A Deep Reinforcement Learning-Based Resource Scheduler for Massive MIMO Networks

Qing An[†], Santiago Segarra[†], Chris Dick[‡], Ashutosh Sabharwal[†], Rahman Doost-Mohammady[†]

[†]Department of Electrical and Computer Engineering, Rice University
[‡]NVIDIA

***Abstract*—The large number of antennas in massive MIMO systems allows the base station to communicate with multiple users at the same time and frequency resource with multi-user beamforming. However, highly correlated user channels could drastically impede the spectral efficiency that multi-user beamforming can achieve. As such, it is critical for the base station to schedule a suitable group of users in each time and frequency resource block to achieve maximum spectral efficiency while adhering to fairness constraints among the users. In this paper, we consider the resource scheduling problem for massive MIMO systems with its optimal solution known to be NP-hard. Inspired by recent achievements in deep reinforcement learning (DRL) to solve problems with large action sets, we propose SMART, a dynamic scheduler for massive MIMO based on the state-of-the-art Soft Actor-Critic (SAC) DRL model and the K-Nearest Neighbors (KNN) algorithm. Through comprehensive simulations using realistic massive MIMO channel models as well as real-world datasets from channel measurement experiments, we demonstrate the effectiveness of our proposed model in various channel conditions. Our results show that our proposed model performs very close to the optimal proportionally fair (Opt-PF) scheduler in terms of spectral efficiency and fairness with more than one order of magnitude lower computational complexity in medium network sizes where Opt-PF is computationally feasible. Our results also show the feasibility and high performance of our proposed scheduler in networks with a large number of users and resource blocks.**

***Index Terms*—Massive MIMO, Resource Scheduling, Deep Reinforcement Learning.**

I. Introduction

MASSIVE multiple-input multiple-output (MIMO) is one of the key technologies poised to radically improve the spectral efficiency of the current 5G networks and beyond. Through the use of tens or hundreds of antennas at the base station, it can perform multi-user beamforming to serve tens of users in the same time-frequency resource block (RB). However, scheduling which users to serve simultaneously in each RB plays an important role in achieving the large throughput gains promised by the massive MIMO technology. Beamforming performance can be significantly degraded if there is a substantial correlation in the wireless channels among the scheduled users, as this correlation makes it challenging to effectively focus signal energy when transmitting toward scheduled users. Similarly, separating the signals received from multiple users becomes challenging when their channels are correlated. In networks with high user mobility, the channels of individual users and their correlations with other users within each RB are rapidly fluctuating. This dynamic nature of channel characteristics substantially increases the challenges associated with achieving optimal resource scheduling for massive MIMO networks. Specifically, fair scheduling of radio resources while maximizing spectral efficiency is essential in real deployments. The formulation of the optimal *Proportionally Fair* (Opt-PF) scheduling problem typically results in an integer linear optimization (ILP) problem with an NP-hard solution [1]. The large complexity associated with solving an ILP, when the number of users and resource blocks is large, prohibits designing optimal yet computationally feasible schedulers that can work in the time-stringent 5G and beyond standards. There is a large body of work [2]–[5] that design heuristics or approximation algorithms with low complexity to optimize the spectral efficiency of the networks. However, they either do not evaluate fairness at all or demonstrate poor fairness. This is due to the fact that designing low-complexity approximation algorithms for multi-objective combinatorial optimization problems is typically hard [6].

In the field of artificial intelligence and machine learning, Markov Decision Processes (MDPs) [7] have emerged as a powerful mathematical framework for modeling decision-making problems under uncertainty. MDPs represent sequential decision processes as a set of states, actions, and transition probabilities, where the goal is to find an optimal policy that maximizes a predefined objective function, such as expected cumulative rewards. However, solving MDPs can be computationally demanding, especially for complex problems with large state and action spaces. To address this challenge, Deep Reinforcement Learning (DRL) [8] has gained significant attention in recent years. DRL combines reinforcement learning algorithms with deep neural networks to approximate value functions or policies, enabling the handling of high-dimensional state spaces. By leveraging the representation power of deep neural networks, DRL algorithms have achieved remarkable successes in solving continuous and discrete action space problems in various domains, including robotics [9], game playing [10], and energy management [11]. Notably, DRL has also been applied to solve complex combinatorial optimization tasks. For instance, [12] has adopted DRL to solve the traveling salesman problem, a classic combinatorial optimization problem. Similarly, [13] solves the covering salesman problem through a DRL model. This motivates the need to explore DRL as a potential tool to solve the optimal proportionally fair resource scheduling for massive MIMO networks. Instead of using an explicit mathematical model, decision optimization in a wireless resource scheduler

can be represented as a Markov Decision Process (MDP) whose observations and actions are guided by a well-defined reward function. A DRL agent can then approach an optimum MDP solution by learning from its interactions with the wireless environment. The choice of the DRL model to solve the resource scheduling problem is crucial in achieving high performance and scalability in terms of the number of users in real-world massive MIMO networks. In the recent years, many DRL models for decision making in discrete action space that fit the resource scheduling problem have been proposed. Deep Q-Network (DQN) [10], Double DQN [14], Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C) [15], Actor-Critic with Experience Replay (ACER) [16], and Proximal Policy Optimization (PPO) [17] are a few examples. However, all these models are shown to struggle with large discrete action spaces that are typically present in combinatorial optimization problems, a phenomenon known as action dimensional disaster [18]. Another class of DRL models that deal with continuous action spaces has been used and adapted for discrete action spaces in various domains. For instance, Deep Deterministic Policy Gradient (DDPG) [19] is a popular continuous-based DRL model used to solve a variety of decision problems with large discrete action spaces [20], including resource scheduling in massive MIMO [21], [22]. However, DDPG is known to be very sensitive to hyper-parameter tuning in actual training, especially in high-dimensional and complicated tasks [23].

In this paper, we present a novel DRL framework for the resource scheduling problem in massive MIMO networks. The novelty of our framework is three-fold:

First, we propose a DRL-based scheduler design named SMART, based on the recently proposed soft actor-critic (SAC) model [24]. The SAC model has superior sample efficiency by incorporating an entropy term in its value function and automatic tuning of hyper-parameters. Therefore, it can converge to the optimal solution in large multi-dimensional action spaces much faster than the existing models such as DDPG. Given that SAC is by design used for continuous space problems, we propose to combine SAC with K-Nearest Neighbors (KNN) algorithm to generate discrete outputs corresponding to user scheduling decisions in massive MIMO networks. To achieve the scalability required for real-world massive MIMO networks with a large number of users, we propose a novel dimension division strategy that maps the discrete action set for scheduling to multiple dimensions.

Second, we significantly reduce the state space and, thus, the complexity of the proposed SMART model for massive MIMO by using user grouping labels as the model states instead of the raw channel state information (CSI) matrix. The user grouping labels indicate which users have less correlated channel vectors, hence, are more suitable to be scheduled at the same time. This reduces the computational complexity of the model in both training and inference by $2\times$ without sacrificing spectral efficiency or fairness.

Third, we demonstrate the scalability of SMART to a large number of resource blocks consistent with 5G systems. We demonstrate that our scheduler framework can operate independently on different resource blocks and, at the same time, achieve close to optimal performance.

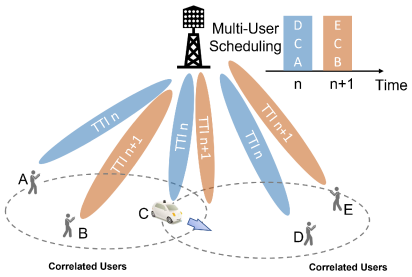


Fig. 1: System Model.

We evaluate the effectiveness of SMART in various channel conditions in both simulated as well as real-world channel traces through a comparison of its performance with state-of-the-art scheduling algorithms, including heuristic-based and DRL-based models. We comprehensively demonstrate the effectiveness of our proposed method in achieving near-optimal spectral efficiency while simultaneously maintaining superior inter-user fairness very close to the Opt-PF scheduler. We experimentally analyze the computational complexity of our method and demonstrate its efficiency. We also provide guidelines on how our proposed system can be deployed on real-world 5G and beyond systems while achieving the latency required for the 5G new radio (NR) standard.

II. System Model and Existing Work

A. System Model

We consider a single-cell network with a massive MIMO base station (BS) with M antennas serving L single-antenna users in its cell. The base station uses orthogonal frequency division multiplexing (OFDM) and performs MU-MIMO transmission and reception to $N < L$ users such that $N \leq M$. We consider time-division duplex (TDD) operation, where all L users periodically send orthogonal pilot sequences to the BS for channel estimation. We assume that the scheduler possesses full knowledge of the channel condition of all users associated with the BS and the channel for each user does not change during a transmission time interval (TTI). Subsequently, the BS selects a set of N users for data transmission and reception through beamforming based on their estimated channel and assigns their modulation schemes, and communicates that information through the control channel. Using their assigned modulation scheme, the selected users will transmit their symbols at the same RB in the uplink and receive them simultaneously in the downlink. A simplified system model is depicted in Fig. 1. For the uplink, we consider the following signal model

$$\mathbf{y} = \mathbf{H}\mathbf{u} + \mathbf{n}, \tag{1}$$

where $\mathbf{y}$ is the $M \times 1$ received signal vector at the BS, $\mathbf{H}$ is the $M \times N$ channel matrix, and $\mathbf{u}$ is the $N \times 1$ transmitted symbols vector by the users. Additionally, $\mathbf{n}$ is $M \times 1$ receiver complex noise vector with a circular Gaussian distribution, $\mathbf{n} \sim \mathcal{CN}(0, \sigma^2\mathbf{I})$ where σ^2 is the noise variance and $\mathbf{I}$ is the identity matrix. Note that, the value of N can vary in each TTI depending on the current channel condition and it can be

bounded by a maximum value $N_{\max}$. We assume the BS uses zero forcing (ZF) for beamforming. The BS calculates the ZF beamformer using the estimated channel $\hat{\mathbf{H}}$ as

$$\mathbf{W} = \hat{\mathbf{H}}(\hat{\mathbf{H}}^H \hat{\mathbf{H}})^{-1}. \tag{2}$$

The BS then performs receive beamforming on the received signal to estimate the transmit symbol vector $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{u}} = \mathbf{W}^H \mathbf{y}. \tag{3}$$

For simplicity, we only consider the uplink, but the above model is extendable to the downlink as well. The above signal model is for a single subcarrier in an OFDM system, but the same model applies to all subcarriers.

