Received 14 October 2023; revised 15 November 2023; accepted 21 November 2023. Date of publication 28 November 2023; date of current version 18 December 2023.

Digital Object Identifier 10.1109/OJCOMS.2023.3337044

Deep Reinforcement Learning Approach for HAPS User Scheduling in Massive MIMO Communications

SARA SHARIFI, HESAM KHOSHKBARI, GEORGES KADDOUM, AND OUASSIMA AKHRIF (Senior Member, IEEE)

Department of Electrical Engineering, École de Technologie Supérieure Montréal, Montreal, QC H3C 1K3, Canada

CORRESPONDING AUTHOR: S. SHARIFI (e-mail: sara.sharifi.1@ens.etsmtl.ca)

This work was supported by the Defence Research and Development Canada.

ABSTRACT In this paper, we devise a deep SARSA reinforcement learning (DSRL) user scheduling algorithm for a base station (BS) that uses a high-altitude platform station (HAPS) as a backup to serve multiple users in a wireless cellular network. Considering a realistic scenario, we assume that only the outdated channel state information (CSI) of the terrestrial base station (TBS) is available in our defined user scheduling problem. We model this user scheduling problem using a Markov decision process (MDP) framework, aiming to maximize the sum-rate while minimizing the number of active antennas at the HAPS. Our performance analysis shows that the sum-rate obtained with our proposed DSRL algorithm is close to the optimal sum-rate achieved with an exhaustive search method. We also develop a heuristic optimization method to solve the user scheduling problem at the BS. We show that for a scenario where perfect CSI is not available, our proposed DSRL algorithm outperforms the heuristic optimization method.

INDEX TERMS HAPS, deep reinforcement learning, user scheduling, Markov decision process.

I. INTRODUCTION

TERRESTRIAL base stations (TBSs) can no longer keep up with the growing data demand due to the rapid developments of new applications, such as virtual reality, smart healthcare, and intelligent transportation systems. To overcome this issue, non-terrestrial base stations (NTBSs), such as unmanned aerial vehicles (UAVs) and high-altitude platform stations (HAPSs) can be used as backup transmitters in the next generation of wireless communications. There are extensive studies on the applications and challenges associated with UAVs [1], [2] and HAPSs [3], [4] in wireless communications. For instance, HAPSs can be used as an alternative in emergencies where the TBS is not available [5]. Also, it can be used to increase the coverage area for mobile users [6] and to serve IoT users in different applications [7]. It is worth mentioning that HAPSs benefit from better uplink communication, quasi-stationary locations, less path loss attenuation, and lower latency than satellites. Also, HAPSs have a significant advantage over UAVs in terms of longer battery life. Thus, HAPSs are great candidates to be used as backup BSs to compensate users with low received SINR/SNR at TBSs [8].

In this context, several works have considered the resource allocation problems in non-terrestrial networks, e.g., the authors in [9] study beamforming techniques to mitigate interference between a HAPS and a terrestrial network. Assuming the coexistence of both TBSs and NTBSs, users should be associated with the proper access network to enhance the system data rate. Consequently, many works study user scheduling in integrated terrestrial and non-terrestrial networks (between NTBSs and terrestrial cells) to integrate NTBSs into existing terrestrial wireless networks [8], [10], [11]. In [11], the resource allocation problem in a vertical heterogeneous network (VHeNet) is studied to maximize the downlink throughput of ground users. The authors solve the problem in two stages: a short-term stage and a long-term stage. Access link association, backhaul link association, and power allocation are performed in the short-term stage while assuming a fixed location for the HAPS. First, the access and backhaul link association are optimized assuming a fixed power allocation. Then, the power allocation is solved using Taylor expansion. In the long-term stage, the shrink-and-realign method is used to determine the location of the HAPS.

All mentioned works use mathematical models that are computationally costly for systems with a large number of users and/or a large number of antennas at the BS/HAPS. Moreover, these methods cannot deal with network uncertainties. Below, we present the recent works that tackle the mentioned issues for solving resource allocation problems with reinforcement learning (RL)/deep RL (DRL) techniques.

II. RELATED WORKS

RL/DRL methods have been used in different communication network applications to solve the resource allocation problem [12], [13], [14], [15]. More specifically, several works have studied user/BS association using DRL in non-terrestrial networks. The authors in [16] study the transmission of information between two TBSs via a number of satellites and one HAPS. They use DRL for satellite association and to determine the location of the HAPS which maximizes the end-to-end rate. In [17], the authors apply DRL for multi-user access in non-terrestrial networks to maximize the sum-rate with minimum handoff requirements. Further, the authors perform user association to maximize the total throughput of the network while avoiding frequent handoffs resulting from the mobility of airborne vehicles. The authors of [18] studied the DRL-based user association in a VHetNet with an emphasize on the role of satellites. In [19], the authors study minimizing the age of information (AoI) in an intelligent transportation system (ITS) where UAVs collect the produced information by sensors on the vehicles to provide up-to-date data. To do so, the authors use DRL to optimize the UAVs' trajectory and schedule the sensor selection to minimize the AoI of the collected data. In a similar work [20], the authors use DRL to optimize the UAVs' trajectory and node selection to collect data. The main purpose in [20] is to minimize the AoI of information collected while accounting for constraints on the nodes' battery level. Similarly, the authors in [21] apply RL to minimize the AoI of the collected information from IoT sensors by determining the UAVs' trajectory and node selection.

Although instructive, the mentioned works do not consider the user scheduling problem in integrated terrestrial and non-terrestrial networks. In [19], [20], [21], there is no TBS and only UAVs perform user scheduling in order to minimize the AoI of the network. The authors do not consider TBSs or how NTBSs and TBSs can cooperate. Moreover, in [17], the authors only study the effect of user association between different airborne vehicles and do not assume any TBSs. Therefore, the authors do not investigate the effect of user association between a TBS and an NTBS to maximize the sum-rate of the network. In the coexistence of both a TBS and a HAPS, some users who experience a poor channel quality should be associated with the HAPS to receive a higher rate. On the other hand, due to the high distance between the HAPS from ground users, compared to the TBS, serving all users via HAPS is not optimal because the users that experience good channel conditions can receive higher rates from the TBS. Furthermore, given the limitations in the power supply and the number of antennas in HAPS, there is a limitation on the number of users that can be associated with HAPS. Moreover, adding an extra layer to the wireless network complicates network management. As a result, managing the inter-layer CSI sharing overhead is the first step in making it easier to integrate HAPS into existing terrestrial networks. Secondly, the proposed resource allocation methods should not impose a higher computational complexity compared with the currently used methods in terrestrial networks. Consequently, an efficient user scheduling policy should be implemented to maximize the network sum-rate while taking into account the limitations imposed by HAPS.

In our previous work [22], we utilized Deep Q-learning (DQL) to perform user association between a TBS and a HAPS. Though we showed promising results, it is noteworthy that the global channel state information (CSI) was used in [22]. Moreover, the agent's performance under imperfect CSI was not satisfactory compared to its performance under perfect CSI. Here, we redesign the state space and propose a novel algorithm that performs well under both perfect and imperfect CSI. Furthermore, we cut the overhead resulting from inter-layer CSI sharing.

III. CONTRIBUTIONS

In this study, considering a user association problem between a TBS and a HAPS where only outdated CSI is available, we use DRL to design a method to maximize the downlink sum-rate of available users in a wireless multi-input multi-output (MIMO) network. More specifically, here, our agent, i.e., the TBS, performs both user association and HAPS antenna selection at each time slot. To avoid sharing information between the HAPS and the TBS, our DRL agent only uses the CSI of the channels between the TBS antennas and the available users to make a decision. It is worth mentioning that, in contrast to current mathematical optimization algorithms that require full CSI knowledge of both the NTBS and TBS, in this work, the agent can make decisions based on the obtained channel coefficients of the link between its antennas and the allocated users only. Moreover, to further reduce the overhead in the network, we assume only the CSI of the served users by the TBS at the previous time slot will be updated, and thus the CSI of the remaining users (served by the HAPS at the previous time slot) remains unchanged. Meaning that the agent makes decisions by using outdated TBS's CSI.

Considering the challenges posed by the outdated CSI from TBS and the unknown CSI from HAPS, two significant issues arise for the DRL agent. The first issue is the tendency to overestimate Q-values during the learning phase due to incomplete information. The second challenge involves the need for extensive exploration to develop an optimal policy. To address these issues, we introduce an

on-policy DRL method employing the state-action-reward-state-action (SARSA) algorithm. This approach, which we term deep SARSA Reinforcement Learning (DSRL), effectively manages the outlined problems. Importantly, DSRL achieves this without increasing computational complexity, especially when compared to traditional DQL methods. Further, we develop a heuristic optimization-based user scheduling algorithm for the sake of comparison and show that under the perfect CSI case, our proposed DSRL approach performs roughly as well as the used heuristic method. For the imperfect CSI scenario, our proposed DSRL method outperforms both convex optimization and methods of optimization with uncertainty bounds. It is noted that the global CSI is required for the heuristic method used.

The main contributions of this work are summarized as follows:

- Our study focuses on optimizing user scheduling between a TBS and a HAPS to enhance the network's sum-rate, considering time-varying channels with factors like geometric attenuation, Doppler frequency, and shadow fading. We also emphasize minimizing circuit power consumption at the HAPS through efficient antenna selection, thereby defining a joint problem of HAPS antenna selection and user scheduling maximization.
- We employ a DSRL approach for solving our optimization problem, which avoids overestimation of Q-values and functions efficiently under imperfect CSI. The state space in our model is formulated to normalize channels between users and TBS, ensuring consistent agent performance even as channel estimation noise varies.
- A key feature of our method is that the DSRL agent relies only on the CSI of the channels between the TBS antennas and the available users for decision-making, thereby eliminating the need for information exchange between the HAPS and the TBS. This contrasts with traditional mathematical optimization algorithms that require complete CSI knowledge of both the NTBS and TBS. Furthermore, to reduce network overhead, our system updates only the CSI of users served by the TBS in the previous time slot, leaving the CSI of users served by the HAPS unchanged. This approach is thoroughly tested through extensive simulations, demonstrating its robustness and scalability when compared to other methods such as exhaustive search, optimization-based scheduling, DQL, and random selection.
- Furthermore, to provide a more realistic and comprehensive evaluation, our study includes scenarios with imperfect CSI. In these scenarios, our DSRL method outperforms traditional convex optimization and convex optimization with uncertainty bounds methods as shown in our comparative analysis. This highlights the stability and effectiveness of our approach in maintaining performance consistency, despite the varying accuracy of CSI data.

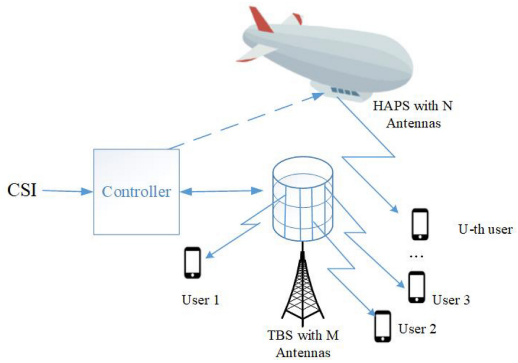

FIGURE 1. System model illustration.

