS run. For each isolated search, the following steps are performed.

Suppose it is the turn of the w -th element of $\mathcal{U}_{cs}^{(v)}$, denoted by $u_{v,w}$, to perform the isolated CS, where $w = 1, 2, \dots, |\mathcal{U}_{cs}^{(v)}|$. We first define $\ell_{v,w}$ to be the near-RT loop in which the agent $u_{v,w}$ begins its search. We also let N_s define the number of near-RT loops to be spent for codebook evaluation. During the first N_s near-RT loops (i.e., $\ell_{v,w} \leq \ell < \ell_{v,w} + N_s$), the quality of current codebook matrix $\mathbf{T}_{v,w}^{\text{old}} = [\phi_{u_{v,w},1}^{(N_s \ell_{v,w})}, \phi_{u_{v,w},2}^{(N_s \ell_{v,w})}, \dots, \phi_{u_{v,w},T_p}^{(N_s \ell_{v,w})}]$ is evaluated by computing

$$\bar{r}_{v,w}^{\text{old}} = \frac{1}{N_s} \sum_{n=0}^{N_s-1} r_{u_{v,w}}^{(\ell_{v,w}+n)}(p), \quad (26)$$

which is the average of the most N_s recent rewards collected at agent $u_{v,w}$ via our PA algorithm. Note that (26) represents the quality of PA performed using the codebook $\mathcal{T}_{u_{v,w}}^{(N_s \ell_{v,w})}$.

After obtaining (26), the agent generates a $T_p \times T_p$ column-normalized zero-mean Gaussian random perturbation matrix $\mathbf{P}_{v,w}$ and computes the rotation matrix as $\mathbf{R}_{v,w} = \sqrt{1 - \eta_{u_{v,w}}^2} \mathbf{I}_{T_p} + \eta_{u_{v,w}} \mathbf{P}_{v,w}$, where $\eta_{u_{v,w}} = 1 - \frac{\ell_{v,w} - \ell_{u_{v,w}}^*}{N/N_s - \ell_{u_{v,w}}^*}$ is the perturbation degree designed to decrease with $\ell_{v,w}$ to obtain a converged solution. Note that larger $\eta_{u_{v,w}}$ results in $\mathbf{R}_{v,w}$ with greater perturbation.

After acquiring $\mathbf{R}_{v,w}$, the agent rotates the current codebook to obtain a new codebook matrix

$$\mathbf{T}_{v,w}^{\text{new}} = \text{proj}(\mathbf{R}_{v,w} \mathbf{T}_{v,w}^{\text{old}}), \quad (27)$$

where $\text{proj}(\cdot)$ is the projection function for which we use the Gram-Schmidt orthogonalization algorithm [48]. The set of T_p columns in $\mathbf{T}_{v,w}^{\text{new}}$ is then used as a new codebook for agent $u_{v,w}$ during the next N_s near-RT loops (i.e., $\ell_{v,w} + N_s \leq \ell < \ell_{v,w} + 2N_s$). After these N_s near-RT loops, where a set of N_s rewards using the new codebook are collected by our PA algorithm, the agent computes

$$\bar{r}_{v,w}^{\text{new}} = \frac{1}{N_s} \sum_{n=N_s}^{2N_s-1} r_{u_{v,w}}^{(\ell_{v,w}+n)}(p), \quad (28)$$

to evaluate the quality of the new codebook. At this point, agent $u_{v,w}$ has evaluated (26) and (28) from using two

different codebooks $\mathbf{T}_{v,w}^{\text{old}}$ and $\mathbf{T}_{v,w}^{\text{new}}$, respectively, and determines which codebook to keep by the end of search using the following criterion

$$\mathbf{T}_{u_{v,w}}^{(N_s(\ell_{v,w}+2N_s))} = \begin{cases} \mathbf{T}_{v,w}^{\text{new}} & \text{if } \bar{r}_{v,w}^{\text{new}} > \bar{r}_{v,w}^{\text{old}}, \\ \mathbf{T}_{v,w}^{\text{old}} & \text{otherwise.} \end{cases} \quad (29)$$

Algorithm 2: Proposed Codebook Search Scheme

```

1 Input: Pilot length  $T_p$ , number of consistent PAs
   required for stability  $N_{cs}$ , codebook evaluation interval
    $N_s$ , number of RT loops  $N$ , set of agents  $\mathcal{U}$ 
2 Initialize CS run index  $v = 0$ , set of agents subject for
   CS  $\mathcal{U}_{cs}^{(v)} = \emptyset$ , the counter for agent  $u$   $a_u = 0$ ,
    $CS_{\text{run}} = 0$ , and  $CS_{\text{iso}} = 0$ ; assign identical codebook
   for all  $u \in \mathcal{U}$ ; capture  $\mathbf{S}_u^{(0)}$  using (13)
3 for  $\ell = 1$  to  $N$  do
4   for  $u \in \mathcal{U}$  do
5     Capture  $\mathbf{S}_u^{(\ell)}$  using (13)
6     if  $\mathbf{S}_u^{(\ell)} = \mathbf{S}_u^{(\ell-1)}$  then  $a_u = a_u + 1$  else
7        $a_u = 0$ ; if  $a_u = N_{cs}$  then  $\ell_u^* = \ell$ 
8   if  $CS_{\text{run}} = 0$  then
9      $\mathcal{U}_{cs}^{(v)} = \{u \in \mathcal{U} | \ell_u^* < \ell\}$ ; if  $|\mathcal{U}_{cs}^{(v)}| > 0$  then
10       $w = 1$  and  $CS_{\text{run}} = 1$ 
11   if  $CS_{\text{run}} = 1$  then
12     if  $CS_{\text{iso}} = 0$  then  $\ell_{v,w} = \ell$ ;  $CS_{\text{iso}} = 1$ 
13     if  $CS_{\text{iso}} = 1$  then
14       if  $\ell = \ell_{v,w} + N_s - 1$  then compute  $\bar{r}_{v,w}^{\text{old}}$ 
15         using (26); apply new codebook  $\mathbf{T}_{v,w}^{\text{new}}$ 
16         using (27)
17       if  $\ell = \ell_{v,w} + 2N_s - 1$  then
18         Compute  $\bar{r}_{v,w}^{\text{new}}$  using (28); decide
19         codebook using (29);  $w = w + 1$  and
20          $CS_{\text{iso}} = 0$ 
21   if  $w > |\mathcal{U}_{cs}^{(v)}|$  then  $v = v + 1$ ;  $CS_{\text{run}} = 0$ 
16 Output: Rotated codebook  $\mathcal{T}_u^{(N)}$ ,  $\forall u \in \mathcal{U}$ 