An RB is the smallest scheduling granularity in 5GNR, which contains resources in the time and frequency domain. One RB in 5G is made up of 12 consecutive subcarriers in the frequency domain [25]. In the time domain, the composition of RBs in 5G is more flexible and can vary between one OFDM symbol and the entire slot (1 ms in numerology 0). The quality of the wireless channel changes dramatically over time, across users, and among different frequency bands. It is shown in [26] that wireless channel capacity might fluctuate by up to 9 times in 20 MHz LTE bandwidth with over 100 RBs. This effect is more pronounced in 5G since it typically has a wider bandwidth (i.e. 40 MHz to 400 MHz). Consequently, user selection decisions will vary across RBs due to the frequency selectivity of the channel. Thus, it is essential to take into account resource scheduling for every RB individually. In our design, we first focus on resource scheduler design on a single RB and then extend to many RBs to show the adaptability of our proposed scheduler to 5G massive MIMO networks.

Optimal Schedulers: In the literature, multiple schedulers are defined as optimal. The rate-optimal scheduler, known as emphOptimal Maximum Rate (Opt-MR), finds the resource scheduling solution in each TTI that maximizes the sum rate

$$\begin{aligned} \underset{x^t_{l,b}}{\text{argmax}} \quad & \sum_{b=1}^{B}\sum_{l=1}^{L} r^t_{l,b}\ x^t_{l,b}, \\ \text{s.t.} \quad & \sum_{l=1}^{L} x^t_{l,b} \le N_{max} \\ & x^t_{l,b} \in \{0,1\} \end{aligned} \tag{4}$$

where $x^t_{l,b}$ represents the binary selection of user l at TTI t and RB b and $r^t_{l,b}$ is the instantaneous rate achieved by user l at TTI t and RB b. We calculate the instantaneous rate as $r^t_{l,b} = log_2(1+\mathrm{SINR}^t_{l,b})$, where $\mathrm{SINR}^t_{l,b}$ is the received signal to interference-plus-noise ratio from each beamformed user l at TTI t and RB b. We consider B as the maximum number of RBs being used in the system.

Simply maximizing the sum rate ignores the notion of fairness where, depending on the channel conditions, some users may never get selected. Therefore, a commonly used scheduler, known as *Optimal Proportionally Fair* (Opt-PF) scheduler, finds the resource scheduling solution that maximizes the following objective [27], [28]

$$\begin{aligned} \underset{x^t_{l,b}}{\text{argmax}} \quad & \sum_{b}^{B}\sum_{l}^{L} w^t_{l,b}\ x^t_{l,b}, \\ \text{s.t.} \quad & \sum_{l=1}^{L} x^t_{l,b} \le N_{max} \\ & x^t_{l,b} \in \{0,1\} \end{aligned} \tag{5}$$

$$w^t_{l,b} = \frac{r^t_{l,b}}{\sum_b^B\ p^t_{l,b}}, \tag{6}$$

$$p^t_{l,b} = \begin{cases} p^{t-1}_{l,b} + r^{t-1}_{l,b},\ if\ x^{t-1}_{l,b} = 1 \\ p^{t-1}_{l,b},\ otherwise \end{cases}$$

where $w^t_{l,b}$ denotes the weighted rate, which we calculate as the ratio of instantaneous rate $r^t_{l,b}$ to received rate $p^t_{l,b}$ until TTI t on all RBs. Normalizing the instantaneous rate with the total received rate guarantees that all users have a fair chance of getting selected by the scheduler even when they are experiencing a poor channel.

Both optimization problems in (4) and (5) are NP-hard since they can be reformulated as an Integer Linear Programming (ILP) problem [29]. Specifically, we can reformat (5) when $B = 1$ as the following ILP problem,

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmax}} \quad & \mathbf{w}^T\ \mathbf{x} \\ \text{s.t.} \quad & \mathbf{J}_{\mathrm{L,L}}\mathbf{x} \le N_{max}\mathbf{J}_{\mathrm{L}} \\ & \mathbf{x} \in \{0,1\}^L \end{aligned} \tag{7}$$

where $\mathbf{w}$ is a vector of all users instantaneous rates, $\mathbf{x}$ is user binary selection vector. Also $\mathbf{J}_{\mathrm{L,L}}$ and $\mathbf{J}_{\mathrm{L}}$ are square matrix and vector of all ones with size L, respectively.

Solving (7) by exhaustively searching through the combinations of vector $\mathbf{x}$ has the complexity of $\mathcal{O}(2^L)$. Solving (4) and (5) through an exhaustive search, when B RBs are considered, the complexity will increase to $\mathcal{O}(2^{LB})$. However, there are approximate algorithms for the Opt-PF problem with polynomial complexity, such as the one proposed in [28]. We discuss and evaluate an approximate algorithm in §IV along with other benchmarks.

B. Existing Work and Motivation

Recent work on resource scheduling in massive MIMO and MU-MIMO can be classified into two general categories: heuristics schedulers, and AI-based schedulers. In this section, we provide an overview of some of the most relevant works in each category.

Heuristics Scheduler Designs: Many existing MU-MIMO scheduling works provide heuristics-based approximations to the Opt-PF scheduler [5], [30], [31]. While they try to strike a balance between complexity and performance, often their complexity does not scale to large networks or they significantly underperform the optimal scheduling policies.

The scheduler proposed in [31] implements a multi-phase optimization to solve Eq. (5) in MU-MIMO settings. It narrows down the exhaustive search needed for the Opt-PF solution

using some relaxations of the optimization problem. For e.g., it decouples the user selection in different RBs. Moreover, in each RB, it reduces the number of choices based on the channel quality of each user before deciding the user selection action based on the correlation of the remaining users. Through these sub-optimal relaxations, their method can be parallelized and efficiently implemented on a powerful GPU, and hence can meet the stringent 5G-NR latency constraints (i.e., nearly 1ms). Despite the low-latency implementation,, this scheduler only scales to $M = 12$ and $N = 4$, and as a result, it has limited scalability to massive MIMO. In [5], two heuristics-based user scheduling algorithms are proposed and evaluated on channel datasets collected from a dense indoor massive MIMO network with stationary users. However, the algorithms sacrifice fairness in favor of spectral efficiency. They are also not evaluated under mobility scenarios. The work in [32] proposed a scheduler for massive MIMO that schedules users with low correlation channels in the same time slot. It first partitions users into groups through a user grouping algorithm. The scheduler then goes through all groups and schedules all users in each group with a rate-fair method. As we discuss later in the paper, this scheduling algorithm fails to work well in fast-varying channel environments when inter-user channel correlations are continuously changing and it is unable to fairly allocate users across channel coherence blocks.

AI-based Scheduler Designs: Due to the huge complexity of the optimization-based methods, several recent works [18], [21], [22], [33]–[36] have proposed DRL models for MIMO scheduling. A Q-learning-based DRL resource scheduling is proposed in [34]. It models the user scheduling problem as a Markov Decision Process (MDP) that outperforms the round-robin scheduler in terms of sum rate. However, the discrete DRL models are known to have difficulty in converging in large action sets [37]. The convergence issue is also true for more advanced discrete DRL models, such as DQN and Double DQN. As such, discrete DRL models have limited scalability to a large number of users for multi-user scheduling in massive MIMO networks. We will also demonstrate these limitations in §IV.

The work in [21] proposes a DDPG-based user scheduler for massive MIMO networks. Its model outputs a probability distribution over all selectable users and chooses the most promising UE combinations at each TTI. However, it includes a raw channel matrix in state space and the number of elements in action space equals the number of UEs. Large state and action spaces hinder its scalability. This algorithm is extended in [22] for both user scheduling and transmit precoding based on DDPG. It considers multiple antennas and antenna correlation on the UE side as well. However, their proposed scheduler has limited scalability and does not consider the evaluation of user fairness. We implement a DDPG-based scheduler as one of our benchmarks and discuss its performance with respect to our proposed scheduler.

A pointer network is investigated in [18] as the actor in an actor-critic framework to convert the combinatorial problem in multi-user scheduling into a sequential selection problem. However, sequential scheduling has slow inference, which makes it undesirable for latency-sensitive 5G networks. Additionally, applying the model to large networks results in a complicated network structure and a long model update time due to the use of a raw channel matrix as the input. This is exacerbated further by complex-valued channels, which need to be separated into real and imaginary parts before being fed to the model. We implement a pointer network-based DRL scheduler as a benchmark and discuss these limitations in more detail in §IV.

Our Proposed Method: We propose SMART, a massive MIMO user scheduler based on the recently proposed soft actor-critic (SAC) DRL model [23], [24] and the KNN algorithm [38]. SAC has gained attraction in several real-time control problems such as robotic locomotion [39]. SAC was originally designed to handle continuous action spaces. However, the user scheduling is a discrete decision problem where an appropriate set of users must be selected at each TTI. The work in [40] provides a modification of SAC for discrete action spaces, but we find that their modification is still not suitable for large discrete action sets as it has serious convergence issues in large-scale networks. Inspired by the approach in [20], we use the KNN [38] to discretize SAC to adapt it to discrete action spaces. The basic idea is to use a continuous-based algorithm to generate an initial or "proto" continuous action first. Then, the K nearest discrete actions are found by using the KNN algorithm. Among the K nearest discrete actions, the one with the maximum Q value is selected. We further propose a novel dimension division strategy that helps to scale up the size of the combinatorial action set (i.e., number of users in the network) and enhance model convergence capability. Using this approach, we enable our model to dynamically select the users to maximize system spectral efficiency and inter-user fairness. More details are illustrated in §III-C. In contrast to prior work, our proposed scheduler is more scalable and performs very close to the Opt-PF solution.

