Dissertation

Deep Reinforcement Learning for Cell-Free Massive MIMO Network Optimization

Author
Charmae Franchesca Mendoza, M.Sc.
Matriculation Number: 11935993

Advisor
Associate Prof. Dipl.-Ing. Dr.techn. Stefan Schwarz

Assisting Advisor
Univ.Prof. Dipl.-Ing. Dr.techn. Markus Rupp

Reviewers
Prof. Dr. André L. F. de Almeida
Prof. Dr. Luís Castedo Ribas

Institute of Telecommunications
Technische Universität Wien
Vienna, Austria

February, 2025

Abstract

Despite the significant advancements in wireless communication technologies, inter-cell interference remains a limiting factor due to the cell-centric design of traditional mobile networks. Cell-free massive multiple-input multiple-output (MIMO) is a paradigm shift in network architecture, where we replace the fixed cell boundaries with a seamless network of cooperating access points (APs) to achieve a uniformly good performance throughout the coverage area. To harness its full potential, it is necessary to address its scalability issue and the need for dynamic optimization based on the current state of the wireless environment. Compared to conventional optimization techniques and (un-)supervised machine learning, deep reinforcement learning (DRL) is capable of operating model-free, without requiring any prior knowledge, including training datasets, and in an online manner, making it an effective tool for real-time network adaptation. Motivated by these advantages, this dissertation leverages DRL for the realization of scalable, self-adapting cell-free massive MIMO. The dissertation consists of three main parts.

The first part focuses on user-centric clustering, where each user equipment (UE) is served by only a subset of APs. We demonstrate that there exists a cluster size that enables the scalable user-centric variant to stay close to the upper bound rate performance, exhibited by the canonical setup, but with significantly fewer AP-UE connections, translating to lower fronthaul requirements. While our proposed iteration-based algorithms have managed to deal with the non-convexity of the optimization problems, such methods are challenging to implement in real-time.

The second part of the dissertation capitalizes on single-agent RL (SARL) for cell-free network optimization. We design a framework that (de-)activates APs by jointly considering the position of all users. We show that by properly identifying the underutilized APs to deactivate, we reduce power consumption while obtaining a quality of service (QoS) that is close to that achieved when all the APs are always turned on. We next propose a SARL system that utilizes the spatial user densities for grouping the APs in a scalable network with multiple central processing units (CPUs). We demonstrate that by tailoring the AP group sizes to the expected user concentrations in different subareas, we improve the network sum rate. We then develop a DRL-based algorithm for user-centric clustering. By optimizing the AP selection and cluster size for each UE, we obtain almost the same performance as the canonical setup while benefiting from the reduced fronthaul capacity usage. Our findings prove that small user-centric clusters are sufficient to achieve good QoS, implying that only a subset of APs substantially contributes to UE rate performance.

The last part of the dissertation promotes distributed learning architectures by

employing multi-agent RL (MARL), in addition to SARL. Within MARL, we further explore two setups, namely centralized training, decentralized execution (CTDE) and personalized federated learning (FedPer). The former implements centralized training, while the latter localizes this and only requires periodic aggregation of local base layer parameters. We equip all three frameworks with prioritized experience replay to accelerate convergence in a dynamic RL environment, which is characterized by the combination of device (de-)activation and user mobility. Our model-free frameworks rely solely on UE rate feedback, making them agnostic to the system model, and operate in an online mode, enabling them to learn and adapt to the unpredictable activation patterns on the go. Our numerical experiments show that the proposed prioritized sampling-based FedPer system boasts near-optimal rate and power performance while incurring the least amount of communication overhead.

Acknowledgments

I would like to express my gratitude to Prof. Stefan Schwarz for giving me the opportunity to pursue this Ph.D. and for his guidance throughout these years. I am grateful for all the knowledge I gained from our fruitful discussions, and it has been a pleasure working with you. My sincere appreciation also extends to Prof. Markus Rupp for his support and valuable feedback. This Ph.D. journey would not have been possible without the financial assistance from the Christian Doppler Research Association and our industrial partners, for which I am truly thankful. I would also like to deeply acknowledge Prof. André L. F. de Almeida and Prof. Luís Castedo Ribas for their time and effort in serving as examiners of this dissertation.

日本で研究を行う機会をいただいた国立情報学研究所(NII)には、大変お世話になりました。特に金子先生には、私をグループに迎え入れてくださり、ご多忙の中、共同研究に多くの時間と労力を割いて、丁寧にご指導いただきました。先生と一緒に研究に取り組む機会をいただけたことを、心より嬉しく思っております。また、滞在中に素晴らしい時間を共にした、日本で出会った友人たち、特に花菜さん、佳丽さん、吉田さんにも、心から感謝申し上げます。

A heartfelt thank you to my colleagues at the Institute of Telecommunications for their constant encouragement, kindness, and stimulating discussions. You have all made this journey enjoyable.

I am sincerely grateful to my amazing friends for all the laughter and much-needed distractions. Thank you for cheering me on and for celebrating the small wins with me.

Finally, my deepest gratitude goes to my family. To my parents, Greg and Beth, and to my sister, Camille, thank you for your unconditional love and for always believing in me. To Andok, who stood by me through thick and thin, I am incredibly thankful for everything.

Contents

1 Introduction

The exponential growth of mobile data traffic, driven by the emergence of data-intensive services and the advancement of device capabilities [1], has led to numerous technological innovations in wireless communications. In order to cope with the traffic demand, the capacity of the cellular network can be primarily increased in three ways: (1) by utilizing more bandwidth, which is costly, (2) by improving the spectral efficiency, enabled by advanced signal processing techniques, and (3) by deploying more cells per unit area, referred to as network densification [2].

A breakthrough technology is massive multiple-input multiple-output (MIMO) [3], shown to significantly increase spectral efficiency [4]. We equip the base station with multiple co-located antennas, called an antenna array, spatially multiplexing several users on the same time-frequency resources. This concept also supports the use of distributed antenna arrays, which offers higher coverage probability by bringing the antennas closer to the users.

Of the three methods identified above, network densification has contributed the most in increasing capacity [5]. After some point, however, further densification does not help anymore, as interference is also worsened in the process [6]. To combat this, cooperation among base stations through the backhaul network has been investigated in coordinated distributed wireless systems [7], such as network MIMO [8], coordinated multipoint (CoMP) [9], and distributed antenna system [10]. In spite of this, a couple of limiting factors were highlighted in [11], including the large amount of overhead incurred from sharing the channel state information (CSI) required for coordination and the persistent inter-cell interference that is inherent in the cell-centric design of traditional mobile networks [12].

1.1 Motivation and Scope of Dissertation

Cell-Free Massive MIMO

Cell-free massive MIMO combines the benefits of massive MIMO and coordinated distributed systems [13, 14]. It consists of a large number of distributed antennas or access points (APs) jointly serving a much smaller number of user equipments

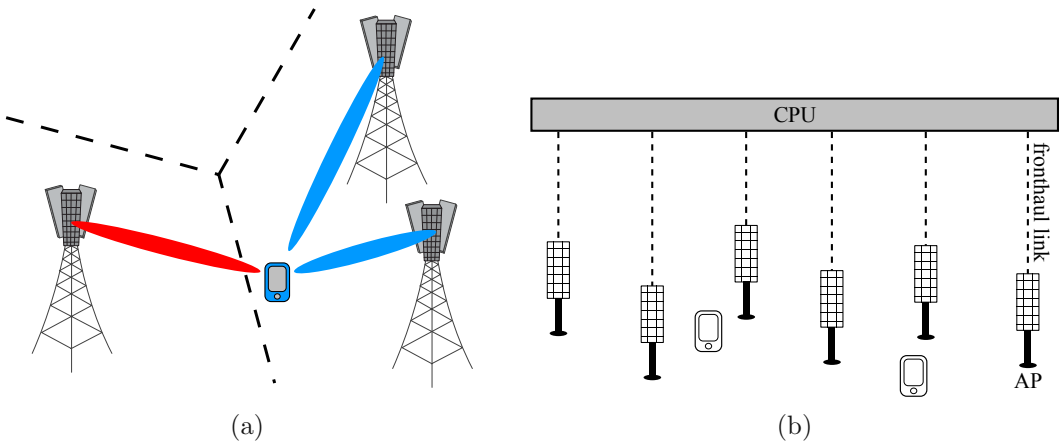


Figure 1.1: CoMP-JT (left) versus cell-free massive MIMO (right).

(UEs) using the same time-frequency resources. This assumption allows us to exploit the channel hardening and favorable propagation properties of massive MIMO [15]. The users benefit from the increased macro-diversity enabled by the coherent joint operation of distributed APs [16], which are connected via the fronthaul links to a central processing unit (CPU). The compatibility of cell-free massive MIMO with other network architectures, such as cloud radio access network (CRAN), offers several advantages. For instance, multiple CPUs may serve as the baseband units that are instantiated on demand, which implies more flexible resource allocation and lower operational costs, thereby opening new possibilities for improved network performance [17]. On the other hand, to realize cell-free massive MIMO, a number of practical deployment issues must be addressed, including challenges related to scalability, backhaul/fronthaul capacity, and latency, as discussed in [18, 19].

Cell-free massive MIMO operates in time-division duplex (TDD) mode [20]. The APs locally estimate the channels through the uplink pilots transmitted by the UEs. By virtue of channel reciprocity, the APs use the estimates for precoding in the downlink and for data detection in the uplink. These estimates, however, may be correlated when the UEs utilize the same pilot sequences, resulting in performance degradation. Pilot contamination effects can be mitigated by properly designing the pilots, such as in [21,22]. In principle, the APs do not need to exchange CSI, leading to reduced overhead compared to prior coordinated distributed systems. In the case of maximum ratio transmission (MRT)/maximum ratio combining (MRC), most processing can be done locally at the APs, and therefore, cell-free massive MIMO can be implemented in a distributed manner [14]. More sophisticated schemes that target the residual inter-user interference, such as zero-forcing [23] and minimum mean-squared error (MMSE) [24], are shown to perform better; however, this is at the expense of higher backhaul requirements.

Inter-cell interference causes the poor cell-edge performance in cellular networks. The joint transmission (JT) variant of CoMP and cell-free massive MIMO are illustrated in Fig. 1.1. In CoMP-JT, the cooperating base stations in a predefined clus-

ter jointly transmit to the UE. Inter-cell or inter-cluster interference still exists and is generated by the non-cooperating base stations. In contrast, this is suppressed in cell-free massive MIMO by eliminating the fixed cell boundaries and replacing them with a seamless network of cooperating APs. A dense AP deployment is also envisioned, such that a user is always surrounded by several APs regardless of geographical location [25]. This enables the cell-free network to provide a uniformly good service throughout the coverage area [26]. Its scalability can be improved by adopting the user-centric approach in [27], where each UE is served by only a subset of APs, and by employing multiple CPUs [11].

Deep Reinforcement Learning

Network optimization has been traditionally carried out by conventional optimization methods. Formulating an appropriate mathematical optimization problem and then solving it using such techniques help us understand the theoretical performance of the system. However, this usually comes with high computational complexity, for instance, in the case of iteration-based approaches [28]. Moreover, existing algorithms commonly require network-wide, up-to-date knowledge, such as CSI, which may be challenging to obtain in practice.

Recent advances in artificial intelligence have promoted the application of machine learning (ML) for realizing intelligent wireless communication networks [29]. Common types of ML include supervised learning, which requires a labeled dataset for training a model, and unsupervised learning, which finds hidden patterns and relationships in unlabeled data [30]. Studies demonstrate that ML-based methods are capable of dealing with the same optimization problems, but with much lower computational complexity while still achieving good performance [31]. However, both types assume that training datasets are available, which in reality, is not often the case. This then dictates the limited types of problems we can solve.

Another variant of ML is reinforcement learning (RL) that operates without needing any prior knowledge, including training datasets [32]. At its core are three components, namely agent, environment, and reward. The agent learns on its own by interacting with the environment and consequently obtaining a reward, which it aims to maximize. Specifically, it looks at the current state of the environment and acts based on this observation. Its learning process is governed by the reward feedback, which quantifies how good the selected action is, allowing the agent to improve its decision-making ability that is represented by its policy. Thus, RL is best described as learning through trial and error. The policy of an agent has been traditionally in the form of a lookup table. However, this proves to be infeasible for more complex problems, where the size of the state and action spaces is too large. In [33], a deep neural network (DNN) acts as a policy function approximator, giving rise to deep reinforcement learning (DRL). While an actual physical environment may serve as the RL environment, it is common practice to initially utilize a simulated environment for training the policy, especially in the case of safety-critical applications [34]. For instance, the role of digital twin (DT) in sixth-generation

(6G) mobile networks was highlighted in [35]. One use case is network planning, in which DT allows operators to first optimize and test different configurations in a virtual setup without disrupting the network operation. The trained model is then deployed and fine-tuned in a real-world environment, referred to as sim-to-real transfer in DRL [34].

Given that the wireless environment is inherently non-static, we highlight another advantage of DRL over existing methods, which is its ability to handle both static and dynamic scenarios [36]. In contrast, traditional algorithms solve the optimization problem for a single snapshot of the wireless environment. This implies having to go through the computationally expensive approach every time the environment changes, as well as knowing when this change will occur in advance. Likewise, (un-)supervised learning requires different training datasets for entirely new environments.

In summary, DRL is capable of operating model-free, requiring no prior information about the environment dynamics, and in an online manner, making it suitable for varying wireless scenarios. It is, therefore, easy to understand the motivation behind DRL being an attractive tool for solving resource allocation problems in telecommunications [37]. Meanwhile, as previously discussed, cell-free massive MIMO has the potential of providing a uniformly good service to everyone, and thus, it is considered one of the key enablers of 6G [38]. However, in order to fully reap its benefits, we must optimize it, for which we can utilize DRL. In this thesis, we specifically focus on the following aspects.

- Network scalability
  The canonical form of cell-free massive MIMO is not scalable, as it assumes that all UEs are served by all the APs, which are connected to a single CPU. In reality, we expect user-centric clustering to be implemented and multiple CPUs to be utilized to cover different geographical areas. Moreover, although fully centralized, network-wide optimization would theoretically give us the best performance, it comes with some downsides, including high latency and signalling overhead [39]. Frameworks geared towards distributed optimization play an important role in network scalability.

- Self-adaptability of network
  Given that the wireless environment is not static, the cell-free network must optimize itself according to the real-time changes in the environment. This not only ensures good quality of service (QoS), but also contributes to creating a more energy-efficient mobile network. A candidate tool for this purpose is DRL. However, it is known to suffer from slow convergence [40]. Thus, incorporating strategies to accelerate learning improves the effectiveness of DRL in dealing with dynamic scenarios.

Scope of Work

In this dissertation, we leverage DRL to address the challenging task of realizing a scalable cell-free massive MIMO network, capable of autonomously adapting itself to the current state of the wireless environment.

- In Chapter 3, we focus on contrasting the canonical and user-centric variants of cell-free massive MIMO to investigate the possible performance degradation with the latter setup, in exchange for improved network scalability.
- In Chapter 4, we utilize single-agent RL (SARL) to optimize the cell-free network according to instantaneous user information. This includes dynamically (de-)activating APs based on the presence of users in an area, forming AP clusters of varying sizes in a multi-CPU system based on spatial user density, and optimizing the AP selection for each user-centric cluster.
- In Chapter 5, we tackle the slow convergence limitation of DRL while solving an uplink power control problem in a dynamic wireless environment. We look into the different distributed learning architectures enabled by multi-agent RL (MARL), where we also utilize federated learning [41]. We compare the fully centralized and decentralized frameworks, providing insights on performance and communication overhead.

1.2 Literature Review

In this section, we provide a literature review on selected topics under cell-free massive MIMO, which are investigated in this dissertation.

1.2.1 User-Centric Clustering

When cell-free massive MIMO was proposed in [13, 14], it was assumed that all users are connected to all the APs. However, this renders the network unscalable, with each AP having to handle the data of all UEs, posing a problem for fronthaul capacity-limited systems. In [27], it was suggested that each UE is served by only a subset of APs, forming what is referred to as a user-centric cluster. Determining which APs make up each cluster is essentially an AP-UE association problem.

Different metrics can be utilized for AP selection. For instance, the APs with the best channel conditions were chosen in [11, 27], while in [42], pilot assignment additionally dictated the AP-UE association decisions. The authors in [43] employed the Hungarian algorithm, which is a combinatorial optimization technique, to assign each UE to one of the predefined virtual clusters, such that the sum rate was maximized. Their method was shown to outperform the case where each user is served by a fixed number of APs with the strongest channel. In [44], energy efficiency was maximized while satisfying per-UE spectral efficiency and per-AP transmit power

constraints. Two AP selection schemes were presented, where a fixed percentage of the APs were chosen based on either the received power or large-scale fading quantities. In [45], k-means was utilized to first cluster the users, such that pilot contamination was minimized, and then each AP was allocated to the user group with the closest cluster centroid. Similarly, the authors in [46] proposed to initially group the users, and then greedily assign each AP based on a fixed ratio between the number of APs and UEs per cluster.

A common denominator for the above-mentioned works is that they involve a fixed number of APs that serve each UE. This then becomes a design parameter, whose predefined value must be properly configured to ensure a sufficiently good performance. This also implies that the user-centric clusters are of equal size. However, this is not a realistic assumption, since the UEs would have different propagation conditions with respect to the APs. Thus, they require a varying number of serving APs in their clusters. In principle, the AP-UE connections can also be decided by defining them as optimization variables. Alternatively, this is treated as a power allocation problem, where zero power signifies that the AP does not serve the user. In the case of non-convexity, the resulting problem is complex and difficult to solve [47, 48]. In Chapters 3, we look at the performance comparison between canonical and user-centric cell-free massive MIMO by investigating their guaranteed rate and total fronthaul requirement [49]. In Chapter 4, we utilize SARL to deal with the non-convexity and form user-centric clusters that vary in size [50].

1.2.2 Dynamic AP Activation

The base station was identified to be taking the largest chunk of the total power consumption of a cellular network in [51]. Motivated by this, different base station ON/OFF strategies were investigated in the literature [52, 53]. Since the traffic demand varies throughout the day, putting some of the base stations into sleep mode during off-peak hours improves energy efficiency. The same idea applies to cell-free massive MIMO, where we expect a dense AP deployment.

In [54], the number of active APs and the transmit powers were jointly optimized to minimize the power consumption of the cell-free network while satisfying user demands. The globally optimal solution was achieved by solving a mixed-integer second-cone program. In addition, a suboptimal low-complexity algorithm that exploits group sparsity was presented. Similarly, in [55], $\ell_{2,1}$-norm-based block sparsity and successive approximation were utilized to jointly optimize power allocation and AP selection, such that energy consumption is minimized while satisfying QoS constraints. Several suboptimal heuristic ON/OFF schemes were proposed in [56]. The goal was to find an algorithm that gives a good balance between performance and computational complexity. In [57], the same group built on their earlier work to consider a non-uniform distribution of users in mmWave cell-free massive MIMO. It was later demonstrated in [58] that heuristic approaches based on effective channel gain, rather than on path loss as in [56, 57], performed better.

Most prior works relied on either heuristics or high-complexity optimization algo-

rithms. In Chapter 4, we take a different approach by utilizing SARL to handle this problem. We present a DRL-based framework that (de-)activates APs based on the presence of users in the area to maximize the minimum user signal-to-interference-plus-noise ratio (SINR) [59]. We propose another framework that makes this decision such that either the QoS demand is satisfied or the network sum rate is maximized in [50].

1.2.3 Multi-CPU System and AP Clustering

Utilizing a single CPU to cover a large geographical area is not realistic and does not scale well, since this CPU would need to handle all the data of all APs. Instead, multiple CPUs are deployed, essentially assigning them to subareas. The authors in [11] demonstrated that a system of multiple independent CPUs, coupled with their proposed distributed power control scheme, behaves close to the fully centralized, single-CPU setup. Meanwhile, a leakage-based optimization problem was solved separately at each CPU to enable a distributed resource allocation in [60].

Several works have focused on inter-CPU communication. In [61], different degrees of CPU cooperation were identified according to the type of information exchanged among the CPUs. It was shown that only a modest performance degradation results from sharing long-term CSI. A cooperation protocol was proposed in [62], where the CPUs share channel estimates and implement either centralized or distributed precoding. The same authors advocated partial information exchange among the CPUs to achieve a balance between performance and overhead in [63]. Specifically, only the large-scale fading coefficients are shared, and based on those, each CPU estimates the AP clustering and power control policies of the other CPUs.

In a multi-CPU cell-free network, each CPU is associated to a group of APs and handles the data of the users served by those APs. It follows that determining how to properly form the AP clusters is a topic that must be investigated. In [64], an optimization problem was first formulated to obtain the optimal number of clusters. A Gaussian mixture model was then employed to group the APs, which was shown to outperform k-means. The number of subnetworks was maximized while satisfying a per-user rate constraint by rewriting the optimization problem to enable bipartite graph partitioning in [65, 66]. In Chapter 4, we leverage SARL and consider the realistic case of heterogeneous user traffic to determine the appropriate cluster sizes that lead to good performance [67].

1.2.4 Power Control

Power control plays an important role both in mitigating inter-user interference and enabling an energy-efficient network. Conventional optimization algorithms were utilized in several works, such as [68–70], to assign UE powers in order to optimize different performance metrics. Meanwhile, ML-based methods are capable of solving power allocation problems with lower computational complexity. They were shown to achieve near-optimal performance in prior studies, including [31,71,72]

that focused on the downlink case and [73–75] that considered the uplink. However, as previously motivated, the main advantage of DRL is that it does not rely on datasets for offline training, making it suitable for non-static environments.

In [76], DRL was utilized to find the beamforming matrix that maximizes the long-term energy efficiency. The AP clusters and beamforming matrix were jointly optimized using DRL to maximize either the sum rate or the minimum user rate in [77]. A framework was proposed in [78] to solve power control problems that are based on the max-min, max-sum, and max-product of the user rates, where the DRL scheme consistently outperformed conventional optimization and deep learning-based systems. The authors in [79] employed DRL for power allocation while balancing between user fairness and sum-rate maximization. These works, however, only considered a fully static scenario. In [80], both static and mobile users were considered when maximizing the sum rate, subject to user rate constraints, while investigating the impact of different state vector definitions.

When the wireless environment changes, the DRL agent must react to this by updating its policy accordingly. Otherwise, its actions will be based on an outdated policy, degrading performance. However, one issue of DRL is slow convergence [40]. Prioritized experience replay was proposed in [81] to speed up learning. In [82], this mechanism was employed for power allocation, where it was also suggested that an optimal value for the prioritization factor exists. However, the study was limited to a fully static, point-to-point scenario. In Chapter 5, we utilize DRL for uplink power control and rely on prioritized experience replay to accelerate convergence. Compared to existing works, the dynamic environment is not only characterized by user mobility, but also by device (de-)activation [83, 84].

We additionally consider MARL for power control in Chapter 5. The authors in [85] utilized MARL for downlink power allocation in cell-free massive MIMO, aiming to maximize the sum spectral efficiency. In [86], it was employed for jointly optimizing the pilot and data powers to maximize either the sum rate or the minimum user rate. In Chapter 5, we investigate the performance of SARL and MARL systems. We also explore combining federated learning [41] with DRL and prioritized experience replay for power allocation [84].

1.3 Structure and Contributions

In this section, we outline the structure and main contributions of this dissertation. We also refer to the publications in which the contributions have been first presented.

Chapter 2 – Preliminaries

The second chapter provides a brief introduction to key concepts. In Sec. 2.1, we formally define cell-free massive MIMO, including its underlying technologies inherited from cellular massive MIMO, as well as its scalability issue. In Sec. 2.2, we give an overview of RL and discuss the DRL algorithms used in this work.

Chapter 3 – User-Centric Clustering

In this chapter, we highlight the importance of user-centric clustering in making the cell-free network scalable. We investigate the impact of cluster formation on the total fronthaul requirement and guaranteed rate of the network. We formulate the corresponding non-convex optimization problems and propose mathematical algorithms to handle them. We show that there exists a cluster size such that the user-centric variant achieves almost the same QoS level as the canonical setup while benefiting from a significant reduction in fronthaul capacity usage.

The algorithms developed in this chapter have been published in [49]:

(i) C. F. Mendoza, S. Schwarz and M. Rupp, "Cluster Formation in Scalable Cell-free Massive MIMO Networks," *16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Thessaloniki, Greece, 2020, pp. 62-67.

Chapter 4 – SARL-based Network Optimization

In Chapter 4, we propose three SARL frameworks that cater to different optimization objectives in the context of cell-free massive MIMO. We design their reward functions to be easily configurable to accommodate varying operating points and performance targets.

We present a DRL system that dynamically activates only the best APs based on the instantaneous user position in Sec. 4.2. We demonstrate that by switching off underutilized APs, we reduce power consumption while still obtaining high user SINR values. In Sec. 4.3, we envision a multi-CPU setup, and for this purpose, develop a DRL scheme that exploits our knowledge of the spatial user density for creating the AP groups associated to the CPUs. We show that by forming bigger groups in subareas with expected higher user concentration, we maximize the network sum rate. Lastly, we develop a DRL framework for AP-UE association and AP (de-)activation, such that either a given user demand is satisfied or the sum spectral efficiency is maximized, in Sec. 4.4. We prove that by optimizing the AP selection and cluster size for each user, we stay close to the upper bound rate performance even with small clusters, which then translates to fronthaul savings.

The contributions of this chapter have been published in [50, 59, 67]:

(i) C. F. Mendoza, S. Schwarz and M. Rupp, "Deep Reinforcement Learning for Dynamic Access Point Activation in Cell-Free MIMO Networks," *25th International ITG Workshop on Smart Antennas (WSA)*, French Riviera, France, 2021, pp. 1-6.

(ii) C. F. Mendoza, S. Schwarz and M. Rupp, "Deep Reinforcement Learning for Spatial User Density-based AP Clustering," *IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, Oulu, Finland, 2022, pp. 1-5.

(iii) C. F. Mendoza, S. Schwarz and M. Rupp, "User-Centric Clustering in Cell-Free MIMO Networks using Deep Reinforcement Learning," *IEEE International Conference on Communications (ICC)*, Rome, Italy, 2023, pp. 1036-1041.

Chapter 5 – Accelerated SARL and MARL for Power Control

In the final chapter, we tackle an uplink power control problem with the goal of maximizing the guaranteed QoS of a dynamic cell-free network. We consider continuous and discrete power values in Sec. 5.4 and 5.5, respectively. One novelty of our work is the ability of our DRL frameworks to effectively handle UE (de-)activation without knowing the device ON/OFF patterns in advance, in addition to user mobility. We incorporate prioritized experience replay and demonstrate that it does not only speed up convergence, but also enables higher user rates and lower power consumption. We apply this strategy to our proposed fully centralized SARL and distributed MARL systems. Within MARL, we further investigate two setups, with one relying on centralized training, and the other keeping the training local at the agents. The latter framework takes advantage of personalized federated learning [87], where each agent only forwards the base layer of its trained local model for periodic aggregation. We also highlight that we specifically designed our systems such that the network entities share user rate information only, which is motivated by privacy preservation and overhead reduction for improved scalability. We show that applying prioritization on the federated learning-based distributed framework enables a near-optimal performance with the least amount of communication overhead.

The methods investigated in this chapter have been presented in [83, 84]:

(i) C. F. Mendoza, M. Kaneko, M. Rupp and S. Schwarz, "Accelerated Deep Reinforcement Learning for Uplink Power Control in a Dynamic Cell-Free Massive MIMO Network," *IEEE Wireless Communications Letters*, vol. 13, no. 6, pp. 1710-1714, Jun. 2024.

(ii) C. F. Mendoza, M. Kaneko, M. Rupp and S. Schwarz, "Enhancing the Uplink of Cell-Free Massive MIMO through Prioritized Sampling and Personalized Federated Deep Reinforcement Learning," accepted in *IEEE Transactions on Cognitive Communications and Networking*.

2
Preliminaries

In this chapter, we introduce the two major concepts central to this dissertation, namely cell-free massive MIMO and DRL. We first provide a formal definition of canonical cell-free massive MIMO, briefly touching upon its underlying technologies. We also shed some light on its scalability issue and discuss how this can be tackled. We next present the basics of RL, which is our main tool for optimizing the cell-free network. We later describe the DRL algorithms employed in this work.

2.1 Cell-Free Massive MIMO

2.1.1 Definition

The canonical cell-free massive MIMO network [13, 14] consists of M geographically distributed APs, having N antennas each, and K single-antenna UEs[1]. We operate in the regime of $MN \gg K$, such that a much larger number of APs or antennas spatially multiplex a relatively smaller number of users on the same time-frequency resources. All APs are connected to a CPU via the fronthaul links, as illustrated in Fig. 1.1b. The CPU is mainly responsible for AP coordination, synchronization, and data processing.

When considering a traditional cellular network architecture, users at the cell centers experience good QoS, while those at the cell edge suffer from poor performance, causing large rate variations and unreliable service [21]. In cell-free massive MIMO, we move away from such cell-centric design and transition to a more user-centric one, comprising of cooperating APs without fixed cell boundaries. The goal is to achieve a uniformly good performance throughout the coverage area by always surrounding each UE with multiple APs regardless of its location [26].

[1] In principle, a UE may have multiple antennas. We, however, make a simplifying assumption that single-antenna UEs are present in the cell-free network.

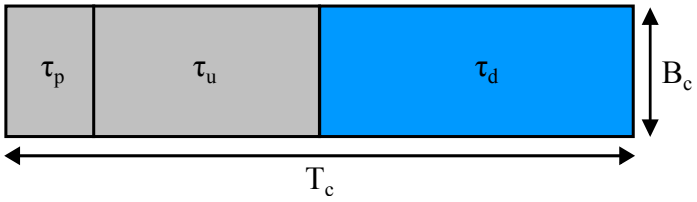


Figure 2.1: TDD frame structure.

2.1.2 Underlying Technologies

We now look into the key massive MIMO technologies that are utilized by the cell-free network [15, 20].

Time-Division Duplex

Obtaining CSI is a prerequisite for the APs to perform certain tasks, such as precoding. The APs first estimate the uplink channels through the pilot signals transmitted by the UEs. In the case of TDD operation, the uplink and downlink are separated in time. By virtue of channel reciprocity, the uplink and downlink channel realizations are considered to be the same. Thus, the downlink channels can also be estimated even without utilizing additional feedback. In contrast, the uplink and downlink are separated in frequency when considering the frequency-division duplex (FDD) mode. The channels are now different, and the reciprocity property can no longer be exploited. This implies that pilots must be sent in both directions, and the UEs must feed back their downlink channel estimates to the APs, resulting in significant overhead [4]. For this reason, cell-free massive MIMO employs TDD [20].

We consider a block-fading channel model. The TDD frame structure for each channel coherence block is shown in Fig. 2.1. A coherence block has a time interval equal to the coherence time T_c (i.e., channel is assumed to be constant) and a frequency interval equal to the coherence bandwidth B_c (i.e., channel is assumed to be frequency flat). It contains $\tau_c = T_c B_c$ complex-valued samples, of which τ_p are for the uplink pilots, τ_u are for uplink data transmission, and τ_d are for downlink data transmission. Note that the coherence block essentially limits the number of mutually orthogonal pilots (equal to τ_p) that can be assigned to the UEs. When the pilot sequences are shared by the users, pilot contamination occurs, where channel estimates become correlated that then degrades performance. Different pilot assignment algorithms have been proposed to alleviate pilot contamination effects, including those in [21, 22].

Channel Hardening

In massive MIMO, we equip the base station with a large number of antennas. Its asymptotic behavior gives rise to two phenomena. One of them is channel hardening [4]. Consider the channel $\mathbf{h}_{k,m} \in \mathbb{C}^{N \times 1}$ between UE k with one antenna and AP m

with N antennas. The following condition holds as $N \to \infty$

$$\frac{\|\mathbf{h}_{k,m}\|^2}{\mathbb{E}\{\|\mathbf{h}_{k,m}\|^2\}} \to 1. \tag{2.1}$$

That is, the fading channel behaves as if it is deterministic. This simplifies resource allocation, as we no longer need to adapt it according to the small-scale fading variations.

Favorable Propagation

Another asymptotic property is favorable propagation [4]. Consider channels $\mathbf{h}_{k,m}$, $\mathbf{h}_{j,m} \in \mathbb{C}^{N\times 1}$ of UEs k and j, respectively. The following is true as $N \to \infty$

$$\frac{|\mathbf{h}_{k,m}^H \mathbf{h}_{j,m}|^2}{\|\mathbf{h}_{k,m}\|^2 \|\mathbf{h}_{j,m}\|^2} \to 0. \tag{2.2}$$

This implies that the channels of the two users are nearly orthogonal. The inter-user interference is then negligible, even with low-complexity processing techniques, such as MRT/MRC. When N is finite, utilizing precoding schemes that suppress inter-user interference [23, 24] and properly grouping the users to be spatially multiplexed [88] improve performance.