```

As the CS described above runs for each agent in $\mathcal{U}_{cs}^{(v)}$, total $2N_s |\mathcal{U}_{cs}^{(v)}|$ near-RT loops are spent to complete the CS run v . For every run, each agent tries a new codebook generated using a random rotation and decides to keep whichever codebook that yields higher reward. The algorithm starts its very first CS run at $\ell = \min_{u \in \mathcal{U}} \ell_u^*$ and continuously conducts each subsequent CS run. By changing the codebook only when it is determined to be better, the algorithm proceeds to find the best set of U codebooks that minimizes the degree of PC. Note that, in order to evaluate the codebooks, our CS scheme utilizes the reward $r_u^{(\ell)}(p)$, which is obtained during our PA scheme. Therefore, no additional information needs to be collected the O-DUs to conduct the CS. The overall procedure for our CS scheme is summarized in Alg 2.

IV. NUMERICAL EVALUATION

In this section, we evaluate our pilot assignment (PA) scheme under O-RAN CFmMIMO channel estimation scenarios with various system parameters. We analyze

both channel estimation performance and computational complexity to discuss the scalability and practicality of our method. In addition, we compare the performance of our proposed approach against different baselines which include [31], [32] among others.

A. Simulation Setup, Performance Metrics, and Baselines

We consider different combinations of O-DUs ($U = 4$), single-antenna O-RUs ($M = 96$), and single-antenna users ($K \in \{24, 36\}$) placed in an area of $100 \text{ m} \times 150 \text{ m}$ geometry to create O-RAN CFmMIMO systems. We assume the same number of O-RUs connected to each O-DU (i.e., $|\mathcal{M}_u^{\text{DU}}| = \frac{M}{U}, \forall u$) and the same number of users paired with each agent in the near-RT RIC (i.e., $|\mathcal{K}_u^{\text{DU}}| = \frac{K}{U}, \forall u$). We set a channel estimation interval $T_e = 1 \text{ ms}$, implying our O-RAN RT loop occurs once every 1 ms. Each scenario is simulated with a maximum $N = 10000$ RT loops, which corresponds to 10 seconds with $T_e = 1 \text{ ms}$. We assume $N_n = 10$ RT loops occur per O-RAN near-RT loop and $L = 9$ internal experience generation per near-RT loop unless stated otherwise. For mobile scenarios, we generate initial ($i = 0$) and final ($i = N$) positions for each user such that the velocity v_k ranges from 0 m/s (or 0 km/h) to 1.4 m/s (or 5 km/h). Then, for each $i = 0, 1, \dots, N$, the position of each user is updated according to v_k . Such a mobile scenario for 96×24 CFmMIMO (where $M \times K$ refers to M O-RUs and K users) with $U = 4$ O-DUs (equivalently, $U = 4$ agents in the near-RT RIC) is demonstrated in Fig. 7. The large-scale fading factor $\beta_{km}^{(i)}, \forall k, m$ is assumed to follow the 3GPP urban-micro line-of-sight pathloss model [40] with carrier frequency $f_c = 2 \text{ GHz}$, O-RU height of 10 m, and user height of 1.5 m. We consider a pilot length of $T_p = 4$ and a RU cluster size of $M_k^{\text{UE}} = 8, \forall k$ unless stated otherwise. For our codebook search (CS) scheme, we consider an agent to be stable if the PA is consistent for $N_{cs} = 100$ consecutive times and assume the codebook evaluation interval $N_s = 5$.

We use the same DQN design for all agents: one convolutional neural network (CNN) with 32 kernels of size $|\mathcal{K}_u^{\text{DU}}| \times T_p$ followed by two fully connected layers of width $|\mathcal{K}_u^{\text{DU}}|T_p$. All layers use ReLU activation and the Adam optimizer with learning rate of 0.001. The discount factor for the weight update is set $\gamma = 0.5$. We also set the size of replay memory $D_m = 1000$ and train the neural network using $D_b = 128$ samples per minibatch. The train network weights are updated via SGD and synchronized with the target network whenever 200 and 400 new additional experiences are stored in the replay memory, respectively. We implement ϵ -greedy action-selection [45] with the probability of selecting a random action in the ℓ -th near-RT loop computed as $\epsilon_\ell = e^{-(\Gamma/N)N_n\ell}$, where $\Gamma = 15$ is the scaling factor.