III. SMART: A Scalable SAC-KNN-based massive MIMO Scheduler

In this section, we first provide a brief introduction to SAC. Subsequently, we describe the design of our proposed scheduler based on the SAC DRL framework. We discuss how we discretize the output of the SAC framework by applying the KNN algorithm and propose a dimension division strategy to scale up the supported size of the action set. We also propose to reduce the complexity of the framework by using the user grouping instead of the raw channel matrix as the input to the framework. Additionally, we discuss how we scale up the model to support as many RBs as needed for realistic 5G networks.

A. A Primer on SAC

SAC is an off-policy Deep Reinforcement Learning (DRL) model that employs a stochastic policy, in contrast to the deterministic policy used in Deep Deterministic Policy Gradient (DDPG). Instead of selecting the optimal action, a stochastic policy outputs probabilities for all possible actions.

The optimal policy in SAC, defined in (8), aims to maximize both the cumulative reward R and the policy entropy H.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[\sum_t R(s_t, a_t) + \alpha H(\pi(\cdot|s_t)) \right]. \tag{8}$$

where the policy entropy H is defined as

$$H(\pi(\cdot|s_t)) = -\sum P(a_t|s_t) \times log(P(a_t|s_t)) \tag{9}$$

By maximizing policy entropy, SAC encourages the model to extensively explore the action space, facilitating the discovery of global optima and enhancing sample efficiency. Moreover, SAC samples transition from replay memory to learn from past experience, similar to other off-policy algorithms like DQN [10] and Double DQN [14]. In contrast to on-policy models such as PPO [17] and A3C [15], which update their policies based on experiences generated by the current policy, SAC has the ability to learn from a broader spectrum of experiences. This characteristic enhances sample efficiency and aids in facilitating convergence, especially in high-dimensional action spaces as demonstrated in [24].

In general, SAC has the following two major benefits:

1) **Strong exploration capability**. SAC does not discard any action, even if it is not the best one. If multiple promising actions are found, the stochastic policy will choose them with equal probability. This feature helps SAC explore more and not easily get trapped in local optima. In contrast, the deterministic policy-based algorithms, such as DDPG [41], save the action with the highest value resulting in fewer exploration opportunities.

2) **High robustness**. Most applications of RL require the agent to perform well in the presence of disturbances in the environment. Because of the adopted stochastic and entropy maximizing algorithm, SAC explores as many potential actions as possible and, hence, it is able to deal with complicated and dynamic environments (e.g., mobility scenarios in wireless communication), including scenarios it has never encountered [42].

Fig. 2 shows the block diagram of the SAC framework. Similar to any actor-critic architecture in DRL, the actor in SAC generates a policy from which an action is drawn based on the current state. The role of the critic is to assess the actor's policy and guide the actor toward the optimal path through feedback. Unlike other actor-critic models, SAC adjusts the Q function by a temperature coefficient (α in (8)), which represents the weight of entropy. Furthermore, in [23], the authors improve SAC with automatic entropy coefficient adjustment. This method significantly reduces the burden of manually adjusting hyper-parameters in training and stabilizes its convergence. In contrast, hyper-parameter tuning and unstable environments are still big challenges for the majority of state-of-the-art DRL models such as DDPG [43]. Another advantage of SAC is its robustness in handling multi-dimensional tasks. High-dimensional tasks are generally challenging to deal with for DRL model due to a phenomenon known as the curse of dimensionality [44]. However, due to the high sample efficiency boosted by entropy maximization, SAC has demonstrated to perform very well in high-dimensional tasks with up to 21 action dimensions [24]. Specifically, SAC is demonstrated to work well in the design of autonomous robots where the actions of multiple parts of the robot must be decided simultaneously. As we discuss later, we use this feature of SAC as our advantage to deal with large discrete action sets in massive MIMO user scheduling.

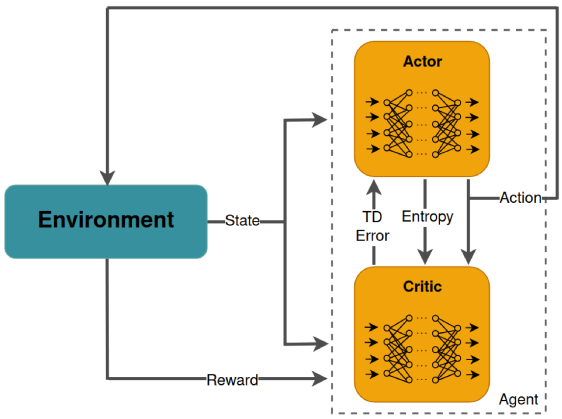


Fig. 2: Soft Actor-Critic Framework.

B. SMART Scheduler Core Design

In this section, we adapt the discretized SAC algorithm [23] to formulate and build a Markov Decision Process (MDP) model to solve the user scheduling problem in massive MIMO networks.

State space. We define the state space of user l at TTI t as $s_t^l := [\gamma_t^l, f_t^l, g_t^l] \in \mathcal{S} := [\Gamma, \mathcal{F}, \mathcal{G}]$, where γ_t^l indicates maximum achievable spectral efficiency of user l at TTI t, f_t^l indicates the total amount of transmitted data by user l up until TTI t, and g_t^l is the user group label of user l at TTI t. The value of γ_t^l can be calculated as the spectral efficiency of user l in SU-MIMO, where only user l is scheduled at TTI t. The users with the same user grouping label g_t^l have low channel correlation so they are preferred to be scheduled together. We will introduce more details on the user grouping strategy in §III-E.

Action space. The action space set $\mathcal{A}$ consists of discrete values, encoding the user-selection decision. We denote the action at time t as $a_t \in \mathcal{A}$. Due to its combinatorial nature, the action set grows exponentially with the number of users in the system. For instance, with a total of L users available, any number of users between 1 and N_{max} can be scheduled at each TTI t, and thus the total number of possible selections is $\sum_{i=1}^{N_{max}} \binom{L}{i}$.

Reward. Our ultimate objective for resource scheduling is to maximize both the system's spectral efficiency and fairness among users. By system spectral efficiency, we refer to the sum rate achieved by all users scheduled together at TTI t. We use a normalized version of this quantity expressed by γ_t^{total}. The normalization factor is calculated as follows. We measure the achievable rates for each user in the system if that user were scheduled individually (SU-MIMO). We then use the sum of the N largest rates out of the total L users as

the normalization factor. This will guarantee a value in [0, 1] which then can be used in the reward function. To quantify fairness, we use Jain's fairness index (JFI) [45], which can be expressed at each TTI t as

$$JFI_t = \frac{\left(\sum_{l=1}^{L} f_l^t\right)^2}{L\sum_{l=1}^{L}\left(f_l^t\right)^2}. \tag{10}$$

As such, we include the normalized spectral efficiency and the JFI in the reward function of the MDP model. The reward R_t achieved at TTI t can be then formulated as

$$R_t = \beta\gamma_t^{total} + (1-\beta)JFI_t. \tag{11}$$

In (11), β determines the relative importance of each item in the reward function based on the preference of the system operator. Note that, both items are the range $[0, 1]$ so that we can effectively adjust their weights in the reward function with parameter β.

C. Discrete Action SAC Design

Originally, SAC is a continuous action space model and thus, it cannot be directly applied to the massive MIMO user scheduling problem. There are existing discrete action space models, such as DQN [10] and Double DQN [14], that could potentially be used to solve the problem. But as we will show in §IV, none of these methods can handle the large action set in massive MIMO user scheduling. Note that, the discrete action space set in multi-user scheduling in massive MIMO increases exponentially as the number of users grows. For example, with $M = 64$ BS antennas and $L = 64$ single-antenna users, or simply a 64×64 network size, and $N_{\max} = 16$ in each TTI, the action set size has up to $\sum_{i=1}^{16}\binom{64}{i} \approx 7 \times 10^{14}$ actions.

Several recent works have attempted to solve the large discrete action space problem by discretizing the continuous-control-based DRL model. In this direction, [20] combines DDPG with KNN to solve problems with large discrete action sets (e.g., recommender systems and language models). More precisely, a KNN approximation [38] is used because of its agile search in logarithmic time. Its fundamental idea is to first generate a so-called proto continuous action (i.e. a real number in $[-1, 1]$) from the continuous action space DRL model. Then, KNN is used to calculate the l^2-norm between the proto action with actions in the discrete space represented by integer numbers corresponding to different actions, sort them in ascending order, and pick the first K ones. Here, K is a system hyper-parameter. Finally, after comparing the Q values of these K discrete actions in the critic network, the one with the highest Q value is chosen as the final action. Similarly, we propose to augment the SAC model with a KNN approximation model that can map the continuous action space to a discrete one. However, the model in [20] is shown to be effective for tasks with up to one million actions, far below the number of scheduling actions encountered in a large massive MIMO network. Next, we propose an idea to scale the feasibility of the model to much larger action sets.

D. Dimension Division

One major drawback of mapping continuous actions to discrete actions is the *decision accuracy loss*. The reason is that, as the size of the discrete action set increases, the corresponding distance between discrete actions in the continuous domain will become extremely small. The precision of each discrete action when mapped from a continuous action space in the range $[-1, 1]$ is equal to $(1-(-1))/2^L$, where 2^L is the total number of discrete actions. When this precision is smaller than the network output precision, it will lead to decision accuracy loss. This precision loss prohibits scaling up the size of the discrete action set. In order to improve the scalability of our model to much larger action sets, i.e. larger number of users, we propose a novel strategy that we call *dimension division*, where we break up the action space into multiple dimensions. As discussed in §III-A, high-dimensional tasks are generally challenging to deal with in DRL models. But here, we particularly rely on the strength of the SAC model in handling multiple dimensions. The difference in our approach is that we use this strength in a multi-dimensional discrete action space. With D dimensions, we can reduce the number of actions in each dimension from 2^L to $(2^L)^{1/D}$ actions. As such, mapping precision is also changed from $(1-(-1))/2^L$ to $(1-(-1))/(2^L)^{1/D}$ in each dimension. Based on this strategy, the continuous-action DRL model will generate proto actions in D dimensions. We apply the approximate KNN to each proto action to generate the K nearest discrete actions in each dimension. Finally, the critic network will pick the discrete action with the maximum Q value to form the final action (i.e. an integer number between 1 and 2^L). This final discrete action is then mapped to a specific user combination from all possible combinations of L users to be scheduled. Fig. 3 illustrates the proposed workflow. In general, to scale up the number of supported users, it is important to strike a balance between the number of dimensions and the size of each dimension. In §IV, we demonstrate that the SMART scheduler is able to perform well with a number of users as high as $L = 128$ whereas DDPG is unable to converge in that scenario.