It has been shown that several factors, including spatial channel correlation, affect the degree of channel hardening and favorable propagation in cell-free massive MIMO. We refer to [15, 21] for a more detailed discussion.

2.1.3 Network Scalability

The canonical cell-free network described in Sec. 2.1.1 assumes that all users are served by all the APs and commonly considers network-wide processing. This does not scale well in terms of computational complexity and required fronthaul capacity, making it hard to realize in practice. Motivated by this, the user-centric concept was proposed in [27], in which each UE is connected to a subset of APs only. The idea is to admit the best APs into the user-centric cluster, with those APs having the greatest impact on user rate performance. The individual clusters of the different UEs are color coded in Fig. 2.2. It follows that each AP now serves a limited number of users, even as the total number of UEs grows. With reference to the analysis in [42], this implies finite complexity and fronthaul requirement per AP, since an AP has to estimate the channels, compute the precoding/combining vectors, and send/receive data corresponding to a finite (small) number of users only. Furthermore, considering the large geographical area to be serviced, as well as the dense AP deployment [25], we expect to utilize multiple CPUs to cover different subareas [11]. In line with this, varying degrees of interconnectivity among the CPUs, as well as their computational complexity, were investigated in [61–63]. This is in contrast to the canonical setup that employs a single CPU only. The envisioned scalable cell-free network is depicted

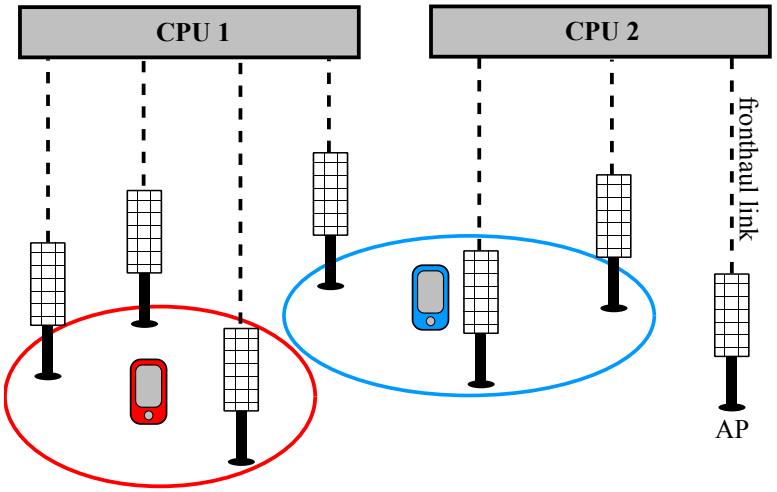


Figure 2.2: Scalable cell-free massive MIMO with user-centric clustering in a multi-CPU environment.

in Fig. 2.2. Here, each UE is served by the APs in its own user-centric cluster, and the APs associated to each CPU are grouped together.

2.2 Reinforcement Learning

Machine learning can be categorized into three types, namely supervised, unsupervised, and RL [30]. While the first two train a model using either labeled or unlabeled data, RL takes a different approach, in that there is no need for a training dataset. A decision-maker, known as the agent, learns autonomously through trial and error by interacting with its environment and receiving a reward feedback. In this section, we provide an overview of key RL concepts. We refer to [32] for a more detailed description.

2.2.1 Agent-Environment Interaction

The agent observes the state $s^{(t)} \in \mathcal{S}$ of the environment at time t, and based on that, decides which action $a^{(t)} \in \mathcal{A}$ to perform. At the next time step $t+1$, the state of the environment changes to $s^{(t+1)}$, which is triggered by the recently applied action, as well as by uncontrolled factors (e.g., other agents acting on the same environment). The agent receives a corresponding numerical reward $r^{(t+1)} \in \mathbb{R}$ that signals how good or bad the action is given the state [32]. This framework is depicted in Fig. 2.3. The experience at time t is represented by the tuple $(s^{(t)}, a^{(t)}, r^{(t+1)}, s^{(t+1)})$. An RL problem is mathematically formalized as a Markov decision process, which satisfies the Markov property that a state transition depends only on the most recent state and action [89]. Moreover, RL can either be model-based or model-free. In the former case, all transition probabilities (i.e., probability of moving from one state

Figure 2.3: Agent-environment interaction in reinforcement learning.

to another) are known beforehand. However, this is unlikely to be true for most of the real-world problems. Therefore, we focus on model-free RL, in which no prior information about the environment dynamics is required [32].

The agent aims to maximize the cumulative reward received over time, or the (discounted) return,

$$\begin{aligned} G^{(t)} &= r^{(t+1)} + \gamma r^{(t+2)} + \gamma^2 r^{(t+3)} + \dots \\ &= \sum_{n=0}^{\infty} \gamma^n r^{(t+n+1)}, \end{aligned} \tag{2.3}$$

where $\gamma \in [0, 1]$ is the discount factor that specifies the present value of future rewards. With smaller γ values, the agent tends to focus on maximizing immediate rewards, while with larger ones, it leans towards strongly considering future rewards. An important RL concept is the exploration-exploitation trade-off. Since the agent relies on collected experiences, it knows little to no information about the environment at the initial phase of the training process. Thus, it must first explore by taking suboptimal actions that result to lower rewards. As the training progresses, it gradually starts exploiting its improved knowledge by selecting better actions that yield higher rewards [32].

2.2.2 Temporal-Difference Learning

The policy π of an agent is a function that maps a state to an action (i.e., it decides the action to take given a state). In order to maximize the expected discounted return, the agent must learn the optimal policy π_*. The state-value function for policy π is the expected discounted return when starting from state s and then acting according to policy π from then on

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[G^{(t)} | s^{(t)} = s] \\ &= \mathbb{E}_\pi \left[\sum_{n=0}^{\infty} \gamma^n r^{(t+n+1)} \middle| s^{(t)} = s \right]. \end{aligned} \tag{2.4}$$

The action-value function for policy π is the expected discounted return when starting from state s, taking action a, and then acting according to policy π from then on

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_\pi[G^{(t)}|s^{(t)}=s, a^{(t)}=a] \\ &= \mathbb{E}_\pi\left[\sum_{n=0}^{\infty}\gamma^n r^{(t+n+1)}\middle| s^{(t)}=s, a^{(t)}=a\right]. \end{aligned} \tag{2.5}$$

This is also known as the Q-function that provides the Q-value of a state-action pair. Calculating the value function for every state or state-action pair is, however, impractical. As derived in [32], the Bellman equation simplifies this computation by estimating the values as

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[G^{(t)}|s^{(t)}=s] \\ &= \mathbb{E}_\pi[r^{(t+1)} + \gamma(r^{(t+2)} + \gamma r^{(t+3)} + \ldots)|s^{(t)}=s] \\ &= \mathbb{E}_\pi[r^{(t+1)} + \gamma G^{(t+1)}|s^{(t)}=s] \\ &= \mathbb{E}_\pi[r^{(t+1)} + \gamma V_\pi(s^{(t+1)})|s^{(t)}=s], \end{aligned} \tag{2.6}$$

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_\pi[G^{(t)}|s^{(t)}=s, a^{(t)}=a] \\ &= \mathbb{E}_\pi[r^{(t+1)} + \gamma(r^{(t+2)} + \gamma r^{(t+3)} + \ldots)|s^{(t)}=s, a^{(t)}=a] \\ &= \mathbb{E}_\pi[r^{(t+1)} + \gamma G^{(t+1)}|s^{(t)}=s, a^{(t)}=a] \\ &= \mathbb{E}_\pi[r^{(t+1)} + \gamma Q_\pi(s^{(t+1)}, a^{(t+1)})|s^{(t)}=s, a^{(t)}=a]. \end{aligned} \tag{2.7}$$

The above results are utilized for temporal-difference (TD) learning, where the value function is updated at each time step as

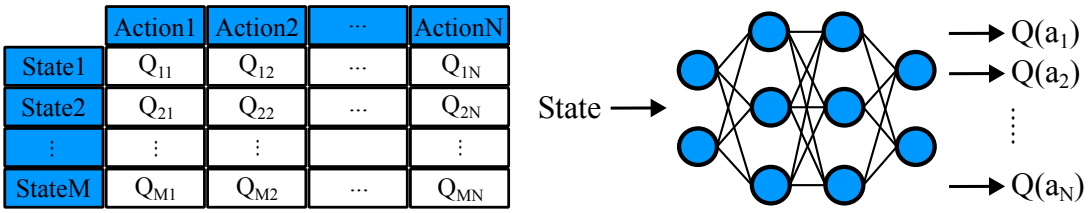
$$Q(s^{(t)},a^{(t)}) \leftarrow Q(s^{(t)},a^{(t)}) + \alpha_{\text{step}}\underbrace{(\underbrace{r^{(t+1)} + \gamma Q(s^{(t+1)}, a_{\max})}_{\text{TD target } (y)} - Q(s^{(t)},a^{(t)}))}_{\text{TD error } (\delta_{\text{TD}})}, \tag{2.8}$$

$$a_{\max} = \arg\max_a Q(s^{(t+1)}, a). \tag{2.9}$$

Here, we define the learning rate or step size $\alpha_{\text{step}} \in (0,1]$. The TD error δ_{TD}, which we aim to minimize, is the difference between the TD target and the current Q-value estimate. The TD target, commonly denoted by y in the RL literature, is obtained by adding the immediate reward and the discounted highest Q-value for the next state [32].

2.2.3 Deep Reinforcement Learning Algorithms

Classical RL algorithms, such as Q-learning [90], store the Q-values of the state-action pairs in a lookup table or Q-table, as illustrated in Fig. 2.4a. This, however,



(a) (b)

Figure 2.4: Q-table (left) versus DNN as a Q-function approximator (right).

is infeasible for high-dimensional problems. Motivated by this, a DNN was used to approximate the Q-function in [33], which we refer to as DRL. In Fig. 2.4b, the DNN takes in the state of the environment and then outputs the Q-values of all the N possible actions.

Deep reinforcement learning algorithms can be categorized in different ways. In order to find the optimal policy π_*, an algorithm can be: (1) policy-based, where we learn the policy directly, (2) value-based, where we learn the optimal value function that results to the optimal policy, or (3) a combination of both. Another classification is whether it is on-policy or off-policy. The former utilizes the same policy for action selection and updating the policy or value function, while the latter relies on separate policies for that purpose [32].

There are several state-of-the-art DRL algorithms in the existing literature [91]. Here, we describe the algorithms used in this dissertation.

Double Deep Q-Network

Double deep Q-network (DDQN) is an off-policy, value-based method that handles continuous/discrete state and discrete action spaces. It was introduced in [92] as an improved version of the earlier deep Q-network (DQN) [33]. Specifically, DQN suffers from the moving target problem, where the same DNN is used to compute both the TD target and current Q-estimate in the TD error in (2.8). This creates an effect that the target is moving, causing oscillations during training [93]. Another issue is the overestimation of Q-values when we again use the same DNN for choosing the best action in (2.9) to determine the TD target [92]. We solve these problems in DDQN by employing two DNNs: primary Q and target Q' networks, parameterized by θ_{prime} and θ_{targ}, respectively. The TD-target calculation is decoupled from action selection by using separate DNNs. The TD target in the update is then

$$y^{(t)}_{\text{targ}} = r^{(t+1)} + \gamma Q'(s^{(t+1)}, a_{\max}; \theta^{(t)}_{\text{targ}}), \tag{2.10}$$

where the action is selected as

$$a_{\max} = \arg\max_a Q(s^{(t+1)}, a; \theta^{(t)}_{\text{prime}}). \tag{2.11}$$

Double DQN relies on the concept of experience replay, such that the agent saves its new experience $(s^{(t)}, a^{(t)}, r^{(t+1)}, s^{(t+1)})$ in its memory or replay buffer $\mathcal{B}$ at each time step [94]. A mini-batch of X experiences is randomly sampled from the buffer when updating the primary network. This mechanism ensures that uncorrelated experiences are used for the update. It also helps to mitigate catastrophic forgetting, where the agent only remembers and learns from its most recent experiences [95].

The primary DNN parameter θ_{prime} is updated by minimizing the loss or the mean-squared error (MSE)

$$L_{\text{DDQN}} = \frac{1}{X}\sum_{i=0}^{X-1}(y_{\text{targ},i} - Q(s_i, a_i; \theta^{(t)}_{\text{prime}}))^2 \tag{2.12}$$

over the sampled mini-batch by performing gradient descent [96]. On the other hand, the target DNN parameter θ_{targ} is updated by copying θ_{prime} periodically or using soft updates, such as Polyak averaging

$$\theta^{(t+1)}_{\text{targ}} = \tau_{\text{pol}}\theta^{(t)}_{\text{prime}} + (1 - \tau_{\text{pol}})\theta^{(t)}_{\text{targ}}, \tag{2.13}$$

with $\tau_{\text{pol}} \in (0, 1)$ being the rate of averaging [97].

As discussed in Sec. 2.2.1, the agent must find a balance between exploration and exploitation. We use the decaying ϵ-greedy algorithm, with $\epsilon \in [0, 1]$, for the DDQN action selection [32]. This is given by

$$a^{(t)} = \begin{cases} \arg\max_a Q(s^{(t)}, a; \theta^{(t)}_{\text{prime}}), & \text{with probability } 1 - \epsilon \\ \text{random action}, & \text{with probability } \epsilon. \end{cases} \tag{2.14}$$

We start with a high ϵ value, which implies that the agent would initially select random actions, corresponding to the exploration phase. We decrease ϵ over time at a rate $\epsilon_{\text{decay}} \in [0, 1]$ until it reaches ϵ_{end} as

$$\epsilon^{(t+1)} = \max\{\epsilon^{(t)}(1 - \epsilon_{\text{decay}}), \epsilon_{\text{end}}\}. \tag{2.15}$$

The agent then exploits its knowledge of improved Q-estimates by picking the action that gives the highest Q-value.

Deep Deterministic Policy Gradient

Deep deterministic policy gradient (DDPG) is an off-policy algorithm that handles continuous/discrete state and continuous action spaces [98]. It combines the benefits of policy-based and value-based methods by learning both an actor (policy) and a critic (Q-value function) [99]. It utilizes four DNNs: primary actor μ (ϕ_{prime}), target actor μ' (ϕ_{targ}), primary critic Q (θ_{prime}), and target critic Q' (θ_{targ}) networks.

The primary actor takes in the state observation, and based on that, outputs a single action that maximizes the long-term reward. The primary critic takes in both

the state and selected action, and outputs the corresponding Q-value that is then passed to the actor. The primary actor utilizes this information to gauge how good or bad the chosen action is given the current state and to update ϕ_{prime} accordingly for improving the policy. This update is carried out by taking the following derivative of the Q-function with respect to the policy parameter (while keeping the Q-parameter constant)

$$\nabla_{\phi_{\text{prime}}} Q \approx \frac{1}{X} \sum_{i=0}^{X-1} \nabla_a Q(s, a; \theta_{\text{prime}})|_{s=s_i, a=\mu(s_i)} \nabla_{\phi_{\text{prime}}} \mu(s; \phi_{\text{prime}})|_{s=s_i} \tag{2.16}$$

and then applying gradient ascent over the sampled mini-batch of X experiences. That is, we aim to learn policy μ that outputs the action maximizing the Q-function [98].

The target networks, which improve learning stability [93], are used to calculate the TD target in DDPG

$$y_{\text{targ}}^{(t)} = r^{(t+1)} + \gamma Q'(s^{(t+1)}, \mu'(s^{(t+1)}; \phi_{\text{targ}}^{(t)}); \theta_{\text{targ}}^{(t)}). \tag{2.17}$$

The primary critic θ_{prime} is updated by minimizing the MSE

$$L_{\text{DDPG}} = \frac{1}{X} \sum_{i=0}^{X-1} (y_{\text{targ},i} - Q(s_i, a_i; \theta_{\text{prime}}^{(t)}))^2 \tag{2.18}$$

over the sampled mini-batch using gradient descent. The target networks are updated using Polyak averaging

$$\phi_{\text{targ}}^{(t+1)} = \tau_{\text{pol}} \phi_{\text{prime}}^{(t)} + (1 - \tau_{\text{pol}}) \phi_{\text{targ}}^{(t)}, \tag{2.19}$$

$$\theta_{\text{targ}}^{(t+1)} = \tau_{\text{pol}} \theta_{\text{prime}}^{(t)} + (1 - \tau_{\text{pol}}) \theta_{\text{targ}}^{(t)}. \tag{2.20}$$

Exploration in DDPG is performed by adding a small amount of noise on top of the selected action. Instead of using Ornstein-Uhlenbeck noise [100] as in the seminal paper [98], we utilize uncorrelated, zero-mean Gaussian noise, which produces good results in our tested environments [101].

Proximal Policy Optimization

Proximal policy optimization (PPO) is an on-policy method that handles continuous/discrete state and action spaces. It aims to improve the learning stability by ensuring that the policy is updated conservatively (i.e., the new policy does not change too much from the old one) [102]. For this, it learns both an actor π (policy) for action selection and a critic V (value function) for updating the policy, which are represented as DNNs with parameters ϕ and θ, respectively.

In [102], the PPO clipped surrogate objective function is given by

$$J^{(t)} = \min\left(r_{\text{PPO}}^{(t)}\hat{A}^{(t)}, \text{clip}\left(r_{\text{PPO}}^{(t)}, 1-\xi_{\text{PPO}}, 1+\xi_{\text{PPO}}\right)\hat{A}^{(t)}\right), \tag{2.21}$$

where the ratio between the current and old policies is expressed as

$$r_{\text{PPO}}^{(t)} = \frac{\pi_\phi(a^{(t)}|s^{(t)})}{\pi_{\phi,\text{old}}(a^{(t)}|s^{(t)})}, \tag{2.22}$$

and the generalized advantage estimator (GAE) with parameter λ_{GAE} [103] is

$$\hat{A}^{(t)} = \sum_{n=0}^{T-1}(\gamma\lambda_{\text{GAE}})^n\delta^{(t+n)}, \tag{2.23}$$

$$\delta^{(t)} = r^{(t)} + \gamma V_\theta(s^{(t+1)}) - V_\theta(s^{(t)}). \tag{2.24}$$

Thus, PPO ensures that the policy update is not too large by clipping the ratio in (2.22) to be within a range specified by ξ_{PPO}. We then take the minimum or lower bound of the resulting unclipped and clipped terms.

The actor ϕ is updated by maximizing (2.21) over the sampled mini-batch of X experiences using gradient ascent. On the other hand, the critic θ is updated by minimizing the MSE

$$L_{\text{PPO}} = \frac{1}{X}\sum_{i=0}^{X-1}(V_\theta(s_i) - G_i)^2 \tag{2.25}$$

over the sampled experiences using gradient descent. Similar to DDPG, we utilize Gaussian noise for exploration.

3

User-Centric Clustering

The previous chapters motivated us to adopt the user-centric approach to cell-free massive MIMO, as it is claimed to be superior to its canonical counterpart in terms of scalability [42]. We must ensure that even with this cell-free variant, the users are still able to experience good service. In this chapter, we aim to gain some insights on the impact of user-centric cluster formation on practical aspects of the cell-free network, namely fronthaul requirement and guaranteed rate. In particular, we are interested in determining the resulting performance gap between the two cell-free setups, in exchange for improved scalability. The methods developed in this chapter have been published in [49].

3.1 Downlink System Model

We consider a downlink cell-free massive MIMO network with M APs, having N antennas each, and K single-antenna UEs. The total number of antennas MN is much greater than the number of users K being served. We define the set of all APs as $\mathcal{M} = \{1, \ldots, M\}$ and the set of all UEs as $\mathcal{K} = \{1, \ldots, K\}$.

The independent and identically distributed (i.i.d.) Rayleigh fading channel between AP m and UE k is given by

$$\mathbf{h}_{k,m} = \sqrt{g_{k,m}}\widetilde{\mathbf{h}}_{k,m} \in \mathbb{C}^{N\times 1}, \tag{3.1}$$

which is a common model in the Sub-6 GHz band assumed in this work. The small-scale fading is denoted by $\widetilde{\mathbf{h}}_{k,m} \sim \mathcal{CN}(\mathbf{0}, \mathbf{I}_N)$. The macroscopic fading coefficient $g_{k,m}$ follows a distance-dependent path loss model with shadowing fading as

$$g_{k,m} = \left(\frac{\lambda_c}{4\pi}\right)^2 \left(\frac{1}{d_{k,m}}\right)^{n_c} s_{k,m}, \tag{3.2}$$

where λ_c is the wavelength of carrier frequency f_c, $d_{k,m}$ is the distance between UE k and AP m, n_c is the path loss exponent, and $s_{k,m} \sim \mathcal{LN}(0, \sigma_c^2)$ is the random lognormally distributed shadow fading. We assume that each AP knows the channels

to the UEs perfectly.

During downlink data transmission, the signal sent by AP m is obtained by summing up the precoded signals for the individual UEs as

$$\begin{aligned}\mathbf{x}_m &= \sum_{k\in\mathcal{K}} \mathbf{w}_{k,m} s_k \delta_{k,m} \\ &= \sum_{k\in\mathcal{K}} \sqrt{\rho_{k,m}}\mathbf{f}_{k,m} s_k \delta_{k,m} \in \mathbb{C}^{N\times 1}.\end{aligned} \tag{3.3}$$

The signal intended for UE k consists of the data symbol s_k with power $\mathbb{E}(|s_k|^2) = 1$ and the precoding vector $\mathbf{w}_{k,m} \in \mathbb{C}^{N\times 1}$, which is made up of the power allocated to UE k by AP m $\rho_{k,m}$ and the normalized vector $\mathbf{f}_{k,m} \in \mathbb{C}^{N\times 1}$. To facilitate user-centric clustering, we introduce variable $\delta_{k,m} \in \{0, 1\}$ that indicates whether or not AP m serves UE k. For ease of notation, we define $\delta_{:,m}$ as the K-element AP-UE association vector of AP m, and $\delta_{k,:}$ as the M-element association vector of UE k. Note that in the case of $\delta_{k,m} = 0$, the corresponding $\rho_{k,m}$ and $\mathbf{f}_{k,m}$ are set to 0.

The input-output relationship of UE k is written as

$$\begin{aligned}y_k &= \sum_{m\in\mathcal{M}} \mathbf{h}_{k,m}^H \mathbf{x}_m + z_k \\ &= \underbrace{\sum_{m\in\mathcal{M}} \sqrt{\rho_{k,m}}\mathbf{h}_{k,m}^H \mathbf{f}_{k,m} s_k \delta_{k,m}}_{\text{desired signal}} + \underbrace{\sum_{\substack{j\in\mathcal{K}\\ j\neq k}} \sum_{m\in\mathcal{M}} \sqrt{\rho_{j,m}}\mathbf{h}_{k,m}^H \mathbf{f}_{j,m} s_j \delta_{j,m}}_{\text{inter-user interference}} + \underbrace{z_k}_{\text{noise}},\end{aligned} \tag{3.4}$$

where z_k is the receiver noise with variance σ_z^2.

The SINR of UE k is expressed as

$$\text{SINR}_k = \frac{\left|\sum_{m\in\mathcal{M}} \sqrt{\rho_{k,m}}\mathbf{h}_{k,m}^H \mathbf{f}_{k,m} \delta_{k,m}\right|^2}{\sum_{\substack{j\in\mathcal{K}\\ j\neq k}} \left|\sum_{m\in\mathcal{M}} \sqrt{\rho_{j,m}}\mathbf{h}_{k,m}^H \mathbf{f}_{j,m} \delta_{j,m}\right|^2 + \sigma_z^2}. \tag{3.5}$$

The signal power is computed by coherently adding the contributions of the APs forming the user-centric cluster. The spectral efficiency of UE k in bps/Hz is

$$\text{SE}_k = \log_2(1 + \text{SINR}_k). \tag{3.6}$$

3.2 Impact of Cluster Formation

In this section, we investigate how user-centric clustering influences the total fronthaul requirement and guaranteed rate of the cell-free network. Specifically, we

formulate the corresponding non-convex optimization problems and propose algorithms to approximately solve them.

3.2.1 Fronthaul Optimization

The fronthaul capacity is limited in practice, despite it being assumed to be an unlimited resource in most research works [104]. Therefore, it has to be taken into account when designing a realistic coordinated distributed wireless system. Motivated by this, we formulate the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}} \delta_{k,m} && \text{(3.7a)} \\ \text{w.r.t.} \quad & \delta_{k,m} \in \{0,1\}, \\ & 0 \le \rho_{k,m} \le \rho_{\max}\delta_{k,m}, \\ \text{subject to} \quad & \sum_{k\in\mathcal{K}} \rho_{k,m}\delta_{k,m} \le \rho_{\max}, \forall m \in \mathcal{M}, && \text{(3.7b)} \\ & \text{SINR}_k \ge \Gamma, \forall k \in \mathcal{K}. && \text{(3.7c)} \end{aligned}$$

The objective in (3.7a) is to minimize the total number of established AP-UE connections. The power constraint in (3.7b) ensures that the sum of the non-negative UE powers allocated by AP m does not exceed its maximum transmit power $\rho_{\max}$. For simplicity, we assume the same value for $\rho_{\max}$ for all M APs. Note that $\rho_{k,m}$ is non-zero only if $\delta_{k,m} = 1$. The per-UE SINR constraint in (3.7c) defines a certain performance level Γ that is guaranteed by the cell-free network.

Problem (3.7) is written in terms of the AP-UE link count and considers neither the type of information nor the actual number of bits transmitted via the fronthaul links. Nevertheless, this aligns with our goal, since minimizing the number of AP-UE connections implicitly minimizes the required fronthaul capacity of the network. Specifically, fewer connections result in smaller user-centric clusters. With the APs serving fewer UEs, less information go through the fronthaul links. More importantly, we do so while making sure that the users still experience good service, as imposed by Constraint (3.7c).

We reformulate the SINR constraint in (3.7c) as

$$\text{SINR}_k = \frac{\left|\sum_{m\in\mathcal{M}} \sqrt{\rho_{k,m}}\mathbf{h}_{k,m}^H\mathbf{f}_{k,m}\delta_{k,m}\right|^2}{\sum_{\substack{j\in\mathcal{K}\\ j\neq k}}\left|\sum_{m\in\mathcal{M}} \sqrt{\rho_{j,m}}\mathbf{h}_{k,m}^H\mathbf{f}_{j,m}\delta_{j,m}\right|^2 + \sigma_z^2} = \frac{|s_k|^2}{\|\mathbf{i}_k\|^2}, \tag{3.8}$$

$$\Re(s_k) \ge \sqrt{\Gamma}\|\mathbf{i}_k\|, \qquad \Im(s_k) = 0 \tag{3.9}$$

following a similar approach as in [105]. This renders the problem solvable using

software tools, including CVX [106] and MOSEK [107]. However, the presented mixed-integer linear programming (MILP) problem is NP-hard, and its complexity grows exponentially with M and K. Thus, we apply linear programming (LP) relaxation by dropping the integer requirement for $\delta_{k,m}^{(\mathrm{MILP})} \in \{0,1\}$ and allowing it to take in continuous values, such that $\delta_{k,m}^{(\mathrm{LP})} \in [0,1]$. The relaxation leads to a convex problem that is solvable in polynomial time, even for a large network of APs and UEs. This transformation is governed by the following relationship, controlled by a threshold variable $\alpha_{\mathrm{thr}} \in [0,1]$,

$$\delta_{k,m}^{(\mathrm{MILP})} = \begin{cases} 1 & \delta_{k,m}^{(\mathrm{LP})} \geq \alpha_{\mathrm{thr}}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

Equation (3.10) provides a straightforward approach for relating the relaxed solution back to the original one. However, its effectiveness is dictated by our choice of the α_{thr} value. We may obtain a feasible but suboptimal solution to the original problem or even an infeasible solution where some of the constraints are violated. One way to determine the best α_{thr} is to sweep over its possible values. We start with $\alpha_{\mathrm{thr}} = 0$ and gradually increase it by a predefined step size until we reach $\alpha_{\mathrm{thr}} = 1$. At each iteration, we first solve the relaxed problem to get $\delta^{(\mathrm{LP})}$, $\forall k \in \mathcal{K}$, $m \in \mathcal{M}$. We then convert them to $\delta^{(\mathrm{MILP})}$ based on the current α_{thr} in (3.10). A connection is established between UE k and AP m when $\delta_{k,m}^{(\mathrm{LP})}$ is at least α_{thr}, where we set $\delta_{k,m}^{(\mathrm{MILP})}$ to 1. The resulting $\delta^{(\mathrm{MILP})}$ values are utilized to compute the user SINRs and the objective function in (3.7a). We check if the SINR constraint in (3.7c) is satisfied for all the UEs, such that after all the α_{thr} update iterations, we know which α_{thr} values provide a feasible solution. Since our objective is to minimize the fronthaul requirement by keeping the AP-UE link count as low as possible, we select the α_{thr} that corresponds to the smallest objective value. The procedure is summarized in Algorithm 1.

Algorithm 1: Generic approach to solve Problem (3.7)

1: Apply LP relaxation to the original MILP problem.
2: Solve the LP relaxed problem to obtain $\delta^{(\mathrm{LP})}$.
3: **for** $(\alpha_{\mathrm{thr}} = 0; \alpha_{\mathrm{thr}} \leq 1; \alpha_{\mathrm{thr}} = \alpha_{\mathrm{thr}} + \mathrm{step\_size})$ **do**
4: **if** $\delta^{(\mathrm{LP})} \geq \alpha_{\mathrm{thr}}$ **then**
5: $\delta^{(\mathrm{MILP})} \leftarrow 1$
6: **else**
7: $\delta^{(\mathrm{MILP})} \leftarrow 0$
8: **end if**
9: Compute the objective function using $\delta^{(\mathrm{MILP})}$.
10: Compute SINR_k, $\forall k \in \mathcal{K}$, using $\delta^{(\mathrm{MILP})}$.
11: $\mathrm{SINR}_{\mathrm{check}} \leftarrow \max_k(\Gamma - \mathrm{SINR}_k)$
12: **end for**
13: Find the feasible points corresponding to $\mathrm{SINR}_{\mathrm{check}} \leq 0$.
14: Select the point with the smallest objective value.

3.2.2 Max-Min SINR Optimization

By eliminating the fixed cell boundaries, cell-free massive MIMO provides a uniformly good service to all UEs, including the cell-edge users who traditionally suffer from poor performance. We formulate the following optimization problem to determine how good the guaranteed rate is

$$\begin{aligned} \text{maximize} \quad & \min_k \mathrm{SINR}_k && \text{(3.11a)} \\ \text{w.r.t.} \quad & \delta_{k,m} \in \{0,1\}, \\ & 0 \leq \rho_{k,m} \leq \rho_{\max}\delta_{k,m}, \\ \text{subject to} \quad & \sum_{k\in\mathcal{K}} \rho_{k,m}\delta_{k,m} \leq \rho_{\max}, \forall m \in \mathcal{M}, && \text{(3.11b)} \\ & \sum_{m\in\mathcal{M}} \delta_{k,m} \leq \mathrm{csize}_{\max}, \forall k \in \mathcal{K}. && \text{(3.11c)} \end{aligned}$$

The objective in (3.11a) is to maximize the minimum user SINR of the network. The per-AP power constraint in (3.11b) ensures that the total allocated power does not exceed the maximum transmit power $\rho_{\max}$. Since practical deployments are typically fronthaul-limited [104], we impose a user-centric cluster size constraint in (3.11c), where $\mathrm{csize}_{\max}$ specifies the maximum number of APs that may serve each UE.

Problem (3.11) can be reformulated as

$$\begin{aligned}
\text{maximize} \quad & t && \text{(3.12a)}\\
\text{w.r.t.} \quad & \delta_{k,m} \in \{0,1\}, \\
& 0 \leq \rho_{k,m} \leq \rho_{\max}\delta_{k,m}, \\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} \rho_{k,m}\delta_{k,m} \leq \rho_{\max}, \forall m \in \mathcal{M}, && \text{(3.12b)}\\
& \sum_{m\in\mathcal{M}} \delta_{k,m} \leq \text{csize}_{\max}, \forall k \in \mathcal{K}, && \text{(3.12c)}\\
& \text{SINR}_k \geq t, \forall k \in \mathcal{K}. && \text{(3.12d)}
\end{aligned}$$

Similar to Sec. 3.2.1, we rewrite Constraint (3.12d) as (3.9) and apply LP relaxation that transforms $\delta_{k,m}^{(\text{MILP})} \in \{0,1\}$ into $\delta_{k,m}^{(\text{LP})} \in [0,1]$. The resulting convex problem can be solved using the bisection method, where at each iteration, a feasibility problem is solved and the value of t is updated accordingly. This iterative process continues as long as the bisection tolerance condition is satisfied [108].