We now describe the baseline methods for performance comparison. We first consider a random assignment

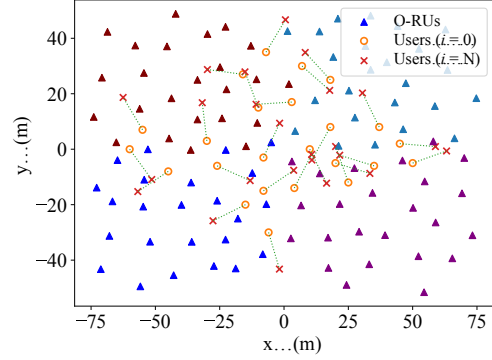


Fig. 7: Geographical layout of O-RAN CFmMIMO with $U = 4$, $M = 96$, and $K = 24$. O-RUs connected to the same O-DU have the same color. Each user moves from the initial (circle) to the final position (cross) in 10 seconds.

strategy (PA-RA) where pilots are assigned randomly for each user. The strategy does not impose any complexity but yields mediocre channel estimation performance. We also consider an exhaustive method (PA-ES) where the entire T_p^K combinations of pilots are searched to find the PA having the lowest MSE, which is evaluated using β_{km} and σ^2 assumed to be known a priori. PA-ES provides the best MSE performance but is considered impractical in terms of computational complexity as the search space exponentially increases with the number of users. We also consider two PA algorithms in the recent literature: PA strategies using Tabu-search [31] and Hungarian [32] methods. To solve our PA problem, we design the algorithms to utilize sum-MSE as the metric. The sum-MSE expression is a function of the assigned pilots and therefore provides an effective metric to optimize the PA. Tabu-search-based PA (PA-TS) utilizes the Tabu-search framework to find the MSE-minimizing pilot combination while the PA using the Hungarian algorithm (PA-HG) iteratively solves a reward matrix to find the PA solution. Both require prior knowledge of β_{km} and σ^2 and have computational complexity that becomes prohibitive as the number of users increases. Also, these methods do not consider practical framework (e.g., decentralized PA) but simply rely on centralized processing, which makes them hard to integrate into O-RAN architecture. Also, they do not take the user mobility into account and fail to adapt to the change imposed by the time-varying dynamics.

We next discuss our PA scheme to be simulated for detailed evaluation. We conduct the learning process described in Sec. III-D with inter-DU message passing (PA-DRL+MSG), i.e., $\tilde{p}_u^{(\ell)}$ is computed by each O-DU and transferred to the agent. In addition, we apply the CS scheme described in Sec. III-E along with PA-DRL+MSG (PA-DRL+MSG+CS) to assess the improvement brought by adjusting the codebook orientation across O-DUs. As our PA scheme is specifically tailored to the O-RAN architecture, practical implementation with scalable computation is possible. Since we base our

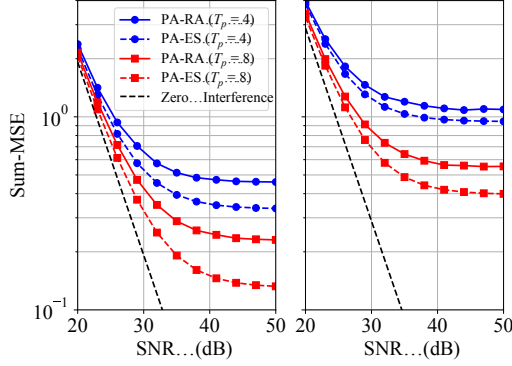


Fig. 8: Sum-MSE vs. SNR with $K = 24$ (left) and $K = 36$ (right).

learning on the DRL framework, which offers training that is adaptive to the dynamic environment, and conduct CS that checks the real-time observation, our PA scheme can reflect the user mobility.

We evaluate the performance of our proposed PA scheme over two different metrics: (i) the sum-MSE defined for the objective function in $\mathcal{P}_u$, i.e., $\sum_{k \in \mathcal{K}} \text{MSE}_k^{(i)}$, and (ii) the runtime it takes to obtain the converged MSE. For the numerical results, we run each scenario 50 times and take their average to make our analysis statistically significant. In each run, we use the same O-RU topology but randomize the locations of K users. All the algorithms were implemented in Python and tested on hardware with a Tesla T4 GPU and 12.7 GB RAM.

B. Performance of O-RAN CFmMIMO

1) *Impact of PA on channel estimation:* We first demonstrate the impact of PA on channel estimation in our O-RAN CFmMIMO system. In Fig. 8, we provide sum-MSE versus signal to noise ratio (SNR) plots for different values of T_p and K , where we define SNR as $\frac{1}{\sigma^2}$, and discuss the following observations. First, we see that $T_p = 8$ yields lower MSE than $T_p = 4$. It is expected since the number of users sharing the same pilot tends to be smaller for larger T_p . Next, for lower SNRs, the MSE gap between PA-RA and PA-ES is not significant since the noise dominantly contributes to channel estimation error. However, as SNR increases, interference due to PC becomes more dominant and forces an error floor, making the curves almost horizontal. For the case of 50 dB SNR, we find that with $T_p = 4$ and $K = 24$, optimizing PA can reduce the sum-MSE up to 27%. For the remaining experiments, we use SNR of 50 dB to focus on the interference-limited regime.

2) *Impact of O-RAN parameters:* We assess the impact of O-RAN-dependent system parameters on the performance of our PA scheme. The sum-MSE curves (moving-averaged with a window size of 500) of PA-DRL+MSG over the O-RAN RT loop for different values of N_n and L are shown in Fig. 9. Recall that N_n is the number of RT loops for a single near-RT loop, and L is the number

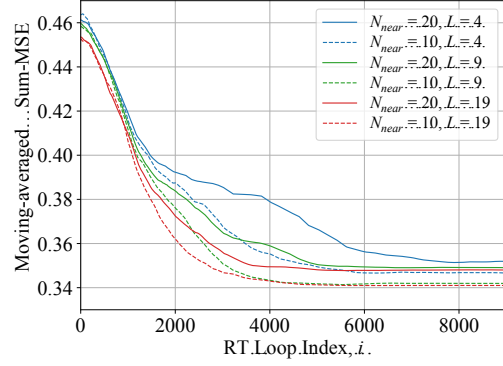


Fig. 9: Sum-MSE vs. RT loop with $K = 24$.

of extra experiences generated per near-RT loop by the agent. Both N_n and L are dependent on the capability of O-RAN in which CFmMIMO network is built.