The rest of the paper is organized as follows. The system model, problem formulation, and objective function of our study are illustrated in Section IV, Section V, and Section VI, respectively. Our two-stage proposed DSRL user scheduling algorithm is explained in Section VIII, and the heuristic optimization-based user scheduling algorithm is developed in Section IX. The simulation results are presented in Section X. Finally, the conclusion is provided in Section XI.

Notations: The upper-case boldfaced letters and lower-case boldfaced letters are used to represent matrices and vectors, respectively; the notation $(\cdot)^T$ is used to show the transpose of a vector/matrix; diag($\mathbf{s}$) creates a diagonal matrix with its diagonal entries being the elements of the vector $\mathbf{s}$. The Calligraphic fonts (e.g., $\mathcal{A}$) are used to signify sets, and the cardinality of a set $\mathcal{A}$, is denoted as $|\mathcal{A}|$. The notation $\mathbb{E}\{\cdot\}$ identifies the mathematical expectation; $[\cdot]_u$ extracts the u-th row of a matrix. The $||\cdot||$ is used for the Frobenius norm.

IV. SYSTEM MODEL

We consider a cellular system model where a HAPS, equipped with a uniform linear antenna array with N antennas, is considered to backup a TBS in the downlink data transmission. As shown in Figure 1, considering a TBS with M antennas in the center of a cell, we denote U as the total number of single-antenna users that demand data. Here, we assume that the system is slotted by T time slots, where $t = 1, 2, \ldots, T$, and it operates in time division duplex (TDD) mode. At each time slot t, the TBS uses the obtained CSI to select a subset of users to be served and assigns the remaining users to be served by the HAPS. Our main goal here is to devise a user scheduling algorithm that can maximize the sum-rate of the system.

In the considered system, we assume that the HAPS is equipped with smart antennas to provide a high quality of service (QoS) with a minimum circuit power consumption. More specifically, at each time slot t, the number of active antennas depends on the number of scheduled users to be served by the HAPS at that time slot. We define $\mathcal{Z}_t$ as a set that contains the indices of the active antennas. Note that,

$|\mathcal{Z}_t|$ is less than or equal to the N available antennas at the HAPS (i.e., $|\mathcal{Z}_t| \leq N$). In the sequel, we provide the problem formulation and objective function of the considered user allocation problem.

Remark 1: In this paper, we operate under the assumption that the back-hauling links within the network are established. This condition facilitates the transmission of control signals from TBS to both HAPS and the satellite. This widely considered assumption aligns with prior works in the literature [11], [23].

V. PROBLEM FORMULATION

To formulate the system model, we define $\mathcal{U}$ as a set that contains all available users in the cell, where $\mathcal{U} = \{u_1, u_2, \ldots, u_U\}$. Assuming that the TBS is located in the center of the cell, we denote $\mathcal{K}_t$ and $\mathcal{K}'_t$ as the sets that contain the users scheduled to be served by the TBS and the HAPS, respectively, at time slot t. Note that, $\mathcal{K}_t$ and $\mathcal{K}'_t$ are complementary sets (i.e., $\mathcal{K}_t \cap \mathcal{K}'_t = 0$). Considering time-varying channels, we use $\mathbf{H}_t \triangleq [\mathbf{h}_{1,t}, \mathbf{h}_{2,t}, \ldots, \mathbf{h}_{U,t}]^T$ as the channel matrix between the TBS and all available users. Here, $\mathbf{h}_{u,t} = [h_{ui,\ell}]_{i=1}^{M}$ is an $M \times 1$ vector of the channel coefficients between the u-th user and the TBS antennas, for $u = 1, 2, \ldots, U$. To define the channel matrix between the HAPS antennas and the users, we use $\mathbf{G}_t \triangleq [\mathbf{g}_{1,t}, \mathbf{g}_{2,t}, \ldots, \mathbf{g}_{U,t}]^T$, where $\mathbf{g}_{u,t} = [g_{uj,\ell}]_{j=1}^{N}$ is an $N \times 1$ vector of the channel coefficients between the u-th user and the HAPS antennas, for $u = 1, 2, \ldots, U$. For given $\mathcal{K}_t$, $\mathcal{K}'_t$, and $\mathcal{Z}_t$, we define $\hat{\mathbf{H}}_t \triangleq [\mathbf{h}_{u,t}]_{u\in\mathcal{K}_t}$ and $\hat{\mathbf{G}}_t \triangleq [g_{uj,t}]_{\substack{j\in\mathcal{Z}_t \\ u\in\mathcal{K}'_t}}$ as the submatrixes of $\mathbf{H}_t$ and $\mathbf{G}_t$ that are the channels between the TBS and the HAPS and their assigned users, respectively.

At time slot t, considering the u-th user that is scheduled to be served by the TBS (i.e., $u \in \mathcal{K}_t$) which is surrounded by L cells (each cell with a TBS) in a cellular system, we denote the received signal $y_{u,t}$ as

$$y_{u,t} = \mathbf{h}_{u,t}^H \mathbf{w}_{u,t}\, s_{u,t} + \underbrace{\sum_{u'\in\mathcal{K}_{\ell,t},u'\neq u} \mathbf{h}_{u,t}^H \mathbf{w}_{u',t}\, s_{u',t}}_{\textit{intra-cell interference}} + \underbrace{\sum_{\ell=1}^{L} \sum_{u'\in\mathcal{K}_{\ell,t},u'\neq u} \left(\mathbf{h}_{u,t}^{\ell}\right)^H \mathbf{w}_{u',t}^{\ell}\, s_{u',t}^{\ell}}_{\textit{inter-cell interference}} + n_{u,t}, \quad (1)$$

where $s_{u,t}$ is the unit-energy transmitted signal to the u-th user by the TBS, $n_{u,t} \sim \mathcal{CN}(0, \sigma_n^2)$ denotes the noise with average power σ_n^2, and $\mathbf{w}_{u,t} \in \mathbb{C}^{M\times 1}$ is the precoding vector. Furthermore, to formulate the inter-cell interference we use $\mathbf{h}_{u,t}^{\ell}$, $\mathbf{w}_{u',t}^{\ell}$, and $s_{u',t}^{\ell}$ to denote the channel between the u-th user and the ℓ-th neighboring TBS antennas, the corresponding precoding vector, and its transmitted signal, respectively. Note that we can rewrite $y_{u,t}$ in (1) as

$$y_{u,t} = \mathbf{h}_{u,t}^H \mathbf{w}_{u,t}\, s_{u,t} + \underbrace{\sum_{u'\in\mathcal{K}_{\ell,t},u'\neq u} \mathbf{h}_{u,t}^H \mathbf{w}_{u',t}\, s_{u',t}}_{\textit{intra-cell interference}} + \hat{n}_{u,t}, \quad (2)$$

here $\hat{n}_{u,t} \sim \mathcal{CN}(0, \hat{\sigma}_n^2)$ is the total received noised at the u-th user served by the TBS, where

$$\hat{\sigma}_n^2 = \sigma_n^2 + \sum_{\ell=1}^{L} \sum_{u'\in\mathcal{K}_{\ell,t},u'\neq u} \mathbb{E}\left\{ |\left(\mathbf{h}_{u,t}^{\ell}\right)^H \mathbf{w}_{u',t}^{\ell}|^2 \right\}. \quad (3)$$

We use Zero-forcing beamforming (ZFB) to cancel the intra-cell interference. Note that the TBS is equipped with a massive number of antennas, and therefore we can assume that ZFB can fully cancel out the intra-cell interference. Thus, considering that $\mathbf{W}_t = [\mathbf{w}_{u,t}]_{u\in\mathcal{K}_t}$ is the percoding ZFB matrix, we can write

$$\mathbf{W}_t = \widetilde{\mathbf{W}}_t \boldsymbol{\mu}_t = \hat{\mathbf{H}}_{\ell,t}\left(\hat{\mathbf{H}}_t^H \hat{\mathbf{H}}_t\right)^{-1} \boldsymbol{\mu}_t, \quad (4)$$

where $\boldsymbol{\mu}_t \in \mathbb{C}^{|\mathcal{K}_t|\times|\mathcal{K}_t|}$ is a diagonal matrix that allows us to determine the power constraint in our problem formulation, such that $\mathbb{E}\{tr\{\mathbf{W}_t\mathbf{W}_t^H\}\} = P$. We can obtain the u-th entry of $\boldsymbol{\mu}_t$ as

$$\mu_{u,t} = \left[\boldsymbol{\mu}_t\right]_{uu} = \sqrt{P|\mathcal{K}_t|^{-1}\left[\, \|\tilde{\boldsymbol{w}}_{u,t}\|^2 \,\right]^{-1}}, \quad (5)$$

where $\tilde{\mathbf{w}}_{u,t}$ is the u-th row of the precoding matrix, obtained as $\tilde{\mathbf{w}}_{u,t} = [\widetilde{\boldsymbol{W}}_t]_u = [\hat{\mathbf{H}}_t(\hat{\mathbf{H}}_t^H\hat{\mathbf{H}}_t)^{-1}]_u$.

Given the above formulation, we denote $\gamma_u(\mathcal{K}_t)$ as the signal to interference noise ratio (SINR) at the u-th user that is served by the TBS (i.e., $u \in \mathcal{K}_t$) and write

$$\gamma_u(\mathcal{K}_t) = \frac{|\mathbf{h}_{u,t}\mathbf{w}_{u,t}|^2}{\hat{\sigma}_n^2}, \quad \text{for} \quad u \in \mathcal{K}_t. \quad (6)$$

Given (6), we can write the obtained t-th time slots's sum-rate for all users served by the TBS as

$$R(\mathcal{K}_t) = \sum_{u\in\mathcal{K}_t} log_2(1 + \gamma_u(\mathcal{K}_t)). \quad (7)$$

In this section, we aim to find the sum-rate of the users served by the HAPS at time slot t. To do so, considering that $n'_{u,t} \sim \mathcal{CN}(0, \sigma_n'^2)$ denotes the noise average power $\sigma_n'^2$, we write the SINR at the u-th user that is served by the HAPS (i.e., $u \in \mathcal{K}'_t$) as

$$\gamma'_u(\mathcal{Z}_t, \mathcal{K}'_t) = \frac{|\hat{\mathbf{g}}_{u,t}\mathbf{w}'_{u,t}|^2}{\sigma_n'^2}, \quad \text{for} \quad u \in \mathcal{K}'_t. \quad (8)$$

where, $\hat{\mathbf{g}}_{u,t}$ is the u-th row of $\hat{\mathbf{G}}_t$ and $\mathbf{w}'_{u,t} = [\mathbf{W}'_t]_u = [\hat{\mathbf{G}}_t(\hat{\mathbf{G}}_t^H\hat{\mathbf{G}}_t)^{-1}]_u$ is the u-th user's ZFB precoding vector. To guarantee that HAPS can eliminate the intra-cell interference, we limit the number of assigned users to the HAPS to the number of available antennas N (i.e., $|\mathcal{K}'_t| \leq N$). Now, we write the sum-rate of the users served by the HAPS at time slot t as

$$R'(\mathcal{Z}_t, \mathcal{K}'_t) = \sum_{u\in\mathcal{K}'_t} log_2(1 + \gamma'_u(\mathcal{Z}_t, \mathcal{K}'_t)). \quad (9)$$

In the next section, we define the objective function of our user scheduling optimization problem under the constraint of minimum circuit power consumption at the HAPS.