The bisection method outputs $\delta_{k,m}^{(\text{LP})}$ of the relaxed problem, which still needs to be related back to $\delta_{k,m}^{(\text{MILP})}$ of the original problem using $\text{csize}_{\max}$. We interpret $\delta_{k,m}^{(\text{LP})}$ as the probability of establishing a connection between AP m and UE k. We, therefore, start the transformation process by sorting the $\delta^{(\text{LP})}$ values of UE k in descending order. After which, we take only the $\text{csize}_{\max}$ largest entries and determine the corresponding AP indices. Those APs are considered to be the best ones for UE k. As such, they make up its user-centric cluster, and we set the corresponding $\delta^{(\text{MILP})}$ variables to 1. This process is done for each UE. We then utilize the resulting $\delta^{(\text{MILP})}$ values to compute the user SINRs. Taking the minimum of those gives us the guaranteed QoS of the network. The procedure is outlined in Algorithm 2.

3.3 Numerical Evaluations

We consider a downlink cell-free MIMO network with $M = 50$ APs, having $N = 10$ antennas each, and a varying number of single-antenna UEs $K \in \{2, \ldots, 15\}$. The simulation parameters are listed in Table 3.1, where the chosen values guarantee the feasibility of the optimization problems. In order to minimize the information exchange among the APs, we utilize MRT, such that $\mathbf{f}_{k,m} = \frac{\mathbf{h}_{k,m}}{\|\mathbf{h}_{k,m}\|}$ in (3.5), as it has been shown to achieve good performance while allowing the precoding to be done locally at the APs [14].

Algorithm 2: Generic approach to solve Problem (3.12)

Apply LP relaxation to the original MILP problem.
Initialize $t_{\min}$, $t_{\max}$, and the bisection tolerance $\epsilon_{\text{tol}} > 0$.
while $t_{\max} - t_{\min} \geq \epsilon_{\text{tol}}$ **do**
  $t \leftarrow \frac{t_{\max}+t_{\min}}{2}$
  Solve the following feasibility problem to obtain $\delta^{(\text{LP})}$.

$$\left.\begin{aligned} &\text{SINR}_k \geq t, \forall k \in \mathcal{K}, \\ &\sum_{k\in\mathcal{K}} \rho_{k,m}\delta_{k,m} \leq \rho_{\max}, \forall m \in \mathcal{M}, \\ &\sum_{m\in\mathcal{M}} \delta_{k,m} \leq \text{csize}_{\max}, \forall k \in \mathcal{K} \end{aligned}\right\}$$

  if feasible **then**
    $t_{\min} \leftarrow t$
  else
    $t_{\max} \leftarrow t$
  end if
end while
for $k = 1$ to K **do**
  Sort $\delta^{(\text{LP})}(k,:)$ in descending order.
  Store in $[\text{val}, \text{AP}_{\text{idx}}]$.
  for $m = 1$ to M **do**
    if $\text{sum}(\delta^{(\text{MILP})}(k,:)) < \text{csize}_{\max}$ **then**
      $\delta^{(\text{MILP})}(k, \text{AP}_{\text{idx}}(m)) \leftarrow 1$
    end if
  end for
end for
Compute SINR_k, $\forall k \in \mathcal{K}$, using $\delta^{(\text{MILP})}$.
Max $\text{SINR}_{\min} \leftarrow \min_k(\text{SINR}_k)$

Table 3.1: Simulation parameters

Parameter	Value
Carrier frequency f_c	2 GHz
Path loss exponent n_c	2
Shadow fading variance σ_c^2	6
Noise variance σ_z^2	10^{-5}
Per-AP maximum transmit power $\rho_{\max}$	1 W

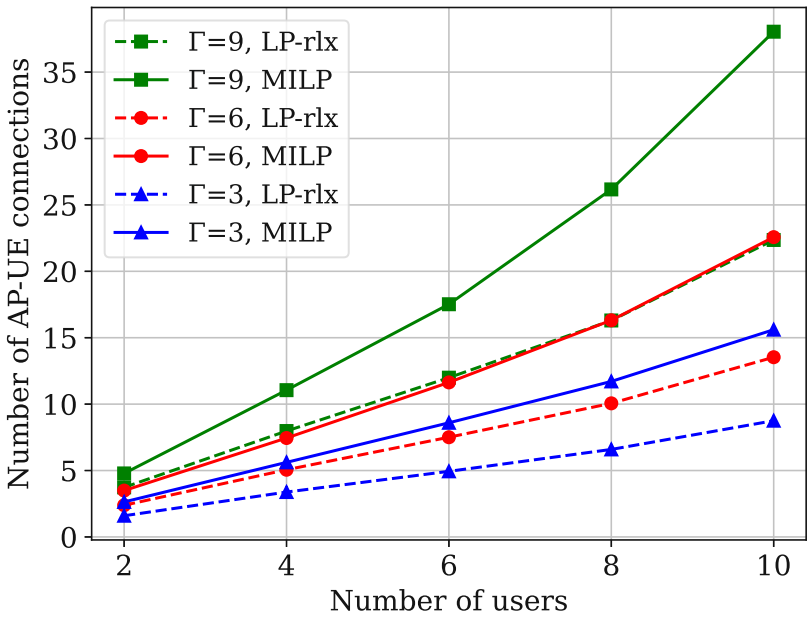


Figure 3.1: Total fronthaul requirement with varying number of users and guaranteed QoS level Γ expressed in dB.

3.3.1 Total Fronthaul Requirement

In Fig. 3.1, we present the results of the fronthual optimization when solving Problem (3.7) through Algorithm 1. The dashed and solid curves correspond to the LP relaxation (*LP-rlx*) and MILP solutions, respectively. For the minimization problem at hand, the LP solution provides a lower bound on that of MILP.

The cell-free network guarantees a performance level Γ in (3.7c). A higher value forces each UE to be served by more APs to meet the stricter QoS requirement. This implies more AP-UE connections are established, and the individual user-centric clusters are bigger. As a consequence, a larger amount of information is exchanged via the fronthaul links. It follows that the required fronthaul capacity increases with Γ.

Admitting more users to the network inevitably increases the total fronthaul requirement for a given Γ value in Fig. 3.1. Every additional UE needs at least one serving AP in its cluster. Moreover, based on the definition of cell-free massive MIMO, a UE is likely to be connected to more than one of the distributed APs to benefit from their joint coherent transmission. The number of APs in the user-centric cluster differs for the UEs. Some will require more to satisfy the same SINR constraint. The AP count depends on several factors, such as the macrosopic path loss of the user with respect to the APs, that then impact the UE rate calculation.

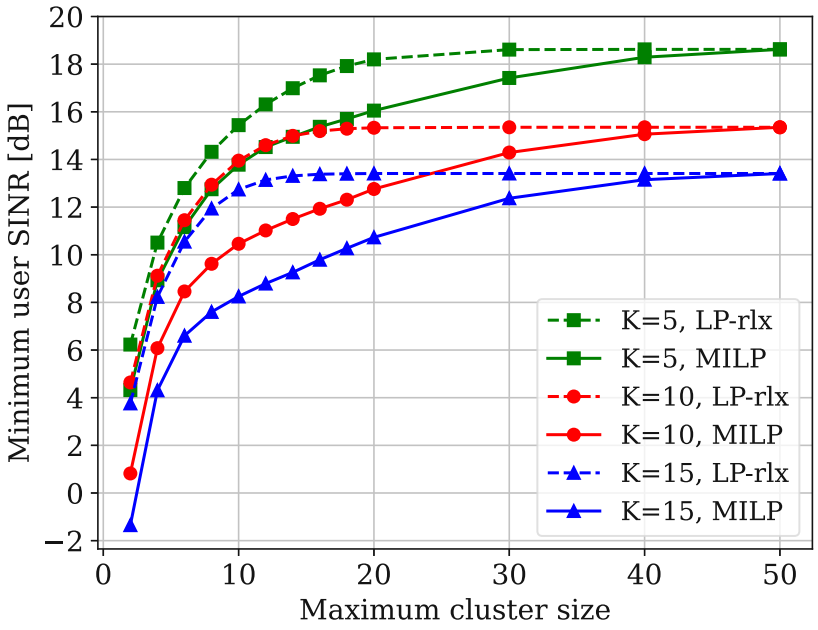


Figure 3.2: Guaranteed QoS with varying number of users and cluster size.

3.3.2 Guaranteed Quality of Service

In Fig. 3.2, we show the results of the max-min SINR optimization when solving Problem (3.12) through Algorithm 2. The dashed and solid curves correspond to the LP relaxation (*LP-rlx*) and MILP solutions, respectively. Note that the plotted SINR values for LP relaxation are obtained by solving the relaxed problem through the bisection method in Algorithm 2 (Lines 1 to 11). On the other hand, we get those of MILP by first converting $\delta^{(\mathrm{LP})}$ to $\delta^{(\mathrm{MILP})}$ and then using $\delta^{(\mathrm{MILP})}$ to calculate the achievable user SINRs (Lines 12 to 22). In this case, we have a maximization problem, where the LP solution serves as an upper bound on that of MILP.

We first focus on the scenario of $K = 10$ users (in red). The per-UE cluster size constraint in (3.12c) limits the maximum number of APs that can serve each user to $\mathrm{csize}_{\max}$. In Fig. 3.2, we observe that for both LP and MILP, the guaranteed rate increases with $\mathrm{csize}_{\max}$. With possibly more APs coherently transmitting to each UE, the individual user rates, as well as the guaranteed QoS, are likely to be higher. However, in the case of LP relaxation, it starts to saturate at $\mathrm{csize}_{\max} = 20$. Beyond this point, increasing $\mathrm{csize}_{\max}$ does not impact performance anymore, and the maximum $\mathrm{SINR}_{\min}$ stays at around 15.5 dB. On the other hand, the MILP curve does not exhibit such behavior. Although, notice that the rate improvement gets less pronounced with increasing $\mathrm{csize}_{\max}$. Specifically, the first 10 APs already provide 10.5 dB, while the next 10 only amount to a 2.5-dB increase. In the extreme case that each UE is connected to all 50 APs, corresponding to the canonical setup, we only achieve 15.5 dB. This is a modest improvement in performance given that

we possibly add 400 AP-UE connections in the process, translating to a significant increase in the required fronthaul capacity. This suggests that when allowing the formation of bigger user-centric clusters, the additional APs are likely to be already far away from the UEs, such that they have little effect on the user rates. Based on these results, the user-centric setup outperforms its canonical counterpart, since in the former case, we are able to experience good performance without being burdened by the fronthaul requirement of the network. Lastly, decreasing the number of users to be served by the same number of APs leads to lower interference, causing an upward shift to the plots in Fig. 3.2.

3.4 Summary

In this chapter, we investigated how user-centric clustering impacts performance, specifically, in terms of the total fronthaul requirement and guaranteed QoS of cell-free massive MIMO. We first formulated and approximately solved an AP-UE link count minimization problem. While we did not consider the actual transmitted bits, we implicitly minimized the fronthaul requirement by establishing the least number of connections possible that satisfied the predefined user demand, resulting in a smaller amount of information being exchanged through the fronthaul links. A stricter QoS constraint necessitated more serving APs per user-centric cluster, subsequently increasing the fronthaul requirement of the network.

We next focused on a max-min SINR optimization problem. We assumed a cluster size constraint that specifies the maximum number of APs available to each user. Here, we had a first look at the performance comparison between the canonical and user-centric cell-free massive MIMO setups. Our numerical experiments demonstrated that there exists a cluster size such that the latter setup is overall better than the former. In reality, only a subset of APs significantly impacts performance. This implies that, even with small cluster sizes (user-centric), we are able to achieve almost the same guaranteed rate as the canonical variant while considerably reducing fronthaul capacity usage.

4

SARL-based Network Optimization

Conventional optimization techniques have been instrumental in solving various problems in wireless communications. Many relevant problems are non-convex and are dealt with using algorithms of high computational complexity. These may be time-consuming to implement in practice [28], including those designed in Chapter 3. Leveraging ML to tackle this challenge has been a leap forward, as not only does it involve low-complexity solutions, but also equips the wireless network with intelligence for potential self-optimization capabilities.

In this chapter, we consider different optimization problems in the context of cell-free massive MIMO and rely on DRL to solve them. We focus on fully centralized frameworks, in which we utilize a single agent. Among the aspects that we will look into is power consumption. As a step towards greener networking, we aim to develop energy-efficient cell-free systems. To this end, we propose methods to dynamically select and activate only a subset of APs based on the instantaneous user positions. Another aspect that we are interested in is network scalability, which is critical in practical coordinated distributed wireless systems. This can be viewed in two distinct levels, namely CPU-AP and AP-UE associations. In the former case, we consider multiple CPUs and group the APs connected to the same CPU. The latter case is synonymous to user-centric clustering, in that we decide which subset of APs serves each UE. In both levels, we employ SARL for cluster formation based on predefined optimization objectives. The frameworks developed in this chapter have been published in [50, 59, 67].

4.1 Extended System Model

We again consider the downlink case of cell-free massive MIMO and extend the system model presented in Sec. 3.1. To accommodate the dynamic AP (de-)activation that we will explore in this chapter, we define a new variable $b_m \in \{0, 1\}$ that indicates whether or not AP m is active. The modified input-output relationship of UE

k is

$$y_k = \underbrace{\sum_{m\in\mathcal{M}} b_m\sqrt{\rho_{k,m}}\mathbf{h}_{k,m}^H\mathbf{f}_{k,m}s_k\delta_{k,m}}_{\text{desired signal}} + \underbrace{\sum_{\substack{j\in\mathcal{K}\\ j\neq k}}\sum_{m\in\mathcal{M}} b_m\sqrt{\rho_{j,m}}\mathbf{h}_{k,m}^H\mathbf{f}_{j,m}s_j\delta_{j,m}}_{\text{inter-user interference}} + \underbrace{z_k}_{\text{noise}}. \quad (4.1)$$

The corresponding instantaneous SINR of UE k is then

$$\text{SINR}_k = \frac{\left|\sum_{m\in\mathcal{M}} b_m\sqrt{\rho_{k,m}}\mathbf{h}_{k,m}^H\mathbf{f}_{k,m}\delta_{k,m}\right|^2}{\sum_{\substack{j\in\mathcal{K}\\ j\neq k}}\left|\sum_{m\in\mathcal{M}} b_m\sqrt{\rho_{j,m}}\mathbf{h}_{k,m}^H\mathbf{f}_{j,m}\delta_{j,m}\right|^2 + \sigma_z^2}. \quad (4.2)$$

Here, we coherently add the contributions of the active APs only. In fact, the base system model in Sec. 3.1 is a special case where $b_m = 1$, $\forall m \in \mathcal{M}$. In this chapter, depending on the optimization objective of a given SARL framework, we configure the values of b_m and $\delta_{k,m}$, $\forall k \in \mathcal{K}$, $m \in \mathcal{M}$, accordingly.

As we will see later when we discuss the proposed frameworks in detail, the SINR has a direct impact on the decision-making process of the DRL agent at each time step. Since the instantaneous SINR in (4.2) is based on the microscopic CSI, the decisions become sensitive even to small user movements. This may lead to an undesirable behavior, such as switching the APs on/off too frequently. Thus, we utilize the average SINR, given by

$$\begin{aligned}\overline{\text{SINR}}_k &= \frac{\mathbb{E}\left\{\left|\sum_{m\in\mathcal{M}} b_m\sqrt{\rho_{k,m}}\mathbf{h}_{k,m}^H\mathbf{f}_{k,m}\delta_{k,m}\right|^2\right\}}{\sum_{\substack{j\in\mathcal{K}\\ j\neq k}}\mathbb{E}\left\{\left|\sum_{m\in\mathcal{M}} b_m\sqrt{\rho_{j,m}}\mathbf{h}_{k,m}^H\mathbf{f}_{j,m}\delta_{j,m}\right|^2\right\} + \sigma_z^2} \\ &= \frac{\sum_{m\in\mathcal{M}} b_m\rho_{k,m}g_{k,m}\delta_{k,m} + \sum_{m\in\mathcal{M}}\sum_{\substack{n\in\mathcal{M}\\ n\neq m}} b_m b_n\sqrt{\rho_{k,m}g_{k,m}\rho_{k,n}g_{k,n}}\delta_{k,m}\delta_{k,n}}{\sum_{\substack{j\in\mathcal{K}\\ j\neq k}}\left(\sum_{m\in\mathcal{M}} b_m\frac{\rho_{j,m}g_{k,m}}{N}\delta_{j,m}\right) + \sigma_z^2}. \quad (4.3)\end{aligned}$$

Recall that we consider an i.i.d. Rayleigh fading channel in (3.1). We implement the precoding locally at the APs and employ MRT, where $\mathbf{f}_{k,m} = \frac{\mathbf{h}_{k,m}}{\|\mathbf{h}_{k,m}\|}$, to minimize the information exchange among the APs [14]. Given these assumptions, we obtain the SINR expression in (4.3), which is based on the macroscopic gain. This allows

for a more stable behavior for a longer period of time, in addition to only having to estimate the gains instead of the channel vectors. The spectral efficiency of UE k in bps/Hz is

$$\mathrm{SE}_k \approx \log_2(1 + \overline{\mathrm{SINR}}_k), \tag{4.4}$$

and the rate in bps is

$$u_k \approx B \log_2(1 + \overline{\mathrm{SINR}}_k), \tag{4.5}$$

where B is the system bandwidth in Hz.

4.2 Dynamic AP Activation

A dense AP deployment is envisioned in cell-free massive MIMO to support the exponential growth in mobile data traffic while enabling a uniformly good service [25]. This, however, triggers a substantial increase in energy consumption, and thus, it is viewed negatively from both economic and environmental standpoints. Since the traffic demand fluctuates throughout the day, one way to tackle this dilemma is to switch off underutilized APs during off-peak hours. In this section, we propose a SARL framework that dynamically (de-)activates APs based on the presence of users in the network [59].

4.2.1 Problem Formulation

Our goal is to derive the best set of APs to activate that maximizes the minimum user SINR as

$$\begin{aligned} \text{maximize} \quad & \alpha \left(\min_k \mathrm{SINR}_k \right) - (1-\alpha) \sum_{m \in \mathcal{M}} b_m && \text{(4.6a)} \\ \text{w.r.t.} \quad & b_m, \\ \text{subject to} \quad & b_m \in \{0,1\}, \forall m \in \mathcal{M}, && \text{(4.6b)} \end{aligned}$$

where $\alpha \in [0,1]$ is a weighting factor. Finding the optimal set of APs is considered an NP-hard, combinatorial problem, where we search over all possible AP activation combinations [56]. As shown in [54], finding the globally optimal solution entails high computational complexity, and thus, suboptimal heuristic algorithms were proposed in [56, 57]. We, therefore, rely on DRL to deal with this challenging problem. Here, we assume that all active APs serve all UEs, such that $\delta_{k,m} = 1, \forall k \in \mathcal{K}, m \in \mathcal{M}_{\mathrm{ON}}$, in Sec. 4.1. We define the set of active APs as $\mathcal{M}_{\mathrm{ON}} = \{m \in \mathcal{M} | b_m = 1\}$.

4.2.2 DDQN Framework for AP Activation

We now present a SARL framework to determine the best APs to turn on by exploiting the available instantaneous user position information. The agent-environment

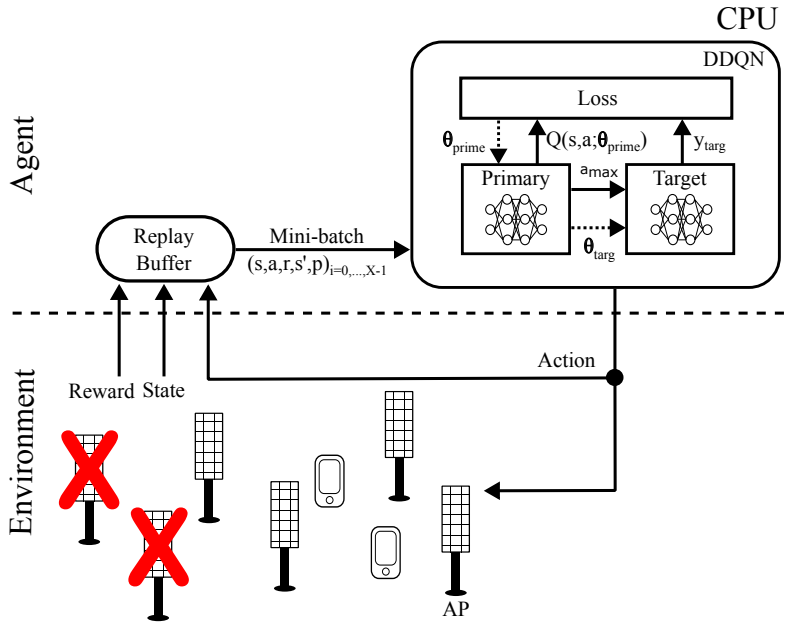


Figure 4.1: SARL-DDQN framework for AP activation based on the presence of users.

scenario is illustrated in Fig. 4.1, where the CPU serves as the agent, and the environment consists of the APs and UEs.

1. State
   The state of the environment at time t is described by

$$\mathbf{s}^{(t)} = [b_1^{(t-1)}, \ldots, b_M^{(t-1)}, g_{1,1}^{(t)}, \ldots, g_{K,M}^{(t)}]. \tag{4.7}$$

   The first M elements signal the current ON/OFF status of the M APs, as a result of the AP selection decision made at the previous time step $t-1$. The last KM elements indirectly represent the position information of the K users, since the path gains are largely determined by the AP-UE distances.

2. Action
   We enumerate all possible AP activation combinations in the discrete action space of size 2^M. Based on its state observation, the agent decides which APs to switch on/off by selecting the combination index as

$$a^{(t)} = \text{ind} \mapsto [b_1, \ldots, b_M]. \tag{4.8}$$

   We then map the scalar ind to the corresponding M-element activation vector. Note that this formulation allows us to change the ON/OFF status of several APs simultaneously. Alternatively, we can reduce complexity by switching only a single AP at the expense of slower convergence. The action selection

process is governed by the ϵ-greedy mechanism in (2.14). We obtain the latest set of active APs, given by $\mathcal{M}_{\text{ON}}^{(t)} = \{m \in \mathcal{M} | b_m^{(t)} = 1\}$.

3. Power allocation
Each active AP divides its transmit power among the K UEs by solving the following max-min SINR optimization problem

$$\begin{aligned} \text{maximize} \quad & \beta^{(t)} && \text{(4.9a)} \\ \text{w.r.t.} \quad & 0 \leq \rho_{k,m}^{(t)} \leq \rho_{\max}, \\ \text{subject to} \quad & \sum_{k \in \mathcal{K}} \rho_{k,m}^{(t)} \leq \rho_{\max}, \forall m \in \mathcal{M}_{\text{ON}}, && \text{(4.9b)} \\ & \text{SINR}_k^{(t)} \geq \beta^{(t)}, \forall k \in \mathcal{K}. && \text{(4.9c)} \end{aligned}$$

Here, we define β as the minimum user SINR to be maximized, which is guaranteed for all K UEs in (4.9c). The power constraint in (4.9b) ensures that the total power allocated by each active AP does not exceed its maximum transmit power $\rho_{\max}$, whose value is the same for all M APs. We use a similar technique as in Chapter 3 of rewriting the SINR constraint as (3.9). The reformulated problem is convex, and it can be solved using the bisection method [108]. As an alternative, we may also employ low-complexity power assignment schemes, for instance, one that is proportional to the channel gains.

4. Reward
The goal of the SARL framework is to determine which APs to activate such that we maximize the guaranteed QoS and improve the energy consumption of the system by switching off underutilized APs. However, these two objectives have contrasting effects on performance. We, therefore, study their trade-off by expressing the reward function as

$$r^{(t+1)} = \underbrace{\alpha \beta_{\text{norm}}^{(t)}}_{\text{QoS}} - \underbrace{(1-\alpha)\rho_{\text{norm}}^{(t)}}_{\text{power}}, \tag{4.10}$$

where

$$\beta_{\text{norm}}^{(t)} = \frac{\beta^{(t)}}{\beta_{\max}}, \tag{4.11}$$

and

$$\rho_{\text{norm}}^{(t)} = \frac{\sum_{m \in \mathcal{M}} b_m^{(t)} \left(\rho_{\text{circuit}} + \sum_{k \in \mathcal{K}} \rho_{k,m}^{(t)} \right)}{M(\rho_{\text{circuit}} + \rho_{\max})}. \tag{4.12}$$

The first term in (4.10) corresponds to QoS, where $\beta^{(t)}$ is obtained by solving Problem (4.9), and $\beta_{\max}$ is the maximum rate when all the APs are turned on. The second term corresponds to the total power consumption of the system, where we define ρ_{circuit} as the power consumed by the circuitry of an AP when

it is active. Note that more sophisticated power consumption models have been presented in the literature [56]. However, in this work, we assume that ρ_{circuit} encompasses all the factors contributing to the additional power utilized by an active AP compared to when it is inactive. We normalize both terms in (4.10) to ensure a fair comparison when investigating the trade-off between rate maximization and power minimization. Moreover, we attach a weighting factor $\alpha \in [0, 1]$ to easily configure the operating point by giving more weight or importance to one of the terms.

5. DRL algorithm
   Given the continuous state space in (4.7) and the discrete action space in (4.8), we utilize the DDQN algorithm, in which we employ separate DNNs for the primary and target networks to stabilize the DRL learning process [92]. We refer to Sec. 2.2.3 for details on DDQN.

Algorithm 3 outlines the procedure used in this work. We first initialize the DRL parameters (Lines 1 to 2). The CPU observes the ON/OFF status of the APs and the presence of users in the coverage area (Line 5). It then decides to (de-)activate certain APs based on its current policy and ϵ value (Lines 6 to 7). After deriving the set of active APs, power allocation is done by centrally solving the max-min SINR optimization problem in (4.9) (Line 8). The CPU receives a reward, which depends on the configured α value (Line 9). It saves the newly acquired experience in its replay buffer or memory $\mathcal{B}$ and proceeds to randomly sample a mini-batch of X experiences (Lines 10 to 11). Those samples are used to calculate the loss and subsequently update the primary network by performing gradient descent (Lines 12 to 13). The target DNN parameter is updated using Polyak averaging (Line 14). Finally, we gradually decrease ϵ until it reaches ϵ_{end} (Line 15).

4.2.3 Numerical Evaluations

We assume a downlink cell-free MIMO network with $M = 8$ APs, having $N = 10$ antennas each, and vary the number of single-antenna UEs $K = \{2, 4, 6\}$ in a 60×60 m^2 area. In this work, we consider a rather small scenario for reasons of complexity and limited compute capability. A higher M corresponds to a significantly larger action space of size 2^M, in addition to solving the max-min SINR optimization problem in (4.9) at each time step. These will be addressed later in the thesis when we utilize other DRL algorithms and transition to distributed architectures. The simulation parameters are summarized in Table 4.1.

Algorithm 3: DDQN for dynamic AP activation

Initialize the DDQN primary (θ_{prime}) and target (θ_{targ}) networks.
Initialize the ϵ-greedy algorithm $\epsilon \leftarrow \epsilon_{\text{start}}$ for RL exploration-exploitation.
for episode $e = 0, \ldots, E-1$ **do**
  for time step $t = 0, \ldots, T-1$ **do**
    Observe the current state $\mathbf{s}^{(t)}$ (4.7).
    Select an action $a^{(t)}$ (4.8) following (2.14).
    (De-)activate the APs accordingly.
    Solve the max-min SINR optimization problem in (4.9) using bisection.
    Observe the reward $r^{(t+1)}$ (4.10) and next state $\mathbf{s}^{(t+1)}$.
    Store experience $\{\mathbf{s}^{(t)}, a^{(t)}, r^{(t+1)}, \mathbf{s}^{(t+1)}\}$ in $\mathcal{B}$.
    Sample a random mini-batch of X experiences from $\mathcal{B}$.
    Compute the loss or MSE (2.12) for all the samples.
    Update θ_{prime} by minimizing the loss using gradient descent.
    Update θ_{targ} using Polyak averaging (2.13).
    Update ϵ if $\epsilon > \epsilon_{\text{end}}$ following (2.15).
  end for
end for

Table 4.1: Simulation parameters

Parameter	**Value**
Carrier frequency f_c	2 GHz
Path loss exponent n_c	2
Shadow fading variance σ_c^2	6
Noise variance σ_z^2	10^{-8}
Per-AP maximum transmit power ρ_{max}	1 W
Per-AP circuit power ρ_{circuit}	1 W
Buffer size	10000
Mini-batch size X	64
Learning rate α_{step}	0.01
Discount factor γ	0.99
$\epsilon_{\text{start}}, \epsilon_{\text{end}}, \epsilon_{\text{decay}}$	1, 0.05, 0.0005
Polyak factor τ_{pol}	0.001

Table 4.2: AP set solutions

	Number of users		
Scheme	2	4	6
Canonical CF	11111111	11111111	11111111
DRL, $\alpha = 0.6$	00110111	00111111	01111111
DRL, $\alpha = 0.5$	00110001	00110011	00010111
DRL, $\alpha = 0.4$	00010000	00010001	00010011
Small cells	00010000	00010001	00010111

We refer to our proposed framework as *DRL*. After experimenting with the DNN architecture and tuning the hyperparameters, the final architecture, employed by the DDQN primary and target networks, is a fully connected DNN with two hidden layers, having 16 neurons each, and the rectified linear unit (ReLU) as the activation function. We utilize the following benchmark schemes to evaluate its performance.

1. *Canonical CF* – All the APs are always active and serve all the users.
2. *Small cells* – Each UE is connected to the best AP with the largest channel gain, and we turn off the unused APs.

The AP set solutions are provided in Table 4.2 for the different schemes and number of users. Each 8-bit solution represents the ON/OFF status of the eight APs.

The guaranteed QoS is depicted in Fig. 4.2. The *Canonical CF* baseline achieves the highest rates, with all the APs contributing to the individual user performance. However, this is at the expense of high power consumption in Fig. 4.3, since all the APs are assumed to be active all the time. Comparing this to *DRL* when $\alpha = 0.6$, we see at most a 1.5-dB gap in terms of QoS while saving up to 6 W. That is, we consume 37.5% more power in exchange for only 7% gain in guaranteed rate with *Canonical CF*, which is rather energy-inefficient. This demonstrates that by exploiting the available user position information to properly select and activate only a subset of APs that significantly contributes to performance, it becomes possible to experience good QoS while keeping power consumption relatively low. Meanwhile, setting α to 0.4, we observe that *DRL* behaves close to *Small cells*, with almost the same APs turned on for the two systems in Table 4.2. Admitting more users to the network decreases the guaranteed rate while increasing the total power consumption for all the schemes.

We investigate the impact of the weighting factor α in (4.10) that controls the trade-off between maximizing the minimum user rate and minimizing power consumption. Configuring α to 0.5 implies that we give equal importance to both objectives. Increasing α signifies that we prioritize improving the guaranteed QoS of the network, as depicted in the upward shift of the curves in Fig. 4.2. However, in order to do so, we are forced to activate more APs in the process. This can be confirmed in Table 4.2 by comparing the number of active APs for different α values.

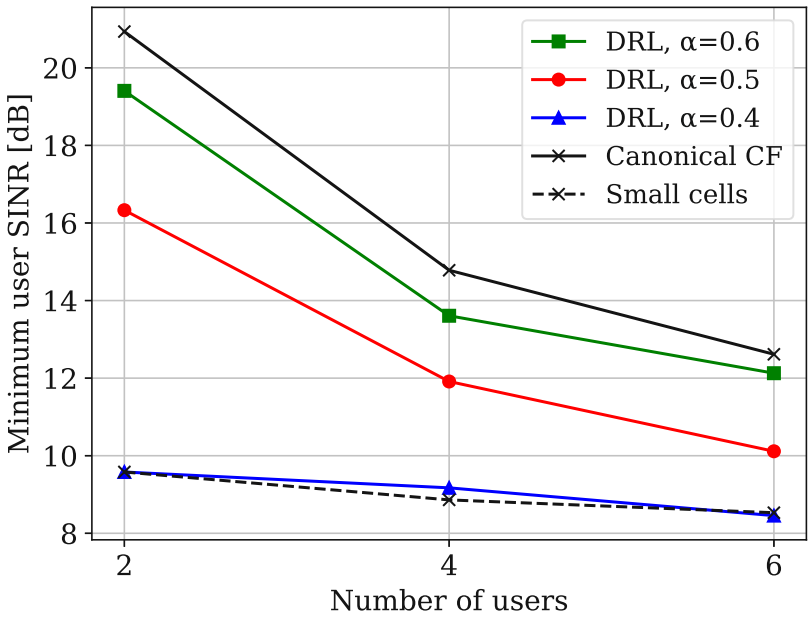


Figure 4.2: Guaranteed QoS for different schemes and α values with a varying number of users.