Now, we make the following observations from Fig. 9. First, regardless of the parameter values, our scheme shows stabilized (i.e., converged) sum-MSE performance, which verifies the effectiveness of our learning when implemented under O-RAN architecture. Second, a lower N_n yields improved MSE regardless of L . Here, lower N_n implies more near-RT loops during the given number of RT loops, allowing agents to interact with the environment more frequently and take more actions to find better solutions. Third, a higher L (more internal loops) allows us to achieve greater sum-MSE reduction in earlier RT loops, validating that more experiences collected in replay memory within the same period are beneficial. Thus, as the size of the dataset increases, our scheme is expected to find the PA faster with low sum-MSE.

C. Performance Comparison Against Different Baselines

Now we assess our proposed PA scheme and compare its performance with several baselines over two metrics: channel estimation MSE and algorithm runtime.

1) *Comparison in MSE:* First, we consider static scenarios, i.e., $v_k = 0, \forall k$. The plots showing sum-MSE performance (moving-averaged with a window size of 500) over RT loops for $K = 24$ and $K = 36$ are presented in Fig. 10 and Figs. 11, respectively. Note that the PA solutions obtained by PA-HG, PA-TS, and PA-ES required true pathloss information and were fixed for the entire RT loops. Among these approaches, it is verified from both figures that PA-ES yields much better MSE performance than PA-TS and PA-HG. We also considered the case where PA-HG and PA-TS are conducted using estimated pathloss, which yields a considerable performance gap compared to the case of using true pathloss knowledge. The estimated pathloss is computed by averaging ten instantaneous power measurements from isolated signal transmissions, which yields around a 25% error magnitude compared to true pathloss. Given that these baselines require prior

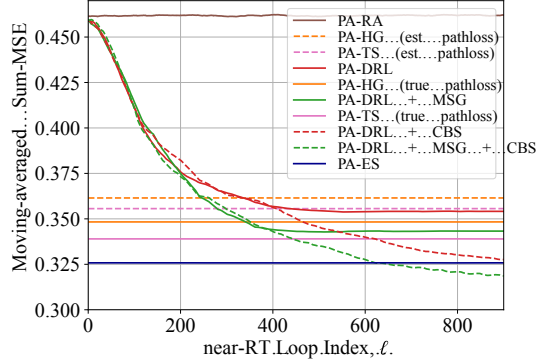


Fig. 10: Sum-MSE performance of different PA schemes over 24 stationary users

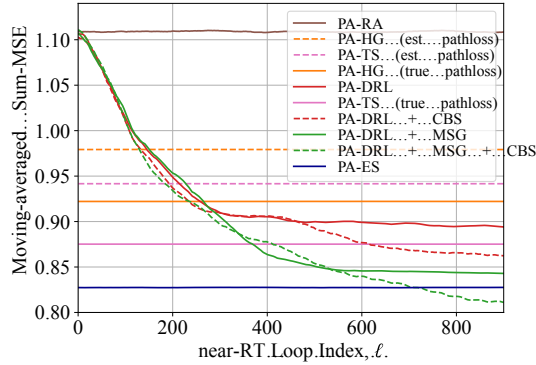


Fig. 11: Sum-MSE performance of different PA schemes over 36 stationary users

knowledge (preferably accurate) to achieve the given performance, our learning-based PA scheme, which does not impose such requirement, is still able to show competitive performance against them. PA-DRL+MSG clearly outperforms PA-HG and PA-TS with estimated pathloss and provides comparable performance with the ones with true pathloss. Once we utilize CS scheme, our proposed PA-DRL+MSG+CBS shows significant improvement and achieves better performance than PA-ES as a result of jointly optimizing both PA and codebook orientation. In Table II, we extend our sum-MSE evaluation up to $K = 72$. We observe that, regardless of K , the relative performance among the algorithms is preserved, which verifies that our proposed scheme obtains consistent improvements as the system size increases.

Note that we compare our proposed scheme only to centralized baselines for two reasons. First, we aim to minimize the performance loss due to decentralization, and thus we can directly evaluate how well our algorithm performs in terms of MSE compared to centralized PA. Second, to the best of our knowledge, there is no existing work on decentralized PA to have a valid comparison.

Next, we consider scenarios in which users move over time (i.e., $\beta_{km}^{(i)}$ changes over i , and $v_k > 0, \forall k \in \mathcal{K}$). Fig. 12 shows the sum-MSE performance (moving-averaged with a window size of 500) of different PA

TABLE II
Sum-MSE performance of various PA schemes over different K values

K	24	36	48	60	72
PA-HG (true pathloss)	0.348	0.949	1.705	2.765	4.179
PA-TS (true pathloss)	0.339	0.875	1.657	2.677	4.049
PA-DRL+MSG+CBS	0.319	0.811	1.455	2.361	3.617

TABLE III
UL and DL SEs in bits/s/Hz for different PA algorithms.