E. User Grouping

Previous works on DRL-based massive MIMO scheduling [18], [21] use the full channel matrix as the input to their DRL model. The size of the channel matrix is $2 \times M \times L$. The factor of 2 denotes the real and imaginary components of the channel estimate since neural networks are usually designed and trained for real values. As the size of the system (M, L) increases and correspondingly the input size of the DRL model grows, the model convergence becomes more difficult. In order to scale up the model to support large network sizes, the input size must be reduced. To reduce the input size, we adopt the user grouping labels calculated from the inter-user channel correlation matrix to guide the DRL model.

The inter-user channel correlation matrix measures the correlation between each pair of users in the network. Specifically, it is calculated as

$$c_{i,j} = \left|\left\langle \frac{\mathbf{h_i}}{\|\mathbf{h_i}\|_2}, \frac{\mathbf{h_j}}{\|\mathbf{h_j}\|_2} \right\rangle\right| = \frac{\left|\mathbf{h_i}^{\mathrm{H}}\mathbf{h_j}\right|}{\|\mathbf{h_i}\|_2 \|\mathbf{h_j}\|_2} \tag{12}$$

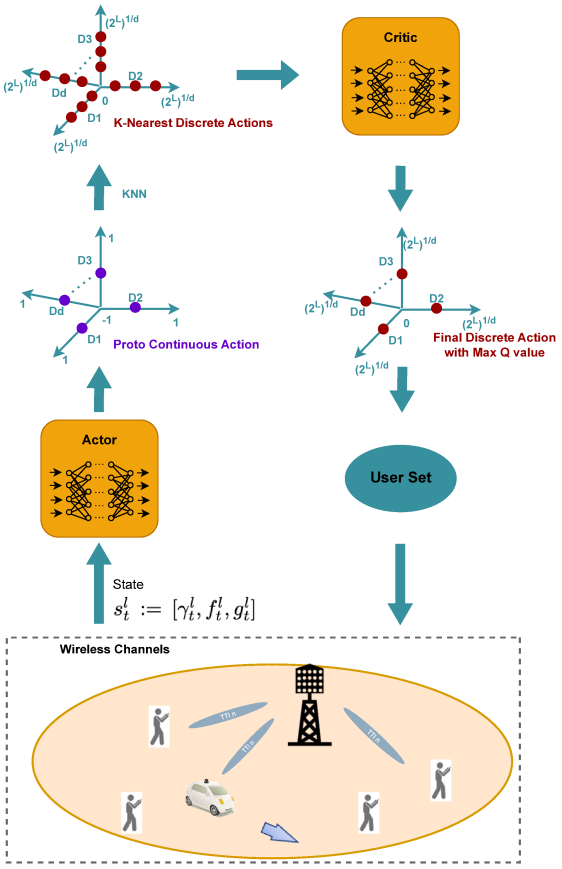


Fig. 3: SMART Architecture.

where $\mathbf{h_i}$ and $\mathbf{h_j}$ are channel vectors of user i and user j in channel matrix $\mathbf{H}$ and $c_{i,j}$ is their channel correlation.

To reduce the complexity of the channel matrix, we adapt a similar user grouping method with [32], as shown in Algorithm 1. The algorithm uses the inter-user channel correlation matrix calculated through equation (12) to partition users with low correlation into separate sets, where the partitioning threshold is c_{th}. During grouping, users in the same group (less correlated users) are assigned the same label. As discussed in §III-B, we only then need to assign a user group label to each user in the state space instead of its complete channel vector. With user grouping labels as input of the DRL model, the state space size will be significantly reduced. As an example, in a 64×64 network size, at each TTI, the state of each user includes three variables: maximum achievable spectral efficiency, the total amount of transmitted data by the user, and user group label. Thus, the total state space size is 192. However, without user grouping, the real and imaginary parts of the raw channel matrix must be fed to the DRL model separately, which leads to a state space size of 8320. Such large-scale inputs will lead to complicated neural network structure, high computation complexity in model updating, and excessive running time (cf. §IV-C).

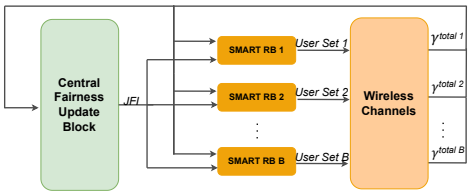
Algorithm 1 User Grouping Algorithm

Input: Channel matrix at TTI t: H_t, user set $\mathcal{L}$ and channel correlation threshold: c_{th}
Output: User group set G

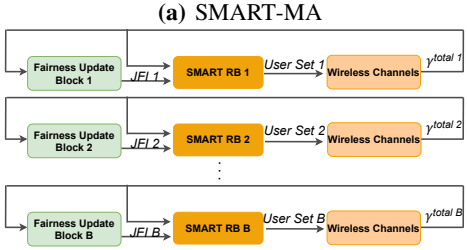
```
Calculate channel correlations of all UE pairs c_{i,j}, ∀i, j ∈ L using Eq (12)
Initialize G = ∅
Let L^c = L
while L^c ≠ ∅ do
    Random pick UE i ∈ L^c and add to the empty user group G_i
    Iteratively search in L^c to find all UEs whose channel correlations with all existing UEs in G_i are smaller than c_th and add them to G_i
    User group G = G ∪ {G_i}
    Update L^c = L^c \ G_i
end while
return User group set G
```

F. Scheduling Across RBs

As mentioned in §II-A, user channel quality varies significantly across RBs. Consequently, the channel correlation among users varies across the RBs as well. This leads to different optimal scheduling solutions for each RB. However, the scheduling decision on each RB will affect the decision on other RBs, particularly as it relates to rate fairness. Since the goal is to maximize both system spectral efficiency and fairness for the whole system, as expressed in equations (4) and (5), the optimal scheduling on all RBs needs to be jointly considered. One way to model this problem is to have independent SMART frameworks, as described in §III-B-§III-E, to make decisions on each RB, with the additional modification that each framework uses the decision from other frameworks running on other RBs to calculate the new fairness in its state space and the new reward, akin to the formulation of weighted rates in Eq. (5). A block diagram of such a model is depicted in Fig. 4a. This model can be regarded as a cooperative multi-agent DRL framework where each SMART framework responsible for a different RB acts as a separate agent that shares its decisions with other agents. We refer to this overall model as SMART-MA. In SMART-MA, agents of RBs are jointly optimized. The instantaneous spectral efficiency of each user is aggregated from all RBs and JFI is updated based on a global user scheduling decision rather than an individual RB's decision. Consequently, the SMART agents of all RBs share the same reward function and engage in cooperative learning. However, multi-agent DRL models are known to be difficult to converge, especially as the number of agents scales up [46]. We demonstrate this by employing a multi-agent model in §IV. For fading channel models, the inter-user channel correlation across RBs will be largely random, and when dealing with a large number of RBs, it is expected that the fairness across RBs will be smoothed out. With this assumption, and given the limitation of the multi-agent model, we propose to use a fully independent model for each RB referred to as SMART-SA and depicted



(a) SMART-MA



(b) SMART-SA

Fig. 4: Fully independent SMART (a), and multi-agent SMART frameworks (b) for scheduling users across RBs.

in Fig. 4b. In the SMART-SA, an independent SMART DRL model is implemented for each RB. Each RB possesses its own distinct state space (not depicted in the diagram) and generates a scheduled user set specific to that RB. Based on the user scheduling decision made by the model, selected users are allocated resources within the wireless environment, and the instantaneous spectral efficiency γ^{total} of each scheduled user can be determined. Sequentially, the accumulated amount of transmitted data and JFI are updated in the respective fairness update block.

In §IV, we demonstrate the effectiveness of SMART-SA for a large number of RBs in getting close-to-optimal results.

IV. Performance Evaluation

In this section, we perform a comprehensive evaluation of our proposed scheduler design. We compare SMART with multiple different schedulers with respect to their achieved normalized spectral efficiency and JFI in various channel conditions. We also provide a comparison of the computational complexity of our DRL-based scheduler with other methods and discuss the feasibility of our scheduler in real-time 5G settings.

A. Experimental Setup

We perform our evaluations in both simulated channels as well as real-world channels measured with a massive MIMO hardware platform. For simulated wireless channels, we use the Quasi Deterministic Radio Channel Generator (QuaDRiGa) [47] software. Specifically, we generate the 3D Urban Micro (UMi) Line Of Sight (LOS) channel model. We consider two channel scenarios: static and mobile. For static channels, we consider two different modes: 1) four user clusters, and 2) random user placement. In the mobile scenario, the base station is positioned at the center of a circular area with a radius of 300 meters. Users within this circle move in various directions at different speeds, with an average speed of 2.8 m/s. The initial positions of the users are randomly assigned, and they will bounce back into the area upon reaching the boundary. We describe the experimental setup for the real-world measured channels in §IV-C4. We implement the system model in §II-A using Python. In terms of modulation scheme, we adopt 16-QAM in our wireless channel simulator and use Error Vector Magnitude (EVM) of the received constellation to derive SNR as demonstrated in [48].

We run our experiments on an NVIDIA DGX A100 server [49]. Both actor and critic networks implement neural nets with two hidden fully connected layers and ReLU activation functions. We use the Adam optimizer [50] to train our DRL model in PyTorch [51]. The most relevant parameters used in our simulations are shown in Table I.