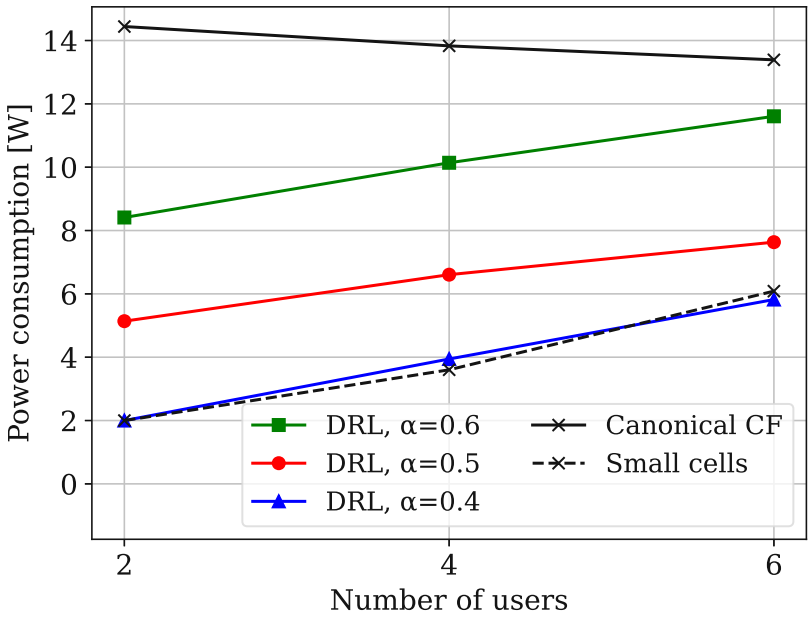


Figure 4.3: Power consumption for different schemes and α values with a varying number of users.

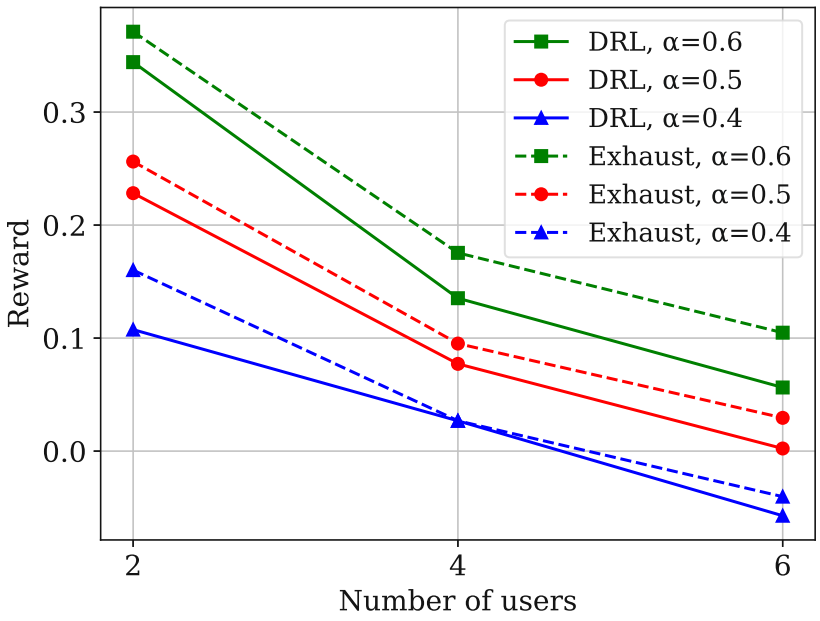


Figure 4.4: Reward comparison for DRL and exhaustive search with a varying number of users.

For instance, in the case of four users, 50% of the APs are turned on when $\alpha = 0.5$, as opposed to 75% when $\alpha = 0.6$. Thus, the system consumes more power in the latter case in Fig. 4.3. If we further increase α, we expect to reach the upper bound, *Canonical CF*. On the other hand, decreasing α signals to the SARL framework that we target lowering power consumption. This then deactivates more APs, as reflected in Table 4.2. For the same example of four users, only 25% of the APs remain turned on when $\alpha = 0.4$. Although this translates to power savings, having fewer APs available to the users leads to a decrease in the guaranteed rate.

The reward in (4.10) that is maximized by the DRL agent during training is shown in Fig. 4.4 for different α settings. With reference to Fig. 4.2 and 4.3, QoS goes down while power goes up for increasing number of users, which explains the decreasing trend in the reward plots. We also performed an exhaustive search to see how close we are to the optimal one. Although our proposed SARL system provides a suboptimal solution, the gap is not large in Fig. 4.4, suggesting that the framework is able to derive a near-optimal set of active APs while accounting for varying performance targets that we configure through α.

4.3 Spatial User Density-based AP Clustering

The realization of scalable cell-free massive MIMO not only requires that each user be served by a subset of APs (user-centric), but also that different geographical areas

be covered by distinct CPUs. Indeed, having a single CPU handle all the data to and from all the APs is infeasible and does not scale well, especially for a dense AP deployment. This was demonstrated in [63], where the computational complexity of single- and multi-CPU setups was investigated. In principle, the CPUs do not need to be hardware entities in the network. They may be instantiated on demand, such as in the case of CRAN, which comes with several benefits [17]. It follows that given a predefined number of CPUs, our next step is to figure out (1) which APs are associated to each CPU (i.e., we group the APs connected to the same CPU) and (2) how large the individual AP groups are. In this section, we propose a SARL framework that considers a multi-CPU environment and forms the AP groups based on the spatial density of users [67].

4.3.1 AP Clustering

We aim to partition the canonical cell-free network into G disjoint groups. We also refer to each group as an AP cluster, which is connected to one of the available G CPUs. We distinguish the AP groups from one another through their identifier, listed in set $\mathcal{G} = \{1, \ldots, G\}$. The group in which AP m belongs to is indicated by variable $c_m \in \mathcal{G}$. We assume that each UE is served by a single AP group at a time, and thus, we introduce variable $d_k \in \mathcal{G}$ to specify the serving group of UE k. We denote the set of APs belonging to group g by $\mathcal{M}_g = \{m \in \mathcal{M} | c_m = g\}$, such that $\mathcal{M} = \cup_{i=1}^{|\mathcal{G}|} \mathcal{M}_i$ and $\mathcal{M}_i \cap \mathcal{M}_j = \emptyset$. Similarly, the set of UEs served by the APs in group g is $\mathcal{K}_g = \{k \in \mathcal{K} | d_k = g\}$, such that $\mathcal{K} = \cup_{i=1}^{|\mathcal{G}|} \mathcal{K}_i$ and $\mathcal{K}_i \cap \mathcal{K}_j = \emptyset$. The AP-UE association is given by

$$\delta_{k,m} = \begin{cases} 1 & d_k = c_m, \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

The above conditions state that UE k is only served by all the APs belonging to group d_k.

The AP clustering is based on the spatial user density, which is characterized by a continuous function over the region of interest [112]. To simplify the problem, we discretize the region into F subareas or subregions. Each subarea f is described by its corresponding spatial density value that provides the expected number of users. Assuming that the users follow a Poisson distribution, this expected value serves as the mean λ_f of the Poisson random variable, which we then use to obtain the instantaneous number of UEs that are uniformly distributed over subregion f. The expected sum rate is

$$u_{\text{exp}} = \mathbb{E}\left\{u_{\text{sum}}\right\} = \mathbb{E}\left\{\sum_{g \in \mathcal{G}} \left(\sum_{k \in \mathcal{K}_g} u_k\right)\right\}, \tag{4.14}$$

where user rate u_k is calculated as in (4.5), and the expectation is with respect to

the random number of users and UE positions.

4.3.2 Problem Formulation

We aim to group the APs such that the expected network sum rate is maximized as follows

$$\begin{aligned} \text{maximize} \quad & u_{\text{exp}} && \text{(4.15a)} \\ \text{w.r.t.} \quad & \delta_{k,m} \in \{0,1\}, \\ & c_1, \ldots, c_M \in \mathcal{G}, \\ & d_1, \ldots, d_K \in \mathcal{G}, \\ \text{subject to} \quad & \delta_{k,m} = \begin{cases} 1 & d_k = c_m, \\ 0 & d_k \neq c_m, \forall k \in \mathcal{K}, m \in \mathcal{M}. \end{cases} && \text{(4.15b)} \end{aligned}$$

Problem (4.15) involves stochastic optimization [111] and is an integer programming problem. A reformulation that could achieve convexity is not known to exist, and thus, we leverage DRL to handle this complex problem. Here, we consider active APs only, such that $b_m = 1$, $\forall m \in \mathcal{M}$, in Sec. 4.1.

4.3.3 DDQN Framework for AP Clustering

In this section, we detail the proposed SARL framework for AP clustering. We assume that the spatial user density information for a given time period is known at the operator side. The system is depicted in Fig. 4.5, where a central node acts as the agent that controls the (instantiated) CPUs, while the APs and UEs make up the environment.

1. State
   The environment at time t is described by the state vector
   $$\mathbf{s}^{(t)} = [c_1^{(t-1)}, \ldots, c_M^{(t-1)}, \lambda_1^{(t)}, \ldots, \lambda_F^{(t)}]. \tag{4.16}$$
   The first M elements indicate the latest AP grouping, which was decided at the previous time step $t-1$. The last F elements provide the spatial user density information of the F subareas at time t.

2. Action
   The agent decides the AP clustering configuration by selecting its index in the discrete action space as
   $$a^{(t)} = \text{ind} \mapsto [c_1, \ldots, c_M]. \tag{4.17}$$
   We then map the scalar ind to the actual M-element AP grouping vector. In total, we have $|\mathcal{G}|^M$ possible AP clustering configurations. Given that we

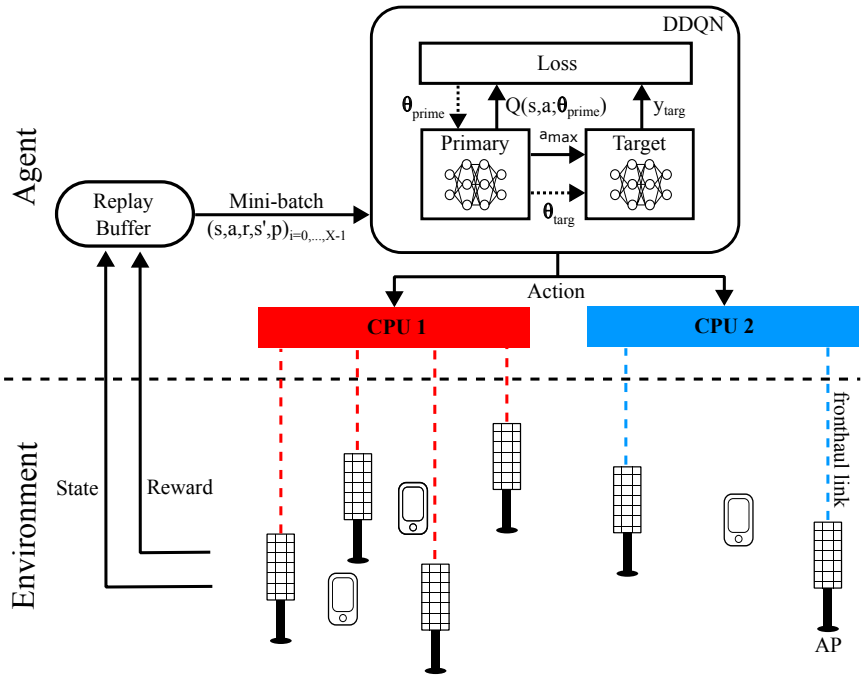


Figure 4.5: SARL-DDQN framework for AP clustering in a multi-CPU network.

envision a dense AP deployment, this implies that the size of the discrete action space would easily blow up. For this reason, we limit ourselves to v valid groupings only. In particular, we designed them in a way that only contiguous APs can be grouped together. This means that no two APs can be in the same group if there are APs between them belonging to a different group. The exploration-exploitation dynamics during action selection is dictated by the ϵ-greedy algorithm in (2.14).

3. AP-UE assignment
   Once the AP groups have been formed, the next task is to decide which group would serve each user. For this purpose, we rely on the channel gains of each UE with respect to all the APs. We first sort the gains in descending order. We consider a predefined search size z, such that we only take the z largest gains. We map them back to the corresponding APs to determine in which group the majority of the z APs belong to. This becomes the serving AP group of the user, and we subsequently set the $\delta_{k,m}$ values accordingly. Note that we designed the framework as such in order to decouple the AP clustering problem from the AP-UE association problem. Ideally, we would handle them jointly; however, the corresponding action space would be too large.

4. Power allocation
   The power assigned to UE k by AP m, assuming equal power allocation, is

expressed as

$$\rho_{k,m}^{(t)} = \delta_{k,m}^{(t)} \frac{\rho_{\max}}{\|\delta_{:,m}^{(t)}\|_0}, \forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \tag{4.18}$$

where $\rho_{\max}$ is the per-AP maximum transmit power. We define $\delta_{:,m}^{(t)}$ as the K-element AP-UE association vector of AP m at time t. We take its zero-norm to obtain the number of users served by AP m. Note that power $\rho_{k,m}^{(t)}$ is non-zero only if UE k and AP m belong to the same group, such that $\delta_{k,m}^{(t)} = 1$. As an alternative to this heuristic approach, we may also solve a power optimization problem, as previously done in Sec. 4.2.2, at the expense of higher complexity.

5. Reward
   The reward maximized by the agent is

$$r^{(t+1)} = u_{\text{exp}} \approx \frac{1}{I} \sum_{i=0}^{I-1} u_{\text{sum}}^{(i)}, \tag{4.19}$$

   where we estimate the expected network sum rate in (4.14) using Monte-Carlo approximation with I random realizations of number of users and UE positions.

6. DRL algorithm
   The problem at hand involves a continuous state space in (4.16) and a discrete action space in (4.17). We, therefore, rely on the DDQN algorithm [92], discussed in Sec. 2.2.3.

Algorithm 4 summarizes the procedure of the proposed system. We start with the initialization phase by setting up the DDQN and ϵ-greedy algorithm parameters (Lines 1 to 2). The agent observes the latest AP grouping and the spatial density of users in different subareas (Line 5). Following this, it decides a new AP clustering configuration according to its current policy and ϵ value (Lines 6 to 7). After applying the action, we implement Monte-Carlo approximation by generating I random realizations of number of users and UE positions. For each realization, we determine the AP-UE assignment, perform equal power allocation, and finally, compute the sum rate (Lines 8 to 12). We take the I resulting sum rates to calculate the average reward (Line 13). The agent receives this numerical reward, which is part of the new experience that it then saves in its replay buffer $\mathcal{B}$ (Lines 14 to 15). It samples X experiences from its buffer that are used to calculate the loss function for updating the DDQN primary network through gradient descent (Lines 16 to 18). On the other hand, the target network is updated using Polyak averaging (Line 19). Lastly, we decrease ϵ if its current value is still higher than ϵ_{end} (Line 20).

Algorithm 4: DDQN for spatial user density-based AP clustering

Initialize the DDQN primary (θ_{prime}) and target (θ_{targ}) networks.
Initialize the ϵ-greedy algorithm $\epsilon \leftarrow \epsilon_{\text{start}}$.
for episode $e = 0, \ldots, E-1$ **do**
  for time step $t = 0, \ldots, T-1$ **do**
    Observe the current state $\mathbf{s}^{(t)}$ (4.16).
    Select an action $a^{(t)}$ (4.17) following (2.14).
    Form the AP groups.
    for random realization $i = 0, \ldots, I-1$ **do**
      Determine $\delta_{k,m}^{(t)}$, $\forall k \in \mathcal{K}, m \in \mathcal{M}$, based on the z largest gains.
      Compute $\rho_{k,m}^{(t)}$, $\forall k \in \mathcal{K}, m \in \mathcal{M}$.
      Compute the network sum rate $u_{\text{sum}}^{(i)}$.
    end for
    Compute the average reward $r^{(t+1)}$ in (4.19).
    Observe the next state $\mathbf{s}^{(t+1)}$.
    Store experience $\{\mathbf{s}^{(t)}, a^{(t)}, r^{(t+1)}, \mathbf{s}^{(t+1)}\}$ in $\mathcal{B}$.
    Sample a random mini-batch of X experiences from $\mathcal{B}$.
    Compute the loss or MSE (2.12) for all the samples.
    Update θ_{prime} by minimizing the loss using gradient descent.
    Update θ_{targ} using Polyak averaging (2.13).
    Update ϵ if $\epsilon > \epsilon_{\text{end}}$ following (2.15).
  end for
end for

4.3.4 Numerical Evaluations

We consider a 300×300 m^2 simulation area, representing a train station scenario. The area is divided into $F = 6$ subregions that serve as train platforms in Fig. 4.6. A platform has five APs, with each having four antennas and a pregenerated spatially correlated shadow fading map [113]. The users in subregion f are Poisson-distributed with mean λ_f. We consider $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6\} = \{60, 60, 20, 10, 10, 10\}$ for the six platforms, respectively. The simulation parameters are listed in Table 4.3.

We refer to our proposed system as *DRL*. The DDQN algorithm employs a fully connected DNN with two hidden layers, having 16 neurons each, and ReLU as the activation function. We evaluate its performance using the following benchmark schemes.

1. *Canonical* – The APs are not clustered in any way. In effect, they belong to a single group.
2. *Small cells (SC)* – Each UE is served by the best AP with the largest gain.
3. *Platform* – The APs on each platform are clustered, creating six static and predefined AP groups.

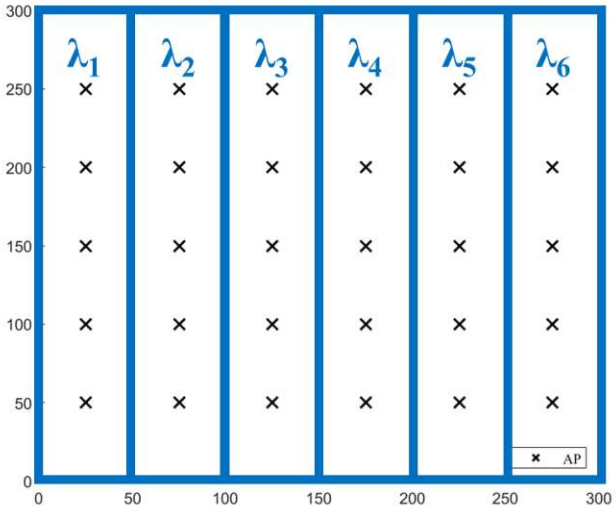


Figure 4.6: Train station scenario with $F = 6$ platforms and $M = 30$ APs.

Table 4.3: Simulation parameters

Parameter	**Value**
Carrier frequency f_c	2 GHz
Bandwidth B	10 MHz
Path loss exponent n_c	2
Shadow fading variance σ_c^2	6
Noise variance σ_z^2	10^{-8}
Per-AP maximum transmit power $\rho_{\max}$	1 W
Number of valid AP groupings v	36
Search size z	5
Buffer size	10000
Mini-batch size X	64
Learning rate α_{step}	0.01
Discount factor γ	0.99
$\epsilon_{\text{start}}, \epsilon_{\text{end}}, \epsilon_{\text{decay}}$	1, 0.0005, 0.0005
Polyak factor τ_{pol}	0.001

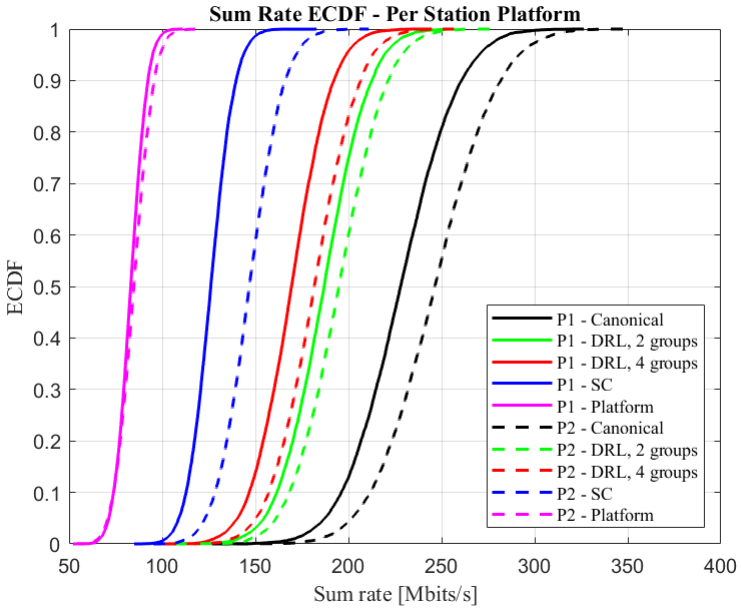
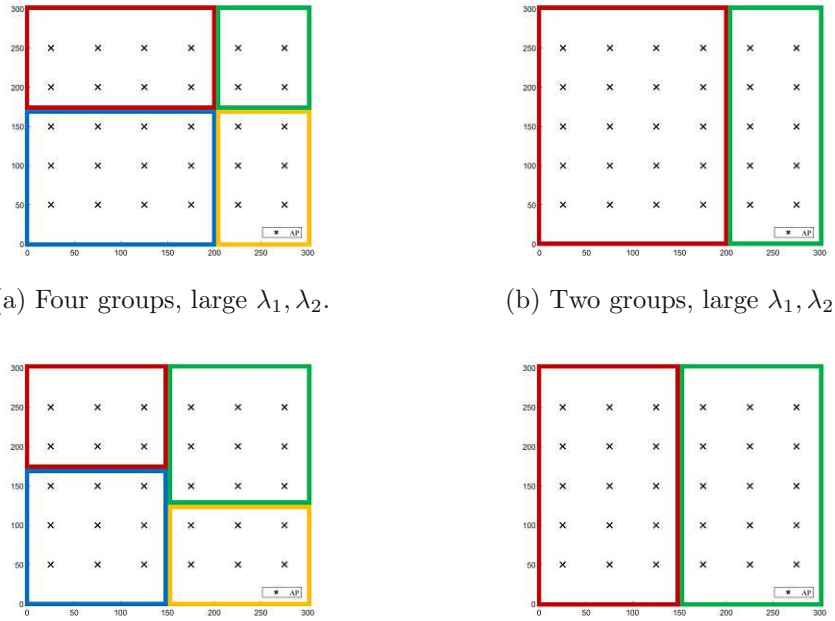


Figure 4.7: Sum rate ECDF (Platforms 1 and 2) for the different schemes.

The empirical cumulative distribution function (ECDF) of the network sum rate for Platforms 1 and 2 (P1 and P2), where majority of the users are located, is shown in Fig. 4.7. Here, we obtain the upper bound performance with the fully centralized *Canonical* baseline by making all the APs available to each UE, resulting in higher individual user rates. This, however, comes with a price, namely longer delays and higher overhead for data co-processing [39]. On the other hand, in distributed architectures such as the proposed *DRL* scheme, these drawbacks are less pronounced, in exchange for a certain degree of performance degradation. For instance, we observe in Fig. 4.7 that forming more AP groups leads to lower rates. Moreover, the performance gap between *Canonical* and *DRL* can be attributed to the restricted design of the discrete action space, where we defined a limited number of AP groupings that the agent can choose from. As motivated in Sec. 4.3.3, we only consider v valid groupings, instead of all the $|\mathcal{G}|^M$ possible combinations. However, this does not guarantee that the optimal grouping is included in the designed action space. Meanwhile, compared to *Small cells*, we achieve better rates with *DRL*, as the users benefit from the joint transmission of multiple distributed antennas or APs. While the *Platform* benchmark also utilizes several antennas, it has the worst performance due to its reliance on static AP groups. In contrast, *DRL* takes advantage of the spatial user density information by allocating more APs to areas with higher user concentration. We also observe that the sum rate of P2 is higher than that of P1 for all schemes. In our simulations, we assume that the users on P2 are served by closer APs on neighboring platforms in both left and right directions (Platforms 1 and 3), while those on P1 are served by more distant APs coming from the same direction (Platforms 2, 3 and so on). Such boundary effects can be reduced by



(a) Four groups, large λ_1, λ_2.

(b) Two groups, large λ_1, λ_2.

(c) Four groups, equal λs.

(d) Two groups, equal λs.

Figure 4.8: Visualization of DRL solution for varying number of AP groups and spatial user densities.

utilizing wraparound techniques in future work [114].

We next illustrate the converged AP clustering solution of *DRL* for different simulated user density values in Fig. 4.8, where the colors represent the AP groups formed. In Fig. 4.8a and 4.8b, we consider the above-mentioned densities, such that $\lambda_1, \lambda_2 \gg \lambda_3, \lambda_4, \lambda_5, \lambda_6$. Since majority of the users are likely to be present on Platforms 1 and 2, *DRL* creates bigger groups in those areas. Allocating more APs to match the anticipated higher number of users leads to better performance. Meanwhile, assuming equal densities for all six platforms in Fig. 4.8c and 4.8d, we see that the agent adapts well by selecting more symmetrical AP groupings to provide uniform performance.

4.4 User-Centric Clustering

In Chapter 3, we had a first look at the scalable user-centric cell-free massive MIMO. While its performance appeared promising, we still relied on conventional iteration-based algorithms. In addition, we considered a cluster size constraint that dictated the number of serving APs per UE. However, this assumption is unrealistic, since the AP-UE propagation conditions differ, which naturally results in clusters of varying sizes. In this section, we propose several SARL frameworks for AP-UE association that consider different optimization objectives without having such size constraint

[50].

4.4.1 Problem Formulation

Our first objective is to keep the individual user-centric clusters as small as possible such that a given user demand is satisfied. This is expressed as

$$\begin{aligned} \text{minimize} \quad & \sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}} \delta_{k,m} && \text{(4.20a)} \\ \text{w.r.t.} \quad & \delta_{k,m} \in \{0,1\}, \\ & 0 \leq \rho_{k,m} \leq \rho_{\max}\delta_{k,m}, \\ \text{subject to} \quad & \sum_{k\in\mathcal{K}} \rho_{k,m}\delta_{k,m} \leq \rho_{\max}, \forall m \in \mathcal{M}, && \text{(4.20b)} \\ & \log_2(1+\text{SINR}_k) \geq \Gamma, \forall k \in \mathcal{K}. && \text{(4.20c)} \end{aligned}$$

Recall that the AP-UE association variable $\delta_{k,m} \in \{0,1\}$ indicates whether or not AP m serves UE k, and $\rho_{k,m}$ is non-zero only if $\delta_{k,m} = 1$. Objective (4.20a) counts these AP-UE connections and minimizes their sum, which in effect, minimizes the fronthaul requirement of the network. Constraint (4.20b) ensures that the total allocated power of AP m does not exceed the maximum transmit power $\rho_{\max}$. Constraint (4.20c) defines the QoS demand Γ in bps/Hz that must be satisfied. In Chapter 3, we proposed an algorithm to find a suboptimal solution to this combinatorial, non-convex problem.

Our second objective is to maximize the sum spectral efficiency of the network as

$$\begin{aligned} \text{maximize} \quad & \sum_{k\in\mathcal{K}} \log_2(1+\text{SINR}_k) && \text{(4.21a)} \\ \text{w.r.t.} \quad & \delta_{k,m} \in \{0,1\}, \\ & 0 \leq \rho_{k,m} \leq \rho_{\max}\delta_{k,m}, \\ \text{subject to} \quad & \sum_{k\in\mathcal{K}} \rho_{k,m}\delta_{k,m} \leq \rho_{\max}, \forall m \in \mathcal{M}. && \text{(4.21b)} \end{aligned}$$

Problem (4.21) is non-convex and NP-hard [109]. Existing schemes find a locally optimal solution to the sum-rate maximization problem [110].

Note that in this work, we assume a single-CPU system, and each UE can be served by any of the APs in the network. In contrast, the proposed scheme in Sec. 4.3.3 grouped the APs associated to the multiple CPUs, and each user was served by only the APs belonging to the same group.

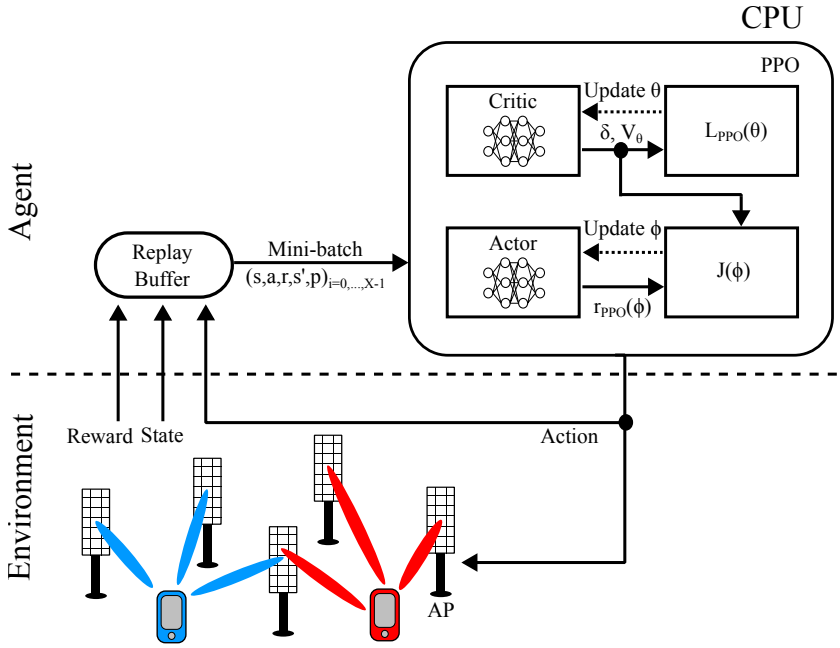


Figure 4.9: SARL-PPO framework for user-centric clustering.

4.4.2 PPO Framework for Minimizing the AP-UE Connections

We now discuss our proposed SARL framework that tackles Problems (4.20) and (4.21). The agent-environment scenario is shown in Fig. 4.9, where the former is represented by the CPU, and the latter includes the APs and UEs. In this case, we consider active APs only, such that $b_m = 1$, $\forall m \in \mathcal{M}$, in Sec. 4.1.

1. State
   The following state vector describes the environment at time t

$$\mathbf{s}^{(t)} = [g_{1,1}^{(t)}, \ldots, g_{K,M}^{(t)}, \mathrm{SE}_1^{(t-1)}, \ldots, \mathrm{SE}_K^{(t-1)}]. \tag{4.22}$$

   The user positions are reflected in the channel gains $g_{k,m}$. The UEs feed back their rate performance through the last K elements of the vector.

2. Action
   The CPU decides which AP-UE links are established, such that the action at time t is written as

$$\mathbf{a}^{(t)} = [\delta_{1,1}^{(t)}, \ldots, \delta_{K,M}^{(t)}]. \tag{4.23}$$

3. Power allocation
   After the connection establishment phase, each AP allocates its transmit power

proportional to the macroscopic channel gains as

$$\rho_{k,m}^{(t)} = \delta_{k,m}^{(t)} \left(\frac{g_{k,m}^{(t)}}{\sum_{j \in \mathcal{K}_m^{(t)}} g_{j,m}^{(t)}} \rho_{\max} \right), \forall k \in \mathcal{K}, m \in \mathcal{M}, \tag{4.24}$$

where $\mathcal{K}_m^{(t)} = \{k \in \mathcal{K} | \delta_{k,m}^{(t)} = 1\}$ is the set of users served by AP m. Power $\rho_{k,m}^{(t)}$ is non-zero only if a link exists between UE k and AP m, such that $\delta_{k,m}^{(t)} = 1$.

Note that while we opted to use a scalable power assignment scheme, it is also possible to employ more sophisticated ones, including network-wide power optimization algorithms, in conjunction with DRL. However, as shown in our previous work [59], this entails higher complexity.

4. Reward
We design two reward functions corresponding to the different optimization goals in Sec. 4.4.1.

In Problem (4.20), we minimize the number of active AP-UE connections while satisfying a given user demand Γ. This translates to the following reward formulation

$$r_{\text{link},1}^{(t+1)} = \begin{cases} KM - \sum_{k \in \mathcal{K}} \sum_{m \in \mathcal{M}} \delta_{k,m}^{(t)}, & \text{if } \min_{k \in \mathcal{K}} (\text{SE}_k^{(t)}) \geq \Gamma \\ 0, & \text{otherwise.} \end{cases} \tag{4.25}$$

The first condition signifies that the agent receives a non-zero reward only if the guaranteed rate of the network is at least Γ bps/Hz. This is equal to the number of inactive links maximized by the agent, which is obtained by taking the difference between the total number of AP-UE links and the actual number of connections established.