VI. OBJECTIVE FUNCTION

At time slot t, the TBS uses the obtained CSI from the uplink training phase to make a decision to cluster the users into two groups to be served by the TBS and the HAPS. More specifically, the TBS uses the estimated channel coefficients to select a subset of users to serve and assigns the remaining ones to the HAPS. To formulate the user scheduling decision, we define an action vector $\mathbf{a}_t = [a_{u,t}]_{u=1}^{U}$, where for $u = 1, 2, \ldots, U$, $a_{u,t} = 1$ if the u-th user is assigned to be served by the HAPS, otherwise $a_{u,t} = 0$, i.e., meaning that the TBS serves the u-th user. Thus, we can obtain $\mathcal{K}_t$ and $\mathcal{K}'_t$, from $\mathbf{a}_t$ as

$$\begin{aligned} \mathcal{K}_t &= \{u \,|a_{u,t} = 0, \text{ for } u = 1, 2, \ldots, U\}, \\ \mathcal{K}'_t &= \{u \,|a_{u,t} = 1, \text{ for } u = 1, 2, \ldots, U\}. \end{aligned} \tag{10}$$

Since the number of assigned users to the HAPS is upper bounded by N, we can write $\mathbf{a}_t \in \{\mathbf{a}|\mathbf{1}^T\mathbf{a} \leq N\}$.

The main goal here is to find the optimal $\mathcal{K}_t$ and $\mathcal{K}'_t$ that maximize the sum-rate of the system while a minimum power consumption is required at the HAPS. To ease the notation, we drop the time index and write our optimization problem as

$$\begin{aligned} \arg\min_{\mathcal{Z}} \ & \left(\arg\max_{\mathcal{K},\mathcal{K}'} \hat{R}(\mathcal{K}, \mathcal{K}', \mathcal{Z})\right) \\ \text{s.t.} \ & \mathcal{K} \bigcap \mathcal{K}' = 0 \quad \text{and} \quad \mathcal{K} \bigcup \mathcal{K}' = U \\ & |\mathcal{K}'| \leq N \\ & |\mathcal{Z}| \leq N \end{aligned} \tag{11}$$

where $\hat{R}(\mathcal{K}, \mathcal{K}', \mathcal{Z}) = R'(\mathcal{Z}, \mathcal{K}') + R(\mathcal{K})$ is the total sum-rate. Using (10), for given $\mathbf{a}$, we can obtain $\mathcal{K}$, and $\mathcal{K}'$.

Thus, $\hat{R}(\mathcal{K}, \mathcal{K}', \mathcal{Z})$ and $\hat{R}(\mathbf{a}, \mathcal{Z})$ can be used interchangeably. Our defined optimization problem in (11) can be viewed as a joint antenna selection and user scheduling problem, which is known to be NP-hard. Specifically, in our problem, the TBS is not aware of the HAPS's CSI. To simplify the problem, we assume the same line of sight (LOS) channels among the antennas at the HAPS due to its high altitude (distance from the users), and thus the antenna selection here only refers to the number of selected antennas at each time slot. In this case, to completely eliminate the intra-cell interference, at each time slot, we consider the number of active antennas at the HAPS is equal to the number of assigned users to be served by the HAPS (i.e., $|\mathcal{Z}_t| = |\mathcal{K}'_t|$). This assumption guarantees that the HAPS provides a high quality of service (QoS) to the assigned users.[1] Therefore, we can rewrite the optimization problem in (11) as

$$\begin{aligned} arg\max_{\mathbf{a}} \ & \hat{R}(\mathbf{a}, \mathcal{Z}) \\ \text{s.t.} \ & |\mathcal{Z}| = \mathbf{1}^T\mathbf{a} \\ & \mathbf{a} = [a_u]_{u=1}^{U}, \quad a_u \in \{0, 1\}. \\ & \mathbf{1}^T\mathbf{a} \leq N \end{aligned} \tag{12}$$

1. To guarantee that zero-forcing beamforming (ZFB) can fully cancel out the intra-cell interference, the number of active antennas at the HAPS should be more than or equal to the number of serving users.

In the next section, we propose a DSRL approach to solve our optimization problem.

VII. MDP FORMULATION

In this section, we first define MDP tuples and then formulate our user scheduling problem using an MDP framework. The below components are used to define an MDP problem:

$$(\mathcal{S}, \mathcal{A}, \mathbf{T}, \mathcal{R}(\mathbf{s}, \mathbf{a}), \pi(\cdot)) \tag{13}$$

where $\mathcal{S}$ is the state space set that contains all possible states; the action space denoted as $\mathcal{A}$ is a set of all possible actions; $\mathbf{T}$ is the state transition probabilities; $R(\mathbf{s}, \mathbf{a})$ is the reward function when action $\mathbf{a} \in \mathcal{A}$ is taken at state $\mathbf{s}$; and $\pi(\cdot)$ stands for policy.

We now formulate our user scheduling problem as an MDP problem.

A. STATE SPACE

Considering the state space $\mathcal{S}$, the state vector $\mathbf{s}_t \in \mathcal{S}$ at time slot t is defined as

$$\mathbf{s}_t = \left[\frac{\mathbf{h'}_t}{\sigma_{\mathbf{h'}_t}}, \mathbf{a}_{t-1}\right] \tag{14}$$

where $\mathbf{a}_{t-1}$ is the action vector executed at time slot $t-1$, $\mathbf{h'}_t = [h'_{1,t} \ \ h'_{2,t} \ \ \cdots \ \ h'_{U,t}]$ is the outdated terrestrial CSI, and $\sigma_{\mathbf{h'}_t}$ is the standard deviation of $\mathbf{h'}_t$. Thus, following the action decision described in the previous section that $a_{u,t-1} = 0$ means the u-th user is assigned to the TBS, the outdated CSI can be defined as below

$$h'_{u,t} = \begin{cases} \|h_{u,t}\|^2, & \text{if} \quad a_{u,t-1} = 0 \\ h'_{u,t-1}, & \text{if} \quad a_{u,t-1} = 1 \end{cases} \tag{15}$$

where $h'_{u,0} = \|h_{u,0}\|^2$. In other words, the outdated terrestrial CSI $h'_{u,t}$ is only updated if the TBS serves the user u at time slot $t-1$. We divide $\mathbf{h'}_t$ by its standard deviation to normalize our outdated terrestrial CSI, which helps our agent maintain its performance under imperfect CSI. Moreover, by providing our agent with the previous action vector, our agent understands whether the available CSI is updated or not. In addition, as stated in [24], under incomplete state space, i.e., the unknown HAPS's CSI in our case, integrating the action into the state space improves the performance of the DRL agent. It is noted that the agent needs to explore more during the training phase to learn the variation pattern of terrestrial CSI. Furthermore, deciding based on outdated CSI while the non-terrestrial CSI is unknown makes the agent overestimate some of the Q-values. To overcome these issues, we resort to DSRL, which, as we show, is comparable to the exhaustive search method.

B. ACTION SPACE

The action space, denoted as $\mathcal{A}$, is a set of all possible actions. For our optimization problem, the action consists of scheduling users to be served by the TBS or the HAPS and selecting a subset of antennas at the HAPS at each time slot. For ease of problem notation, let us break down the action of our joint antenna selection and user scheduling problem into two separate actions: the user scheduling action and the antenna selection action at the HAPS. Note that the decisions for both actions are made at the TBS.

We first denote the user scheduling set that contains all possible $U \times 1$ user scheduling vectors as $\mathcal{A}$. The i-th possible user scheduling vector is denoted as $\mathbf{a}_i = [a_{1,i} \quad a_{2,i} \quad \dots a_{U,i}]^T$, where $a_{u,i} = 1$ means that the u-th user is scheduled to be served by the HAPS at i-th possible vector. Otherwise, $a_{u,i} = 0$. Note that at most, N users can be scheduled to be served by the HAPS at each time slot. Based on these limitations, the action space for all possible user scheduling vectors can be written as

$$\mathcal{A} = \left\{ \mathbf{a}_i \mid \mathbf{1}^T \mathbf{a}_i \leq N \right\}. \tag{16}$$

We then denote $\tilde{\mathcal{A}}$ as the set that contains all possible antenna selection vectors at the HAPS. We define the j-th possible antenna selection action as an $N \times 1$ vector denoted as $\tilde{\mathbf{a}}_j = [\tilde{a}_{1,j} \quad \tilde{a}_{2,j} \quad \dots \quad \tilde{a}_{N,j}]^T$, where $\tilde{a}_{n,j} \in \{0, 1\}$, for $n = 1, 2, \dots, N$. Here, $\tilde{a}_{n,j} = 1$ means that the n-th antenna at the HAPS is selected to participate in the data exchange at time slot t, otherwise $\tilde{a}_{n,j} = 0$. Note that in our problem formulation, we assume that the number of active antennas at the HAPS is equal to the number of assigned users to be served by the HAPS. Moreover, because we assume the same LoS component of the channel links between the users and the HAPS's antennas is approximately the same, for simplicity, antenna selection can be simplified to find the optimal number of active antennas at each time slot. More specifically, at each time slot t, we can use $\mathbf{a}_t$ to find the number of assigned users to be served by the HAPS (i.e., $\mathbf{1}^T \mathbf{a}_t$) and obtain $\tilde{\mathbf{a}}_t$ as

$$\tilde{a}_{n,t} = \begin{cases} 1, \text{ if } n \in \left\{1, 2, \dots, \mathbf{1}^T \mathbf{a}_t\right\} \\ 0, \text{ otherwise} \end{cases} \tag{17}$$

Hence, in the sequel, the action vector only refers to the user scheduling vector $\mathbf{a}_t$, as we can easily use (17) to obtain the antenna selection vector $\tilde{\mathbf{a}}_t$, at each time slot t.

C. REWARD FUNCTION

The function $R(\mathbf{s}, \mathbf{a})$ represents the immediate reward function for taking action $\mathbf{a}$. Note that at each time slot, the u-th element of the state is the channel gain between the u-th user and the TBS antennas at that time slot, for $u = 1, 2, \dots, U$. We can write the immediate reward function as

$$R(\mathbf{s}, \mathbf{a}) = \sum_{u \in \mathcal{K}} log_2(1 + \gamma_u(\mathcal{K})) + \sum_{u \in \mathcal{K}'} log_2\big(1 + \gamma'_u(\mathcal{Z}, \mathcal{K}')\big). \tag{18}$$

where, $\mathcal{K}_t$ and $\mathcal{K}'_t$ are the user scheduling sets, which can be obtained from (10); $\gamma_u(\mathcal{K})$ and $\gamma'_u(\mathcal{Z}, \mathcal{K}')$ are SINR for the TBS and the HAPS, respectively. Moreover, we use (17) to obtain the antenna selection vector at the HAPS (i.e., $\tilde{\mathbf{a}}$) and its corresponding antenna selection set $\mathcal{Z}$.

Remark 2: In our MDP problem formulation, HAPs's circuit power consumption is not included in the reward function as it is already factored into the definition of the action.