Algorithm	$K = 24$		$K = 36$	
	UL	DL	UL	DL
PA-HG (true pathloss)	9.79	9.03	7.65	7.10
PA-TS (true pathloss)	9.89	9.11	7.81	7.23
PA-DRL + MSG + CBS	10.18	9.40	8.06	7.55

algorithms with $K = 24$ evaluated at three different user velocities: 1, 5, and 10 km/h. The values of user velocity were selected so that the users still remain in the coverage area after their movement. PA solution obtained by the baselines at the beginning (i.e., $i = 0$) becomes less effective as time advances, showing a different degree of steady increase by the velocity. Unlike the baselines, as our schemes make their decisions based on the real-time observations, in PA-DRL+MSG+CBS, PAs can be performed in an adaptive manner, maintaining its performance as shown in Fig. 12. Hence, our scheme can yield competitive performance with the prior knowledge-constrained baselines under a dynamic environment.

While our evaluation assumes ideal link connections, imperfect connection links are a significant factor for practical systems. Hence, we extended our experiment to assume probabilistic link failures on O-FH and inter-DU connections and analyze their impact on the channel estimation performance. We observe that the overall sum-MSE increases with the increase of link failure probabilities. Such a result is expected as failed connections prevent the agents from collecting necessary observations and computing accurate rewards. For more details, see Appendix A in our online technical report [49].

To consider a wider range of scenarios, we also considered our experiment under three additional setups: non-uniform user distribution, Rician channel fading with different k -factor values [39], and correlated pathloss with different shadowing variance [26]. We observe that the sum-MSE performance of our scheme and the baselines remains unchanged except for the case of increased shadowing. With greater shadowing, the expected degree of PC increases, and this results in increased sum-MSE of channel estimation for all algorithms. For more details, see Appendix B in our online technical report [49].

Overall, our scheme provides satisfactory performance in MSE as it exploits the decentralized architecture of O-RAN CFmMIMO via distributed learning and CS.

2) *Comparison in SE*: We evaluate uplink (UL) and downlink (DL) achievable SEs by computing $\sum_{k \in \mathcal{K}} R_k^u$ and $\sum_{k \in \mathcal{K}} R_k^d$, respectively, for different PA algorithms. The result is provided in Table III, and we make the following observations. First, the SE performance with

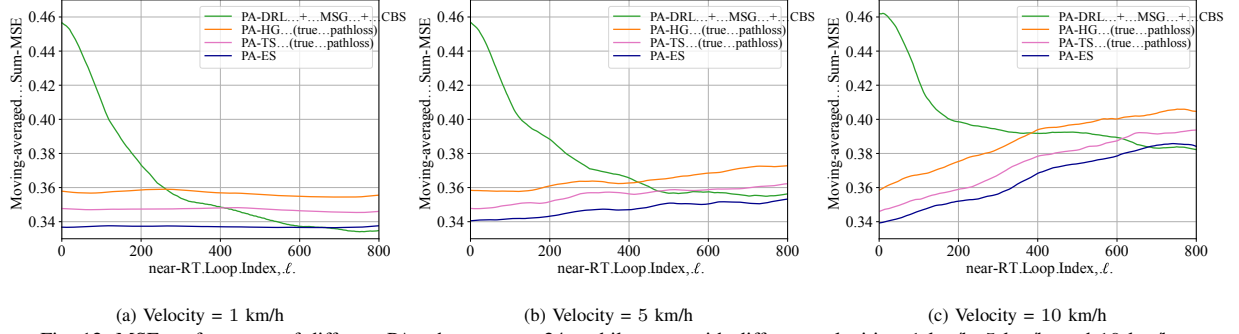


Fig. 12: MSE performance of different PA schemes over 24 mobile users with different velocities: 1 km/h, 5 km/h, and 10 km/h.

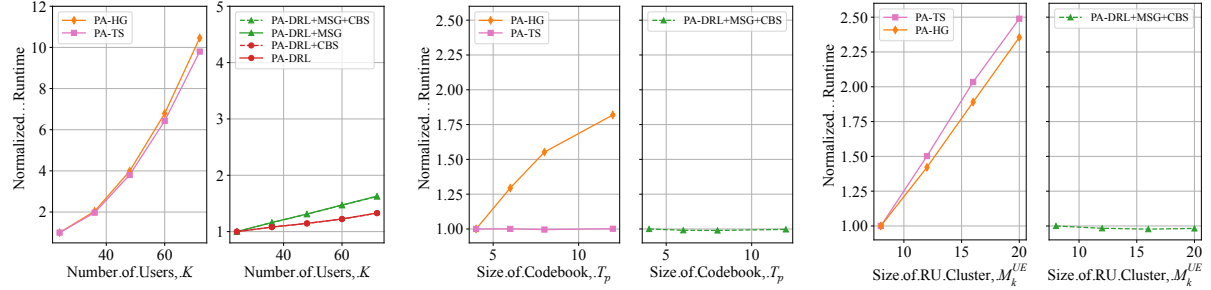


Fig. 13: Relative runtime measurements of various PA schemes over different system parameters. Measurements for each PA algorithm are normalized to the case of the smallest parameter value.

$K = 36$ is lower than the one with $K = 24$, which is resulted from increased user density imposing a larger degree of PC. Second, the performance order we observe in Figs. 10 and 11 is preserved for the sum-rate performance. This verifies that improving channel estimation accuracy via effective PA results in the SE improvement in both uplink and downlink phases.