We may also maximize the sum spectral efficiency of the network as in Problem (4.21), resulting in the following reward expression

$$r_{\text{link},2}^{(t+1)} = \alpha \overline{\sum_{k \in \mathcal{K}} \text{SE}_k^{(t)}} - (1 - \alpha) \overline{\sum_{k \in \mathcal{K}} \sum_{m \in \mathcal{M}} \delta_{k,m}^{(t)}}. \tag{4.26}$$

In this case, however, we see two contrasting objectives, and thus, we seek to find a balance between them. The first term is the network sum rate that we want to maximize, while the second term represents the link count that we want to minimize. To investigate their trade-off, we introduce a weighting factor $\alpha \in [0, 1]$, which allows us to adapt the system to work on varying operating points. We normalize both terms to ensure a fair comparison. The first term is normalized by the sum rate when considering a canonical setup,

where all APs serve all UEs, while the second one by the maximum number of links KM. A special case is $\alpha = 1$ that simplifies (4.26) to correspond to purely a sum-rate maximization problem.

5. DRL algorithm
   We employ the PPO algorithm, which is suitable for continuous state and discrete action spaces. This is motivated by the ability of PPO to handle a multi-dimensional space of actions. In our case, we have a KM-dimensional action vector, where each element takes a value of either 0 or 1 (i.e., $\delta_{k,m} \in \{0, 1\}$). On the other hand, DDQN only supports one-dimensional spaces, implying that we have to enumerate all 2^{KM} possible AP-UE association combinations in a large action space. In addition, PPO has been shown to perform well while requiring less hyperparameter tuning compared to other state-of-the-art DRL algorithms [102]. We refer to Sec. 2.2.3 for the description of PPO.

4.4.3 PPO Framework for Minimizing the Active APs

The proposed system in Sec. 4.4.2 is geared towards reducing the fronthaul requirement of the network. However, it does not consider the total power consumption, and as such, there is a possibility of operating underutilized APs. For instance, we observed in our initial experiments that some APs remained active even while serving only a single UE or no user at all. With this, we propose an alternative framework that focuses on minimizing the number of active APs to achieve power savings.

1. State
   The state vector is left unchanged and follows (4.22).

2. Action
   The CPU decides the set of active APs, denoted by $\mathcal{M}_{\text{ON}}^{(t)} = \{m \in \mathcal{M} | b_m^{(t)} = 1\}$, through the action vector

$$\mathbf{a}^{(t)} = [b_1^{(t)}, \ldots, b_M^{(t)}]. \tag{4.27}$$

   We consider that each active AP serves all the users, and therefore, $\delta_{k,m}$ is no longer part of the action. We assume $\delta_{k,m}^{(t)} = 1$, $\forall k \in \mathcal{K}$, $m \in \mathcal{M}_{\text{ON}}^{(t)}$.

3. Power allocation
   Each active AP divides its transmit power based on the channel gains as

$$\rho_{k,m}^{(t)} = \frac{g_{k,m}^{(t)}}{\sum\limits_{j \in \mathcal{K}_m^{(t)}} g_{j,m}^{(t)}} \rho_{\max}, \forall k \in \mathcal{K}, m \in \mathcal{M}_{\text{ON}}^{(t)}. \tag{4.28}$$

4. Reward
   While we keep the same objectives discussed in Sec. 4.4.1, we reformulate the original reward functions in (4.25) and (4.26), which aim to minimize the AP-UE link count, to now focus on minimizing the number of active APs.

   As the CPU starts to deactivate more APs, the network may not be able to guarantee at least Γ bps/Hz to all UEs. This violates the constraint in (4.20c), and therefore, we penalize the agent by giving it a zero reward. On the other hand, satisfying the QoS demand leads to a non-zero reward equal to the number of inactive APs, obtained by taking the difference between the total number of APs and the actual number of APs activated. These conditions are expressed as

$$r_{\mathrm{AP},1}^{(t+1)} = \begin{cases} M - \sum\limits_{m \in \mathcal{M}} b_m^{(t)}, & \text{if } \min\limits_{k \in \mathcal{K}} (\mathrm{SE}_k^{(t)}) \geq \Gamma \\ 0, & \text{otherwise.} \end{cases} \tag{4.29}$$

   We may also maximize the network sum rate while attempting to save some power by minimizing the number of active APs with the following reward

$$r_{\mathrm{AP},2}^{(t+1)} = \alpha \overline{\sum_{k \in \mathcal{K}} \mathrm{SE}_k^{(t)}} - (1 - \alpha) \overline{\sum_{m \in \mathcal{M}} b_m^{(t)}}. \tag{4.30}$$

   The first term is normalized by the resulting sum spectral efficiency when all UEs are served by all APs (canonical setup), while the second one by the maximum number of APs M. Their trade-off is controlled by the weighting factor $\alpha \in [0, 1]$.

Algorithm 5 outlines the steps of the proposed frameworks in Sec. 4.4.2 and 4.4.3. Both systems utilize the PPO DRL algorithm, and they mainly differ on the action and reward definitions. We start by initializing the DNNs of the policy and value function (Line 1). For each iteration, we first form a trajectory by collecting experiences for a specified number of time steps using the current policy (Line 3). For each experience, the agent observes the latest channel gains and previous user rate performance. Depending on which framework is operated, it decides either how the individual user-centric clusters are formed or which APs to activate. After applying an action, it obtains a reward that is based on either maximizing the network sum rate or satisfying a given user demand. The new experience is then saved in the replay buffer $\mathcal{B}$ (Lines 5 to 9). This is followed by the computation of the PPO advantage estimate and the return for all time steps using the current value function (Lines 11 to 12). Lastly, we update the DNN parameters by first sampling a mini-batch of experiences from the buffer, and then maximizing the PPO objective while minimizing the PPO loss function over the samples (Lines 14 to 16).

Algorithm 5: PPO for minimizing the AP-UE connections/active APs

Randomly initialize the PPO actor (ϕ) and critic (θ) networks.
for iteration $j = 1, 2, ...$ **do**
  Collect experiences for T time steps using the current policy π_{ϕ_j}.
  for time step t **to** $t + T$ **do**
    Observe $\mathbf{s}^{(t)}$ (4.22), then select and apply $\mathbf{a}^{(t)}$: (4.23) or (4.27).
    Compute $\rho_{k,m}^{(t)}, \forall k, m$: (4.24) or (4.28).
    Compute the reward $r^{(t+1)}$: (4.25), (4.26), (4.29), or (4.30).
    Observe the next state $\mathbf{s}^{(t+1)}$.
    Store the experience $\{\mathbf{s}^{(t)}, \mathbf{a}^{(t)}, r^{(t+1)}, \mathbf{s}^{(t+1)}\}$ in $\mathcal{B}$.
  end for
  Compute the advantage estimate $\hat{A}^{(t)}$ using the current value function V_{θ_j}.
  Compute the return $G^{(t)} = \hat{A}^{(t)} + V_{\theta_j}(\mathbf{s}^{(t)})$.
  for epoch $e = 1$ **to** E **do**
    Sample a random mini-batch of X experiences from $\mathcal{B}$.
    Update ϕ by maximizing the PPO objective (2.21) using gradient ascent.
    Update θ by minimizing the PPO loss (2.25) using gradient descent.
  end for
end for

4.4.4 Numerical Evaluations

We consider a downlink cell-free MIMO network with $M = 10$ APs, having $N = 10$ antennas each, and a varying number of single-antenna UEs $K = \{2, 4, \ldots, 10\}$. All network elements are uniformly distributed over a 300×300 m^2 area. The simulation parameters are summarized in Table 4.4. We refer to the SARL frameworks in Sec. 4.4.2 and 4.4.3 as *minLink* and *minAP*, respectively. The PPO algorithm employs a fully connected DNN with two hidden layers, having 64 neurons each, and the hyperbolic tangent as the activation function.

- User demand
  We first investigate the behavior of the two SARL frameworks when tasked to satisfy a given user demand Γ in Fig. 4.10 and 4.11. We observe that *minLink* significantly outperforms *minAP* in terms of the link count in Fig. 4.10, since the former is designed to achieve the objective with the least number of AP-UE connections possible in (4.25). Take for example the case when $\Gamma = 1$ and $K = 10$, we see that *minAP* requires around 60 connections, while *minLink* sets up only 20 of them, amounting to just one third of the other. Meanwhile, when looking at the number of APs activated for that same example in Fig. 4.11, we observe that *minLink* uses all the 10 APs, while *minAP* only needs 60% to be running, with the latter designed to save power by activating as few

Table 4.4: Simulation parameters

Parameter	Value
Carrier frequency f_c	2 GHz
Path loss exponent n_c	2
Shadow fading variance σ_c^2	6
Noise variance σ_z^2	10^{-8}
Per-AP maximum transmit power $\rho_{\max}$	1 W
Mini-batch size X	64
Learning rate α_{step}	0.0003
Discount factor γ	0.99
Horizon T	2048
Number of epochs E	10
PPO clip parameter ξ_{PPO}	0.2
GAE parameter λ_{GAE}	0.95

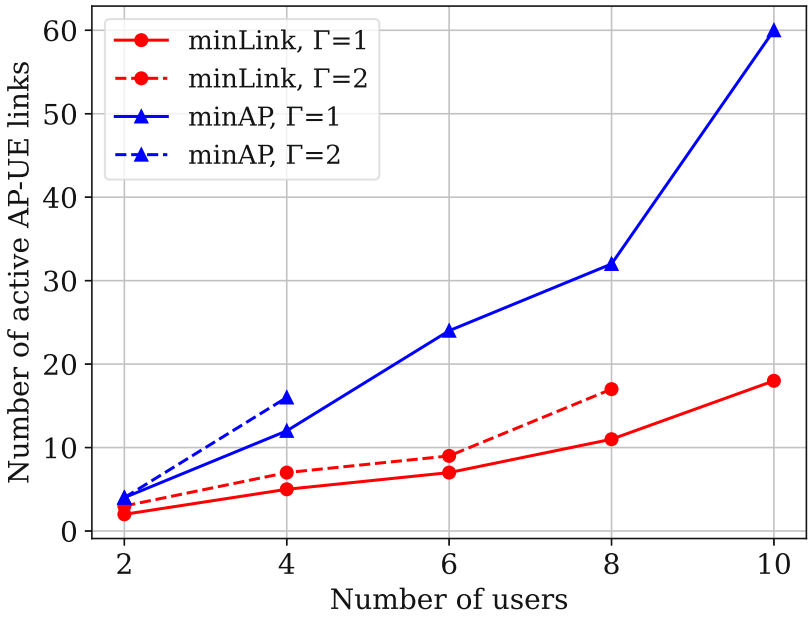


Figure 4.10: Number of AP-UE connections established by minLink and minAP for different Γ values and number of users.

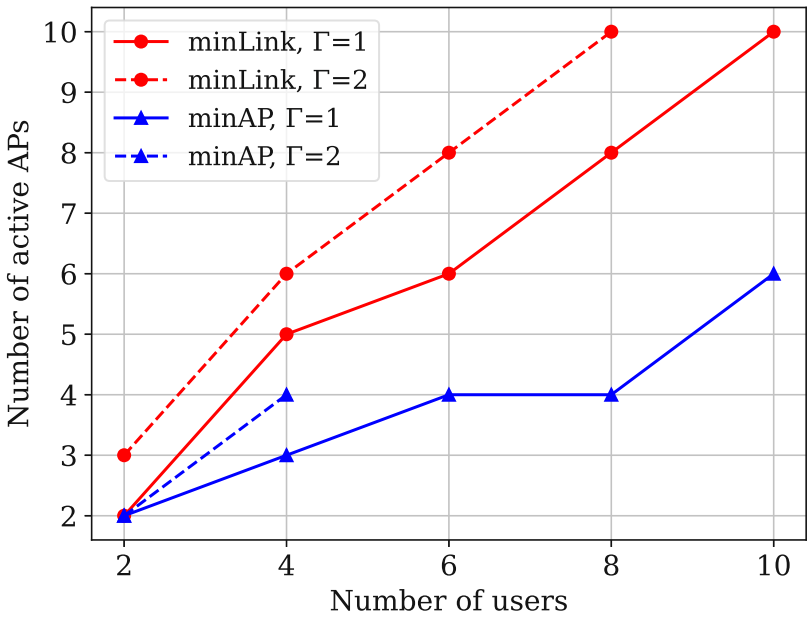


Figure 4.11: Number of APs activated by minLink and minAP for different Γ values and number of users.

APs as possible in (4.29).

When we start increasing the Γ value, we see a corresponding increase in both the number of active links and APs in Fig. 4.10 and 4.11, respectively. This can be explained by the fact that with a stricter QoS constraint, each UE may need more serving APs in its cluster. Consequently, a larger number of APs is activated in the process. Although, after some point, we do not get a feasible solution anymore. This phenomenon occurs faster for *minAP*, as the framework derives the set of active APs to which all the UEs are connected. This means there is little to no flexibility for the users to select the best APs for them, leading to a performance degradation. Note that this happens due to the small network of APs in the simulated scenario, such that adding more APs allows us to obtain a feasible solution even for higher Γ values.

Increasing the number of UEs admitted to the network increases the number of active connections and APs, since a new user automatically establishes at least one additional link to a serving AP. It is worth point out that the number of APs in the user-centric clusters formed by *minLink* varies for the individual users, as this depends on the AP-UE distances and propagation conditions that then impact the UE rate performance.

- Sum spectral efficiency

  We next focus on the performance of the two SARL frameworks when maxi-

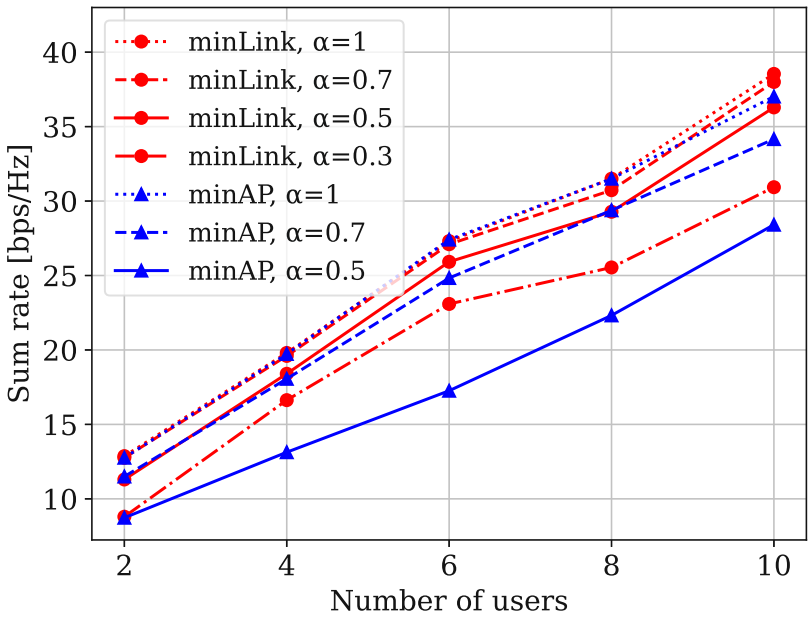


Figure 4.12: Sum rate achieved by minLink and minAP for different α values and number of users.

mizing the network sum rate in Fig. 4.12, 4.13 and 4.14. A general observation is that, as we increase the value of the weighting factor α in rewards (4.26) and (4.30), we see the curves being shifted up in all three figures. The higher the α is, the less penalty the agent receives from either establishing more AP-UE links or activating more APs as long as the sum rate is maximized.

Interestingly, in the case of *minLink* in Fig. 4.12 (in red), the rate improvement gets smaller as we increase α. For instance, the jump in sum spectral efficiency when changing α from 0.3 to 0.5 is more pronounced than when going from 0.5 to 0.7. Recall that we designed our reward to allow us to adapt our system according to some performance target. For a fronthaul-constrained network, this implies putting a limit on the number of AP-UE connections established, which can be easily done by setting an appropriately small value for α in (4.26), as depicted in Fig. 4.13. When such limit on the AP-UE link count exists, the DRL framework is able to identify which links significantly contribute to the user rate performance, and those are prioritized when actually establishing a relatively small number of AP-UE connections. This explains why increasing α further does not impact the sum rate that much anymore.

In the case of *minAP* in Fig. 4.12 (in blue), we observe that configuring α to 0.7 gives a performance that is close to that of $\alpha = 1$, with the latter being 1 to 2 bps/Hz better. However, looking at Fig. 4.14, this gap is at the expense of having all the APs active all the time, as opposed to only 60 to 80% of

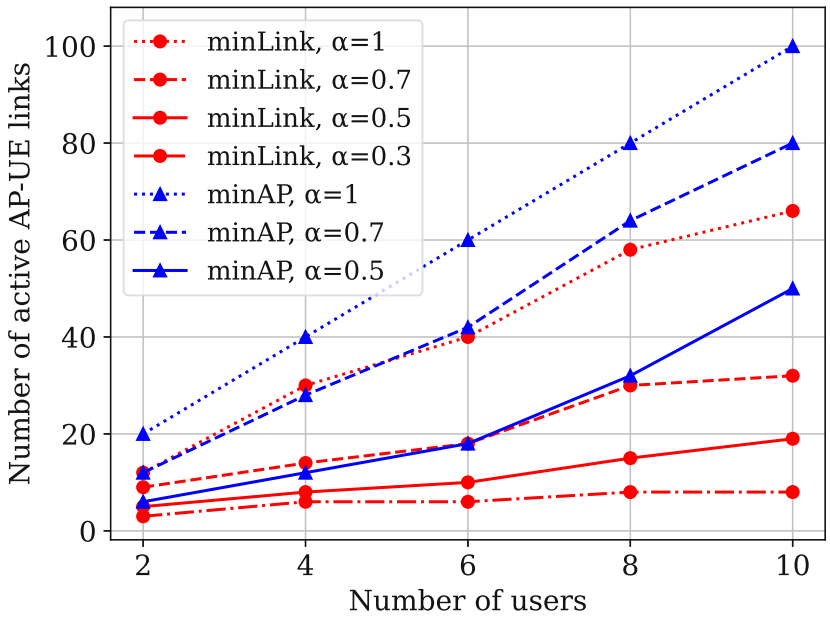


Figure 4.13: Number of AP-UE connections established by minLink and minAP for different α values and number of users.

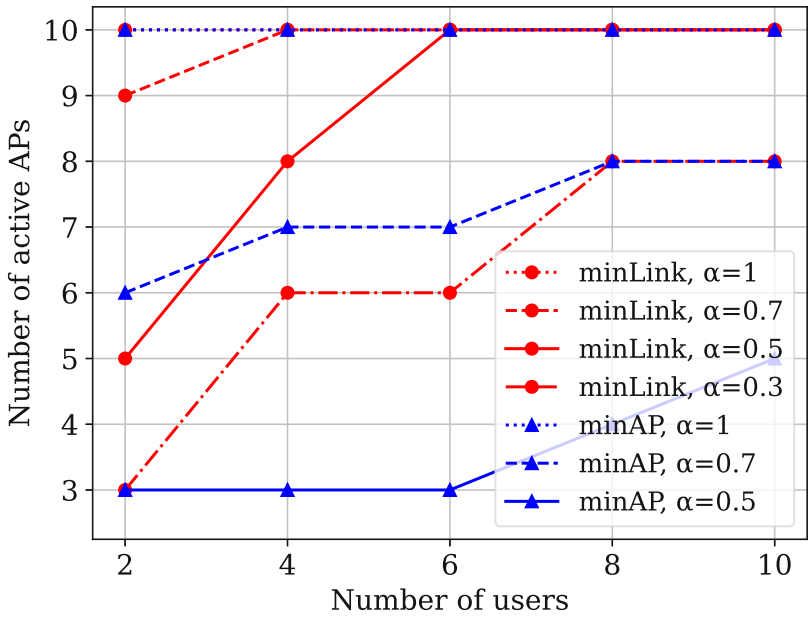


Figure 4.14: Number of APs activated by minLink and minAP for different α values and number of users.

them when $\alpha = 0.7$. This demonstrates the ability of our DRL framework to decrease power consumption without substantially losing on rate performance.

- Comparison with baseline schemes

  We now compare our proposed methods with known benchmarks in the context of cell-free massive MIMO. It was shown in [49] that the canonical setup acts as an upper bound for evaluating performance. As depicted in Fig. 4.14, setting α to 1 in reward (4.30) of *minAP* activates all the APs, which then serve all the users in the network. This, therefore, represents the canonical baseline. Our simulation results in Fig. 4.12 and 4.14 suggest that for appropriate α values, the system behaves close to the upper bound while only activating a subset of APs, improving energy efficiency.

  When $\alpha = 1$ for *minLink*, reward (4.26) is transformed into purely a sum-rate maximization problem, giving us the highest sum spectral efficiency attained by the framework. In Fig. 4.12 and 4.13, we observe that by configuring α properly, we can achieve the same rate performance as the canonical benchmark with much fewer AP-UE connections. By optimizing the AP selection for each user or cluster, the user-centric design effectively outperforms the canonical setup. This translates to lower fronthaul requirements for the network and contributes to making it scalable.

  Another baseline to compare with is the seminal work [27] on the user-centric concept, which assumes that the clusters are of fixed size N_c. A larger N_c implies that there are more serving APs in each cluster, resulting in higher user rates, and thus, narrows the performance gap with our DRL framework. However, assuming the same cluster size for all UEs is inefficient in terms of the fronthaul capacity, since the users require varying number of serving APs in practice.

4.5 Summary

In this chapter, we started to leverage DRL for network optimization. We formulated various optimization problems and proposed SARL-based solutions, which are geared towards the realization of energy-efficient, scalable cell-free massive MIMO.

The mobile data traffic fluctuates throughout the day, and thus, some APs are underutilized during off-peak hours. We developed a DRL approach that selects the best APs to activate given the current user position information. We highlight that the system jointly considers the position of all users, which avoids having an AP switched on even when only a single UE is connected to it. Moreover, we designed our framework to be flexible enough to accommodate different operating points in terms of the guaranteed QoS and power consumption. When we prioritized maximizing the minimum user SINR, we forced more APs to be activated, which increased power consumption. Our simulation results demonstrated that by carefully choosing the

limited number of APs to operate, we were able to obtain almost the same rate performance as when all APs are active while achieving power savings.

The network scalability issue of the canonical cell-free network was tackled by exploring SARL-based clustering algorithms. In practice, multiple CPUs are expected to cover different subareas. Given a predefined number of CPUs, we presented a DRL system to cluster or group the APs associated to the same CPU. In order to determine the size of each AP group, we relied on the current spatial user density information of the subregions. This allowed us to form bigger AP groups in areas where there are likely to be more users. Our numerical evaluations showed that tailoring the group sizes as such led to a higher network sum rate.

We next focused on DRL-based user-centric clustering. Assuming a practical fronthaul-constrained network, we either maximized the sum spectral efficiency or satisfied a QoS demand, while forming the least number of AP-UE links possible. Our numerical experiments proved that the framework is capable of determining the best APs per user that enables us to stay close to the upper bound performance, exhibited by the canonical setup, but with much lower fronthaul requirement. In fact, when we allowed more AP-UE connections to be established by increasing α, we observed insignificant rate improvements, suggesting that only a small number of APs contributed to performance. This supports our findings in Chapter 3. However, instead of utilizing an iteration-based approach that is commonly of high computational complexity, as well as imposing a cluster size constraint, we relied on DRL to intelligently form the user-centric clusters of varying sizes.

5

Accelerated SARL and MARL for Power Control

In Chapter 4, we applied SARL to approximately solve different non-convex optimization problems in cell-free massive MIMO, which would have been rather complex in the case of traditional optimization techniques. While our findings align with our goal of improving the scalability and adaptability of the network, we have mostly limited our experiments to static scenarios. This assumption is unrealistic, given that the wireless environment changes over time, and it does not harness the full potential of DRL. Moreover, we relied on a single agent in all our prior DRL frameworks. A direct consequence of this is that the size of the action space is too large for certain DRL algorithms, making it infeasible for most frameworks to handle large-scale network simulations.

In this chapter, we move away from the above-mentioned limitations. Unlike existing works that only considered user mobility, we assume a dynamic cell-free environment that is characterized by the combination of UE (de-)activation and mobility. Our main motivation for tackling such challenging scenario is its relevance to practical use cases, including those of internet of things (IoT), where battery-limited user devices switch to sleep mode in order to conserve power and prolong their battery life. Furthermore, when dealing with a non-static environment, we inevitably face the issue of slow convergence in DRL [40]. Specifically, the system enters a re-training phase whenever the environment changes. A slow convergence implies that by the time the re-trained model converges, the environment may have changed again, leading to an outdated DNN policy selecting bad actions that then impact performance. Thus, in this chapter, we utilize a technique known as prioritized experience reply [81] in an attempt to accelerate learning. In addition, we transition to distributed DRL architectures by exploring both SARL and MARL, including federated learning [41], to solve an uplink power control problem that maximizes the guaranteed rate of the cell-free network. The algorithms developed in this chapter have been published in [83, 84].

5.1 Uplink System Model

We consider an uplink cell-free massive MIMO network with M single-antenna APs and K single-antenna UEs, such that $M \gg K$. The set of all APs is denoted by $\mathcal{M} = \{1, \ldots, M\}$, the set of all UEs by $\mathcal{K} = \{1, \ldots, K\}$, and the set of active UEs by $\mathcal{K}_{\text{ON}} \subseteq \mathcal{K}$. We assume i.i.d. Rayleigh fading as in (3.1). Given a block-fading channel, each coherence block contains $\tau_c = \tau_p + \tau_u + \tau_d$ samples, of which τ_p are for uplink training, τ_u are for uplink data transmission, and τ_d are for downlink data transmission.

In the uplink training phase, all active UEs transmit their pilot sequences simultaneously. The pilot sequence of UE k is denoted by $\boldsymbol{\phi}_k \in \mathbb{C}^{\tau_p \times 1}$. The APs obtain a superposition of the sequences. The $(\tau_p \times 1)$-pilot signal received by AP m is

$$\mathbf{y}_m = \sum_{k \in \mathcal{K}_{\text{ON}}} \sqrt{\tau_p \rho_p} h_{k,m} \boldsymbol{\phi}_k + \mathbf{n}_m, \tag{5.1}$$

where ρ_p is the pilot transmit power, and $\mathbf{n}_m$ is the noise vector with i.i.d. elements following $\mathcal{CN}(0, \sigma_n^2)$.

In order to estimate the user channel, AP m projects the pilot signal onto $\boldsymbol{\phi}_k^H$ as

$$\hat{y}_{k,m} = \boldsymbol{\phi}_k^H \mathbf{y}_m = \sum_{k' \in \mathcal{V}_k} \sqrt{\tau_p \rho_p} h_{k',m} + \boldsymbol{\phi}_k^H \mathbf{n}_m, \tag{5.2}$$

where $\mathcal{V}_k$ represents the set of active users utilizing the same pilot sequence $\boldsymbol{\phi}_k$. Here, we consider random pilot allocation. Note, however, that different pilot assignment schemes have been presented in the literature to mitigate pilot contamination effects [21, 22]. The MMSE channel estimate [14, 80] is then

$$\hat{h}_{k,m} = \frac{\sqrt{\tau_p \rho_p} g_{k,m}}{\sum_{k' \in \mathcal{V}_k} \tau_p \rho_p g_{k',m} + \sigma_n^2} \hat{y}_{k,m} = c_{k,m} \, \hat{y}_{k,m}, \tag{5.3}$$

with $\hat{h}_{k,m} \sim \mathcal{CN}(0, V_{k,m})$ and

$$V_{k,m} = \frac{\tau_p \rho_p g_{k,m}^2}{\sum_{k' \in \mathcal{V}_k} \tau_p \rho_p g_{k',m} + \sigma_n^2}. \tag{5.4}$$

During uplink data transmission, all active UEs send their data simultaneously. The data symbol of UE k is x_k with power $\mathbb{E}\{|x_k|^2\} = 1$. The APs receive a superposition of the symbols. In the case of AP m, this is given by

$$y_m^{(u)} = \sum_{k \in \mathcal{K}_{\text{ON}}} h_{k,m} \sqrt{\rho_k} x_k + n_m^{(u)}, \tag{5.5}$$

where ρ_k is the uplink transmit power of UE k, and $n_m^{(u)}$ is the noise at AP m.

Access point m utilizes its local channel estimate $\hat{h}_{k,m}$ to obtain $\hat{h}_{k,m}^* y_m^{(u)}$, which is then forwarded to the CPU for data detection. The received signal at the CPU is

$$\begin{aligned} z_k^{(u)} &= \sum_{m\in\mathcal{M}} \hat{h}_{k,m}^* y_m^{(u)} \\ &= \underbrace{\sum_{m\in\mathcal{M}} \hat{h}_{k,m}^* h_{k,m}\sqrt{\rho_k} x_k}_{\text{desired signal}} + \underbrace{\sum_{m\in\mathcal{M}} \sum_{\substack{k'\in\mathcal{K}_{\text{ON}} \\ k'\neq k}} \hat{h}_{k,m}^* h_{k',m}\sqrt{\rho_{k'}} x_{k'}}_{\text{inter-user interference}} + \underbrace{\sum_{m\in\mathcal{M}} \hat{h}_{k,m}^* n_m^{(u)}}_{\text{noise}}. \end{aligned} \tag{5.6}$$

The SINR of UE k is expressed as

$$\begin{aligned} \text{SINR}_k &= \frac{\mathbb{E}\left\{\left|\sum_{m\in\mathcal{M}} \hat{h}_{k,m}^* h_{k,m}\sqrt{\rho_k} x_k\right|^2\right\}}{\mathbb{E}\left\{\left|\sum_{m\in\mathcal{M}} \sum_{\substack{k'\in\mathcal{K}_{\text{ON}} \\ k'\neq k}} \hat{h}_{k,m}^* h_{k',m}\sqrt{\rho_{k'}} x_{k'}\right|^2\right\} + \mathbb{E}\left\{\left|\sum_{m\in\mathcal{M}} \hat{h}_{k,m}^* n_m^{(u)}\right|^2\right\}} \\ &= \frac{\rho_k \left(\sum_{m\in\mathcal{M}} V_{k,m}\right)^2 + \sum_{m\in\mathcal{M}} \rho_k V_{k,m} g_{k,m}}{\tau_p \sum_{k'\in\mathcal{V}_k} \rho_{k'}\rho_p \left(\sum_{m\in\mathcal{M}} c_{k,m} g_{k',m}\right)^2 + \sum_{\substack{k'\in\mathcal{K}_{\text{ON}} \\ k'\neq k}} \rho_{k'} \sum_{m\in\mathcal{M}} g_{k',m} V_{k,m} + \sum_{m\in\mathcal{M}} V_{k,m}\sigma_n^2}, \end{aligned} \tag{5.7}$$

which is based on the power of the three terms in (5.6) following the analysis in [14, 80]. The rate of UE k in bps is

$$u_k \approx B\left(1 - \frac{\tau_p}{\tau_c}\right)\log_2(1 + \text{SINR}_k), \tag{5.8}$$

where B is the system bandwidth in Hz.

This system model is more accurate than the one described in Sec. 3.1, as it accounts for channel estimation and pilot contamination. However, we also emphasize that one of the advantages of DRL is its ability to operate online and model-free. In fact, the presented system model is for evaluation purposes only, meaning, the user rate feedback in our simulations is implemented following this model. In a real-world deployment, our proposed frameworks would consider the actual achieved UE rates, such that they are agnostic to the system model in use. This also implies that they would be able to take into account several practical factors affecting the rate calculation, which are not included in most models.

In this chapter, we consider dynamic device (de-)activation. We refer to the changing of UE ON/OFF states as UE toggling with parameters T_{tog} and K_{tog}. The number of episodes over which the ON/OFF status remains constant is specified by T_{tog}, and the number of UEs that switch from active to inactive mode (or vice versa) every T_{tog} is indicated by K_{tog}.

5.2 Problem Formulation

Inter-user interference may lead to severe performance degradation, highlighting the need for effective power control strategies. We, therefore, formulate the following uplink power control problem that maximizes the guaranteed rate of the cell-free network

$$\begin{aligned} \text{maximize} \quad & \min_{k \in \mathcal{K}_{\text{ON}}} u_k && \text{(5.9a)} \\ \text{w.r.t.} \quad & \rho_k, \\ \text{subject to} \quad & 0 \leq \rho_k \leq \rho_{\max}, \forall k \in \mathcal{K}_{\text{ON}}. && \text{(5.9b)} \end{aligned}$$

The max-min objective is expressed in (5.9a). Constraint (5.9b) specifies the valid range for the power values, where $\rho_{\max}$ is the maximum transmit power that we assume to be the same for all UEs.