D. POLICY

The Policy $\pi(\cdot)$ maps the state to the action. For a given policy and state $\mathbf{s} \in \mathcal{S}$, we can obtain the corresponding action as $\mathbf{a} = \pi(\mathbf{s})$.

So far, we formulated our user scheduling problem as an MDP framework to find the optimal policy that solves the maximization problem in (12). The optimal policy can be defined as follows:

$$\pi^*(\mathbf{s}) = \text{argmax}_{\mathbf{a}}\, Q(\mathbf{s}, \mathbf{a}) \tag{19}$$

where $Q(\mathbf{s}, \mathbf{a})$ is the state-action value function for $\mathbf{s} \in \mathcal{S}$ and $\mathbf{a} \in \mathcal{A}$. At time slot t, we can obtain the state-action value function as:

$$Q(\mathbf{s}, \mathbf{a}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \varrho^t R(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0, \mathbf{a}_0 \right] \tag{20}$$

where $\varrho^t \in [0, 1]$ is a discount factor that shows the trade-off between immediate and future rewards. In the DQL algorithm, a deep neural network (DNN) is used to approximate the Q-function. We denote $\boldsymbol{\theta}$ as the weights of the DNN such that, for a given $\mathbf{s} \in \mathcal{S}$ which is the input of the DNN, the Q-function denoted as $Q(\mathbf{s}, \mathbf{a} \mid \boldsymbol{\theta})$ is returned as the output. Given the highly dynamic environment resulting from time-varying channels, outdated TBS's CSI, and incomplete information due to unknown HAPS's CSI, solving the optimization problem (12) is a challenging issue. As we will show, the conventional DQL method learns a suboptimal policy due to the overestimation of Q-values and a lack of exploration. To overcome these issues, we propose a DSRL method that manages to reach optimal policy without introducing additional computational complexity compared to the DQL method.

VIII. DSRL USER SCHEDULING ALGORITHM

A. BACKGROUND

While the DQL method has gained a lot of popularity for wireless communication problems, it has some limitations in coping with problems that need more exploration and where the state space consists of incomplete information. At each state, the DQL agent updates its Q-values by employing the next state-action value function. Although the next state $\hat{\mathbf{s}}$ is known, the DQL agent is unaware of the next action $\hat{a}$. Thus, the DQL agent always takes the greedy action and chooses the best successor action based on its current knowledge, which is not always optimal and can be misleading in our

case. Based on this, the weights of the DNN are updated using the following loss function:

$$\overline{L}(\boldsymbol{\theta}) = \mathbb{E}\Big[(\overline{\mathbf{y}} - Q(\mathbf{s}, \mathbf{a} \mid \boldsymbol{\theta}))^2\Big] \tag{21}$$

where $\boldsymbol{\theta}$ denotes the DNN's weights, and $\overline{\mathbf{y}}$ is defined as:

$$\overline{\mathbf{y}} = R(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a} \in \mathcal{A}} Q\big(\hat{\mathbf{s}}, \mathbf{a} \mid \boldsymbol{\theta}\big) \tag{22}$$

where $\hat{\mathbf{s}}$ is the next state and $\max_{\mathbf{a}\in\mathcal{A}} Q(\hat{\mathbf{s}}, \mathbf{a} \mid \boldsymbol{\theta})$ is the estimate of the next Q-function. The DQL method is an off-policy DRL method that updates its policy based on the maximum Q-value of the next state (22), regardless of the action taken in the environment. In other words, the behaviour policy (the policy used for selecting actions) and the target policy (the policy used for learning) are different. Consequently, in our case, where only the outdated TBS's CSI is given and the HAPS's CSI is unknown, using the maximum operation in (22) leads to an overestimation of Q-values since the maximum Q-value may not always be optimal due to insufficient information. Furthermore, since the CSI of previously served users is only known by the agent (15), more exploration is needed during the training phase to ensure that the agent can get an overview of the instantaneous CSI of each user. However, the incomplete state space in our problem means that the agent has limited information about some state-action pairs, and since the exploration is only based on the behaviour policy, the agent may not sufficiently explore the less visited states.

To overcome the above-mentioned issues, we resort to the on-policy method called SARSA, which uses the same policy for executing actions and updating its policy. To enable our agent to profit from the advantages of DNN, we use the approach proposed in [25]. In contrast to DQL, here, the agent updates the Q-values based on the actions that are really taken in the environment. As a result, the loss function in DSRL is modified as

$$L(\boldsymbol{\theta}) = \mathbb{E}\Big[(\mathbf{y} - Q(\mathbf{s}, \mathbf{a} \mid \boldsymbol{\theta}))^2\Big] \tag{23}$$

where $\boldsymbol{\theta}$ denotes the DNN's weights, and $\mathbf{y}$ is defined as:

$$\mathbf{y} = R(\mathbf{s}, \mathbf{a}) + \gamma Q\big(\hat{\mathbf{s}}, \hat{\mathbf{a}} \mid \boldsymbol{\theta}\big) \tag{24}$$

where $\hat{\mathbf{a}}$ is the next action executed in the environment by the agent. Thus, the DSRL agent avoids the bias resulting from the maximum operation, which has a considerable impact in our case. Moreover, using the same policy for executing actions and updating the DNN naturally encourages the agent to explore more. Finally, updating the DNN using the observed SARSA transition makes the agent decide more carefully, which, as we will show later, improves the agent's performance under imperfect CSI.

To solve our problem using DSRL, we first model the actual environment of our system model using an MDP framework defined in Section VII. Then, we call each wireless network configuration an episode, where each episode demonstrates a wireless network with various users' locations and their corresponding channel coefficients. Considering that each episode consists of L time slots ($t = 1, 2, \ldots, L$), starting from an initial state $\mathbf{s}_0$, the environment receives the executed action $\mathbf{a}_t$, returns the reward value $R(\mathbf{s}_t, \mathbf{a}_t)$, and moves to the new state $\mathbf{s}_{t+1}$. To benefit from offline training and at the same time make real-time decisions, our learning process consists of two phases: an offline training phase and an online testing phase. In the training phase, we train our DSRL agent to learn the optimal policy using the generated data from a variety of simulated environments. Note that the data's correlation increases the probability of DRL divergence in online training. To decorrelate the data, we use a buffer denoted as $\mathcal{D}$ to store the tuple $(\mathbf{s}_t, a_t, R(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, d_L)$, where $d_L = 1$ if $t = L$, otherwise $d_L = 0$. Furthermore, we use a shuffled random mini-batch of data to update the DNN's weights. Due to the non-linear activation functions in DNN, the similarity between $\mathbf{y}$ and $Q(\mathbf{s}, \mathbf{a} \mid \boldsymbol{\theta})$ in (23) can lead to instability in the training phase. This is due to the fact that small changes in Q-values can lead to significant changes in the agent's policy [26]. In DQL, to tackle this issue, we use a target network. The target network is a delayed copy of the main network, which is used to estimate $Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1} \mid \tilde{\boldsymbol{\theta}})$, where $\tilde{\boldsymbol{\theta}}$ denotes the weights of the target network.

Following the provided explanations, equations (23) and (24) are modified as:

$$L(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{s},\mathbf{a},R(\mathbf{s}.\mathbf{a}),\hat{\mathbf{s}},\hat{\mathbf{a}})\sim\mathcal{D}}\Big[(\mathbf{y} - Q(\mathbf{s}, \mathbf{a} \mid \boldsymbol{\theta}))^2\Big] \tag{25}$$

where $\mathcal{D}$ is the buffer, $\boldsymbol{\theta}$ is DNN's weights, and $\mathbf{y}$ is defined as:

$$\mathbf{y} = R(\mathbf{s}, \mathbf{a}) + (1 - \mathbf{d}_L) \times \gamma Q\Big(\hat{\mathbf{s}}, \hat{\mathbf{a}} \mid \tilde{\boldsymbol{\theta}}\Big) \tag{26}$$

where $\mathbf{d}_L$ determines the last time slot. Then, in the testing phase, we use the trained agent to take real-time actions from the obtained state information at each time slot.

In the sequel, we elaborate on our proposed DSRL user scheduling algorithm.

B. OUR PROPOSED DSRL METHOD

To define the environment of our system model, we assume a TBS, located in the center of a cell with radios r, that serves a subset of U available users and schedules the remaining ones to be served by the HAPS. Let us denote $\sigma^2_{\mathrm{h},u}$ as the variance of the channel between the u-th user and the antennas at the TBS. Here, to model user u's variance, we use the path loss model

$$\sigma^2_{h,u}(dB) = 10\log_{10}\rho - 25\log_{10} d_u - \psi_{dB}, \tag{27}$$

where the path loss exponent is 2.5, d_u is the distance between the u-th user and the TBS, ρ is the path loss constant, and ψ_{dB} is a Gaussian random variable with mean zero and variance $\sigma^2_{\psi_{dB}}$. At each episode, for a given distribution of the users' location, we generate d_u for $u = 1, 2, \ldots, U$. Using (27), we can obtain $\boldsymbol{\sigma}^2_{h,u} = \sigma^2_{h,u}\boldsymbol{I}_{M\times 1}$. Here, we assume that the channel vector $\mathbf{h}_{u,t} \sim$

$\mathcal{CN}(\mathbf{0}_{M\times 1}, \boldsymbol{\sigma}^2_{h,u})$ evolves according to the first-order Gauss-Markov channel model as

$$\mathbf{h}_{u,t} \triangleq \xi_u \mathbf{h}_{u,t-1} + \sqrt{1-\xi_u^2}\mathbf{z}_{u,t}, \quad \text{for } u = 1, \ldots, U. \tag{28}$$

The independent and identically distributed (i.i.d.) innovation sequence $\mathbf{z}_{u,t} \sim \mathcal{CN}(\mathbf{0}_{M\times 1}, \boldsymbol{\sigma}^2_{h,u})$ is independent of the channel vector $\mathbf{h}_{u,t}$, for $u = 1, \ldots, U$. Moreover, $\xi_u \in [0, 1]$ is the fading correlation coefficient corresponding to the u-th user. Note that the value of ξ_u depends on the maximum Doppler frequency [27]. More specifically, for users with no movements (i.e., a static channel model), $\xi_u = 1$ holds true, and $\xi_u = 0$ presents the channel model with i.i.d. channels over t. At each time slot t, considering that the TBS is the agent making the user scheduling decision, the state is the channel gain vector between the users and the TBS, obtained as $\mathbf{s}_t = [|\mathbf{h}_{u,t}|^2]_{u=1}^{U}$. We assume that all users have the same LoS component denoted as g_{LoS}. Moreover, we use the first-order Gauss-Markov channel to model the small variations in channel links due to users' movements, reflections, etc. Thus, we can write

$$\mathbf{g}_{u,t} \triangleq \mathbf{g}_{\text{LoS}} + \xi_u \mathbf{g}_{u,t-1} + \sqrt{1-\xi_u^2}\mathbf{z}'_{u,t}, \ \ u = 1, \ldots, U. \tag{29}$$

where $\mathbf{g}_{\text{LoS}} = (g_{\text{LoS}})\mathbf{I}_{N\times 1}$, $\mathbf{g}_{u,t} \sim \mathcal{CN}(\mathbf{0}_{N\times 1}, \boldsymbol{\sigma}'^2_{h,u})$, and $\mathbf{z}'_{u,t} \sim \mathcal{CN}(\mathbf{0}_{N\times 1}, \boldsymbol{\sigma}'^2_{h,u})$. Here, $\boldsymbol{\sigma}'^2_{h,u} = \sigma'^2_{h,u}\boldsymbol{I}_{N\times 1}$ where $\sigma'^2_{h,u}$ is the channel variance between the u-th user and HAPS antennas, which can be obtained from (27). The rest of this section explains the training and testing procedures.