3) *Behavior comparison in relative runtime:* Now, we evaluate the computational complexity behavior of different PA algorithms. In Fig. 13a, we vary the number of users K from 24 to 72 and measure the relative runtimes (i.e., the increase of runtime with respect to the case $K = 24$) of different PA methods. We do not report absolute runtimes as our implementations do not factor in inter-node bandwidth, latency, and other implementation/hardware-specific factors that can impact head-to-head comparisons among the algorithms. In Fig. 13a, we see that both centralized PA-TS and PA-HG exhibit a polynomial increase in K . Hence, these centralized algorithms can be rendered impractical when PA needs to be executed over a CFmMIMO network with a growing network size. On the other hand, our PA algorithm shows a linear increase in the relative runtime. Overall, combined with the MSE results, we see that our decentralized PA approach obtains advantages both in terms of pilot contamination and scalability. The steady increase in runtimes from our PA scheme are due to the utilization of (i) O-RAN architecture where duration-varying tasks are distributed across the network

and (ii) DNNs of fixed size which only perform a forward computation to determine each pilot update step over near-RT loop. We observe a slight increase in runtime when we consider inter-DU messages into our PA scheme because generating a new set of messages imposes extra computations. Note that our CS scheme barely adds any runtime as it utilizes the rewards already computed during our PA scheme. We hence conclude that our low-complexity PA scheme is a scalable strategy that supports large-scale CFmMIMO systems. As we have previously shown, our PA scheme provides consistently strong performance in terms of sum-MSE regardless of K , which highlights the scalability advantage of our approach, especially for large-scale systems. Note that PA-ES, which is the best baseline in MSE minimization, requires an extreme amount of runtime as it searches over all T_p^K combinations of PA. On the other hand, PA-RA requires no extra runtime but shows much worse MSE performance than other PA schemes (Figs. 10 and 11).

Next, we assess the runtime required to conduct PA schemes over different values of T_p (Fig. 13b) and M_k^{UE} (Fig. 13c), where we normalize the measurements in the same way as Fig. 13a. For varying T_p (the size of codebook), only PA-HG shows undesirable behavior in complexity since the size of the reward matrix used in the Hungarian algorithm depends on T_p . With respect to M_k^{UE} (the size of RU cluster), both PA-TS and PA-HG display a linear increase. Meanwhile, our proposed scheme provides consistent runtimes for both parameters,

which verifies their scalability to support a network with large system parameters.

V. CONCLUSION

In this paper, we developed a learning-based PA scheme for the decentralized CFmMIMO system framed in O-RAN. We adopted O-RAN as a practical system architecture where distinct network functions and multi-timescale control loops efficiently govern the framework of our scheme. After formulating the PA problem and designing the corresponding Markov game model, we developed a PA algorithm based on the MA-DRL framework. We also developed a CS scheme that accelerates our learning-based PA in MSE-minimization without any significant additional complexities. Compared to the state-of-the-art baselines, our approach provided satisfactory performance in terms of both channel estimation MSE and computational scalability. Furthermore, unlike most of the existing PA strategies, our scheme does not require any prior channel knowledge.