For the considered single-antenna setup, Problem (5.9) is equivalent to maximizing the minimum user SINR, which after some reformulation, can be solved centrally using conventional iteration-based optimization algorithms, including those presented in [14, 115]. However, in this chapter, we add another layer of complexity by considering device (de-)activation. This is represented by set $\mathcal{K}_{\text{ON}}$ in (5.9) that consists of the currently active UEs only. In this case, existing algorithms require knowing the precise UE ON/OFF states in advance before being able to carry out network-wide power allocation. However, this is non-trivial, as the unknown and dynamic UE activation patterns depend on several factors, such as battery life and event-triggered updates of IoT sensor devices, which are challenging to predict and may change rapidly. In addition, the said algorithms are commonly of high computational complexity and time-consuming to implement in practice. Motivated by this, we leverage model-free DRL to solve the uplink power control problem in an online manner, without requiring prior knowledge of the UE activation status, by relying solely on UE rate feedback for user privacy preservation and communication overhead reduction.

5.3 Prioritized Experience Replay

Deep reinforcement learning is known to suffer from slow convergence that makes it less competitive as an optimization tool in practical deployments. This shortcoming

is magnified when dealing with a dynamic environment, where re-training is necessary whenever changes in the environment are detected. In this section, we discuss prioritized experience replay [81] or prioritized sampling, which is a technique to accelerate convergence. By default, DRL algorithms assume a uniform distribution when sampling the mini-batch of experiences for updating their DNNs, as detailed in Sec. 2.2.3. In the case of prioritized sampling, we instead give more importance to certain experiences to help the agent learn faster. While there exist different variants of prioritized experience replay, we specifically utilize TD error-based prioritization, since the TD errors are automatically computed when updating the DRL algorithms that we employ in this chapter, namely DDPG and DDQN. Therefore, no additional computational complexity is incurred in the process.

Recall that at each time step, a new experience is saved by the agent in its buffer $\mathcal{B}$. An experience is represented by the tuple (s, a, r, s'). With prioritized sampling, we extend this to include a priority information p. Thus, the ith modified experience tuple stored in $\mathcal{B}$ is given by $(s_i, a_i, r_i, s'_i, p_i)$. The priority value p_i is transformed into the actual probability $P(i) \in [0, 1]$ used during sampling as

$$P(i) = \frac{p_i^\alpha}{\sum_{j=0}^{\text{len}(\mathcal{B})-1} p_j^\alpha}, \tag{5.10}$$

where $\text{len}(\mathcal{B})$ is the current length of the buffer, and $\alpha \in [0, 1]$ is the prioritization factor that controls how much we rely on prioritization. The higher the α is, the greater the degree of prioritization, while plugging $\alpha = 0$ into (5.10) corresponds to the default case of uniform sampling. We see from the above relationship that the priority value dictates how likely an experience will be selected for replay. Therefore, a new experience is always assigned the current maximum priority $p_{\max}$ to increase its chances of being sampled.

The priorities are obtained from the TD errors as

$$p_i = |\delta_{\text{TD},i}| + \varsigma, \tag{5.11}$$

where

$$\delta_{\text{TD},i} = y_{\text{targ},i} - Q(\mathbf{s}_i, a_i; \theta^{(t)}_{\text{prime}}) \tag{5.12}$$

in (2.8) of Sec. 2.2.2, and $\varsigma > 0$ is a small number to avoid dividing by zero in (5.10). In DRL, we aim to minimize the TD error. Thus, we prioritize those experiences with large δ_{TD}, such that they are more likely to be drawn frequently, implying more attempts of minimizing the TD error.

A consequence for implementing such prioritization mechanism is that we introduce a distribution bias that can be compensated and corrected by assigning

importance sampling weights to the saved experiences, expressed as

$$w_i = \left(\frac{1}{\text{len}(\mathcal{B}) \cdot P(i)} \right)^{\beta^{(t)}}, \tag{5.13}$$

where $\beta \in [0, 1]$ is a correction parameter, with $\beta = 0$ corresponding to the case where no correction is made. We anneal β over N_{ts} time steps during the learning process as

$$\beta^{(t+1)} = \beta^{(t)} + \frac{\beta_{\text{end}} - \beta_{\text{start}}}{N_{\text{ts}}}, \tag{5.14}$$

where we initially set β to β_{start} and gradually increase its value until it reaches β_{end}. We note the inverse relationship between the probabilities and weights in (5.13). Since experiences with higher priority and probability are likely to be oversampled, we correct the distribution bias by downweighting them. Lastly, in order to ensure that the weights are within the valid range $w_i \in [0, 1]$, we also normalize them as

$$w_i = \frac{w_i}{\max_j w_j}. \tag{5.15}$$

These weights appear in the following modified loss function

$$L_{\text{priority}} = \frac{1}{X} \sum_{i=0}^{X-1} w_i (y_{\text{targ},i} - Q(\mathbf{s}_i, a_i; \theta^{(t)}_{\text{prime}}))^2, \tag{5.16}$$

which is used for updating the DDPG and DDQN algorithms.

5.4 DDPG with Prioritized Sampling for Power Control

In this section, we present our first SARL framework for uplink power control, which we equip with TD error-based prioritized experience replay to enable our system to quickly adapt to the dynamic and unknown UE activation patterns [83]. The agent-environment scenario is depicted in Fig. 5.1. The CPU acts as the agent that houses the DDPG algorithm and prioritized sampling mechanism, while the APs and users belong to the environment.

5.4.1 SARL-DDPG Framework

- DRL components

  1. State
     The (de-)activation activity of the UEs is monitored by the agent through

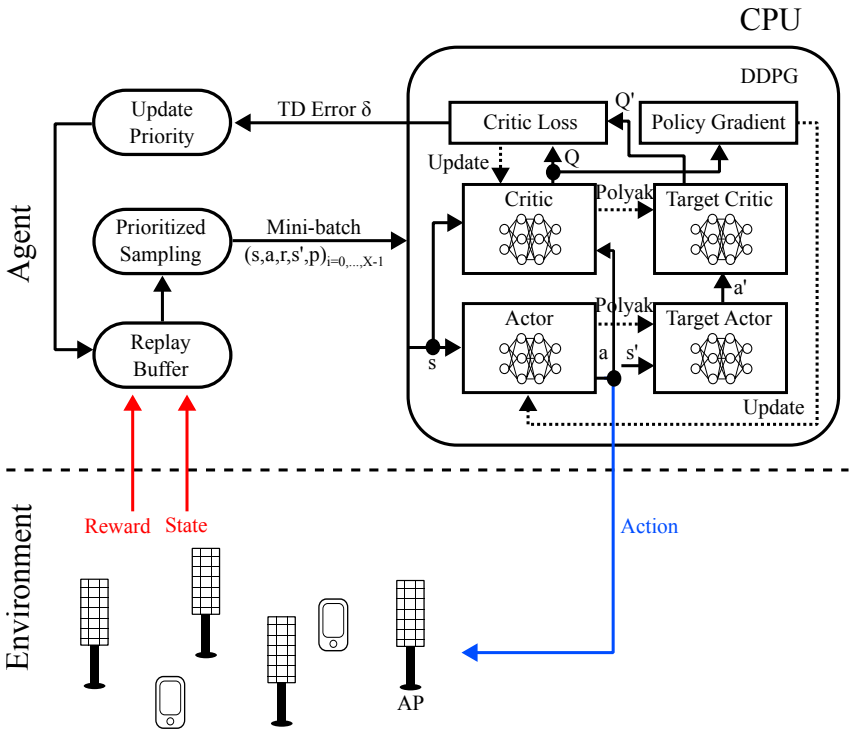


Figure 5.1: SARL-DDPG with prioritized sampling framework for power control.

the state vector

$$\mathbf{s}^{(t)} = [d_1^{(t)}, \ldots, d_K^{(t)}, u_1^{(t-1)}, \ldots, u_K^{(t-1)}], \tag{5.17}$$

where the ON/OFF status of UE k at time t is given by $d_k^{(t)} \in \{0, 1\}$. The user rate performance at the previous time step $t-1$ is indicated by the last K vector elements.

2. Action
The CPU decides the uplink transmit power of all K UEs as

$$\mathbf{a}^{(t)} = [\rho_1^{(t)}, \ldots, \rho_K^{(t)}]. \tag{5.18}$$

We ensure that the power constraint is satisfied by allocating non-zero power within range $\xi \leq \rho_k^{(t)} \leq \rho_{\max}$, with $0 < \xi \ll 1$, to the active UEs and assigning zero power to the inactive ones. We implement a post-correction phase for the latter in case the agent selects a non-zero power for an inactive UE.

3. Reward
The agent receives the following reward that it aims to maximize, corre-

sponding to the guaranteed rate in Problem (5.9),

$$r^{(t+1)} = \min_{k' \in \mathcal{K}_{\text{ON}}^{(t)}} u_{k'}^{(t)}. \tag{5.19}$$

Here, we define the set of active UEs as $\mathcal{K}_{\text{ON}}^{(t)} = \{k \in \mathcal{K} | d_k^{(t)} = 1\}$.

- Proposed SARL-DDPG algorithm
  Given that our problem deals with continuous state and action spaces, we utilize the DDPG algorithm [98], presented in Sec. 2.2.3. Unlike its vanilla form that uniformly samples the experiences used for updating its DNNs, we augment it with prioritized sampling [81]. We refer to Sec. 5.3 for the description of prioritized experience replay. Algorithm 6 summarizes the procedure of the proposed method, which we detail below.
  Step 1: We start with the initialization phase. We set up the DDPG primary actor and critic networks, and configure the same DNN parameters for the corresponding target networks. We next initialize the prioritized sampling mechanism and the noise for action exploration (Lines 1 to 4).
  Step 2: The CPU observes the latest device activation status and the user performance at the previous time step. Based on this information, it decides the non-zero power values for the active UEs. This triggers a reward feedback from the environment, which transitions into a new state at the next time step (Lines 5 to 9).
  Step 3: A basic tuple $(\mathbf{s}^{(t)}, \mathbf{a}^{(t)}, r^{(t+1)}, \mathbf{s}^{(t+1)})$ for the new experience is created. Different from the DRL frameworks that we have seen so far, the prioritized experience replay takes into effect by extending the tuple and attaching a priority information p, which is equal to the current maximum priority $p_{\max}$. The agent saves the modified tuple $(\mathbf{s}^{(t)}, \mathbf{a}^{(t)}, r^{(t+1)}, \mathbf{s}^{(t+1)}, p_{\max})$ in its replay buffer $\mathcal{B}$ (Line 10).
  Step 4: The priority values of the experiences in $\mathcal{B}$ are converted to probabilities following (5.10), which serve as a basis for calculating the importance sampling weights in (5.13) (Lines 11 to 12).
  Step 5: The agent now samples a mini-batch of X experiences using the probabilities associated to the stored experiences in $\mathcal{B}$. For each sample, the TD error δ_{TD} is determined. As the computation differs according to the DRL algorithm in use, we recall that the TD error for DDPG is

$$\delta_{\text{TD}}^{(t)} = y_{\text{targ}}^{(t)} - Q(\mathbf{s}^{(t)}, \mathbf{a}^{(t)}; \theta_{\text{prime}}^{(t)}),$$

  where

$$y_{\text{targ}}^{(t)} = r^{(t+1)} + \gamma Q'(\mathbf{s}^{(t+1)}, \mu'(\mathbf{s}^{(t+1)}; \phi_{\text{targ}}^{(t)}); \theta_{\text{targ}}^{(t)}).$$

  The target actor and critic networks are used to calculate the target value y_{targ}. The weighted loss in (5.16) is computed based on the TD errors of the X samples and is minimized using gradient descent to update the primary critic

Algorithm 6: SARL-DDPG with prioritized sampling for power control

1: Initialize the DDPG primary actor $\mu(\mathbf{s}|\phi_{\text{prime}})$ and critic $Q(\mathbf{s},\mathbf{a}|\theta_{\text{prime}})$ with weights ϕ_{prime} and θ_{prime}. Initialize the target networks μ' and Q' with weights $\phi_{\text{targ}} \leftarrow \phi_{\text{prime}}$ and $\theta_{\text{targ}} \leftarrow \theta_{\text{prime}}$.
2: Initialize the prioritized experience replay parameters $\beta \leftarrow \beta_{\text{start}}$ and $p_{\max} \leftarrow 0$.
3: **for** episode $e = 0, \ldots, E-1$ **do**
4: Initialize a random process $\mathcal{N}$ for action exploration.
5: Initialize state $\mathbf{s}^{(0)}$.
6: **for** time step $t = 0, \ldots, T-1$ **do**
7: Observe the current state $\mathbf{s}^{(t)}$ (5.17).
8: Select and apply action $\mathbf{a}^{(t)} \leftarrow \mu(\mathbf{s}^{(t)}|\phi_{\text{prime}}) + \mathcal{N}^{(t)}$ (5.18).
9: Observe the reward $r^{(t+1)}$ (5.19) and next state $\mathbf{s}^{(t+1)}$.
10: Store experience $(\mathbf{s}^{(t)}, \mathbf{a}^{(t)}, r^{(t+1)}, \mathbf{s}^{(t+1)}, p_{\max})$ in $\mathcal{B}$.
11: Compute the probabilities (5.10) of the experiences.
12: Compute their weights (5.13), (5.15).
13: Sample a mini-batch of X experiences from $\mathcal{B}$ based on the calculated probabilities.
14: Compute their TD error (5.12) and the resulting weighted loss (5.16).
15: Update the critic by minimizing the loss using gradient descent.
16: Update the priorities (5.11) of the samples.
17: Update $p_{\max} \leftarrow \max_{j \in \{0,\ldots,\text{len}(\mathcal{B})\}} p_j$.
18: Update the actor by maximizing the gradient (2.16) using gradient ascent.
19: Update the target networks using Polyak averaging (2.19), (2.20).
20: Update β as in (5.14).
21: **end for**
22: **end for**

(Lines 13 to 15).
Step 6: The newly calculated TD errors are utilized to update the priorities of the sampled experiences as in (5.11). If follows that we also update the current maximum priority $p_{\max}$ (Lines 16 to 17).
Step 7: Similar to vanilla DDPG, we update the DNN parameter of the primary actor by maximizing the DDPG gradient in (2.16). After which, we perform soft updates for the two target networks using Polyak averaging (2.19), (2.20) (Lines 18 to 19).
Step 8: We update the prioritized sampling correction parameter by gradually increasing β following (5.14) (Line 20).

Table 5.1: Simulation parameters

Parameter	Value
Carrier frequency f_c	1.9 GHz
Bandwidth B	20 MHz
Path loss exponent n_c	2
Shadow fading standard deviation	8 dB
Noise figure	9 dB
Pilot transmit power ρ_p	0.1 W
Per-UE maximum transmit power $\rho_{\max}$	0.1 W
Buffer size	$1e6$
Mini-batch size X	100
Learning rate for actor, critic α_{step}	0.001
Discount rate γ	0.9
Polyak factor τ_{pol}	0.005
Prioritization factor α	0.5
Correction parameter β_{start}, β_{end}	0.4, 1

5.4.2 Numerical Evaluations

We consider an uplink cell-free MIMO network with $M = 30$ single-antenna APs and $K = 10$ single-antenna UEs. All network elements are uniformly distributed over a 500×500 m^2 area. The simulation parameters are listed in Table 5.1.

We refer to our proposed SARL-DDPG with prioritized sampling framework in Sec. 5.4.1 as *Prop. DDPG-PS*. We consider the same DNN architecture for both the actor and critic networks, where we utilize a fully connected DNN with two hidden layers, having 64 neurons each. We experimented with different prioritization factor α values to determine the most appropriate one for our system. We achieved the best performance with $\alpha = 0.5$, and we, therefore, use this value in our analysis.

We evaluate the performance of *Prop. DDPG-PS* using the following benchmark schemes.

1. *Ref. Max-min* – Problem (5.9) is solved as in [14].
2. *Ref. DDPG-Uni* – Vanilla DDPG employs uniform sampling as in [98].
3. *Ref. Full power* – Each UE transmits with $\rho_{\max}$.

Furthermore, we test the robustness of our framework using various combinations of UE toggling and user mobility settings. The mobile scenario follows a random walk model, where each user selects a direction (left, right, up, down) and a speed from 0 to 1 m/s at each time step, assuming a uniform distribution.

- Fully static scenario
  We start with the fully static case, where the users are non-mobile, and the

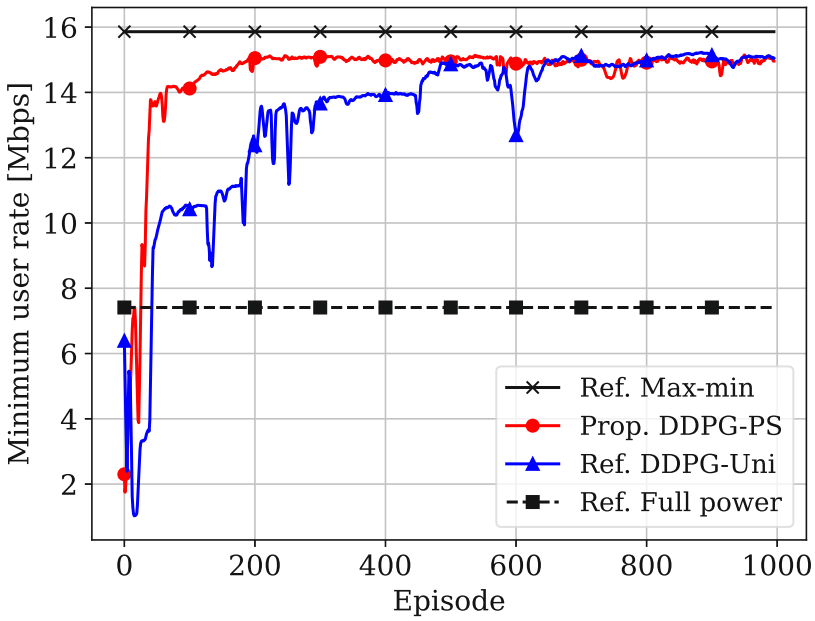


Figure 5.2: Guaranteed rate for the fully static scenario.

UE ON/OFF states remain constant throughout the learning process. The guaranteed rate for the different schemes is depicted in Fig. 5.2. The *Ref. Max-min* baseline provides the upper bound performance. However, it requires knowing the exact UE activation patterns in advance to solve Problem (5.9) using conventional optimization techniques, which is challenging in practice. We observe that our *Prop. DDPG-PS* framework behaves close to optimal, with only a difference of 0.9 Mbps or 5.66%. More importantly, it does so without prior knowledge of the UE ON/OFF states by relying solely on the user rate feedback. The *Ref. DDPG-Uni* benchmark converges to the same rate of 15 Mbps, albeit at a slower pace, which is 300 episodes later compared to *Prop. DDPG-PS*. The worst-performing scheme is *Ref. Full power* due to the increased inter-user interference.

Fig. 5.3 shows the total power consumption for the different setups, obtained by summing up the powers of the active UEs. In the case of the DDPG-based schemes, power consumption goes down while rate improves in Fig. 5.2 during training. This implies that the agent learns to select better power values or actions as it further interacts with the environment by observing the resulting rewards. Moreover, while power reduction is not explicit in the reward function in (5.19), it is accounted for when maximizing the guaranteed rate, as inter-user interference is minimized in the process. In addition, we observe that as opposed to vanilla DDPG, applying prioritization enables our proposed scheme to reach *Ref. Max-min* faster at episode 200. As expected, *Ref. Full power*

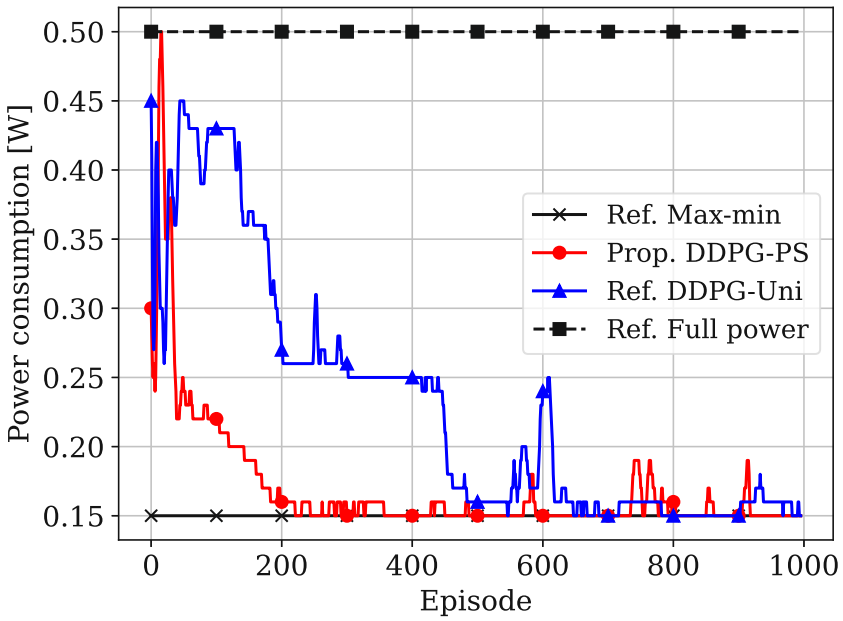


Figure 5.3: Total power consumption for the fully static scenario.

consumes the most power.

- Static users with UE toggling scenario
  We next allow UE toggling for the non-mobile users. Specifically, $K_{\text{tog}} = 0.1K$ UEs switch from active to inactive mode (or vice versa) every $T_{\text{tog}} = 500$ episodes. This explains the sudden activity in the plots of Fig. 5.4 and 5.5 at episode 500. After the toggle, we observe that *Prop. DDPG-PS* is quick to recover from this environment change, already converging to a new rate at episode 600 in Fig. 5.4. In contrast, this only occurs more than 200 episodes later for *Ref. DDPG-Uni*. We also highlight the benefits of prioritized sampling in this case. Unlike in our results for the fully static scenario, we not only achieve accelerated convergence, but also better rate performance, as exhibited by the 7.4% rate increase, compared to vanilla DDPG. This advantage extends to the total power consumption in Fig. 5.5, with *Prop. DDPG-PS* being evidently more power-efficient, consuming almost the same power as the optimal *Ref. Max-min*.

  We now consider the case where UE toggling occurs more frequently by lowering T_{tog} to 350, signifying that the change in the device activation status happens at episodes 350 and 700 in Fig. 5.6 and 5.7. After the first toggle, we observe that our *Prop. DDPG-PS* framework quickly achieves convergence at a rate of 14.5 Mbps at episode 450 in Fig. 5.6. In contrast, without prioritization, *Ref. DDPG-Uni* settles for a lower rate of 12.8 Mbps, amounting to an 11.72% rate decrease, only 100 episodes later. After the second toggle, we ex-

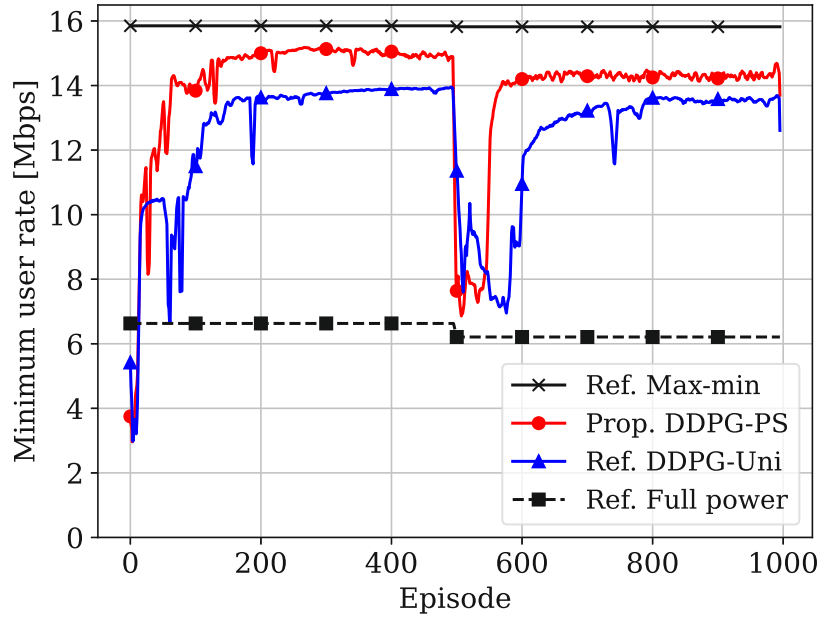


Figure 5.4: Guaranteed rate for the static, $T_{\text{tog}} = 500$, $K_{\text{tog}} = 0.1K$ scenario.

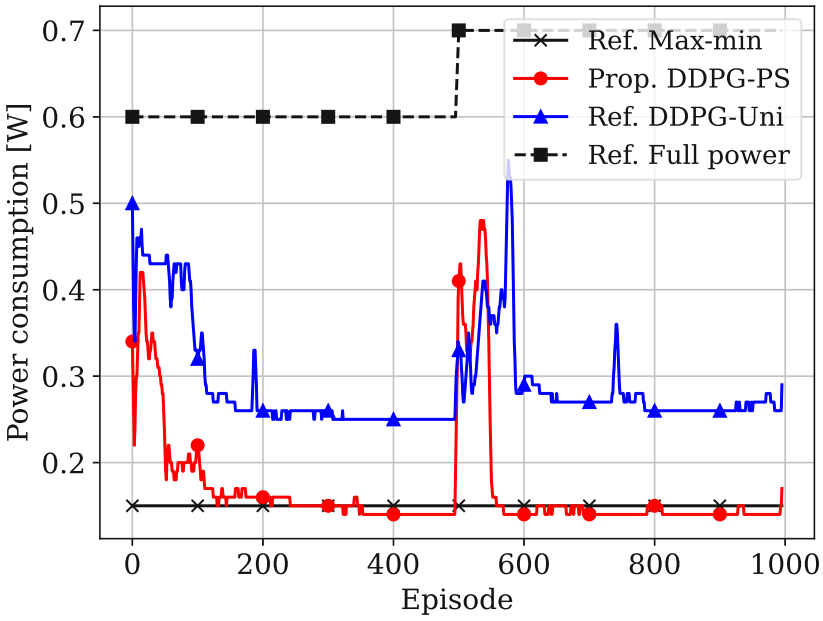


Figure 5.5: Total power consumption for the static, $T_{\text{tog}} = 500$, $K_{\text{tog}} = 0.1K$ scenario.

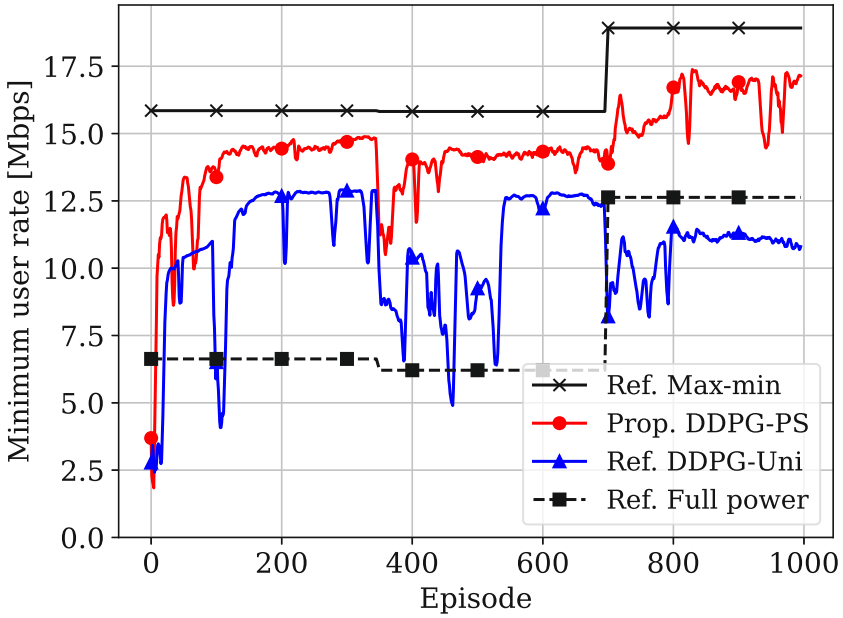


Figure 5.6: Guaranteed rate for the static, $T_{\text{tog}} = 350$, $K_{\text{tog}} = 0.1K$ scenario.

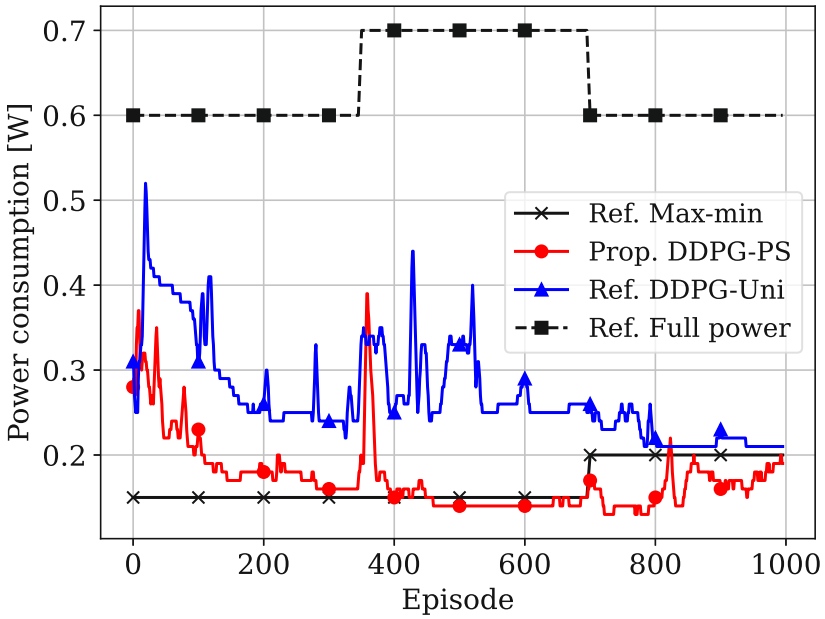


Figure 5.7: Total power consumption for the static, $T_{\text{tog}} = 350$, $K_{\text{tog}} = 0.1K$ scenario.

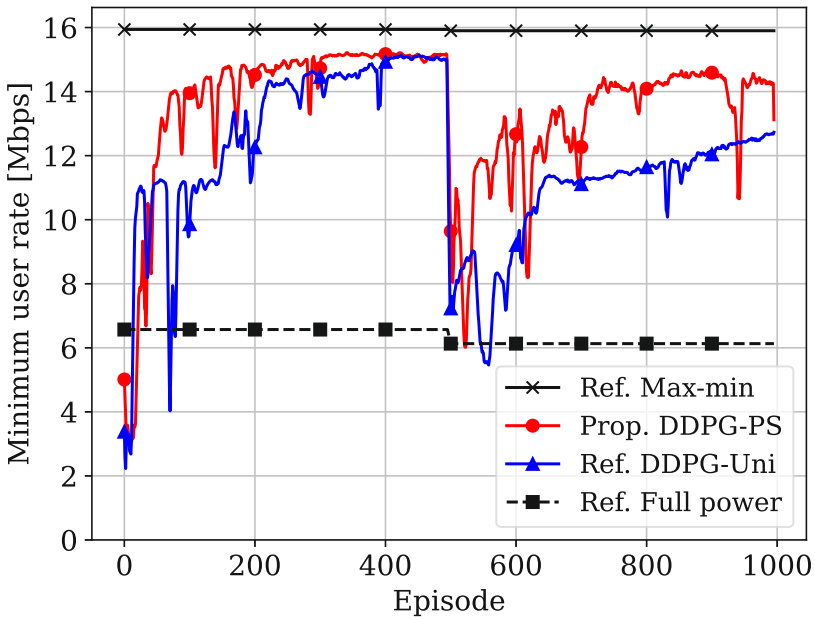


Figure 5.8: Guaranteed rate for the mobile, $T_{\text{tog}} = 500$, $K_{\text{tog}} = 0.1K$ scenario.

pect the guaranteed rate to increase, as depicted in the plots of *Ref. Max-min* and *Ref. Full power*. Our framework detects the change in the environment and is able to react accordingly with a corresponding increase in the minimum user rate. On the other hand, vanilla DDPG likely requires more time to do so. In terms of power consumption, we see that our proposed scheme still behaves close to *Ref. Max-min* in Fig. 5.7.