Training Procedure: Algorithm 1 illustrates the steps of the training process in our proposed user scheduling framework. In the first step, we initialize the buffer and define the main and target networks. To do so, we construct a DNN with weights $\boldsymbol{\theta}$ for the main network and initialize its weights with random values. We also define the target network with the same weights $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}$ and construct it as the main network, i.e., with the same number of layers and neurons per layer. Since the path loss in (27) varies according to the users' location, to prevent our DSRL agent from overfitting on a specific network configuration, we reset the environment, initialize $\mathbf{s}_0$ at the start of each episode, and train our agent for I_{episode} episodes to ensure it can adapt to different large scale fading values in the test phase. To dynamically control the exploration probability during the training procedure, we update the probability of exploration ε_i at the start of each episode where $i \in \{1, 2, \ldots, I_{\text{episode}}\}$. In this context, starting from $\varepsilon_1 = 1$, we decrease the value of ε_i step by step during the first I' episodes. After that, we explore the environment with a constant probability $\varepsilon^{\text{final}}$ that denotes the final probability of exploration. Then, before starting the episode, the agent executes the action a_t where $t = 0$. This is necessary because we need to update the agent's policy with $\hat{\mathbf{a}}$ in (25).

At the i-th episode, based on the ε-greedy exploration strategy, in each time slot t, we choose a random action with a probability of ε_i. Otherwise, with a probability $1-\varepsilon_i$, the state $\mathbf{s}_{t+1}$ is inputted to the DNN, and we choose the action with the highest Q-value, i.e., $a_{t+1} = \arg\max_a Q(\mathbf{s}_t, \mathbf{a}_t \mid \boldsymbol{\theta})$ where a_{t+1} is the next action $\hat{\mathbf{a}}$ in (25). Afterward, the action $\mathbf{a}_{t+1}$ is executed, the environment moves to the next state $\mathbf{s}_{t+2}$, and the reward value $R(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$ form (18) is obtained by the DSRL agent. In the next step, we store the SARSA transition $(\mathbf{s}_t, a_t, R(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, d_L)$ in the buffer.

To update the DSRL agent's policy, we sample a mini-batch of information from the buffer every η time slots. Then, we compute the loss function in (21) and apply the ADAM optimizer to update the DNN's weights $\boldsymbol{\theta}$. As mentioned in Section VIII-A, we keep a delayed copy of our main network to ensure the stability of our DNN during the training process. Therefore, for every η' time slot, where

Algorithm 1: Training Algorithm of the Proposed DSRL Framework

Input: learning frequency η, target update frequency η', saving model frequency η'', maximum number of episodes I_{episode}, maximum number of time slots L, number of users U, number of HAPS antenna N, number of TBS antenna M, users' locations, probability of exploration at the i-th episode ε_i, final probability of exploration $\varepsilon^{\text{final}}$, final episode of updating probability of exploration I';

Initialize buffer D, weights of the main network $\boldsymbol{\theta}$ with random values, and wights of the target network $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}$
for $i \in 1$ *to* I_{episode} **do**
  Generate d_u, for $u = 1, 2, \ldots, U$ and reset environment using (27)
  $t = 0$
  Initialize $\mathbf{s}_t$
  Update ε_i: $\varepsilon_i = \begin{cases} 1 - \frac{1-\varepsilon^{\text{final}}}{I'} & \text{if } 1 < i < I' \\ \varepsilon^{\text{final}} & I' < i < I_{\text{episode}} \end{cases}$
  $a_t \leftarrow \arg\max_a Q(\mathbf{s}_t, \mathbf{a}_t \mid \boldsymbol{\theta})$
  Execute action a_t, move to the new state $\mathbf{s}_{t+1}$, and observe the reward value $R(\mathbf{s}_t, \mathbf{a}_t)$ using (18)
  for $t \in 0$ *to* $L-1$ **do**
    Select a random action with a probability of ε_i otherwise:
    $a_{t+1} \leftarrow \arg\max_a Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1} \mid \boldsymbol{\theta})$
    Execute action a_{t+1}, move to the new state $\mathbf{s}_{t+2}$, and observe the reward value $R(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$ using (18)
    Store $(\mathbf{s}_t, a_t, R(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, d_L)$ in $\mathcal{D}$
    if $t \ \% \ \eta = 0$ **then**
      Sample mini-batch of data from $\mathcal{D}$
      Update DNN's weights using equation (25)
    if $t \ \% \ \eta' = 0$ **then**
      Update target network: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$
  if $i \ \% \ \eta'' = 0$ **then**
    Save the learned model

Output: Learned DSRL model $Q(\mathbf{s}, \mathbf{a} \mid \boldsymbol{\theta})$

$\eta' > \eta$, we copy the main network to the target network and replace $\tilde{\boldsymbol{\theta}}$ by $\boldsymbol{\theta}$. To use our model for real-time decisions during the test phase, we save our main network every η'' episode. We upload this model during the test phase and evaluate the performance of our DSRL agent. In the sequel, we explain the test phase for our proposed DSRL algorithm.

Testing procedure: In the test phase, we first upload the trained DSRL model, which is established in the TBS. Here, the TBS uses the trained DNN to determine the best user scheduling action, i.e., whether users should be served by the TBS or HAPS. To evaluate the performance of our agent over different path loss values, we run our agent for I_{episode} episodes and reset the environment at the start of each episode, i.e., distribute users to compute $\sigma^2_{h,u}$ using (27). After that, $\mathbf{s}_0 = [\|h_{u,0}\|^2]_{u=1}^{U}$ is initialized, where $\mathbf{h}_{u,0} \in \mathcal{CN}(\mathbf{0}_{M\times 1}, \boldsymbol{\sigma}^2_{h,u})$.

In each time slot t, state $\mathbf{s}_t$ is computed as $\mathbf{s}_t = [\|h_{u,t}\|^2]_{u=1}^{U}$ and fed to the established DNN in the TBS. Then, the DNN returns the Q-values and the action with the highest state-action value such that $a_t = \arg\max_a Q(\mathbf{s}_t, \mathbf{a}_t \mid \boldsymbol{\theta})$. The obtained action assigns a subset of users to be served by the HAPS and the remaining ones are served by the TBS. We use (18) to obtain the reward value $R(\mathbf{s}_t, \mathbf{a}_t)$. Afterward, the time-varying channels evolve based on (28). Algorithm 2 shows a detailed description of our testing process.

Algorithm 2: Testing Algorithm of the Proposed DQL Framework

Input: trained DSRL model $Q(\mathbf{s}, \mathbf{a} \mid \boldsymbol{\theta})$, maximum number of episodes I_{episode}, maximum number of time slots L, number of users U, number of HAPS antenna N, number of TBS antenna M, users' locations, channel matrix $\mathbf{H}_t \triangleq [\mathbf{h}_{1,t}, \mathbf{h}_{2,t}, \cdots, \mathbf{h}_{U,t}]^T$;

Upload trained DQL model $Q(\mathbf{s}, \mathbf{a} \mid \boldsymbol{\theta})$ in the TBS
for $i \in 1$ *to* I_{episode} **do**
  Generate d_u, for $u = 1, 2, \ldots, U$ and reset environment using (27)
  Initialize $\mathbf{s}_0 = \left[\|h_{u,0}\|^2\right]_{u=1}^{U}$ where $\mathbf{h}_{u,0} = \mathcal{CN}(\mathbf{0}_{M\times 1}, \boldsymbol{\sigma}^2_{h,u})$
  for $t \in 1$ *to* L **do**
    $\mathbf{s}_t = \left[\|h_{u,t}\|^2\right]_{u=1}^{U}$
    $a_t \leftarrow \arg\max_a Q(\mathbf{s}_t, \mathbf{a}_t \mid \boldsymbol{\theta})$
    Extract users' received rate and obtain $R(\mathbf{s}_t, \mathbf{a}_t)$ using (18)
    Obtain $\mathbf{H}_{t+1}$ using (28)

Output: Executed actions $\mathbf{a}$ and their associated reward values $R(\mathbf{s}, \mathbf{a})$

IX. HEURISTIC OPTIMIZATION FOR USER SCHEDULING

In this section, we propose a heuristic optimization algorithm to find the best user scheduling action that maximizes the sum-rate of the system. To formulate our optimization problem, let us define diagonal user selection matrices for the TBS and the HAPS as $\Delta_t = \text{diag}(\mathbf{1}_{\text{U}\times 1} - \mathbf{a}_{\text{t}})$ and $\Delta'_t = \text{diag}(\mathbf{a}_{\text{t}})$, respectively. In addition, we define an $N \times N$ diagonal antenna selection matrix at the HAPS, denoted as $\Delta''_t = \text{diag}(\tilde{\mathbf{a}}_t)$. Now, we can write

$$\mathbf{F}_t = \Delta_t \mathbf{H}_t, \quad \text{and } \mathbf{F}'_t = \Delta'_t \mathbf{G}_t \Delta''_t \tag{30}$$

where F_t and F'_t are the channel matrices between the TBS and the HAPS and their corresponding served users, respectively, and we can write

$$\begin{aligned}\mathbf{F}_t &= \mathbf{B}\begin{bmatrix}\hat{\mathbf{H}}_t \\ \mathbf{0}_{(U-|\mathcal{K}_t|)\times M}\end{bmatrix} = \mathbf{B}\tilde{\mathbf{H}}_t \\ \mathbf{F}'_t &= \mathbf{B}'\begin{bmatrix}\hat{\mathbf{G}}_t & \mathbf{0}_{U\times(N-|\mathcal{Z}_t|)} \\ \mathbf{0}_{(U-|\mathcal{K}_t|)\times M}\end{bmatrix} = \mathbf{B}'\tilde{\mathbf{G}}_t\end{aligned} \tag{31}$$

Here, $\mathbf{B}$ and $\mathbf{B}'$ are permutation matrices (i.e., $\mathbf{B}^H\mathbf{B} = \mathbf{I}_M$). Thus, we can write

$$\begin{aligned}\mathbf{F}_t^H\mathbf{F}_t &= \tilde{\mathbf{H}}_t^H\mathbf{B}^H\mathbf{B}\tilde{\mathbf{H}}_t = \tilde{\mathbf{H}}_t^H\tilde{\mathbf{H}}_t = \hat{\mathbf{H}}_t^H\hat{\mathbf{H}}_t. \\ \mathbf{F}_t'^H\mathbf{F}'_t &= \tilde{\mathbf{G}}_t^H\mathbf{B}'^H\mathbf{B}'\tilde{\mathbf{G}}_t = \tilde{\mathbf{G}}_t^H\tilde{\mathbf{G}}_t = \hat{\mathbf{G}}_t^H\hat{\mathbf{G}}_t.\end{aligned} \tag{32}$$