REFERENCES

- [1] M. Z. Chowdhury, M. Shahjalal, S. Ahmed, and Y. M. Jang, "6G wireless communication systems: Applications, requirements, technologies, challenges, and research directions," *IEEE Open J. the Commun. Soc.*, vol. 1, pp. 957–975, 2020.
- [2] Y. L. Lee, D. Qin, L.-C. Wang, and G. H. Sim, "6G massive radio access networks: Key applications, requirements and challenges," *IEEE Open J. Veh. Technol.*, vol. 2, pp. 54–66, 2021.
- [3] S. K. Singh, R. Singh, and B. Kumbhani, "The evolution of radio access network towards open-RAN: Challenges and opportunities," in *IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2020, pp. 1–6.
- [4] S. Niknam, A. Roy, H. S. Dhillon, S. Singh, R. Banerji, J. H. Reed, N. Saxena, and S. Yoon, "Intelligent O-RAN for beyond 5G and 6G wireless networks," 2020. [Online]. Available: <https://arxiv.org/abs/2005.08374>
- [5] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," 2022. [Online]. Available: <https://arxiv.org/abs/2202.01032>
- [6] 3GPP, "NG-RAN; architecture description," Tech. Rep. TS 38.401 V17.2.0, Sep 2022.
- [7] O-RAN Alliance, "O-RAN architecture description," Tech. Rep. V07.00, 2022.
- [8] M. Mohsin, J. M. Batalla, E. Pallis, G. Matorakis, E. K. Markakis, and C. X. Mavroumoustakis, "On analyzing beamforming implementation in O-RAN 5G," *Electronics*, vol. 10, no. 17, 2021.
- [9] T. Hewavithana, A. Chopra, B. Mondal, S. Wong, A. Davydov, and M. Majmundar, "Overcoming channel aging in massive MIMO basestations with open RAN fronthaul," in *IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2022, pp. 2577–2582.
- [10] O-RAN Alliance, "O-RAN working group 1 massive MIMO use cases," Tech. Rep. V01.00, 2022.
- [11] 3GPP, "Study on new radio access technology: Radio access architecture and interfaces," Tech. Rep. TR 38.801 V14.0.0, March 2017.
- [12] N.-N. Dao, Q.-V. Pham, N. H. Tu, T. T. Thanh, V. N. Q. Bao, D. S. Lakew, and S. Cho, "Survey on aerial radio access networks: Toward a comprehensive 6G access infrastructure," *IEEE Commun. Surv. & Tut.*, vol. 23, no. 2, pp. 1193–1225, 2021.
- [13] C. Pham, F. Fami, K. K. Nguyen, and M. Cheriet, "When RAN intelligent controller in O-RAN meets multi-UAV enable wireless network," *IEEE Trans. Cloud Comput.*, pp. 1–15, 2022.
- [14] O-RAN Alliance, "O-RAN working group 1 use cases detailed specification," Tech. Rep. V09.00, 2022.
- [15] C. Studer, S. Medjkouh, E. Goultas, T. Goldstein, and O. Tirkkonen, "Channel charting: Locating users within the radio environment using channel state information," *IEEE Access*, vol. 6, pp. 47 682–47 698, 2018.
- [16] G. Interdonato, E. Björnson, H. Q. Ngo, P. Frenger, and E. G. Larsson, "Ubiquitous cell-free massive MIMO communications," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 197, 2019.
- [17] J. Zhang, S. Chen, Y. Lin, J. Zheng, B. Ai, and L. Hanzo, "Cell-free massive MIMO: A new next-generation paradigm," *IEEE Access*, vol. 7, pp. 99 878–99 888, 2019.
- [18] J. Zhang, E. Björnson, M. Matthaiou, D. W. K. Ng, H. Yang, and D. J. Love, "Prospective multiple antenna technologies for beyond 5G," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 8, pp. 1637–1660, 2020.
- [19] E. Björnson and L. Sanguinetti, "Making cell-free massive MIMO competitive with MMSE processing and centralized implementation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 77–90, 2020.
- [20] H. Yang and T. L. Marzetta, "Energy efficiency of massive MIMO: Cell-free vs. cellular," in *IEEE 87th Veh. Technol. Conf. (VTC Spring)*, 2018, pp. 1–5.
- [21] E. Björnson and L. Sanguinetti, "Scalable cell-free massive MIMO systems," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 4247–4261, 2020.
- [22] G. Interdonato, P. Frenger, and E. G. Larsson, "Scalability aspects of cell-free massive MIMO," in *IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.
- [23] H. He, X. Yu, J. Zhang, S. H. Song, and K. B. Letaief, "Cell-free massive MIMO for 6G wireless communication networks," *J. Commun. Inf. Netw.*, vol. 6, pp. 321–335, 2021.
- [24] H. A. Ammar, R. Adve, S. Shahbazpanahi, G. Boudreau, and K. V. Srinivas, "User-centric cell-free massive MIMO networks: A survey of opportunities, challenges and solutions," *IEEE Commun. Surv. & Tut.*, vol. 24, no. 1, pp. 611–652, 2022.
- [25] H. Yin, D. Gesbert, and L. Cottatellucci, "Dealing with interference in distributed large-scale MIMO systems: A statistical approach," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 942–953, 2014.
- [26] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta, "Cell-free massive MIMO versus small cells," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1834–1850, March 2017.
- [27] R. Sabbagh, C. Pan, and J. Wang, "Pilot allocation and sum-rate analysis in cell-free massive MIMO systems," in *IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.
- [28] S. Chen, J. Zhang, E. Björnson, J. Zhang, and B. Ai, "Structured massive access for scalable cell-free massive MIMO systems," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1086–1100, 2021.
- [29] M. Attarifar, A. Abbasfar, and A. Lozano, "Random vs structured pilot assignment in cell-free massive MIMO wireless networks," in *IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2018, pp. 1–6.
- [30] H. Liu, J. Zhang, S. Jin, and B. Ai, "Graph coloring based pilot assignment for cell-free massive MIMO systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9180–9184, 2020.
- [31] H. Liu, J. Zhang, X. Zhang, A. Kurniawan, T. Juhana, and B. Ai, "Tabu-search-based pilot assignment for cell-free massive MIMO systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2286–2290, 2020.
- [32] S. Buzzi, C. D'Andrea, M. Fresia, Y.-P. Zhang, and S. Feng, "Pilot assignment in cell-free massive MIMO based on the hungarian algorithm," *IEEE Wireless Commun. Lett.*, vol. 10, no. 1, pp. 34–37, 2021.
- [33] W. Li, W. Ni, H. Tian, and M. Hua, "Deep reinforcement learning for energy-efficient beamforming design in cell-free networks," in *IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2021, pp. 1–6.
- [34] F. Fredj, Y. Al-Eryani, S. Maghsudi, M. Akrou, and E. Hossain, "Distributed beamforming techniques for cell-free wireless net-

works using deep reinforcement learning,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 2, pp. 1186–1201, 2022.

- [35] Y. Zhao, I. G. Niemegeers, and S. M. H. De Groot, “Dynamic power allocation for cell-free massive MIMO: Deep reinforcement learning methods,” *IEEE Access*, vol. 9, pp. 102 953–102 965, 2021.
- [36] V. Ranjbar, A. Girycki, M. A. Rahman, S. Pollin, M. Moonen, and E. Vinogradov, “Cell-free mMIMO support in the O-RAN architecture: A PHY layer perspective for 5G and beyond networks,” *IEEE Commun. Standards Mag.*, vol. 6, no. 1, pp. 28–34, 2022.
- [37] 3GPP, “NR; radio resource control (RRC) protocol specification,” Tech. Rep. TS 38.331, Sep 2022.
- [38] T. Kim, D. J. Love, and B. Clerckx, “MIMO systems with limited rate differential feedback in slowly varying channels,” *IEEE Trans. Commun.*, vol. 59, no. 4, pp. 1175–1189, 2011.
- [39] D. Tse and V. Pramod, *Fundamentals of Wireless Communication*. New York, NY, USA: Cambridge University Press, 2005.
- [40] 3GPP, “Evolved universal terrestrial radio access (E-UTRA); further advancements for E-UTRA physical layer aspects,” Tech. Rep. TR 36.814 V9.2.0, March 2017.
- [41] Y. Liu, Z. Tan, H. Hu, L. J. Cimini, and G. Y. Li, “Channel estimation for OFDM,” *IEEE Commun. Surv. & Tut.*, vol. 16, no. 4, pp. 1891–1908, 2014.
- [42] C. Wang, Z. Zhang, and H. C. Papadopoulos, “On-the-fly uplink training and pilot code design for massive MIMO cellular networks,” *2020 Information Theory and Applications Workshop (ITA)*, pp. 1–6, 2020.
- [43] A. Chowdhury, P. Sasmal, and C. R. Murthy, “Comparison of orthogonal vs. union of subspace based pilots for multi-cell massive MIMO systems,” in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020, pp. 1–5.
- [44] G. Qu, A. Wierman, and N. Li, “Scalable reinforcement learning for multi-agent networked systems,” *Operations Research*, vol. 70, no. 6, pp. 3601–3628, 2022.
- [45] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
- [46] A. Feriani and E. Hossain, “Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial,” *IEEE Commun. Surv. & Tut.*, 2021.
- [47] J. Ge, Y.-C. Liang, J. Joung, and S. Sun, “Deep reinforcement learning for distributed dynamic MISO downlink-beamforming coordination,” *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6070–6085, 2020.
- [48] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. The Johns Hopkins University Press, 1996.
- [49] M. S. Oh, A. B. Das, S. Hosseinalipour, T. Kim, D. J. Love, and C. G. Brinton, “A decentralized pilot assignment algorithm for scalable O-RAN cell-free massive MIMO,” 2023. [Online]. Available: <https://arxiv.org/abs/2301.04774>