TABLE I: Simulation and Training Parameters

Parameter	Value
Channel Model	3GPP_3D_UMi_LOS
System Bandwidth	20 MHz
System Carrier Frequency	3.6 GHz
TTI Duration	1 ms
Modulation	16QAM
Cell Radius	300 m
UE Speed	0 & 2.8 m/s
Number of BS Antennas	16 & 64
Number of UEs	16 & 64
Batch Size	256
Actor Learning Rate	5e-4
Critic Learning Rate	5e-4
Alpha Learning Rate	3e-4
Automatic Entropy Tuning	True
Optimizer	Adam
Episodes	800
Iterations In Episode	400
Correlation Threshold c_{th} in Algorithm 1	0.5
β in Eq. (11)	0.5

B. Benchmarks

In order to do a thorough comparison, we implement various scheduler models as benchmarks including classical and heuristics-based schedulers, discrete-control-based DRL schedulers, continuous-control-based DRL schedulers, and attention-mechanism-based RL schedulers.

Classical Scheduler: We consider Opt-PF, Opt-MR, an approximate PF (Approx-PF) and a heuristics-based algorithm as classical schedulers. Algorithms of Opt-PF and Opt-MR are introduced in §II-A. Given the exceedingly high computational complexity involved in employing optimal schedulers for large-scale networks, we devise a variation of an approximate Proportional Fairness (Approx-PF) scheduler in [28] that offers reduced complexity from Opt-PF presented in §II-A. The algorithmic details of this particular implementation can be found in Algorithm 2. In this approach, we first calculate a weighted-rate matrix similar to Opt-PF in (5) and then select N_{max} users with the highest weighted rates. Consequently, the computational complexity is reduced significantly from $\mathcal{O}(2^L)$ to $\mathcal{O}(2^{N_{max}})$. However, this is still too complex in large-scale networks and thus needs to be simplified further. Unlike the approximate scheduler described in [28], we do not consider the individual data load of each user in our work. Instead, we implement the user grouping in Algorithm 1 in

Algorithm 2 Approximate Proportional Fairness (Approx-PF) Algorithm

Input: Resource block set $\mathcal{B}$, Channel matrix of resource block b at TTI t: $H_{t,b}$ and user set $\mathcal{L}$
Output: Scheduled user set on resource block b: $\mathcal{U}_b$
1: Calculate weighted rate $w_{l,b}^t$ for all L users on resource block b at TTI t using (5)
2: Sort and select N users with the highest weighted rate on resource block b to construct a subset of user $\mathcal{N}_b$
3: Do user grouping in user subset $\mathcal{N}_b$ as Algorithm 1
4: Find the user group $\mathcal{U}_b$ with the most users as the scheduled user set on resource block b at TTI t
5: **return** $\mathcal{U}_b$

this user subset and select the group with the most users. User grouping strategy helps Approx-PF to avoid scheduling highly inter-correlated users, thereby improving overall system performance and releasing the heavy complexity to $\mathcal{O}(N^2)$.

As for the heuristics-based benchmark, we use the algorithm in [32]. This algorithm groups users based on their channel correlation and allocates power to the users in the selected group. It then proposes to schedule the groups in a round-robin fashion. We implement a variation of the scheduler proposed in [32]. We assume perfect power control in our model to enable fair comparison with the modified algorithm. We refer to this benchmark algorithm as RR-UG. As we demonstrate later, this algorithm, while effective in static user scenarios, becomes ineffective in highly mobile channel scenarios where channel correlations are continuously changing. We expect a similar behavior by other heuristic methods that rely on channel correlation-based user grouping.

Discrete-control-based DRL Scheduler: There are several DRL models for discrete action spaces in the literature. We select DQN [10] and Double DQN [14] with Prioritized Experience Replay Buffer (PERB) [52] as two representative discrete-control-based DRL algorithms. The study in [16] shows a comparison of these two model with other discrete DRL models such as ACER and A3C and shows the superior performance and convergence of our selected benchmarks. We implement both discrete-control-based DRL models as benchmarks and refer to them as PRTY-DQN and PRTY-DDQN. To balance exploration and exploitation, we adopt the epsilon-greedy algorithm in both models. For fair comparison against other benchmarks, we tune the hyper-parameters so as to achieve the best possible performance [16], [17], [24]. Because of the simple neural network structure of PRTY-DQN and PRTY-DDQN, we adopt grid search to comprehensively identify the optimal hyper-parameters. For PRTY-DQN, we implement 2-hidden-layer neural networks with 32 neurons in each layer. We use the same settings in the main network and the target network of PRTY-DDQN. For both models, we set the same state space, action space, and reward function as our proposed scheduler.

Continuous-control-based DRL Scheduler: Similar to SAC, DDPG is also a continuous-control-based DRL model that has been used to solve optimization problems with large action sets, e.g., on massive MIMO user scheduling [20], [21]. To compare SAC with a DDPG-based scheduler, we replace the SAC module in our design with DDPG and use it as our benchmark. For fairness of comparison, this benchmark adopts the same dimension division strategy as our design to generate multi-dimensional scheduling actions, particularly in evaluating 64×64 network size. Furthermore, we use the same state space and reward function as well as the epsilon-greedy algorithm for this benchmark algorithm as in our proposed scheduler.

Attention-mechanism-based RL Schedulers: We implement a pointer-network-based scheduler (PN) as proposed in [18] in an actor-critic architecture. The PN is used as the actor network, which consists of a long short-term memory (LSTM)-based encoder and decoder. The critic network is a multi-layer perceptron (MLP) and is trained using stochastic gradient descent. A limitation of this model is that the number of scheduled users needs to be fixed. Thus, in our evaluation of the PN scheduler, we set the number of scheduler users N to be so that $M/N \approx 4.5$ which is shown to be the near-optimal number for the ZF beamformer [53].

Our Proposed Scheduler: We implement two variants for our scheduler: 1) a variant with raw channel matrix as input that we call SMART-Vanilla, and 2) a variant with user grouping labels as input (as described in §III-E) that we simply call SMART.

In our evaluations, the Opt-PF scheduler serves as the optimal benchmark for fairness while the Opt-MR scheduler is optimal for spectral efficiency. For thoroughness, we first rule out the discrete DRL-based scheduler, i.e. DQN and Double DQN, due to their inability to scale to large network sizes. Second, we compare the remaining benchmarks in a medium 16×16 network size and in different channel conditions. This allows comparison of the AI-based benchmarks with Opt-PF and Opt-MR schedulers when they are still in a computationally feasible range. Lastly, we increase the size of the network to 64×64, which we consider a real-world network size. In this network size, both Opt-PF and Opt-MR schedulers become computationally infeasible and thus, we only compare our proposed schedulers with PN, DDPG, and RR-UG.

C. Results

1) Model Training and Convergence

We trained the SMART model, in a 64×64 network size, for 800 epochs with 400 iterations in each epoch. To ensure model convergence and learning performance, we divide 8 dimensions in action space and 256 actions in the action set of each dimension, as discussed in III-C. The training takes about five hundred epochs which is when the DRL model converges. During the training process, we employ the epsilon-greedy algorithm to effectively manage the trade-off between exploration and exploitation. This is achieved by selecting random actions or utilizing learned actions that yield the highest reward. The value of epsilon denotes the probability of selecting random actions for exploration purposes. Initially, we set epsilon to 1, and gradually decrease it to zero over a span of five hundred epochs.

We also trained SMART for a 128×128 network. To deal with this extremely large action set, we break it down into 16 dimensions with 256 actions in each dimension for sufficient decision accuracy. With these parameters, we find that our DRL model can still converge. Conversely, all other RL-based benchmarks, except PN, fail to converge in this scenario. However, as we show later, the training and inference time for PN is significantly larger and its performance in terms of fairness is inferior to our scheduler. It is important to highlight that SMART-Vanilla cannot converge in networks of this size either due to the excessive state space. This observation further emphasizes the motivation behind incorporating user grouping in SMART.

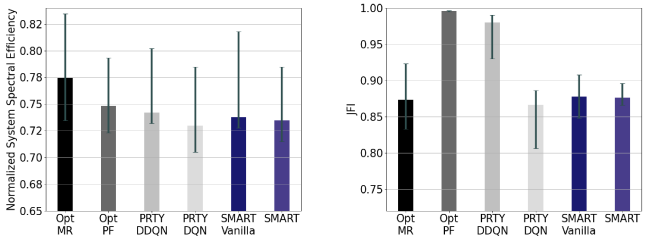
Convergence of PRTY-DQN and PRTY-DDQN: Discrete-based DRL is intuitively a suitable choice to deal with discrete combinatorial optimization problems, such as resource scheduling, by modeling them as MDPs. However, in problems with large action sets, the discrete-based DRL model is shown unable to converge during the training process [18], [54], an effect known as the action dimension disaster [18]. We also demonstrate this effect by training PRTY-DQN and PRTY-DDQN on multiple network sizes. Our experiments show that the largest network size that these models could converge is 4×4, and $N_{\max} = 2$. In this configuration, the size of the action set is 10.

2) Performance Comparison in Various Network Sizes

In the testing phase, we run our simulation environment for additional 400 TTIs in the same cell and use the trained model to schedule users while recording the spectral efficiency and the JFI values across TTIs. For a fair comparison, we use the exact same channels generated as input to all benchmarks. It is important to note that partial or outdated channel information could impair the performance of the resource scheduler, particularly in scenarios involving high-speed mobility. This impacts any system that relies on the channel information for scheduling decisions and thus is beyond the scope of our work. Nevertheless, in this case, complementary methods that perform channel prediction based on the partial or outdated channel information such as the ones proposed in [55]–[57] can be used to enhance the performance of the scheduler. In the following, we provide evaluation results of various benchmarks in multiple network sizes. In each network size, we plot the average spectral efficiency and JFI over all TTIs. We also display error bars in each plot indicating the minimum and maximum values of results across TTIs.

Small network size: We consider the 4×4 network configuration in a mobile scenario, to compare the performance of PRTY-DQN and PRTY-DDQN with our proposed scheduler.

Fig. 5 shows that PRTY-DDQN outperforms PRTY-DQN and SMART-Vanilla on both spectral efficiency and JFI. This is due to decision accuracy loss imposed by mapping the SAC output from continuous space to discrete space in our scheduler, as discussed in §III-C. However, the limitation on the scalability of PRTY-DDQN makes it impractical to use in real-world network sizes. mportantly, we observe that the performance of SMART is almost the same as SMART-Vanilla. This is an important finding since it shows using user grouping labels as input to our model instead of the raw



(a) **(b)**

Fig. 5: Spectral Efficiency and JFI comparison of SMART with DQN and Double DQN in user mobility scenario and 4×4 network size.

channel matrix as in SMART-Vanilla simplifies neural network structure while not impairing model performance.