We use (32) to write the sum-rate of our system in (33), shown at the bottom of the page, where P and P' denoted the total power at the TBS and the HAPS, respectively.

$$\begin{aligned}\check{R}(\Delta_t, \Delta'_t, \Delta''_t) &= \log_2\left(\det\left(\mathbf{I}_M + \frac{P}{|\mathcal{K}_t|\hat{\sigma}_n^2}\hat{\mathbf{H}}_t^H\hat{\mathbf{H}}_t\right)\right) + \log_2\left(\det\left(\mathbf{I}_{|\mathcal{Z}_t|} + \frac{P'}{|\mathcal{K}'_t|\sigma_n'^2}\hat{\mathbf{G}}_t^H\hat{\mathbf{G}}_t\right)\right) \\ &= \log_2\left(\det\left(\mathbf{I}_M + \frac{P}{|\mathcal{K}_t|\hat{\sigma}_n^2}\mathbf{F}_t^H\mathbf{F}_t\right)\right) + \log_2\left(\det\left(\mathbf{I}_N + \frac{P'}{|\mathcal{K}'_t|\sigma_n'^2}\mathbf{F}_t'^H\mathbf{F}'_t\right)\right) \\ &= \log_2\left(\det\left(\mathbf{I}_M + \frac{P}{|\mathcal{K}_t|\hat{\sigma}_n^2}\mathbf{H}_t^H\underbrace{\Delta_t^H\Delta_t}_{=\Delta_t}\mathbf{H}_t\right)\right) + \log_2\left(\det\left(\mathbf{I}_N + \frac{P'}{|\mathcal{K}'_t|\sigma_n'^2}\Delta_t''^H\mathbf{G}_t^H\underbrace{\Delta_t'^H\Delta'_t}_{=\Delta'_t}\mathbf{G}_t\Delta''_t\right)\right) \\ &= \log_2\left(\det\left(\mathbf{I}_U + \frac{P}{|\mathcal{K}_t|\hat{\sigma}_n^2}\Delta_t\mathbf{H}_t\mathbf{H}_t^H\right)\right) + \log_2\left(\det\left(\mathbf{I}_N + \frac{P'}{|\mathcal{K}'_t|\sigma_n'^2}\underbrace{\Delta_t''^H\mathbf{G}_t^H}_{\triangleq\,\mathbf{C}_t^H}\underbrace{\Delta_t'^H\Delta'_t}_{=\Delta'_t}\underbrace{\mathbf{G}_t\Delta''_t}_{\triangleq\,\mathbf{C}_t}\right)\right).\end{aligned} \tag{33}$$

Considering that $\mathbf{C}$ is a $U \times N$ matrix and using the following properties

$$\det\left(\mathbf{I}_N + \mathbf{C}_t^H \Delta_t' \mathbf{C}_t\right) = \det\left(\mathbf{I}_U + \Delta_t' \mathbf{C}_t \mathbf{C}_t^H\right) \tag{34}$$

where $\mathbf{C}_t = \mathbf{G}_t \Delta_t''$ and $\mathbf{C}_t \mathbf{C}_t^H = \mathbf{G}_t \underbrace{\Delta_t'' \Delta_t''^H}_{=\Delta_t''} \mathbf{G}_t^H$, we rewrite the sum-rate as

$$\begin{aligned} &\breve{R}\left(\Delta_t, \Delta_t', \Delta_t''\right) \\ &= \log_2\left(\det\left(\mathbf{I}_U + \frac{P}{|\mathcal{K}_t|\hat{\sigma}_n^2} \Delta_t \mathbf{H}_t \mathbf{H}_t^H\right)\right) \\ &\quad + \log_2\left(\det\left(\mathbf{I}_U + \frac{P'}{|\mathcal{K}_t'|\sigma_n'^2} \Delta_t' \mathbf{G}_t \Delta_t'' \mathbf{G}_t^H\right)\right). \end{aligned} \tag{35}$$

Given the fact that $F(\mathbf{X}) = \log_2(\det(\mathbf{X}))$ is concave in the entries of $\mathbf{X}$ and that the sum of concave functions is a concave function, we propose the use of interior-point algorithms [28] to solve (35). However, in our scenario, the entries of Δ_t, Δ_t', and Δ_t'' are binary values, and thus the selection problem is NP-hard. To overcome this issue, we propose to relax the binary values of the selection matrices. More specifically, the u-th diagonal entry of the TBS user selection matrix Δ_t, denoted as $\Delta_{uu,t}$ is $\Delta_{uu,t} = 1 - a_{u,t}$, meaning that $\Delta_{uu,t} \in \{0, 1\}$. To relax our selection constraints, we allow $\Delta_{uu,t} \in [0, 1]$ and $\Delta_{uu,t}' \in [0, 1]$. To account for the limitation imposed on the number of scheduled users to the HAPS, we add the constraint $\text{trace}(\Delta_t') = \text{N}$ to our optimization problem. Moreover, to further simplify our optimization problem, we assume $\Delta_{nn,t}'' = 1$, for $n = 1, 2, \ldots, N$ (i.e., no antenna selection at the HAPS), and thus, $\Delta_t' \mathbf{G}_t \Delta_t'' \mathbf{G}_t^H = \Delta_t' \mathbf{G}_t \mathbf{G}_t^H$.

For each time slot t, we can write our maximization problem as

$$\begin{aligned} \max \quad & \log_2\left(\det\left(\mathbf{I}_U + \frac{P}{|\mathcal{K}_t|\hat{\sigma}_n^2} \Delta_t \mathbf{H}_t \mathbf{H}_t^H\right)\right) \\ & + \log_2\left(\det\left(\mathbf{I}_U + \frac{P'}{|\mathcal{K}_t'|\sigma_n'^2} \Delta_t' \mathbf{G}_t \mathbf{G}_t^H\right)\right) \\ \text{s.t.} \quad & 0 \le \Delta_{uu,t} \le 1, \ \text{for } u = 1, 2, \ldots, U \\ & 0 \le \Delta_{uu,t}' \le 1, \ \text{for } u = 1, 2, \ldots, U \\ & \text{trace}(\Delta_t) = \text{U} - \text{N} \\ & \text{trace}\left(\Delta_t'\right) = \text{N} \end{aligned} \tag{36}$$

The Interior-point algorithm can be used to solve the above equation. Then, the indices corresponding to $(U-N)$ largest values of $\Delta_{uu,t}$ are selected as the indices of the users that are assigned to be served by the TBS and the rest of N indices (that are equivalent to indices of the N largest values of $\Delta_{uu,t}'$) are the indices of the users that are selected to be served by the HAPS.

The Interior-point algorithm can be used to solve (36). We now propose a heuristic user scheduling algorithm, written in Algorithm 3. At each time slot t, the users' channel links between the TBS and the HAPS (i.e., $\mathbf{H}_t$ and $\mathbf{G}_t$, respectively) are inputted to the algorithm of our algorithm. By first solving (36), we obtain Δ_t and Δ_t'. Note that Δ_t' is a diagonal user selection matrix for the HAPS. Thus, by sorting the diagonal elements of Δ_t', the u-th element of the action vector $\mathbf{a}_t$ is equal to 1 (i.e., $a_{u,t} = 1$) if it is among the first N highest values in the sorted Δ_t' entries. Otherwise, $a_{u,t} = 0$. Given the obtained action vector $\mathbf{a}_t$, we use (10) to obtain $\mathcal{K}_t$ and $\mathcal{K}_t'$, which correspond to the sets of users assigned to the TBS and the HAPS, respectively. Then, by finding $\hat{\mathbf{H}}_t \triangleq [\mathbf{h}_{u,t}]_{u \in \mathcal{K}_t}$ and $\hat{\mathbf{G}}_t \triangleq [\mathbf{g}_{u,t}]_{u \in \mathcal{K}_t'}$ and their corresponding beamforming matrices $\mathbf{W}_t$ and $\mathbf{W}_t'$, respectively, the sum-rate of the system can be measured. It is noted that we only perform user scheduling and assume that all the HAPS's antennas are active at each time slot, meaning that $\mathcal{Z}_t = \{1, 2, \ldots, N\}$. Using (7) and (9), we can obtain $R(\mathcal{K}_t)$ and $R'(\mathcal{Z}_t, \mathcal{K}_t')$, respectively. At each time slot t, the output of Algorithm 3 is the total sum-rate of the system $\hat{R}(\mathcal{K}_\mathbf{t}, \mathcal{K}_\mathbf{t}', \mathcal{Z}) = R'(\mathcal{Z}_t, \mathcal{K}_t') + R(\mathcal{K}_t)$.

Algorithm 3: Heuristic Optimization User Selection Algorithm

Initialization: Number of users U, number of HAPS antennas N, number of TBS antennas M, total TBS transmit power P, and total HAPS transmit power P'.
Input: $\mathbf{H}_t$ and $\mathbf{G}_t$
1. Solve (32) to obtain Δ_t and Δ_t'.
2. Sort the diagonal entries of Δ_t'.
3. Obtain the entries of the action vector $\mathbf{a}_t$ as

$$a_{u,t} = \begin{cases} 1, & \text{if } \Delta_{uu,t}' \text{ is among } N \text{ highest entries of } \Delta_t' \\ 0, & \text{otherwise} \end{cases} \tag{37}$$

4. Use (10) to obtain $\mathcal{K}_t$ and $\mathcal{K}_t'$.
5. Obtain $\hat{\mathbf{H}}_t \triangleq [\mathbf{h}_{u,t}]_{u \in \mathcal{K}_t}$ and $\hat{\mathbf{G}}_t \triangleq [\mathbf{g}_{u,t}]_{u \in \mathcal{K}_t'}$ and their corresponding beamforming matrices $\mathbf{W}_t$ and $\mathbf{W}_t'$, respectively.
6. Set $\mathcal{Z}_t = \{1, 2, \ldots, N\}$ and use (7) and (9) to obtain $R(\mathcal{K}_t)$ and $R'(\mathcal{Z}_t, \mathcal{K}_t')$, respectively.
Output: Sum-rate $\hat{R}(\mathcal{K}_\mathbf{t}, \mathcal{K}_\mathbf{t}', \mathcal{Z}) = R'(\mathcal{Z}_t, \mathcal{K}_t') + R(\mathcal{K}_t)$

X. SIMULATION RESULTS

In this section, we present our simulation results to demonstrate the performance of our proposed DSRL user scheduling method and its advantages over traditional convex optimization algorithms.

A. SIMULATION CONFIGURATION

We consider a TBS with $M = 16$ antennas at the center of a cell with a radius of 3 km, a HAPS with $N = 2$ antennas, and $U = 6$ single-antenna users that are distributed uniformly in the cell. The TBS power supply is set to 10 dB (i.e., $P = 10$ dB) and 5 dB for HAPS's supply power (i.e., $P' = 5$ dB). Furthermore, we use Jake's model [29] to obtain

TABLE 1. Training parameters of our environment.

Parameter	Value	Parameter	Value
P	10db	N	3
P'	5db	$\sigma_n'^2, \hat{\sigma}_n^2$	0.1
M	16	U	6

TABLE 2. DNN training setting and input parameters in Algorithm 1.