- Mobile users with UE toggling scenario
  Lastly, we combine user mobility with UE (de-)activation in Fig. 5.8 and 5.9, assuming $T_{\text{tog}} = 500$ and $K_{\text{tog}} = 0.1K$. Both DDPG-based schemes now take a longer time to converge to a solution compared to their static counterparts in Fig. 5.4 and 5.5. In particular, we observe an additional 100 episodes for *Prop. DDPG-PS*, while *Ref. DDPG-Uni* has yet to reach convergence even at episode 1000 in Fig. 5.8. The same phenomenon can be perceived in Fig. 5.9, where the power consumption plots are slower to settle to their converged values. In this more challenging scenario, the agent has to deal with the added mobility, on top of detecting the environment change triggered by UE (de-)activation, with both simultaneously affecting its action or power decisions. Nonetheless, our results demonstrate the robustness of our proposed framework, as we retain the performance gain in terms of rate, power, and convergence speed compared to vanilla DDPG while approaching the performance of *Ref. Max-min*.

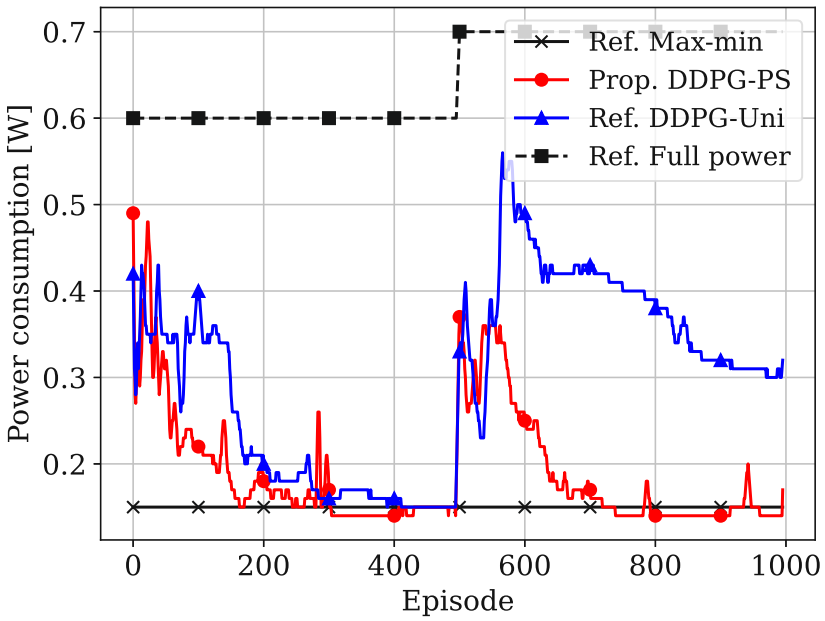


Figure 5.9: Total power consumption for the mobile, $T_{\text{tog}} = 500$, $K_{\text{tog}} = 0.1K$ scenario.

5.5 DDQN with Prioritized Sampling for Power Control

In the previous section, we utilized DDPG that employs four DNNs to solve the uplink power control problem in (5.9). Meanwhile, it was shown in [116] that the DNN processes involved in operating DRL algorithms consume a huge amount of energy. This is not an issue for a SARL system that considers the CPU as the agent, since the CPU has sufficient computing power to keep the DDPG algorithm running. In this section, however, we explore distributed architectures in line with our goal of creating a scalable cell-free network. Specifically, in addition to SARL, we propose two MARL systems, namely centralized training, decentralized execution (CTDE) and personalized federated learning (FedPer), that we later detail. In the case of MARL, each UE serves as one of the multiple agents. In practical deployments, the DRL algorithm would be implemented in battery-limited user devices, which then poses a problem due to the expected high power consumption of the DRL operation. We, therefore, utilize DDQN that considers a discrete action space and utilizes only two DNNs with relatively few parameters [92], in order to conserve energy and prolong the battery life of UEs. Moreover, the use of discrete power levels is a practical assumption specifically for IoT scenarios that commonly involve low-complexity devices [117]. Similar to the prior DDPG framework, we leverage TD error-based prioritized experience replay for improved performance [84].

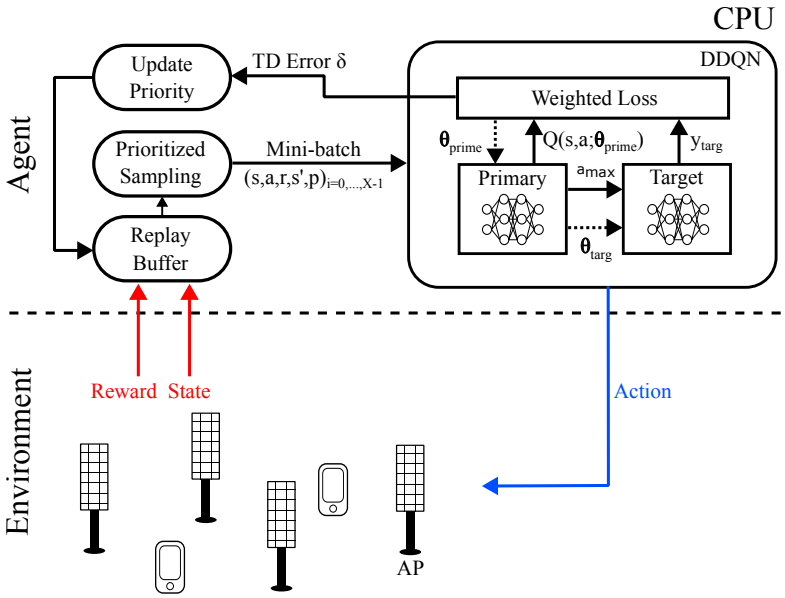


Figure 5.10: SARL-DDQN with prioritized sampling framework for power control.

5.5.1 SARL-DDQN Framework

We present our first DDQN framework for power control, which is based on a fully centralized SARL architecture [84]. The CPU serves as the agent that runs the prioritized sampling-based DDQN, while the APs and UEs are part of the environment, as illustrated in Fig. 5.10.

- DRL components

  1. State
     The state of the environment at time t is given by

$$\mathbf{s}^{(t)} = [d_1^{(t)}, \ldots, d_K^{(t)}, v_1^{(t-1)}, \ldots, v_K^{(t-1)}, u_1^{(t-1)}, \ldots, u_K^{(t-1)}]. \tag{5.20}$$

     The first K elements specify the latest activation state of the UEs, with $d_k^{(t)} \in \{0, 1\}$. Initially, the agent does not know how to interpret its state observations. It is, therefore, prone to committing power allocation errors (PAEs), where it assigns non-zero power even to inactive UEs. This negatively impacts performance, as it also affects the power allocated to the active UEs. In order to avoid this, we define the following PAE indicator

$$v_k^{(t-1)} = \begin{cases} 1, & \text{if } \rho_k^{(t-1)} > 0, d_k^{(t-1)} = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5.21}$$

     The total PAE at time t is

$$\text{PAE}^{(t)} = \sum_{k \in \mathcal{K}} v_k^{(t)}. \tag{5.22}$$

By observing the PAE vector elements, as well as the RL reward, the agent is able to autonomously learn the UE activation patterns. Thus, we expect the PAE to go down to zero as the agent further interacts with the environment. The last K elements in (5.20) provide the user rate performance.

2. Action
In DDQN, we consider a one-dimensional discrete action space. Since we have to determine the uplink transmit power of all K UEs, we enumerate all possible power combinations in the action space. The agent selects the index of one of these combinations as

$$a^{(t)} = \text{ind} \mapsto [\rho_1, \ldots, \rho_K]. \tag{5.23}$$

We then map the scalar ind to the corresponding K-element power vector. The possible values for power ρ_k are listed below

$$\rho_k = \left\{0, \frac{\rho_{\max}}{N_{\text{pow}} - 1}, \frac{2\rho_{\max}}{N_{\text{pow}} - 1}, \ldots, \rho_{\max}\right\}, \tag{5.24}$$

where N_{pow} denotes the number of power levels. The size of the SARL action space is $(N_{\text{pow}})^K$.

3. Reward
We formulate the following reward

$$r^{(t+1)} = \min_{j \in \mathcal{K}_{\text{ON}}^{(t)}} u_j^{(t)} - \text{PAE}^{(t)}, \tag{5.25}$$

which is made up of the guaranteed rate that we aim to maximize and a penalty term to minimize PAE by assigning zero power to inactive UEs.

- Proposed SARL-DDQN algorithm
We combine DDQN and prioritized experience replay, described in Sec. 2.2.3 and 5.3, respectively. Algorithm 7 outlines the steps implemented by the proposed SARL framework, which we detail below.

Step 1: We first initialize the DDQN primary and target networks, the ϵ-greedy algorithm for RL exploration-exploitation, and the prioritized sampling parameters (Lines 1 to 3).
Step 2: The agent makes its state observation, comprising of the latest UE ON/OFF states and the previous PAEs and user rates. It then applies its selected UE power combination. The CPU obtains a corresponding reward that is designed to maximize the minimum user rate while keeping PAE as low as possible. The environment transitions into a new state at the next time step (Lines 4 to 9).
Step 3: Before saving its newly acquired experience in its memory $\mathcal{B}$, the

CPU attaches a priority to it that is equal to the current maximum priority $p_{\max}$ (Line 10).
Step 4: As described in Sec. 5.3, the prioritized experience replay mechanism transforms the priorities of the stored experiences into probability values that are then used to compute the importance sampling weights. These relationships are governed by (5.10) and (5.13) (Lines 11 to 12).
Step 5: We next sample a mini-batch of X experiences using the previously calculated probabilities. For each sample, we compute the TD error δ_{TD}, which recall in DDQN is given by

$$\delta_{\text{TD}}^{(t)} = y_{\text{targ}}^{(t)} - Q(\mathbf{s}^{(t)}, a^{(t)}; \theta_{\text{prime}}^{(t)}),$$

where

$$y_{\text{targ}}^{(t)} = r^{(t+1)} + \gamma Q'(\mathbf{s}^{(t+1)}, a_{\max}; \theta_{\text{targ}}^{(t)}),$$

and

$$a_{\max} = \arg\max_{a} Q(\mathbf{s}^{(t+1)}, a; \theta_{\text{prime}}^{(t)}).$$

The target network calculates the target value y_{targ}, while the primary network is responsible for action selection. The resulting TD errors, together with the weights, are utilized to form the weighted loss function in (5.16). The primary DNN is updated by minimizing the loss over the sampled experiences using gradient descent. On the other hand, the target DNN is updated through Polyak averaging (2.13) (Lines 13 to 16).
Step 6: We perform several parameter value updates. We modify the priorities of the samples based on the new TD error values (5.11), as well as the current maximum priority $p_{\max}$. We anneal the correction parameter β according to (5.14). We update the ϵ-greedy algorithm to gradually shift from exploration to exploitation as in (2.15) (Lines 17 to 20).

Algorithm 7: SARL-DDQN with prioritized sampling for power control

Initialize the DDQN primary (θ_{prime}) and target (θ_{targ}) networks.
Initialize the ϵ-greedy algorithm $\epsilon \leftarrow \epsilon_{\text{start}}$ for RL exploration-exploitation.
Initialize the prioritized experience replay parameters $\beta \leftarrow \beta_{\text{start}}$ and $p_{\max} \leftarrow 0$.
for episode $e = 0, \ldots, E-1$ **do**
  Initialize state $\mathbf{s}^{(0)}$.
  for time step $t = 0, \ldots, T-1$ **do**
    Observe the current state $\mathbf{s}^{(t)}$ (5.20).
    Select and apply an action $a^{(t)}$ (5.23) according to (2.14).
    Observe the reward $r^{(t+1)}$ (5.25) and next state $\mathbf{s}^{(t+1)}$.
    Store experience $(\mathbf{s}^{(t)}, a^{(t)}, r^{(t+1)}, \mathbf{s}^{(t+1)}, p_{\max})$ in $\mathcal{B}$.
    Compute the probabilities (5.10) of the experiences.
    Compute their weights (5.13), (5.15).
    Sample a mini-batch of X experiences from $\mathcal{B}$ based on the calculated probabilities.
    Compute their TD error (5.12) and the resulting weighted loss (5.16).
    Update θ_{prime} by minimizing the loss using gradient descent.
    Update θ_{targ} using Polyak averaging (2.13).
    Update the priorities (5.11) of the samples.
    Update $p_{\max} \leftarrow \max_{j \in \{0,\ldots,\text{len}(\mathcal{B})\}} p_j$.
    Update β as in (5.14).
    Update ϵ if $\epsilon > \epsilon_{\text{end}}$ following (2.15).
  end for
end for

5.5.2 MARL-CTDE Framework

Our first distributed MARL setup is based on the CTDE approach in [118,119], comprising of a central trainer and multiple agents [84]. Unlike our prior SARL systems, where the CPU makes the power decision for all users, we distribute this task across the agents, in this case, the K UEs. At every time step, each agent selects its local action and subsequently acquires a new local experience, which is forwarded to the central trainer. The trainer gathers the experiences, and together with previously stored ones, utilizes them for training a single DDQN. This architecture creates an effect that the agents are learning together to accelerate convergence. The updated DNN model is then fed back to the agents for local action selection, and therefore, this is commonly referred to as parameter sharing [120]. Although the same trained DNN policy is used in the decision-making of multiple agents, the distributed DNNs take in different local state observations, resulting in distinct experiences for the agents [121]. The CTDE agent-environment scenario is shown in Fig. 5.11. The CPU serves as the central trainer that implements DDQN with prioritized experience replay [122]. Each UE acts as one of the K MARL agents that utilizes its own

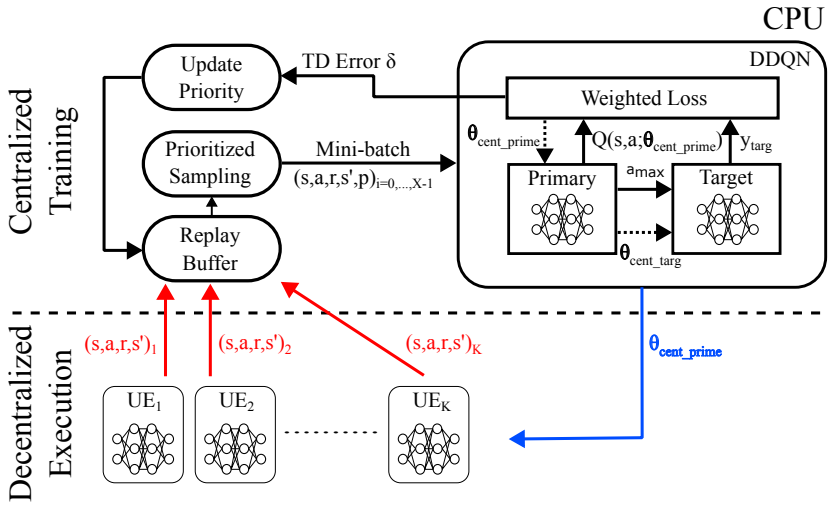


Figure 5.11: MARL-CTDE with prioritized sampling framework for power control.

DNN for local action selection.

- DRL components

  1. State
     The local observation of agent k at time t is described by

     $$\mathbf{s}_k^{(t)} = [d_k^{(t)}, u_k^{(t-1)}, u_{\substack{j\in\mathcal{K}\\ j\neq k}}^{(t-1)}]. \tag{5.26}$$

     Since agent k is an actual UE in the network, it knows the true activation status of UE k, represented by $d_k^{(t)}$. When a UE is inactive, it does not participate in the RL process, and its transmit power is automatically set to zero. As such, unlike the earlier SARL system, there would be no power violations, and we do not need to keep track of the PAE.

  2. Action
     The local action of agent k is

     $$a_k^{(t)} = \rho_k^{(t)}. \tag{5.27}$$

     The agent selects a single transmit power value among those listed in (5.24). The size of the action space is $|\mathcal{A}_{\text{CTDE}}| = N_{\text{pow}}$, which is considerably smaller than $|\mathcal{A}_{\text{SARL}}| = (N_{\text{pow}})^K$, highlighting the need for MARL architectures to improve network scalability.

  3. Reward

The reward is expressed as

$$r_k^{(t+1)} = \begin{cases} \min\limits_{j \in \mathcal{K}_{\text{ON}}^{(t)}} u_j^{(t)}, & d_k^{(t)} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{5.28}$$

We consider a global reward that is equal to the minimum user rate that we aim to maximize, which is received by the active UEs following the first condition in (5.28).

- Proposed MARL-CTDE algorithm
  We detail the procedure employed by the proposed CTDE framework, given in Algorithm 8.

  Step 1: We first initialize the DDQN DNNs and the prioritized sampling system at the central trainer. We also initialize the distributed DNN policies and the ϵ-greedy algorithm at the individual agents (Lines 1 to 2).
  Step 2: Each active UE observes the previous rate performance of the other UEs and decides on its own latest transmit power. It obtains a centrally computed reward that indicates the current guaranteed rate of the network. It then forwards its newly created experience to the CPU and updates the ϵ value if necessary. In contrast, inactive UEs do not undergo any of these steps (Lines 3 to 16).
  Step 3: The CPU attaches priority $p_{\max}$ to all the gathered local experiences from the active UEs before storing them in its replay buffer $\mathcal{B}$ (Line 17).
  Step 4: Since CTDE only distributes the action selection task and still keeps the training centralized, the entire prioritized experience replay mechanism is implemented at the central trainer (Lines 18 to 26).
  Step 5: Finally, the CPU sends a copy of the updated DNN policy $\theta_{\text{cent\_prime}}$ to all K UEs for them to use at the next time step (Lines 27 to 29)

$$\theta_{\text{prime},k}^{(t+1)} \leftarrow \theta_{\text{cent\_prime}}^{(t)}. \tag{5.29}$$

Algorithm 8: MARL-CTDE with prioritized sampling for power control

Initialize the DDQN primary ($\theta_{\text{cent\_prime}}$) and target ($\theta_{\text{cent\_targ}}$) networks, as well as the prioritized experience replay parameters $\beta \leftarrow \beta_{\text{start}}$ and $p_{\max} \leftarrow 0$, of the CPU (central trainer).
Initialize the distributed policy network ($\theta_{\text{prime},k}$) and the ϵ-greedy algorithm $\epsilon_k \leftarrow \epsilon_{\text{start},k}$, $\forall k \in \mathcal{K}$, of the UEs (agents).
for episode $e = 0, \ldots, E-1$ **do**
  Initialize state $\mathbf{s}_k^{(0)}$, $\forall k \in \mathcal{K}$.
  for time step $t = 0, \ldots, T-1$ **do**
    for UE $k = 0, \ldots, K-1$ **do**
      if $d_k^{(t)} = 1$ **then**
        Observe the current local state $\mathbf{s}_k^{(t)}$ (5.26).
        Select and apply a local action $a_k^{(t)}$ (5.27) according to (2.14).
        Observe the global reward $r_k^{(t+1)}$ (5.28) and next state $\mathbf{s}_k^{(t+1)}$.
        Forward experience $(\mathbf{s}_k^{(t)}, a_k^{(t)}, r_k^{(t+1)}, \mathbf{s}_k^{(t+1)})$ to the CPU.
        Update ϵ_k if $\epsilon_k > \epsilon_{\text{end},k}$ following (2.15).
      else
        Pass
      end if
    end for
    /* At the CPU */
    Store $(\mathbf{s}_k^{(t)}, a_k^{(t)}, r_k^{(t+1)}, \mathbf{s}_k^{(t+1)}, p_{\max})$, $\forall k \in \mathcal{K}_{\text{ON}}^{(t)}$, in $\mathcal{B}$.
    Compute the probabilities (5.10) of the experiences.
    Compute their weights (5.13), (5.15).
    Sample a mini-batch of X experiences from $\mathcal{B}$ based on the calculated probabilities.
    Compute their TD error (5.12) and the resulting weighted loss (5.16).
    Update $\theta_{\text{cent\_prime}}$ by minimizing the loss using gradient descent.
    Update $\theta_{\text{cent\_targ}}$ using Polyak averaging (2.13).
    Update the priorities (5.11) of the samples.
    Update $p_{\max} \leftarrow \max_{j \in \{0,\ldots,\text{len}(\mathcal{B})\}} p_j$.
    Update β as in (5.14).
    Send the latest $\theta_{\text{cent\_prime}}$ to all K UEs.
    for UE $k = 0, \ldots, K-1$ **do**
      Update local DNN $\theta_{\text{prime},k}$ (5.29).
    end for
  end for
end for

5.5.3 MARL-FedPer Framework

Our second distributed MARL setup combines federated learning [41] with DRL [84]. Federated learning is motivated by user privacy preservation, which is made possible by keeping the data samples local at the clients. In contrast to CTDE, the agents do not forward their RL experiences to a central trainer. They instead utilize them for training locally and only send a copy of their local model to a central server. The server implements model aggregation before feeding back the updated global model to the clients. We employ FedPer, where only the base layers are forwarded to the server, while the upper or personalized layers are kept local. This federated learning variant has been shown to outperform the default federated averaging-based aggregation (FedAvg) [87, 123, 124].

The weight matrix and bias vector of a single base layer of the local DNN of client k are denoted by $\mathbf{W}_k$ and $\mathbf{b}_k$, respectively. We define the model aggregation period T_{aggr} as the time duration over which independent local training takes place at each client. Every T_{aggr} time steps, the clients send the base layer parameters to the server, which then averages them as

$$\mathbf{W}_{\text{glob}} \leftarrow \frac{1}{|\mathcal{K}_{\text{ON}}|} \sum_{j \in \mathcal{K}_{\text{ON}}} \mathbf{W}_j, \tag{5.30}$$

$$\mathbf{b}_{\text{glob}} \leftarrow \frac{1}{|\mathcal{K}_{\text{ON}}|} \sum_{j \in \mathcal{K}_{\text{ON}}} \mathbf{b}_j. \tag{5.31}$$

While our work considers a single base layer only, the above definitions can be easily extended to support the aggregation of multiple base layers per client [87]. The proposed FedPer system is illustrated in Fig. 5.12, where the CPU acts as the server, and the UEs serve as the K MARL agents or clients. Compared to CTDE, each UE houses its own prioritized sampling-based DDQN.

- DRL components

  1. State
     The local observation of agent k is given by

     $$\mathbf{s}_k^{(t)} = [d_k^{(t)}, u_k^{(t-1)}, u_{j \in \mathcal{N}_k}^{(t-1)}], \tag{5.32}$$

     where we define $\mathcal{N}_k \subseteq \mathcal{K} \backslash \{k\}$ as the neighborhood of UE k containing its closest $|\mathcal{N}_k|$ neighbors. Setting $\mathcal{N}_k = \mathcal{K} \backslash \{k\}$ implies that UE k monitors all the other $K - 1$ users, as in the case of CTDE.
  2. Action
     Agent k selects its own UE transmit power as its local action

     $$a_k^{(t)} = \rho_k^{(t)}, \tag{5.33}$$

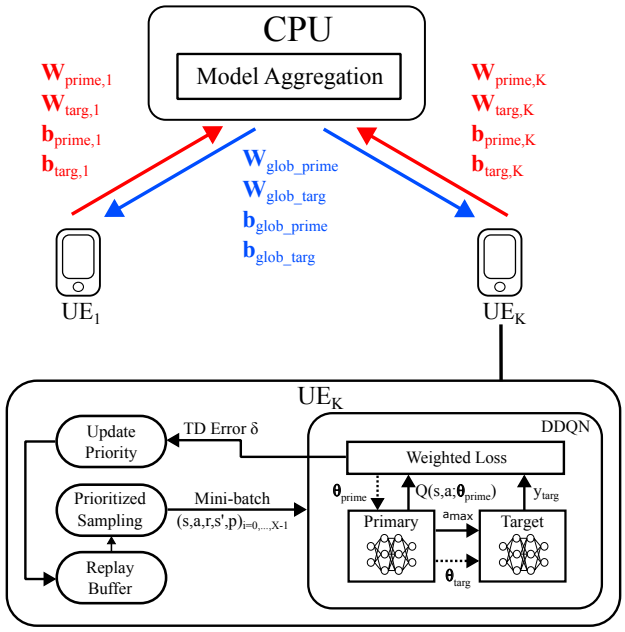


Figure 5.12: MARL-FedPer with prioritized sampling framework for power control.

whose value is among those enumerated in (5.24). Thus, the size of the action space is $|\mathcal{A}_{\text{FedPer}}| = N_{\text{pow}}$.

3. Reward
The local reward of agent k is

$$r_k^{(t+1)} = \begin{cases} \min\limits_{j \in \mathcal{N}_{k,\text{ON}}^{(t)}} u_j^{(t)}, & d_k^{(t)} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{5.34}$$

The first condition states that each active UE receives a reward equal to the local minimum user rate, considering itself and its monitored neighboring active UEs.

- Proposed MARL-FedPer algorithm
Algorithm 9 summarizes the procedure utilized by the proposed FedPer framework that we discuss below.

Step 1: We start with the initialization phase, where we first set up the global models for the DDQN primary and target networks at the server or aggregator node. These are sent out to the clients, which use them as a starting point for their local models. We locally initialize the ϵ-greedy algorithm and prioritized experience replay mechanism at each agent. To facilitate the FedPer model aggregation, we also set up global_ts that keeps track of the global time step count (Lines 1 to 2).
Step 2: Each active UE k observes the previous rate performance of the users

in its predefined neighborhood $\mathcal{N}_k$. It then decides its transmit power and consequently receives a local reward that is based on the resulting rates of its monitored active neighbors. These form a new local experience to which agent k attaches its current maximum priority $p_{\max,k}$ before saving it in its private buffer $\mathcal{B}_k$ (Lines 3 to 12).

Step 3: Unlike CTDE, FedPer localizes both action selection and prioritized sampling. Thus, the prioritized experience replay mechanism described in Sec. 5.3 is implemented at each agent (Lines 13 to 22).

Step 4: The model aggregation timing is dictated by global_ts and T_{aggr}. Every T_{aggr}, each active UE forwards the base layer parameters of its primary and target DNNs to the CPU. In turn, the server aggregates the collected local models as

$$\mathbf{W}_{\text{glob\_prime}} \leftarrow \frac{1}{|\mathcal{K}_{\text{ON}}|} \sum_{j \in \mathcal{K}_{\text{ON}}} \mathbf{W}_{\text{prime},j}, \tag{5.35}$$

$$\mathbf{b}_{\text{glob\_prime}} \leftarrow \frac{1}{|\mathcal{K}_{\text{ON}}|} \sum_{j \in \mathcal{K}_{\text{ON}}} \mathbf{b}_{\text{prime},j}, \tag{5.36}$$

$$\mathbf{W}_{\text{glob\_targ}} \leftarrow \frac{1}{|\mathcal{K}_{\text{ON}}|} \sum_{j \in \mathcal{K}_{\text{ON}}} \mathbf{W}_{\text{targ},j}, \tag{5.37}$$

$$\mathbf{b}_{\text{glob\_targ}} \leftarrow \frac{1}{|\mathcal{K}_{\text{ON}}|} \sum_{j \in \mathcal{K}_{\text{ON}}} \mathbf{b}_{\text{targ},j}. \tag{5.38}$$

The resulting global models are sent back to the clients for them to utilize at the next time step

$$\mathbf{W}^{(t+1)}_{\text{prime},k} \leftarrow \mathbf{W}^{(t)}_{\text{glob\_prime}}, \tag{5.39}$$

$$\mathbf{b}^{(t+1)}_{\text{prime},k} \leftarrow \mathbf{b}^{(t)}_{\text{glob\_prime}}, \tag{5.40}$$

$$\mathbf{W}^{(t+1)}_{\text{targ},k} \leftarrow \mathbf{W}^{(t)}_{\text{glob\_targ}}, \tag{5.41}$$

$$\mathbf{b}^{(t+1)}_{\text{targ},k} \leftarrow \mathbf{b}^{(t)}_{\text{glob\_targ}}. \tag{5.42}$$

We highlight that, in contrast to CTDE, the local experiences are kept private by the UEs. The agent coordination is enabled by the periodic sharing of the local base layer only, instead of the entire DNN as in the case of FedAvg (Lines 23 to 33).

Algorithm 9: MARL-FedPer with prioritized sampling for power control

Initialize the global models ($\mathbf{W}_{\text{glob\_prime}}$, $\mathbf{b}_{\text{glob\_prime}}$, $\mathbf{W}_{\text{glob\_targ}}$, $\mathbf{b}_{\text{glob\_targ}}$) at the CPU (server) and send them to the K UEs (clients). Set global_ts $\leftarrow 0$.
Initialize the distributed DDQN primary ($\theta_{\text{prime},k}$) and target ($\theta_{\text{targ},k}$) networks, the ϵ-greedy algorithm $\epsilon_k \leftarrow \epsilon_{\text{start},k}$, and the prioritized experience replay parameters $\beta_k \leftarrow \beta_{\text{start},k}$ and $p_{\text{max},k} \leftarrow 0$, $\forall k \in \mathcal{K}$.
for episode $e = 0, \ldots, E-1$ **do**
  Initialize state $\mathbf{s}_k^{(0)}$, $\forall k \in \mathcal{K}$.
  for time step $t = 0, \ldots, T-1$ **do**
    global_ts $\leftarrow$ global_ts $+ 1$
    for UE $k = 0, \ldots, K-1$ **do**
      if $d_k^{(t)} = 1$ **then**
        Observe the current local state $\mathbf{s}_k^{(t)}$ (5.32).
        Select and apply a local action $a_k^{(t)}$ (5.33) according to (2.14).
        Observe the local reward $r_k^{(t+1)}$ (5.34) and next state $\mathbf{s}_k^{(t+1)}$.
        Store $(\mathbf{s}_k^{(t)}, a_k^{(t)}, r_k^{(t+1)}, \mathbf{s}_k^{(t+1)}, p_{\text{max},k})$ in $\mathcal{B}_k$.
        Compute the probabilities (5.10) of the experiences.
        Compute their weights (5.13), (5.15).
        Sample a mini-batch of X experiences from $\mathcal{B}_k$ based on the calculated probabilities.
        Compute their TD error (5.12) and the resulting weighted loss (5.16).
        Update $\theta_{\text{prime},k}$ by minimizing the loss using gradient descent.
        Update $\theta_{\text{targ},k}$ using Polyak averaging (2.13).
        Update the priorities (5.11) of the samples.
        Update $p_{\text{max}} \leftarrow \max_{j \in \{0,\ldots,\text{len}(\mathcal{B}_k)\}} p_j$.
        Update β_k as in (5.14).
        Update ϵ_k if $\epsilon_k > \epsilon_{\text{end},k}$ following (2.15).
        if global_ts $\mod T_{\text{aggr}} == 0$ **then**
          Send $\mathbf{W}_{\text{prime},k}$, $\mathbf{b}_{\text{prime},k}$, $\mathbf{W}_{\text{targ},k}$ and $\mathbf{b}_{\text{targ},k}$ to the CPU.
        end if
      else
        Pass
      end if
    end for
    if global_ts $\mod T_{\text{aggr}} == 0$ **then**
      The server aggregates the local models as (5.35), (5.36), (5.37), (5.38) and feeds back the updated parameters to the K agents.
      for UE $k = 0, \ldots, K-1$ **do**
        Update the local DNNs as (5.39), (5.40), (5.41), (5.42).
      end for
    end if
  end for
end for

Table 5.2: Communication overhead
(with $y > z$ and $x > z$)

	UE-CPU		
	Downlink	**Uplink**	**UE-UE**
SARL	action [Kx bits]	state, reward, next state [$(6K+1)x$ bits]	-
CTDE	$\theta_{\text{cent\_prime}}$ [y bits]	$(\mathbf{s}, a, r, \mathbf{s}')$ [$\lvert\mathcal{K}_{\text{ON}}\rvert(2K+4)x$ bits]	u_k [$\lvert\mathcal{K}_{\text{ON}}\rvert v$ bits]
FedPer	$\mathbf{W}_{\text{glob\_prime}}$, $\mathbf{b}_{\text{glob\_prime}}$, $\mathbf{W}_{\text{glob\_targ}}$, $\mathbf{b}_{\text{glob\_targ}}$ [z bits]	$\mathbf{W}_{\text{prime},k}$, $\mathbf{b}_{\text{prime},k}$, $\mathbf{W}_{\text{targ},k}$, $\mathbf{b}_{\text{targ},k}$ [$\lvert\mathcal{K}_{\text{ON}}\rvert z$ bits]	u_k [$\lvert\mathcal{K}_{\text{ON}}\rvert v$ bits]

5.5.4 Communication Overhead

In this section, we compare the DDQN-based SARL and MARL frameworks in terms of their incurred communication overhead. We distinguish the information exchange occurring in two segments: (1) between each UE and the CPU, and (2) among the UEs. Table 5.2 summarizes the type of information and the number of bits required for transmission.