Medium network size: For thorough comparison of all the other benchmarks, we consider the case for medium 16×16 network size, and $N_{\max} = 4$. We only compare the benchmarks with SMART-Vanilla for a fair comparison with other AI-based schedulers which use the raw channel matrix as input. To be able to reason about the performance of each scheduler, we start with a toy network scenario where the users are static and placed in four clusters (4-cluster). The users in each cluster share the same scatters and experience similar small-scale fading, and thus their channel vectors are highly correlated. Fig. 6 shows the spectral efficiency and JFI results in the four-cluster channel mode. It is evident from Fig. 6a that SMART-Vanilla performs very close to Opt-PF scheduler, which shows SMART-Vanilla is able to converge to the Opt-PF solution almost perfectly. In terms of JFI, Fig. 6b shows that SMART-Vanilla closely follows the Opt-PF scheduler as well. Both schedulers underperform the Opt-MR scheduler in terms of spectral efficiency, but the Opt-MR scheduler is not doing well with respect to JFI as expected, since it is only optimizing the spectral efficiency. Interestingly, Fig. 6a also shows the DDPG-based scheduler significantly under-perform SMART-Vanilla. That shows DDPG fails to explore widely enough because of its sample inefficiency and therefore gets stuck in a local optimal. Lastly, we observe that RR-UG achieves a good spectral efficiency and is almost close to SMART-Vanilla. This is expected as the user grouping algorithm groups the users into exactly four groups based on four clusters. Since the users do not move, RR-UG will continue to serve each group at a time. The results also show that SMART-Vanilla can learn the inter-user correlation well, despite using the raw channel matrix from each user. PN is able to achieve near-optimal spectral efficiency but undesirable JFI. The reason is that PN can not deal with varying state representations of the input [58]. Specifically, sequentially selecting the users will affect the fairness in the state space of the MDP model. Therefore, PN fails to optimize the JFI, while still performing well in terms of achieved spectral efficiency.

Figs. 7a and 7c show the normalized spectral efficiency for random placement of static users in the cell and mobile users moving in random directions within the cell, respectively. In both scenarios, we observe that SMART-Vanilla

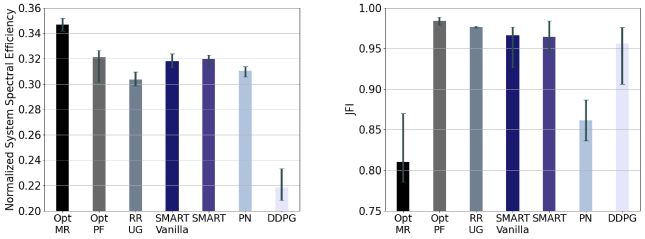


Fig. 6: Spectral Efficiency and JFI comparison of SMART and existing methods in 16×16 network size and $N_{\max} = 4$ in 4-clusters topology.

still performs very closely to the Opt-PF scheduler, while the DDPG scheduler significantly underperforms SMART-Vanilla. The PN performance also slightly drops compared to the 4-cluster scenario. This can be attributed to the limitation of this scheduler with respect to its predefined number of selected users. Note that in the 4-cluster scenario, the predefined number of scheduled users for PN is exactly the same as the number of users in each user group where users have very low correlation. However, in the random placement scenario, this condition does not necessarily hold and the number of scheduled users by PN could be smaller or larger than the optimal set of users. The PN performance gets worse in the mobility scenario since user grouping is changing over time. For instance, PN could select user sets with high correlation in most cases.

RR-UG achieves a relatively good performance in random placement topology, but it does not achieve the same level of performance as in the 4-cluster channel mode. The reason is that in the setups with random user locations, the user groups could include a larger number of users than $N_{\max} = 4$, and thus the groups have to be broken into smaller subgroups to be scheduled sequentially. This impairs the performance of RR-UG. In the mobility scenario, the performance of RR-UG drops even more. This is due to the variations in channels and user groupings caused by mobility in each TTI. It shows that while RR-UG might be a favorable scheduler in static scenarios (due to its lower computational complexity as we show later), in the mobility scenarios, it does not perform that well. In Figs. 7b and 7d, we see SMART-Vanilla and DDPG achieve high fairness values. A good fairness result for DDPG is expected as fairness is accounted for in the reward function. Opt-MR and RR-UG do not achieve high fairness in both scenarios. For RR-UG, the fairness drops since the user groupings change continuously, and thus the rate fairness cannot be met efficiently despite the time fairness due to the Round-Robin scheduling of groups. It is evident that PN performs very poorly with respect to JFI, as discussed earlier.

Real-world network size: We consider a more realistic network size with a 64-antenna massive MIMO base station[1] at the center of the cell. We also consider $L = 64$ connected

[1]Most commercial deployments of massive MIMO include 64-antenna base stations

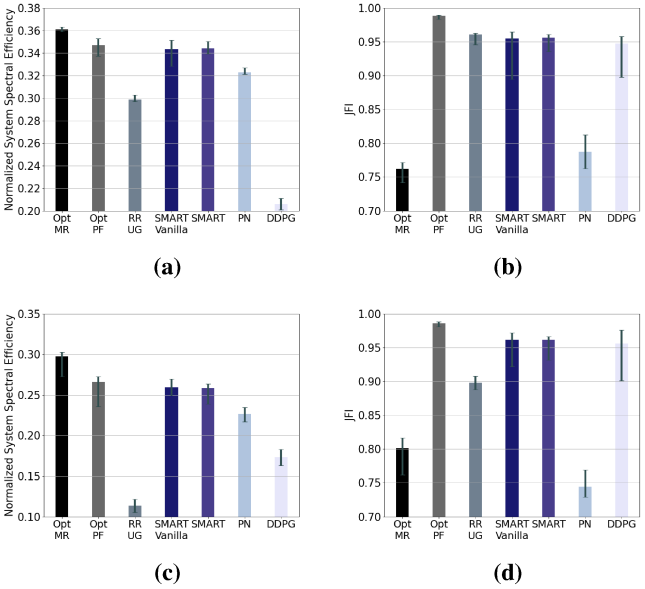


Fig. 7: Spectral Efficiency and JFI comparison of SMART and existing methods in 16×16 network size and $N_{\max} = 4$ in static random user topology (a) and (b), and user mobility scenario (c) and (d).

users which is also a realistic number in small cells [5]. In this case, we assume $N_{\max} = 64$ which means the scheduler can choose to beamform to up to all 64 users in one TTI. In this network size, the complexity of calculating the results for Opt-MR and Opt-PF is too high.Thus, we include Approx-PF as a benchmark instead of Opt-PF along with the results for SMART-Vanilla, SMART, PN, DDPG, RR-UG. As shown in Figs. 8a and 8c, SMART-Vanilla outperforms PN, DDPG, RR-UG, and Approx-PF. By foregoing the exhaustive search, Approx-PF aims to reduce computational complexity. However, we can see that its performance falls short compared to SMART. Similar to our earlier results for medium network size, the performance of RR-UG is close to SMART-Vanilla in static random user placement but drops significantly in the mobility scenario. To enable DDPG to converge in this scenario, we applied the dimension division presented in III-C to its implementation. However, DDPG is unable to perform well in multi-dimensional action sets as discussed earlier. This explains the observation that DDPG does not perform well in terms of spectral efficiency. As we observed in the small and medium networks, the performance of SMART is comparable to that of SMART-Vanilla in both channel scenarios. It demonstrates the effectiveness of using user grouping labels in the state space of SMART.

All schedulers, except PN, achieve high fairness in the static random user placement scenario. In the mobility scenario, the fairness for RR-UG also drops significantly due to varying user groupings across TTIs. Here, PN has the worst JFI for the same reason as we mentioned for the medium network size.

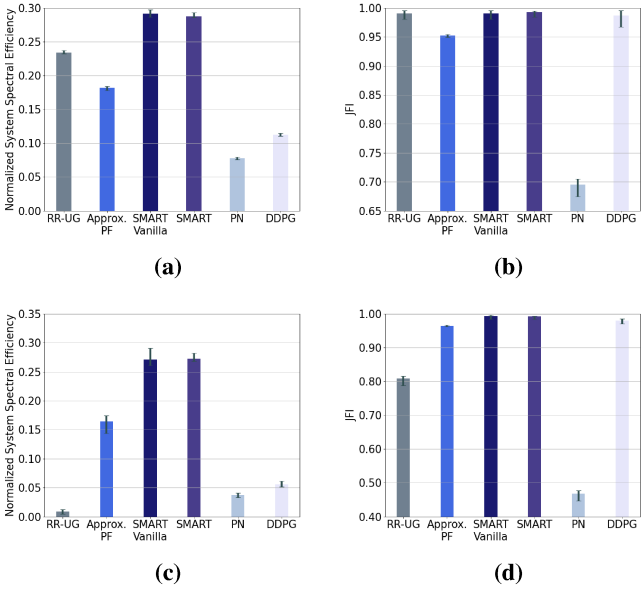


Fig. 8: Spectral efficiency and JFI comparison of SMART and existing methods in 64×64 network size in random user topology (a) and (b), and user mobility scenario (c) and (d).

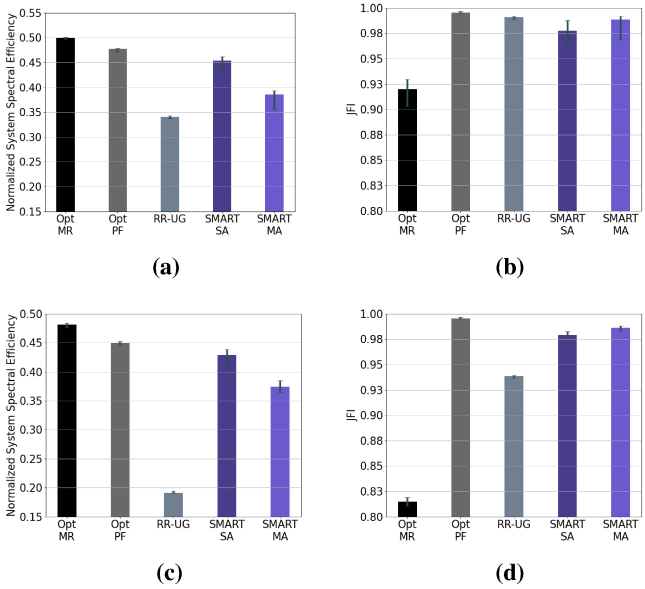


Fig. 9: Spectral Efficiency and JFI comparison of SMART and existing methods in 8×8 network size and $N_{\max} = 4$ with 2 Resource Blocks in static random user topology (a) and (b), and user mobility scenario (c) and (d).