Parameter	Value	Parameter	Value
dimension of input layer	$2 \times U$	I_{episode}	10000
dimension of output layer	$\lvert\mathcal{A}\rvert$	I'	1000
dimension of hidden layers	256, 512	L	100
α	0.001	η	4
mini-batch size	64	η'	20
optimizer	ADAM	η''	10
γ	0.5	$\varepsilon^{\text{final}}$	0.1

ξ_u for $u = 1, 2, \ldots, U$, such that $\xi_u = J_0(2\pi f_d T_s)$. Here, $J_0(\cdot)$ is the zero-order Bessel function, f_d is the maximum Doppler frequency, and T_s is the maximum time interval between adjacent instants. Here, we assume $T_s = 20$ ms and $f_d = 10$ Hz [30]. We use (27), where d_u is the distance between the u-th user and the TBS, and the path loss constant, ρ, is chosen such that at the boundary of our cell, $\sigma_{h,u}^2/\hat{\sigma}_n^2 = -5$ dB holds true. Table 1 summarises the environment parameters during the training of our DSRL method (Algorithm 1).

We implement a DNN with four layers as the main network: an input layer of dimension $2 \times U$, an output layer of size $|\mathcal{A}|$, and two hidden layers with 256, and 512 neurons, respectively. We use the Rectified Linear Unit (ReLU) activation function for the input and hidden layers and the linear activation function for the output layer. We adopt an ADAM optimizer with a learning rate of $\alpha = 0.001$ and use a sequence of information with a mini-batch size of sixty four to train our DNN. All DSRL implementations, i.e., DNN and the ADAM optimizer, are performed using PyTorch. Moreover, we solve the relaxed convex optimization problem using the cvxp library in Python. We train our DSRL agent for $I_{\text{episode}} = 10000$ episodes, where each episode contains $L = 100$ time slots. During the first $I' = 1000$ episodes, we decrease the exploration probability ε_i step by step, as mentioned in Algorithm 1, and we continue with a constant exploration probability $\varepsilon^{\text{final}} = 0.1$ for the rest of the training process. The DNN training setup and input parameters in Algorithm 1 are tabulated in Table 2.

To demonstrate the convergence of our DSRL agent, we plot the evaluation of the loss function in (21) during the training procedure. Figure 2 illustrates the average value of the loss function at each episode. As can be seen from this figure, our DSRL agent successfully converges and manages to learn the optimal policy.

In the next section, we use the trained DSRL agent to evaluate the performance of our proposed Algorithm 2, which is capable of making real-time decisions.

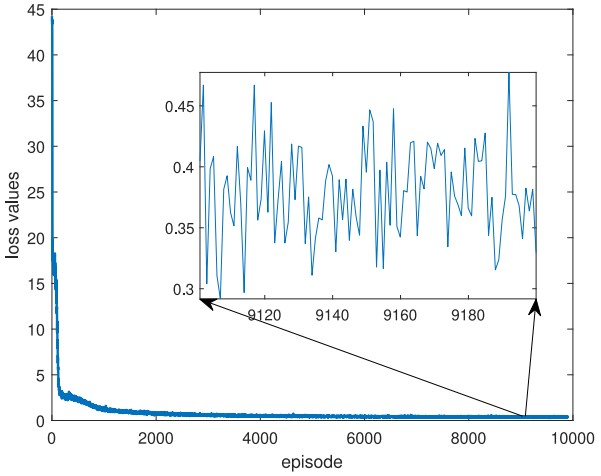

FIGURE 2. Evaluation of the loss value for the DQN agent.

B. PERFECT CSI SCENARIO

In this subsection, we investigate the impact of different network configurations on the performance of our proposed DSRL user scheduling framework when perfect CSI is available. We compare the results of our DSRL algorithm with four different methods: an exhaustive search (ES) method, the proposed heuristic optimization in Algorithm 3 (the CO method), the DQL method, and a random selection method. Here, in the ES method, we explore all possible actions (i.e., all possible user scheduling actions in the action space $\mathbf{a} \in \mathcal{A}$) at each time slot t and choose the one that leads to the highest reward value $R(\mathbf{s}, \mathbf{a})$, meaning that the result of the exhaustive search is the upper-bound sum-rate.[2] In the CO method, the instantaneous full CSI of the HAPS and TBS is required for user scheduling decisions at each time slot. These requirements are incompatible with our assumption, and thus we only show this method as a benchmark. For the random selection approach, we randomly select one of the possible actions $\mathbf{a} \in \mathcal{A}$ at each time slot, meaning that it demonstrates the lower-bound sum-rate. The presented results in this section are averaged over 100 Monte-Carlo results.

Considering parameters in Table 1, Fig. 3 illustrates the averaged sum-rate for different $\hat{\sigma}_n^2$ values, where $\hat{\sigma}_n^2$ represents both inter-cell and intra-cell interference. As can be seen from this figure, the performance gap between the DSRL and ES methods is approximately 0.3 bit-per-channel-use (bcu), and the gap between the DSRL and the random selection is approximately 5.5 bcu, where our proposed DSRL method maintains its performance over different values of $\hat{\sigma}_n^2$. Moreover, we can see that the DSRL method yields an improved performance compared to the DQL method (approximately 0.5 bcu advantage), while both have similar computational complexities.

2. It is noted that the ES method is not practical for real-time decision making as it has to search among all possible actions. Meanwhile, the channels evolve, and the provided CSI becomes outdated. Thus, we only use ES as a comparison method that provides upper-bound.

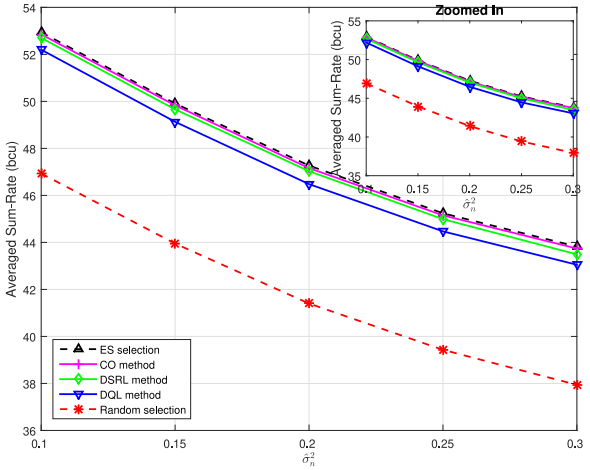

FIGURE 3. Averaged sum-rate versus $\hat{\sigma}_n^2$ values.

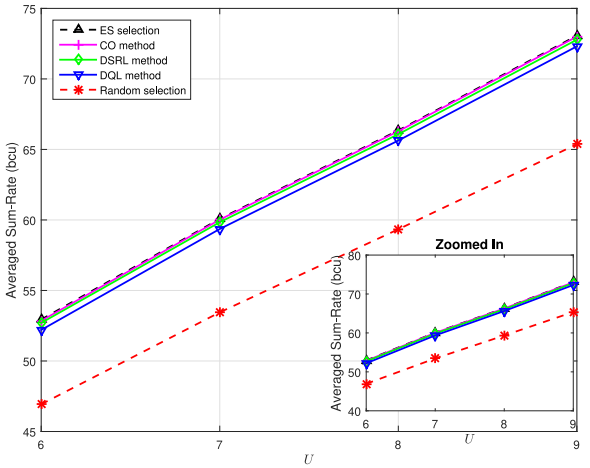

FIGURE 4. Average sum-rate for different numbers of users *U*.

To further investigate the robustness of the DSRL agent in different scenarios, we also study the effect of the user density. In Fig. 4, we plot the averaged sum-rate versus the number of available users U, for $U \in \{6, 7, 8, 9\}$. The provided results show that by increasing U, the averaged sum-rate increases for all the methods. The average sum-rate gap between the ES method and our proposed DSRL method is 0.3 (bcu). In addition, the performance of the DSRL is 0.5 (bcu) higher than that of the DQL method when increasing the number of users; the performance gap between other methods and the DSRL approach remains unchanged. This demonstrates the stability of our proposed method against changes in the number of users.

We also compare the averaged sum-rate of our proposed method versus the number of antennas at the HAPS N in Fig. 5, for $M = 16$ and $U = 9$. It is noted that $N = 0$ means that all the users are served by the BS (i.e., there is only one possible action at each time slot). As shown in Fig. 5, increasing the number of antennas at the HAPS can significantly enhance the averaged sum-rate of the system for all the methods. Adding more antennas (N) increases

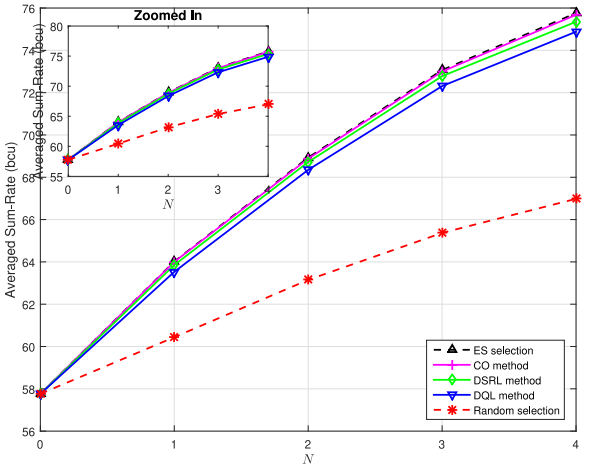

FIGURE 5. Average sum-rate for different numbers of HAPS's antennas *N* at the HAPS.

the performance gap between the ES method and DSRL; however, we expected this result due to the fact that, in the DSRL method, decisions are made only based on TBS CSI. Thus, increasing the HAPS antennas is equivalent to more unknown information for decision-making compared to the CO method, which uses full CSI. This shows that in our proposed DSRL method, by slightly sacrificing performance, we can cut the overhead of the network in half by avoiding sharing information between HAPS and TBS. From this figure, we can also see that the gap between ES selection and the DQL method increases more noticeably compared to the performance gap between ES and the DSRL method. This is due to the fact that the DSRL method is an on-policy DRL method, which prevents our agent from overestimating the Q-values, which has a negative impact on our problem.

So far, we mainly focused on comparisons between our DSRL algorithm and other methods by evaluating their performance for different parameters. However, we assumed that perfect CSI is available, which is not always possible in practice. Thus, we next define a real scenario in which channel estimation errors are considered in the performance analysis.

C. IMPERFECT CSI SCENARIO

In this section, we evaluate the performance of our proposed user scheduling algorithm under imperfect CSI. To do so, we define $\breve{\mathbf{h}}_{u,t} \in \mathcal{CN}(\mathbf{0}_{M\times 1}, \epsilon_{h,u}\mathbf{1}_{M\times 1})$ as the channel estimation error between the u-th user and the BS antennas. Thus, for $u = 1, 2, \ldots, U$, we can write $\mathbf{h}_{u,t} = \ddot{\mathbf{h}}_{u,t} + \breve{\mathbf{h}}_{u,t}$, where $\ddot{\mathbf{h}}_{u,t}$ is the estimated channel vector between the u-th user and the BS antennas. Note that $\ddot{\mathbf{h}}_{u,t}$ and $\breve{\mathbf{h}}_{u,t}$ are i.i.d.