Myeung Suk Oh (Student Member, IEEE) received the B.S. degree from Georgia Institute of Technology, USA and the M.S. degree from the Korea Advanced Institute of Science and Technology (KAIST) in 2013 and 2015, respectively. He is currently pursuing his Ph.D. degree in electrical engineering at Purdue University, IN, USA. His research interests include learning-aided channel estimations and sensor-based localization techniques in future wireless systems.

Anindya Bijoy Das (Member, IEEE) received the B.Sc. degree from the Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2014, and the M.Engg. and Ph.D. degrees from the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA, in 2018 and 2022, respectively. He is currently a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His research interests include federated learning, coding theory, signal processing and machine learning. He is the recipient of the Karas Award 2022 in Iowa State University for his outstanding thesis in the area of Mathematical and Physical Sciences and Engineering. He also received research excellence and teaching excellence awards from the Department of Electrical and Computer Engineering, Iowa State University.

Seyyedali Hosseinalipour (Member, IEEE) received the B.S. degree in electrical engineering from Amirkabir University of Technology, Tehran, Iran, in 2015 with high honor and top-rank recognition. He then received the M.S. and Ph.D. degrees in electrical engineering from North Carolina State University, NC, USA, in 2017 and 2020, respectively. He was the recipient of the ECE Doctoral Scholar of the Year Award (2020) and ECE Distinguished Dissertation Award (2021) at North Carolina State University. He was a postdoctoral researcher at Purdue University, IN, USA from 2020 to 2022. He is currently an assistant professor at the Department of Electrical Engineering at the University at Buffalo (SUNY). His research interests include the analysis of modern wireless networks, synergies between machine learning methods and fog computing systems, distributed machine learning, and network optimization.

Taejoon Kim (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA. He is currently an Associate Professor and Chair’s Council Faculty of electrical engineering and computer science at the University of Kansas (KU). Prior to joining KU, he was a Senior Researcher at the Nokia Bell Laboratories, Berkeley, CA, a Postdoctoral Researcher at KTH, Stockholm, Sweden, and an Assistant Professor at the City University of Hong Kong. His research interests at the intersection of optimization, machine learning, and communications. He holds 29 issued U.S. patents. He was a recipient of the Harry Talley Excellence in Teaching Award and Miller Professional Development Award in Research from the KU School of Engineering. Along with the coauthors, he won The IEEE Communications Society Stephen O. Rice Prize in 2016 and IEEE PIMRC 2012 Best Paper Award.

David J. Love (Fellow, IEEE) received the B.S. (with highest honors), M.S.E., and Ph.D. degrees in electrical engineering from the University of Texas at Austin in 2000, 2002, and 2004, respectively. Since 2004, he has been with the Elmore Family School of Electrical and Computer Engineering at Purdue University, where he is now the Nick Trbovich Professor of Electrical and Computer Engineering. He was a member of the Executive Committee for the National Spectrum Consortium. He holds 32 issued U.S. patents. His research interests are in the design and analysis of broadband wireless communication systems, beyond-5G wireless systems, multiple-input multiple-output (MIMO) communications, millimeter wave wireless, software defined radios and wireless networks, coding theory, and MIMO array processing. Dr. Love is a Fellow of the American Association for the Advancement of Science (AAAS) and was named a Thomson Reuters Highly Cited Researcher (2014 and 2015). Along with his co-authors, he won best paper awards from the IEEE Communications Society (2016 Stephen O. Rice Prize and 2020 Fred W. Ellersick Prize), the IEEE Signal Processing Society (2015 IEEE Signal Processing Society Best Paper Award), and the IEEE Vehicular Technology Society (2010 Jack Neubauer Memorial Award).

Christopher G. Brinton (Senior Member, IEEE) is an Assistant Professor in the Elmore Family School of Electrical and Computer Engineering (ECE) at Purdue University. His research interest is at the intersection of networking, communications, and machine learning, specifically in fog/edge network intelligence, distributed machine learning, and data-driven wireless network optimization. Since joining Purdue ECE in fall 2019, Dr. Brinton has won the NSF CAREER Award (2022), ONR Young Investigator Program (YIP) Award (2022), DARPA Young Faculty Award (YFA, 2022), Intel Rising Star Faculty Award (2022), and roughly \$10M in sponsored research projects as a PI or co-PI. He has also been awarded Purdue College of Engineering Faculty Excellence Awards in Early Career Research (2023), Early Career Teaching (2023), and Online Learning (2022), as well as the Purdue ECE Outstanding Faculty Mentor Award (2020), Ruth and Joel Spira Outstanding Teacher Award (2020), and Purdue Seed for Success Award (2019). Dr. Brinton received the PhD (with honors) and MS Degrees from Princeton in 2016 and 2013, respectively, both in Electrical Engineering.