3) Multi-RB Scheduling Performance

Here, we consider the multi-RB scenario and evaluate the performance of our model presented in III-F. As discussed, the multi-agent DRL models are generally difficult to converge. In fact, our SMART-MA model only converged with 2 RBs ($B = 2$) when $M = 8$, $L = 8$, and $N_{\max} = 4$. Thus, we use this configuration to demonstrate the efficacy of SMART-SA, with respect to SMART-MA. Computational complexities of Opt-PF and Opt-MR were also acceptable in this configuration as presented in §II-A, and thus, we include them in the evaluation along with RR-UG. Since we showed the underwhelming performance of DDPG and PN in the single-RB case, we exclude them from this evaluation. Fig. 9 shows the experiment results for $B = 2$. It is evident that SMART-SA outperforms SMART-MA on spectral efficiency but has a slightly lower JFI. The reason is that SMART-SA tries to maximize spectral efficiency on each RB and sacrifices fairness as opposed to SMART-MA which balances the two metrics across RBs. SMART-SA performs much better in terms of both JFI and spectral efficiency compared to RR-UG. For $B > 2$, SMART-MA, Opt-PF, and Opt-MR become infeasible. However, to demonstrate the performance of SMART-SA, we evaluate it for $B = 100$ with a 64×64 network size and compare it with RR-UG. The evaluation results are shown in the simulation column of Table II. For the results, it is evident that a large number of RBs will not degrade JFI in SMART-SA while still maintaining desirable spectral efficiency. It also reaffirms our previous finding on the low performance of RR-UG in the mobility scenario.

4) Real-World Data Evaluation

To evaluate our proposed scheduler in real-world environments, we conducted a massive MIMO channel measurement experiment in an indoor setting on the Rice University campus. We used a 64-antenna RENEW [59] software-defined massive MIMO base station and seven software-defined clients in a large open area inside a building hall. We fixed six of the clients in a circle, 15m away from the base station. The seventh node was placed on a robot where we moved the robot across the hall starting from the location of the first client to the last. A drawing of the BS and client placements are shown in Fig. 10. We moved the robot along the path with different speeds, i.e. with 0.5m/s, 1m/s, and 2m/s. The mobile node's antenna was facing the base station in all the experiments (LoS channel). We repeated the experiments to measure both LoS and NLoS channels for the fixed clients. In each measurement, we transmitted time-orthogonal uplink pilots from all clients to the BS. The uplink pilots were based on the 802.11 LTS OFDM signal, which contains 52 non-zero subcarriers. We consider each subcarrier as an RB in our evaluation, i.e. $B = 52$. Based on the collected real-world dataset, we train and evaluate the performance of SMART in the 64×7 MIMO configuration with 52 RBs in a slow-speed mobility scenario.

Using these datasets, we evaluate the performance of SMART. Due to convergence issues and excessive computational complexity of other schedulers for $B > 2$ as discussed in §II-A, we are only comparing SMART-SA with RR-UG. The results, listed in Table II, show that RR-UG underperforms SMART-SA in both spectral efficiency and JFI. More importantly, SMART-SA is capable of achieving near-optimal (i.e. about 0.996) JFI, which demonstrates the effectiveness of SMART-SA when applied to multiple RBs. However, we can anticipate that RR-UG performance will get worse as the number of mobile users increases, which is consistent with the results of mobility scenarios in medium

TABLE II: Spectral Efficiency and JFI comparison of SMART and RR-UG with multiple RBs in simulation discussed in §IV-C3 and with real-world data discussed in §IV-C4

Performance Metrics	Simulation with $B = 100$				Real-world Data with $B = 52$					
	Random Placement		Mobility Scenario		LoS Slow-speed		LoS High-speed		NLoS Slow-speed	
	SMART-SA	RR-UG	SMART-SA	RR-UG	SMART-SA	RR-UG	SMART-SA	RR-UG	SMART-SA	RR-UG
Normalized System Spectral Efficiency	0.500	0.254	0.400	0.063	0.713	0.662	0.670	0.584	0.488	0.481
JFI	0.977	0.940	0.950	0.696	0.996	0.952	0.995	0.951	0.986	0.980

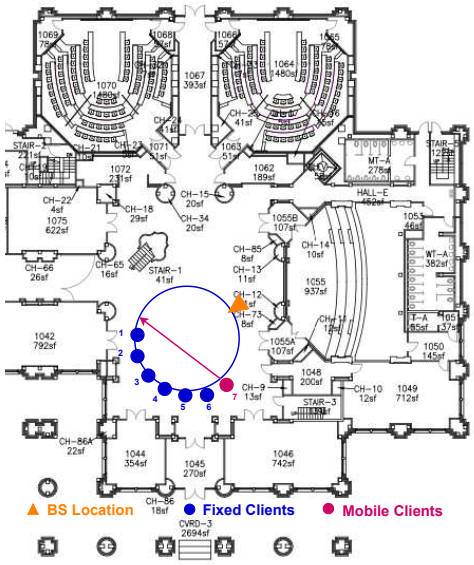


Fig. 10: Topology of the real-world indoor experiment.

and real network size experiments. By running Algorithm 1 on the datasets, we observe just one or two user groups in most TTIs. Thus, RR-UG schedules all seven clients in one or sometimes two TTIs. Therefore, RR-UG is rather competitive as SMART-SA here. For the purpose of showing the generality of our model, we use the model trained on the LoS slow-speed dataset and test it in the LoS high-speed mobility. The results in Table II demonstrate the adaptability of SMART-SA to different mobility scenarios. Compared with the slow-speed mobility scenario, it is obvious that the performance gap between SMART and RR-UG in the high-speed scenario is larger. This is because a high speed makes channel condition and inter-user channel correlation vary more quickly than the slow speed. Faster varying inter-user channel correlation results in quicker variations of user grouping, which makes it challenging for RR-UG to adapt fast enough. However, SMART is capable of dealing with this rapid change. For comprehensiveness, we also test the trained model on NLoS slow-speed topology. The results in Table II show SMART-SA's superiority over RR-UG and its generality in real-world data, albeit not as good as it is in LoS high-speed.

5) Computational Complexity

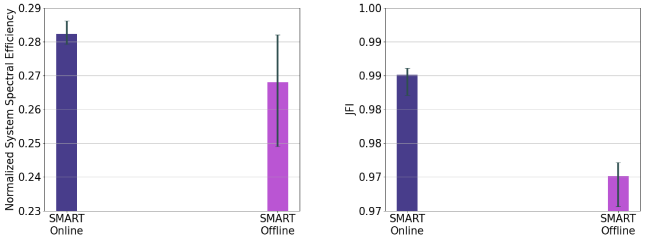
We measure average wall-clock time per TTI for all the schedulers discussed in §IV-C2. For comparison fairness, we run all implementations on a single CPU core on the NVIDIA DGX server. The runtime values are listed in Table III for three network sizes considered in §IV-C2. The results show the runtimes of the schedulers are widely different and they also vary with the network size. For Opt-MR and Opt-PF, the runtime increases exponentially with the network size and thus is not listed for network sizes beyond 16×16. Even though Approx-PF is feasible in real-world size networks with much less complexity than Opt-PF, it still takes about $20\times$ times longer than SMART to execute. Regarding other schedulers, the runtime seems to increase linearly. Both DDPG and SMART-Vanilla show similar results. Comparing SMART and SMART-Vanilla results show that using user grouping labels instead of the raw channel matrix reduces the runtime of the model up to 50%. Tuning hyper-parameters to achieve the best performance for both SMART and SMART-Vanilla, SMART has 3 fewer hidden layers and half the number of neurons in each layer to remain on par with the performance of SMART-Vanilla. However, user grouping requires only an additional 3.5 ms in 64×64 network size, a negligible portion of the total runtime. The runtime for PN is about 1.6x and 4x running time of SMART-vanilla in 16×16 and 64×64, respectively. This is due to the fact that pointer networks are auto-regressive and make decisions sequentially and thus have slow inference. RR-UG shows the smallest runtime among all, but it is not as spectrally efficient as SMART, especially in mobility scenarios.

TABLE III: Wall-clock time in seconds per TTI

System Configuration	Scheduler							
	Opt-MR	Opt-PF	Approx-PF	RR-UG	DDPG	PN	SMART-Vanilla	SMART
16×16	0.15	0.21	-	0.0013	0.034	0.059	0.036	0.024
64×64	-	-	0.604	0.0043	0.058	0.235	0.057	0.030
128×128	-	-	-	-	-	-	-	0.071

D. Discussion and Future Work

The results presented earlier offer good insights into the performance and computational complexity of the proposed SMART scheduler with respect to the existing methods. However, an important question is whether SMART can be deployed to operate in time-stringent 5G-NR systems. For a realistic network size, Table III shows SMART takes as much as 30 ms to run an iteration, $30\times$ longer than one TTI in the least time-stringent mode of 5G-NR [31]. This may seem problematic for the adoption of SMART. To investigate this, we run an experiment in a mobility scenario. We first train SMART offline as before and test the trained model on the testing dataset without online updates to the model. We compare the spectral efficiency results for the offline trained model with the previously presented results that include the online updates. The results are shown in Fig. 11. We observe that, even when we use the offline trained model with no online updates, the performance is remarkably close to when



(a) **(b)**

Fig. 11: Evaluation of SMART with and without a model ((online vs. offline) update in user mobility scenario and 64×64 network size.

the model is continuously updated. The performance can get even closer when we do updates every few tens of TTIs. This finding means that we can only look into the inference time of the model as the scheduling decision time. For 16×16 and 64×64 network sizes, the inference times for SMART are 5.4 and 8.7 ms. Running the model on a single GPU core on the NVIDIA DGX A100 server reduces the inference time values to 1.2 and 1.6 ms, respectively. The inference runtime values can be further reduced to sub-millisecond levels, as required in 5G-NR, by a more efficient implementation such as with CUDA [60] framework and parallelizing the DRL model on several GPU cores. More importantly, the reassuring performance of SMART-SA, demonstrated in §IV-C3, shows that we can get similar runtime values for 100s of RBs, as its architecture allows us to fully parallelize it on different GPU cores.