For further investigation, we add a method to our imperfect CSI comparisons used in [31] that uses uncertainty bounds to solve the convex optimization problem under imperfect CSI. We denote this method as convex optimization with uncertainty bound (COUB). Here, the agent is trained considering perfect CSI (i.e., $\epsilon_h = 0$) in Algorithm 1 and

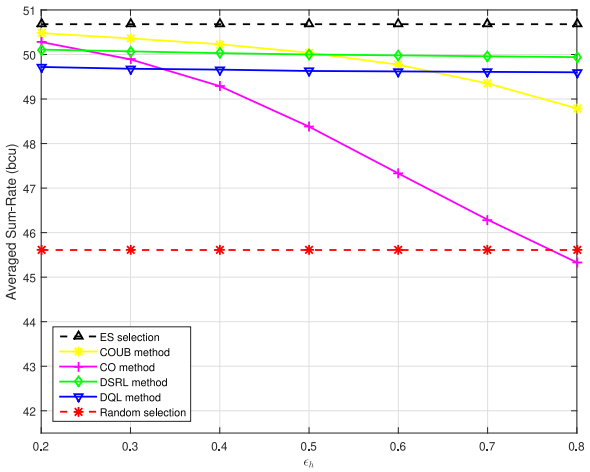

FIGURE 6. Average sum-rate for different channel estimation error ϵ_h values.

then tested assuming different values of ϵ_h. More specifically, in Algorithm 2 and Algorithm 3, the input data is the estimated channel vectors $\ddot{\mathbf{H}}_t = [\ddot{\mathbf{h}}_{u,t}]_{u=1}^{U}$. However, the averaged sum-rate of the system at each time slot is obtained according to the actual data rate (using the true channel matrix $\mathbf{H}_t = [\mathbf{h}_{u,t}]_{u=1}^{U}$).

The results are presented in Fig. 6. As shown in the figure, for small estimation errors, the averaged sum-rate for the COUB method is up to 0.6 dB higher than that of our method. However, for higher ϵ_h (more than 0.5), our proposed method outperforms other mathematical optimization approaches. Hence, in a realistic case where perfect CSI is not available, CO cannot provide the optimal solution. In addition, for ϵ_h higher than 0.4, the COUB method cannot tolerate it, and the averaged sum-rate drops significantly. Note that in the COUB method, full HAPS CSI and TBS CSI are required. In contrast, our proposed DSRL algorithm is shown to be a promising solution that can perform well in worst-case scenarios.

Remark 3: The good performance of the DSRL agent under imperfect CSI is due to the fact that we accurately designed the state vector. In our previous work [22], the DQL agent only received the global CSI (i.e., the CSI of the TBS and the CSI of the HAPS) directly. Although it outperformed the CO method with imperfect CSI, it dropped for large noise values. To solve this issue, we normalize the CSI values to their standard deviation to improve the performance of our proposed DSRL agent under imperfect CSI.

D. COMPUTATIONAL COMPLEXITY

The computational complexity of the DSRL depends on the DNN, which is used as the main network, where the computational complexity of any DNN depends on its architecture. In other words, the total number of neurons affects the computational complexity of the DNN. Consequently, for our DNN with an input size of $2 \times |U|$, output size of $|A|$, and hidden layers with dimensions W_1, and W_2, the computational complexity is computed as $O(|U|W_1 + |A|W_2 + W_1W_2)$. Therefore, for the parameters in Table 1 and Table 2, the computational complexity of our DSRL method is approximately $O(|A|^{3.199})$. For $U = 9$ and the same values of M and N, the computational complexity of the DSRL method is about $O(|A|^{2.51})$. In other words, as our network grows, using DSRL becomes more beneficial. The computational complexity of our proposed optimization-based user scheduling approach in Algorithm 3 is $O(|A|^{3.5})$ [28]. However, it is noted that, after training, no learning and backpropagation is needed in Algorithm 2 for real-time decision-making. In contrast, the optimization-based user scheduling method has to run at each time slot.

E. DISCUSSION

For traditional optimization-based user scheduling methods, such as our proposed method in Algorithm 3, knowledge of perfect CSI is essential to guarantee a reliable result. In addition, both channel links between users-TBS and users-HAPS are required to solve the optimization problem. Furthermore, at each time slot, an optimization problem needs to be solved to obtain the best user scheduling action for that time slot as the channels constantly evolve. To overcome these challenges, in this paper, we propose a DRL user scheduling method that can provide a high performance and reliable results under imperfect CSI between the users and the TBS (no channel links information between users and the HAPS is required).

XI. CONCLUSION

In this paper, considering a cellular system, we studied the user scheduling problem between a TBS and a HAPS with the main goal of maximizing the sum-rate with a minimum power consumption at the HAPS. In this context, we assumed that only the TBS's previous time slot CSI is available for user scheduling at the current time slot. Considering the above assumption, we formulate our optimization problem using an MDP framework and proposed a DSRL user scheduling algorithm. We also developed a heuristics optimization method for the sake of comparison. Moreover, the exhaustive search method and random selection were introduced in the simulation results to show the upper-bound and lower-bound sum-rates, respectively. The provided results validate our claims about the near-optimality, stability, and scalability of our proposed DSRL algorithm. We also showed that in the presence of channel estimation errors, the heuristic optimization method can fail, leading to results similar to random selection scheme. Meanwhile, our proposed DSRL user scheduling algorithm was shown to outperform the heuristics optimization method and maintain a reliable performance.

REFERENCES

[1] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2334–2360, 3rd Quart., 2019.

[2] L. Zhang et al., "A survey on 5G millimeter wave communications for UAV-assisted wireless networks," *IEEE Access*, vol. 7, pp. 117460–117504, 2019.
[3] G. Karabulut Kurt et al., "A vision and framework for the high altitude platform station (HAPS) networks of the future," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 729–779, 2nd Quart., 2021.
[4] M. S. Alam, G. K. Kurt, H. Yanikomeroglu, P. Zhu, and N. D. Đào, "High altitude platform station based super macro base station constellations," *IEEE Commun. Mag.*, vol. 59, no. 1, pp. 103–109, Jan. 2021.
[5] O. Anicho, P. B. Charlesworth, G. S. Baicher, and A. Nagar, "Autonomously coordinated multi-HAPS communications network: Failure mitigation in volcanic incidence area coverage," in *Proc. IEEE Int. Conf. Commun., Netw. Satell. (Comnetsat)*, 2019, pp. 79–84.
[6] O. Anicho, P. B. Charlesworth, G. S. Baicher, A. Nagar, and N. Buckley, "Comparative study for coordinating multiple unmanned HAPS for communications area coverage," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, 2019, pp. 467–474.
[7] S. Sibiya and O. O. Olugbara, "Reliable Internet of Things network architecture based on high altitude platforms," in *Proc. Conf. Inf. Commun. Technol. Soc. (ICTAS)*, 2019, pp. 1–4.
[8] D. J. Birabwa, D. Ramotsoela, and N. Ventura, "Service-aware user association and resource allocation in integrated terrestrial and non-terrestrial networks: A genetic algorithm approach," *IEEE Access*, vol. 10, pp. 104337–104357, 2022.
[9] K. Tashiro, K. Hoshino, and A. Nagate, "Nullforming-based precoder for spectrum sharing between HAPS and terrestrial mobile networks," *IEEE Access*, vol. 10, pp. 55675–55693, 2022.
[10] Y. Yuan, L. Lei, T. X. Vu, Z. Chang, S. Chatzinotas, and S. Sun, "Adapting to dynamic LEO-B5G systems: Meta-critic learning based efficient resource scheduling," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9582–9595, Nov. 2022.
[11] A. Alsharoa and M.-S. Alouini, "Improvement of the global connectivity using integrated satellite-airborne-terrestrial networks with resource optimization," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5088–5100, Aug. 2020.
[12] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.
[13] N. C. Luong et al., "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.
[14] A. Alwarafy, M. Abdallah, B. S. Ciftler, A. Al-Fuqaha, and M. Hamdi, "Deep reinforcement learning for radio resource allocation and management in next generation heterogeneous wireless networks: A survey," 2021, *arXiv:2106.00574*.
[15] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, 2nd Quart., 2021.
[16] J.-H. Lee, J. Park, M. Bennis, and Y.-C. Ko, "Integrating LEO satellite and UAV relaying via reinforcement learning for non-terrestrial networks," in *Proc. IEEE Global Commun. Conf.*, 2020, pp. 1–6.
[17] Y. Cao, S.-Y. Lien, and Y.-C. Liang, "Deep reinforcement learning for multi-user access control in non-terrestrial networks," *IEEE Trans. Commun.*, vol. 69, no. 3, pp. 1605–1619, Mar. 2021.
[18] H. Khoshkbari and G. Kaddoum, "Deep recurrent reinforcement learning for partially observable user association in a vertical heterogenous network," *IEEE Commun. Lett.*, early access, Nov. 8, 2023, doi: 10.1109/LCOMM.2023.3331216.
[19] M. Samir, C. Assi, S. Sharafeddine, D. Ebrahimi, and A. Ghrayeb, "Age of information aware trajectory planning of UAVs in intelligent transportation systems: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12382–12395, Nov. 2020.
[20] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.
[21] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, "Deep reinforcement learning for fresh data collection in UAV-assisted IoT networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2020, pp. 716–721.
[22] H. Khoshkbari, S. Sharifi, and G. Kaddoum, "User association in a VHetNet with delayed CSI: A deep reinforcement learning approach," *IEEE Commun. Lett.*, vol. 27, no. 8, pp. 2257–2261, Aug. 2023.
[23] F. Pervez, L. Zhao, and C. Yang, "Joint user association, power optimization and trajectory control in an integrated satellite-aerial-terrestrial network," *IEEE Trans. Wireless Commun.*, vol. 21, no. 5, pp. 3279–3290, May 2022.
[24] P. Zhu, X. Li, P. Poupart, and G. Miao, "On improving deep reinforcement learning for pomdps," 2017, *arXiv:1704.07978*.
[25] D. Zhao, H. Wang, K. Shao, and Y. Zhu, "Deep reinforcement learning with experience replay based on SARSA," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, 2016, pp. 1–6.
[26] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[27] I. Abou-Faycal, M. Medard, and U. Madhow, "Binary adaptive coded pilot symbol assisted modulation over rayleigh fading channels without feedback," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 1036–1046, Jun. 2005.
[28] A. Dua, K. Medepalli, and A. Paulraj, "Receive antenna selection in MIMO systems using convex optimization," *IEEE Trans. Wireless Commun.*, vol. 5, no. 9, pp. 2353–2357, Sep. 2006.
[29] G. Caire, N. Jindal, M. Kobayashi, and N. Ravindran, "Multiuser MIMO achievable rates with downlink training and channel state feedback," *IEEE Trans. Inf. Theory*, vol. 56, no. 6, pp. 2845–2866, Jun. 2010.
[30] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6255–6267, Oct. 2020.
[31] K. Pathak, S. S. Kalamkar, and A. Banerjee, "Optimal user scheduling in energy harvesting wireless networks," *IEEE Trans. Commun.*, vol. 66, no. 10, pp. 4622–4636, Oct. 2018.