Lastly, we have only considered saturated traffic for each user. A more generic design should consider the incoming traffic model as well as the quality of service (QoS) requirements, e.g. data rate and latency, for each user. Formulation of the scheduling problem and formally solving it using optimization techniques or heuristics-based approximation is a difficult task. We believe AI-based methods such as the one proposed in this paper provide a more promising avenue for solving the generic case if enough training data exists. We leave the design of a more comprehensive scheduler that considers parameters in the higher layers of the network such as traffic models and QoS constraints as future work.

V. Conclusion

In this paper, we presented SMART, a resource scheduler for massive MIMO networks based on the soft actor-critic DRL model. We demonstrated the effectiveness of our scheduler in achieving both spectral efficiency as well as fairness very close to the optimal proportionally fair scheduler. We also showed that our model outperforms state-of-the-art massive MIMO schedulers in all scenarios, and particularly in mobility scenarios. We removed the need for raw channel matrices in training our DRL model by utilizing a user grouping algorithm based on the inter-user correlation matrix and, thus, we significantly reduced the complexity of our model. We also provided guidelines as to how our scheduling model can be deployed in time-stringent 5G-NR systems.

References

[1] L. Li, M. Pal, and Y. R. Yang, "Proportional fairness in multi-rate wireless LANs," in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 2008, pp. 1004–1012.

[2] S. Huang, H. Yin, J. Wu, and V. C. M. Leung, "User selection for multiuser MIMO downlink with zero-forcing beamforming," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 7, pp. 3084–3097, 2013.

[3] K. Ko and J. Lee, "Multiuser MIMO user selection based on chordal distance," *IEEE Transactions on Communications*, vol. 60, no. 3, pp. 649–654, 2012.

[4] N. Prasad, H. Zhang, H. Zhu, and S. Rangarajan, "Multiuser scheduling in the 3GPP LTE cellular uplink," *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 130–145, 2014.

[5] C.-M. Chen, Q. Wang, A. Gaber, A. P. Guevara, and S. Pollin, "User scheduling and antenna topology in dense massive MIMO networks: An experimental study," *IEEE Transactions on Wireless Communications*, vol. 19, no. 9, pp. 6210–6223, 2020.

[6] A. Chassein, M. Goerigk, A. Kasperski, and P. Zieliński, "Approximating combinatorial optimization problems with the ordered weighted averaging criterion," *European Journal of Operational Research*, vol. 286, no. 3, pp. 828–838, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221720303520

[7] R. BELLMAN, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957. [Online]. Available: http://www.jstor.org/stable/24900506

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[9] A. Dargazany, "Drl: Deep reinforcement learning for intelligent robot control – concept, literature, and future," 2021.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[11] P. Lissa, C. Deane, M. Schukat, F. Seri, M. Keane, and E. Barrett, "Deep reinforcement learning for home energy management system control," *Energy and AI*, vol. 3, p. 100043, 2021.

[12] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[13] K. Li, T. Zhang, R. Wang, Y. Wang, Y. Han, and L. Wang, "Deep reinforcement learning for combinatorial optimization: Covering salesman problems," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13 142–13 155, 2022.

[14] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[16] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," 2017.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[18] L. Chen, F. Sun, K. Li, R. Chen, Y. Yang, and J. Wang, "Deep reinforcement learning for resource allocation in massive MIMO," in *2021 29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1611–1615.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.02971

[20] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," 2015. [Online]. Available: https://arxiv.org/abs/1512.07679

[21] X. Guo, Z. Li, P. Liu, R. Yan, Y. Han, X. Hei, and G. Zhong, "A novel user selection massive MIMO scheduling algorithm via real time DDPG," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[22] H. Chen, Y. Liu, Z. Zheng, H. Wang, X. Liang, Y. Zhao, and J. Ren, "Joint user scheduling and transmit precoder selection based on DDPG for uplink multi-user MIMO systems," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. IEEE, 2021, pp. 1–5.

[23] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018. [Online]. Available: https://arxiv.org/abs/1812.05905

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018. [Online]. Available: https://arxiv.org/abs/1801.01290

[25] "5G NR physical channels and modulation (3GPP TS 38.211 version 16.2.0 release 16)," https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/16.02.00_60/ts_138211v160200p.pdf, accessed: 2023-02-13.

[26] Y. Chen, R. Yao, H. Hassanieh, and R. Mittal, "Channel-aware 5G RAN slicing with customizable schedulers."

[27] V. Lau, "Proportional fair space-time scheduling for wireless communications," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1353–1360, 2005.

[28] P. R. M., M. R., A. Kumar, and K. Kuchi, "Downlink resource allocation for 5g-nr massive mimo systems," in *2021 National Conference on Communications (NCC)*, 2021, pp. 1–6.

[29] C. Blair, "Theory of linear and integer programming (alexander schrijver)," *SIAM Review*, vol. 30, no. 2, pp. 325–326, 1988. [Online]. Available: https://doi.org/10.1137/1030065

[30] H. Liu, H. Gao, S. Yang, and T. Lv, "Low-complexity downlink user selection for massive MIMO systems," *IEEE Systems Journal*, vol. 11, no. 2, pp. 1072–1083, 2017.

[31] Y. Chen, Y. Wu, Y. T. Hou, and W. Lou, "mCore: Achieving Sub-millisecond Scheduling for 5G MU-MIMO Systems," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[32] H. Yang, "User scheduling in massive MIMO," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.

[33] J. Shi, W. Wang, J. Wang, and X. Gao, "Machine learning assisted user-scheduling method for massive MIMO system," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2018, pp. 1–6.

[34] G. Bu and J. Jiang, "Reinforcement learning-based user scheduling and resource allocation for massive MU-MIMO system," in *2019 IEEE/CIC International Conference on Communications in China (ICCC)*, 2019, pp. 641–646.

[35] V. H. L. Lopes, C. V. Nahum, R. M. Dreifuerst, P. Batista, A. Klautau, K. V. Cardoso, and R. W. Heath, "Deep reinforcement learning-based scheduling for multiband massive MIMO," *IEEE Access*, vol. 10, pp. 125 509–125 525, 2022.

[36] C.-W. Huang, I. Althamary, Y.-C. Chou, H.-Y. Chen, and C.-F. Chou, "A DRL-based automated algorithm selection framework for cross-layer QoS-aware scheduling and antenna allocation in massive MIMO systems," *IEEE Access*, vol. 11, pp. 13 243–13 256, 2023.

[37] Z. Zhao, Y. Liang, and X. Jin, "Handling large-scale action space in deep Q network," in *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2018, pp. 93–96.

[38] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[39] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," 2018. [Online]. Available: https://arxiv.org/abs/1812.11103

[40] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019. [Online]. Available: https://arxiv.org/abs/1910.07207

[41] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.02971

[42] B. Eysenbach and S. Levine, "Maximum entropy RL (provably) solves some robust RL problems," *arXiv preprint arXiv:2103.06257*, 2021.

[43] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[44] X. Hao, H. Mao, W. Wang, Y. Yang, D. Li, Y. Zheng, Z. Wang, and J. Hao, "Breaking the curse of dimensionality in multi-agent state space: A unified agent permutation framework," 2022. [Online]. Available: https://arxiv.org/abs/2203.05285

[45] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *CoRR*, vol. cs.NI/9809099, 1998. [Online]. Available: https://arxiv.org/abs/cs/9809099

[46] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[47] S. Jaeckel, L. Raschkowski, K. Börner, and L. Thiele, "QuaDRiGa: A 3-d multi-cell channel model with time evolution for enabling virtual field trials," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 6, pp. 3242–3256, 2014.

[48] R. A. Shafik, M. S. Rahman, A. R. Islam, and N. S. Ashraf, "On the error vector magnitude as a performance metric and comparative analysis," in *2006 International Conference on Emerging Technologies*, 2006, pp. 27–31.

[49] "NVIDIA DGX Station A100," https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/dgx-station/nvidia-dgx-station-a100-datasheet.pdf, accessed: 2022-07-27.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[51] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[52] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2016.

[53] E. Björnson, E. G. Larsson, and T. L. Marzetta, "Massive MIMO: ten myths and one critical question," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 114–123, 2016.

[54] T. V. de Wiele, D. Warde-Farley, A. Mnih, and V. Mnih, "Q-learning in enormous action spaces via amortized approximate maximization," 2020.

[55] C. Wu, X. Yi, Y. Zhu, W. Wang, L. You, and X. Gao, "Channel prediction in high-mobility massive mimo: From spatio-temporal autoregression to deep learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 1915–1930, 2021.

[56] Y. Han, S. Jin, C.-K. Wen, and X. Ma, "Channel estimation for extremely large-scale massive mimo systems," *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 633–637, 2020.

[57] C.-J. Chun, J.-M. Kang, and I.-M. Kim, "Deep learning-based channel estimation for massive mimo systems," *IEEE Wireless Communications Letters*, vol. 8, no. 4, pp. 1228–1231, 2019.

[58] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takác, "Deep reinforcement learning for solving the vehicle routing problem," *arXiv preprint arXiv:1802.04240*, 2018.

[59] R. Doost-Mohammady, O. Bejarano, L. Zhong, J. R. Cavallaro, E. Knightly, Z. M. Mao, W. W. Li, X. Chen, and A. Sabharwal, "RENEW: Programmable and observable massive MIMO networks," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 1654–1658.

[60] "Cuda toolkit documentation," https://docs.nvidia.com/cuda/, accessed: 2022-08-01.