We start with the UE-CPU segment, which can be further divided into its downlink and uplink communications. In downlink, the SARL agent applies its K-element action vector, consisting of the UE power values. Assuming that each element corresponds to x bits, this then amounts to Kx transmission bits in total. In CTDE, the central trainer utilizes y bits to broadcast the updated DNN policy ($\theta_{\text{cent\_prime}}$). Meanwhile, it takes z bits for the FedPer server to broadcast the updated global model or the aggregated base layer of the local DNNs ($\mathbf{W}_{\text{glob\_prime}}$, $\mathbf{b}_{\text{glob\_prime}}$, $\mathbf{W}_{\text{glob\_targ}}$, $\mathbf{b}_{\text{glob\_targ}}$). Here, we note that $z < y$ [1]. In uplink, the SARL agent observes the current and next $3K$-element state vectors. Together with the scalar reward feedback, these cost $(6K+1)x$ bits in total. In the case of the MARL setups, recall that only the active UEs participate in the RL process, such that $|\mathcal{K}_{\text{ON}}|$ users transmit to either the central trainer or the aggregator server. Each active CTDE agent spends $(2K+4)x$ bits to forward its new experience $(\mathbf{s}, a, r, \mathbf{s}')$, with the state vectors having $K+1$ elements each. In FedPer, z bits are required for each active client to send the base layer parameters of its local DNN ($\mathbf{W}_{\text{prime},k}$, $\mathbf{b}_{\text{prime},k}$, $\mathbf{W}_{\text{targ},k}$, $\mathbf{b}_{\text{targ},k}$). Here, we assume that $(2K+4)x > z$, as it takes more bits to encode an entire RL experience than solely the base layer parameters of a DNN [125, 126]. We next proceed with the UE-UE segment. Each UE consumes v bits when broadcasting its rate performance to the other users. This corresponds to

[1] Given that $y = f(K)$ and $z = f(|\mathcal{N}_k| + 1)$, the gap between y and z is dictated by $\mathcal{N}_k$. In practice, each UE would only track a limited number of neighboring UEs in $\mathcal{N}_k \subset \mathcal{K}\backslash\{k\}$, implying a smaller local DNN per user and a lower overhead for FedPer.

$|\mathcal{K}_{\text{ON}}|v$ bits for both CTDE and FedPer.

Our analysis above shows that CTDE has a larger communication overhead than FedPer. In the case of SARL, a single RL experience is in effect transmitted when taking into account both downlink and uplink segments. In contrast, multiple RL experiences are forwarded in CTDE, and thus, it generally has a higher overhead than SARL. Note, however, that SARL suffers from scalability issues when considering large network deployments. Meanwhile, FedPer keeps the experiences local at the agents and only involves a periodic sharing of local models for aggregation. More importantly, we point out that T_{aggr} is not necessarily equal to 1, such that we do not aggregate at each time step. In fact, FedPer utilizes either $|\mathcal{K}_{\text{ON}}|(z+v)+z$ bits when aggregation is carried out or only $|\mathcal{K}_{\text{ON}}|v$ bits in time instances without aggregation. This implies that FedPer incurs the least amount of overhead among the three proposed methods. This advantage is more pronounced for larger aggregation periods.

5.5.5 Numerical Evaluations

We initially consider an uplink cell-free MIMO network with $M = 10$ single-antenna APs and $K = 5$ single-antenna UEs. All of which are uniformly distributed over a 100×100 m^2 area. A simulation consists of 500 episodes, having 100 time steps each. The agents make power decisions every time step, which we assume to last for 1 ms. In the case of FedPer, we start the aggregation at episode 10 to allow the agents to stabilize first. The simulation parameters are summarized in Table 5.3.

We refer to our proposed prioritized sampling-based DDQN frameworks in Sec. 5.5.1, 5.5.2, and 5.5.3 as *Prop. SARL-PS*, *Prop. CTDE-PS*, and *Prop. FPer-PS*, respectively. In FedPer, we additionally distinguish between two cases: (1) *Prop. FPer-Comp-PS*, where each agent has complete information on all user rates ($\mathcal{N}_k = \mathcal{K}\backslash\{k\}$), and (2) *Prop. FPer-Part-PS*, where each agent has partial information only ($\mathcal{N}_k \subset \mathcal{K}\backslash\{k\}$). We implement the DDQN primary and target networks using a fully connected DNN with two hidden layers, having 64 neurons each, and ReLU as the activation function. In FedPer, the first hidden layer serves as the base layer, which is forwarded for aggregation, and the second one as the personalized layer, which is kept local at each agent.

Dynamic device (de-)activation has not yet been considered in prior works, and thus, there is no algorithm in the existing literature to compare with. For instance, while MARL downlink power allocation in cell-free MIMO was investigated in [85], it employed uniform sampling and did not consider possible UE toggling. We evaluate the performance of our proposed frameworks using the following baseline schemes.

1. *Ref. Exhaustive* - Problem (5.9) is solved by considering discrete power values and performing an exhaustive search over all possible UE power combinations.

2. *Ref. Full Power* - Each UE transmits with $\rho_{\max}$.

Table 5.3: Simulation parameters

Parameter	**Value**
Carrier frequency f_c	1.9 GHz
Bandwidth B	20 MHz
Path loss exponent n_c	2
Shadow fading standard deviation	8 dB
Noise figure	9 dB
Pilot transmit power ρ_p	0.1 W
Per-UE maximum transmit power $\rho_{\max}$	0.1 W
Number of power levels N_{pow}	5
Buffer size	$1e6$
Mini-batch size X	100
Learning rate α_{step}	0.001
Discount rate γ	0.9
Polyak factor τ_{pol}	0.005
Prioritization factor α	0.7
(5-UE scenario) Correction parameter β_{start}, β_{end}	0.4, 1
(10-UE scenario) Correction parameter (fixed) β	0.4

3. *Ref. Uni.* - Vanilla DDQN relies on uniform sampling as in [92]. We refer to the corresponding DRL benchmarks as *Ref. SARL-Uni*, *Ref. CTDE-Uni*, *Ref. FPer-Comp-Uni*, and *Ref. FPer-Part-Uni*.

4. *Ref. Ind. MARL* - The K agents are fully independent and do not exchange any information. Each agent selects its transmit power to maximize its own rate.

We test our proposed algorithms using different combinations of UE toggling and user mobility settings. In the mobile scenario, each user randomly selects a direction (left, right, up, down) and a speed from 0 to 1 m/s at each time step, assuming a uniform distribution.

We compared the performance of FedPer, FedAvg [41], and the case with no aggregation. The results in Fig. 5.13 motivate us to employ FedPer, which achieves the highest guaranteed rates. Meanwhile, we observe that averaging random local models at each time step in FedAvg leads to unstable learning or even divergence, as investigated in previous works [87]. Lastly, we get the worst rate without aggregation. In FedPer, we experimented with various aggregation period T_{aggr} values. It turned out that we obtain comparable performance for $T_{\text{aggr}} = 1$ and $T_{\text{aggr}} = 10$. Thus, we utilize $T_{\text{aggr}} = 10$ in our simulations to enable a good performance with relatively low overhead by aggregating less frequently.

We experimented with different numbers of discrete power levels N_{pow} for FedPer in Fig. 5.14. Here, we include the *Continuous* benchmark, where Problem (5.9) is solved while considering continuous power values as in [14]. A lower N_{pow} implies

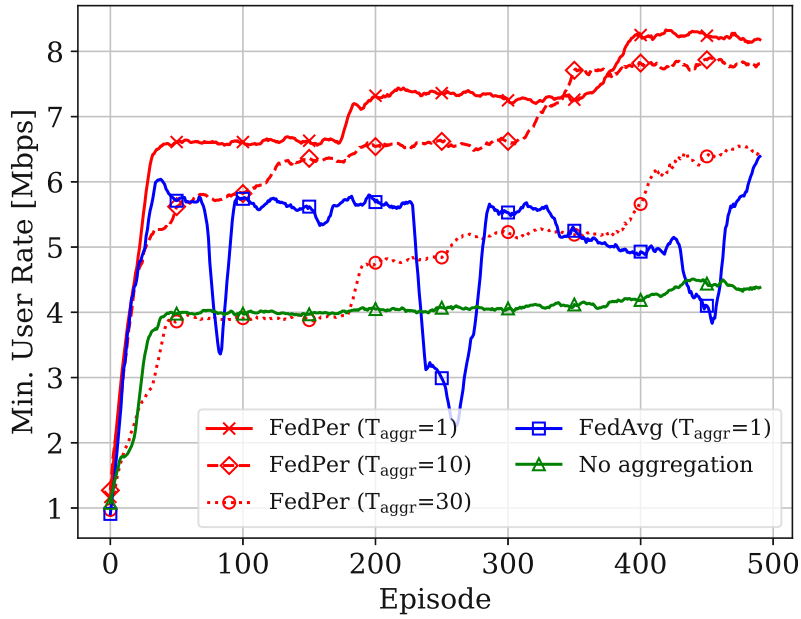


Figure 5.13: Performance comparison between FedPer and FedAvg.

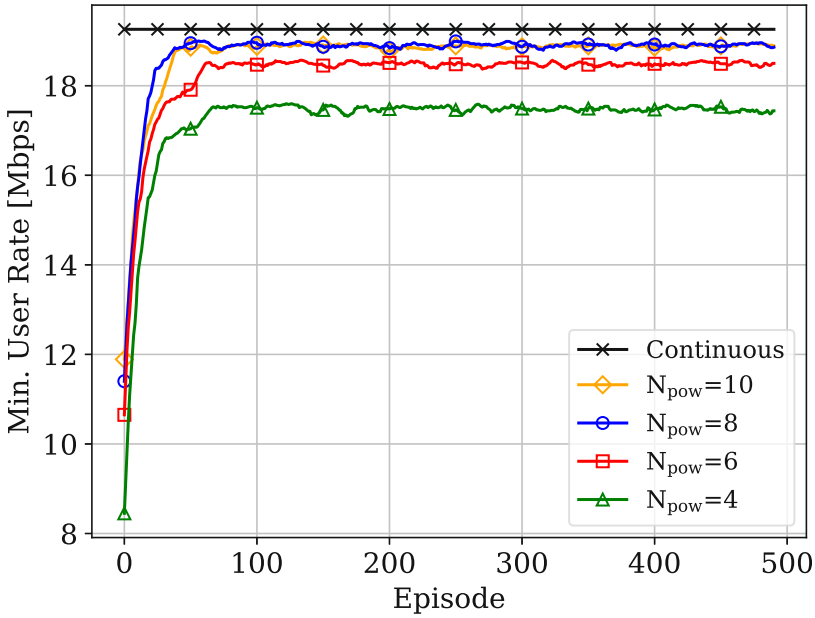


Figure 5.14: Impact of number of discrete power levels N_{pow}.

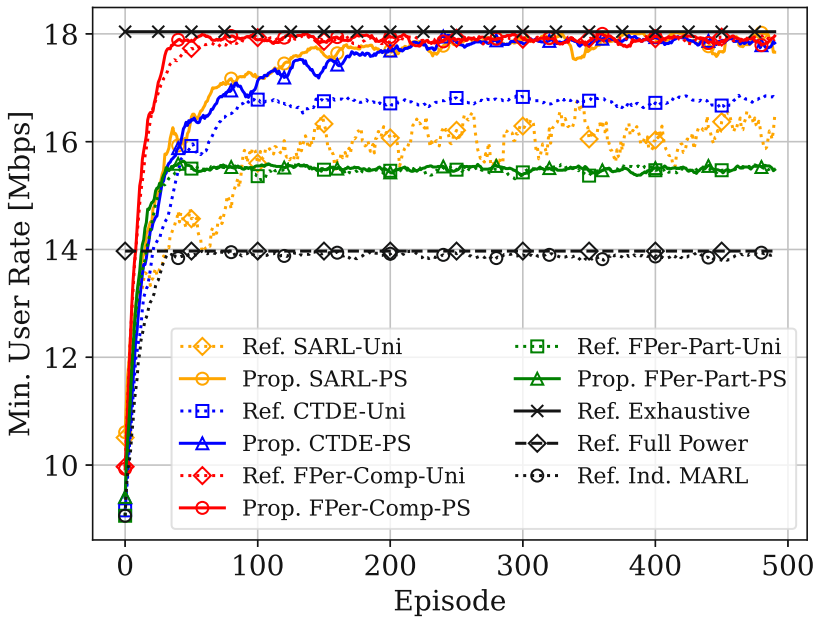


Figure 5.15: Guaranteed rate for the fully static scenario with $K = 5$.

coarser power quantization that results in a performance degradation. On the other hand, better guaranteed rates are achieved with higher N_{pow} values. In the following numerical evaluations, we utilize $N_{\text{pow}} = 5$ to limit the size of the SARL action space, allowing us to compare the SARL and MARL frameworks. Note, however, that with sufficiently large N_{pow}, we would obtain a performance that is closer to the *Continuous* benchmark, as depicted in Fig. 5.14.

- Fully static scenario
  We first consider the fully static case, where the users are not moving and the UE activation states remain constant throughout the simulation. The minimum user rates achieved by the different schemes are depicted in Fig. 5.15, where *Ref. Exhaustive* provides us with the upper bound performance. We obtain 16 Mbps and 18 Mbps with *Ref. SARL-Uni* and *Prop. SARL-PS*, respectively. This demonstrates how prioritization enables SARL to achieve the optimal rate, as well as a more stable learning curve. We next compare *Ref. CTDE-Uni* and the fully centralized *Ref. SARL-Uni* system, where we get a slightly higher rate in the former. This is likely due to the significantly larger action space for SARL, with $|\mathcal{A}_{\text{SARL}}| = 3125$ compared to just $|\mathcal{A}_{\text{CTDE}}| = 5$ per agent in CTDE. Nevertheless, prioritized sampling again makes a difference, allowing *Prop. CTDE-PS* to also approach *Ref. Exhaustive*. Meanwhile, both *Ref. FPer-Comp-Uni* and *Prop. FPer-Comp-PS* exhibit the optimal rate performance. When we limit the information exchange among the agents in the case of *Ref. FPer-Part-Uni* and *Prop. FPer-Part-PS*, such that each UE

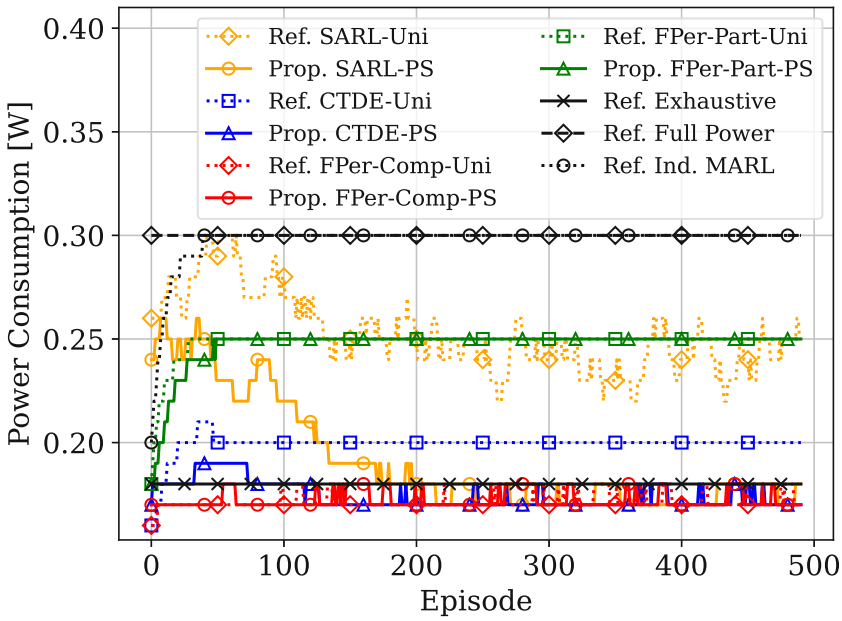


Figure 5.16: Total power consumption for the fully static scenario with $K = 5$.

tracks a limited number of neighboring UEs with $|\mathcal{N}_k| = 0.4K$, the minimum user rate decreases by 13.89%. While we get marginal benefits from equipping FedPer with prioritized experience replay for this simple scenario, we will see later in this section that this is not the case when considering more complicated scenarios. Lastly, we get the worst guaranteed rates with *Ref. Full Power* and *Ref. Ind. MARL* due to the increased inter-user interference, since the UEs decide on their transmit power without regard to the other users.

We show the total power consumption of the different schemes in Fig. 5.16. These are calculated by summing either the UE power values in the resulting action vector of SARL or the individual powers of the multiple agents in CTDE and FedPer. We first observe that power consumption goes down while the guaranteed rate increases for all DRL-based methods. This suggests that the agents are able to improve their DNN policy, allowing them to select better actions or power values, as they further interact with the environment by observing the states and reward feedback. Baselines *Ref. SARL-Uni* and *Ref. CTDE-Uni* converge to around 0.24 W and 0.20 W, respectively. In contrast, *Prop. SARL-PS* and *Prop. CTDE-PS*, as well as both *Ref. FPer-Comp-Uni* and *Prop. FPer-Comp-PS*, achieve the optimal power performance at 0.18 W, as indicated by *Ref. Exhaustive*. Having partial network information in *Ref. FPer-Part-Uni* and *Prop. FPer-Part-PS* causes a 38.89% increase in power consumption compared to their FPer-Comp equivalents. Finally, we observe that *Ref. Ind. MARL* behaves close to *Ref. Full Power*, with the agents likely

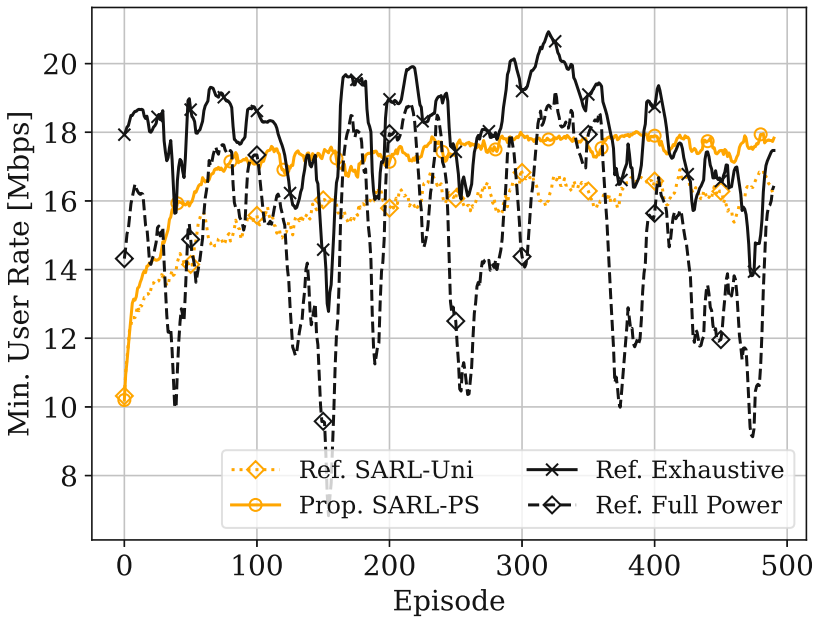
to utilize their maximum transmit power to maximize their own rate. This, however, only leads to power inefficiency, as evident in Fig. 5.16.

- Mobile users without UE toggling scenario

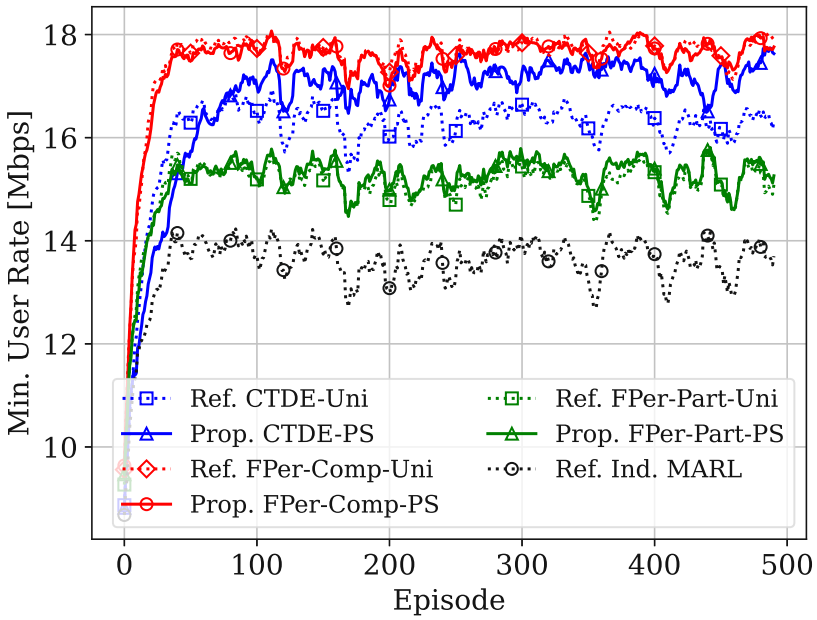
  We next add user mobility while keeping the UE ON/OFF states unchanged during the learning process. We observe that the plots in Fig. 5.17 and 5.18 are noisier than their fully static counterparts. This can be attributed to the random user movements at each time step that directly impact the channel gains and UE rate calculations. Furthermore, we consider a small number of discrete power levels in our simulations for reasons of feasibility and limited compute capability. This may have also contributed to the rate fluctuations, since there is a relatively large gap between any two power levels, causing significant variations in the user rates from one time step to the next.

  User mobility makes the power selection process more challenging for the agents. This is exhibited by the PAE plots of *Ref. SARL-Uni* and *Prop. SARL-PS* in Fig. 5.19. Recall that the ideal case would be a PAE of 0, meaning, the agents have learned to assign zero power to all inactive UEs. In the case of static users (in blue), *Ref. SARL-Uni* settles for a PAE of 0.2, while prioritized sampling makes it possible to bring the PAE to 0 for *Prop. SARL-PS*. However, with user mobility (in red), we observe higher PAEs for both schemes. With prioritization, the PAE is still likely to reach 0, albeit at a slower pace. As previously mentioned, a non-zero PAE leads to poor power performance, since the active UEs are assigned suboptimal power values. This phenomenon is illustrated in Fig. 5.16 and 5.18, where prioritized sampling-based frameworks are more power-efficient. Note that PAE does not exist for CTDE and FedPer, since the transmit power of inactive UEs is automatically zero.

  Recall that we purposely designed our proposed frameworks to only include rate information in the state vector to minimize information exchange, which is motivated by improved scalability and user privacy preservation. This is, however, insufficient when it comes to high-mobility scenarios. Additional information, including position- and channel-related ones, may help improve the DRL performance in such scenarios, as investigated in [80]. Nonetheless, our results in Fig. 5.17, 5.18, and 5.19 demonstrate the potential of prioritization.



(a)



(b)

Figure 5.17: Guaranteed rate for the mobile, no UE toggling scenario with $K = 5$.

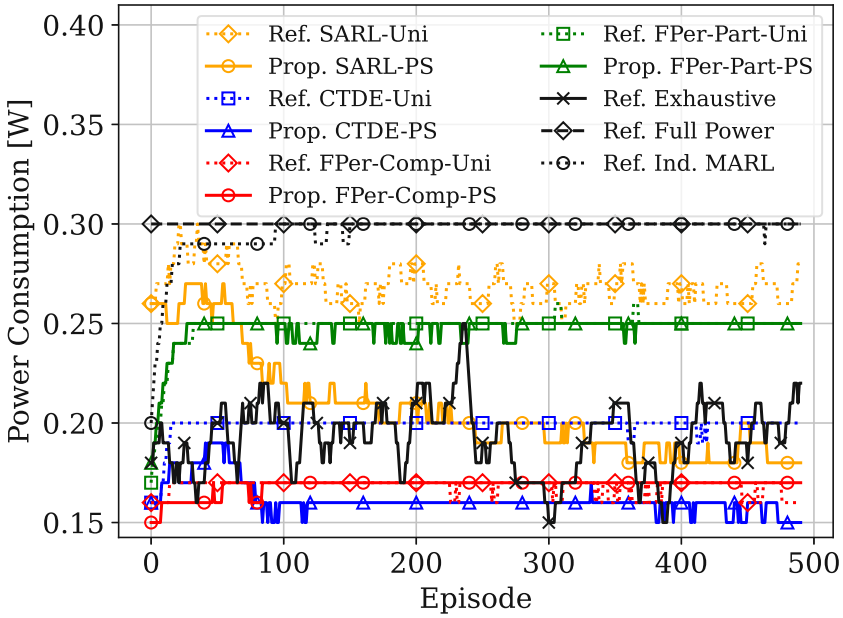


Figure 5.18: Total power consumption for the mobile, no UE toggling scenario with $K = 5$.

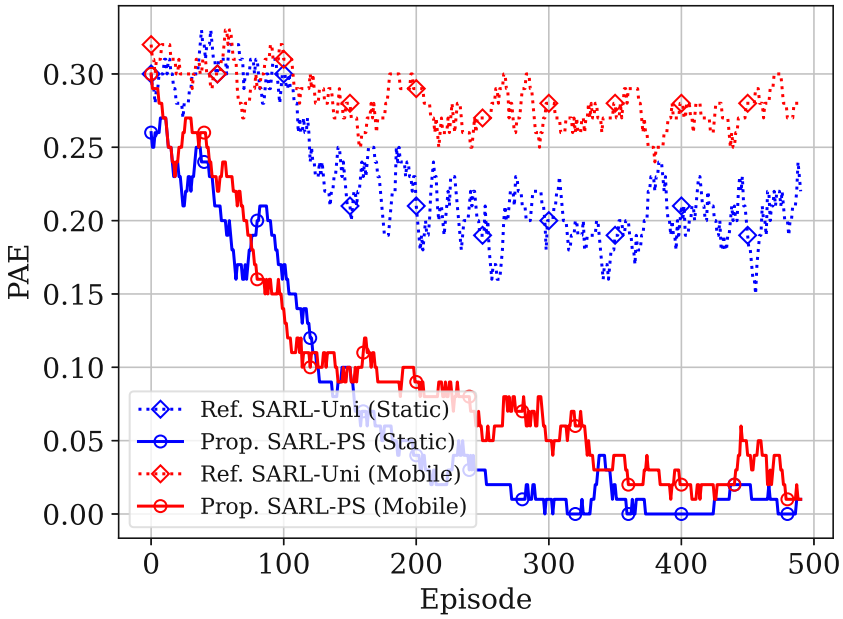


Figure 5.19: Impact of user mobility on PAE.

Table 5.4: Convergence analysis

	Before Toggle			After Toggle		
	Rate [Mbps]	**Episode Number**	**Time [s]**	**Rate [Mbps]**	**Episode Number**	**Time [s]**
Ref. SARL-Uni	16.0	150	15	15.5	450	20
Prop. SARL-PS	17.9	175	17.5	15.5	260	1
Ref. CTDE-Uni	16.8	100	10	11.75	400	15
Prop. CTDE-PS	17.2	125	12.5	12.5	375	12.5
Ref. FPer-Comp-Uni	17.9	50	5	15.25	450	20
Prop. FPer-Comp-PS	17.9	50	5	15.25	270	2
Ref. FPer-Part-Uni	15.5	50	5	12.5	400	15
Prop. FPer-Part-PS	15.5	25	2.5	14.0	450	20

- Static users with UE toggling scenario

  We next investigate the performance of our systems considering the complex case of device (de-)activation. Here, we set $T_{\text{tog}} = 250$ and $K_{\text{tog}} = 0.2K$ for the static users. This means that some of the UEs change their ON/OFF status at episode 250; hence, we see abrupt movements at the 250th mark in Fig. 5.20 and 5.21. We also summarize the converged guaranteed rates and convergence time before and after the UE toggle in Table 5.4. The convergence time is expressed in terms of the episode at which convergence was reached and the elapsed time from the latest UE toggle until convergence.

  While both uniform and prioritized sampling-based SARL and FPer-Comp schemes converge to near-optimal rates and power values after the device (de-)activations at episode 250, convergence occurs 200 episodes earlier when applying prioritization in *Prop. SARL-PS* and *Prop. FPer-Comp-PS*. However, when we factor in the other benefits of the distributed *Prop. FPer-Comp-PS*, namely reduced communication overhead and improved network scalability, it outperforms the fully centralized *Prop. SARL-PS*. When the agents have partial information only in their state vector, the system suffers from performance degradation, as exhibited by *Ref. FPer-Part-Uni*. This can be improved by equipping the system with prioritized experience replay, where we observe a 12% rate increase in *Prop. FPer-Part-PS*. While this advantage extends to *Prop. CTDE-PS*, all CTDE-based schemes require more time steps to find better DRL solutions for this scenario.

  We emphasize that the agents have no prior knowledge of the device activation patterns that are by themselves unpredictable, as they would depend on the UE battery life in practice. Thus, the agents have no way of anticipating the sudden changes in the ON/OFF status of the other UEs. Our simulation results demonstrate that our proposed prioritized sampling-based frameworks are capable of adapting to new and unknown environments, which are in this case, triggered by UE toggling.

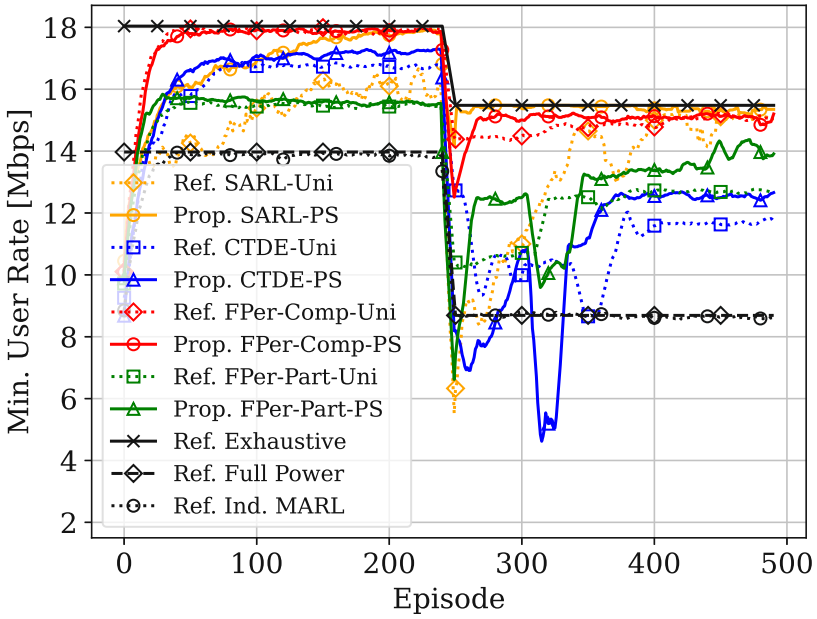


Figure 5.20: Guaranteed rate for the static, $T_{\text{tog}} = 250$, $K_{\text{tog}} = 0.2K$ scenario with $K = 5$.

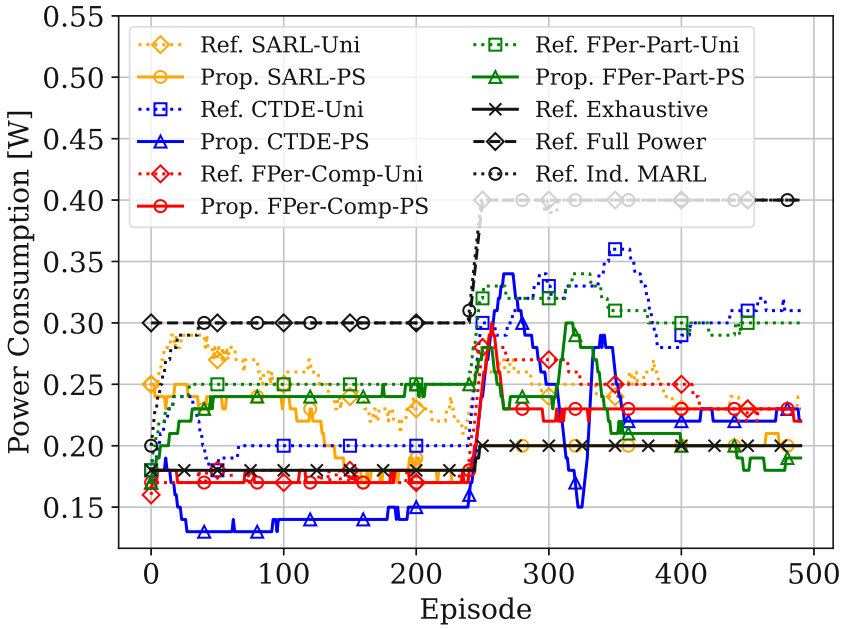
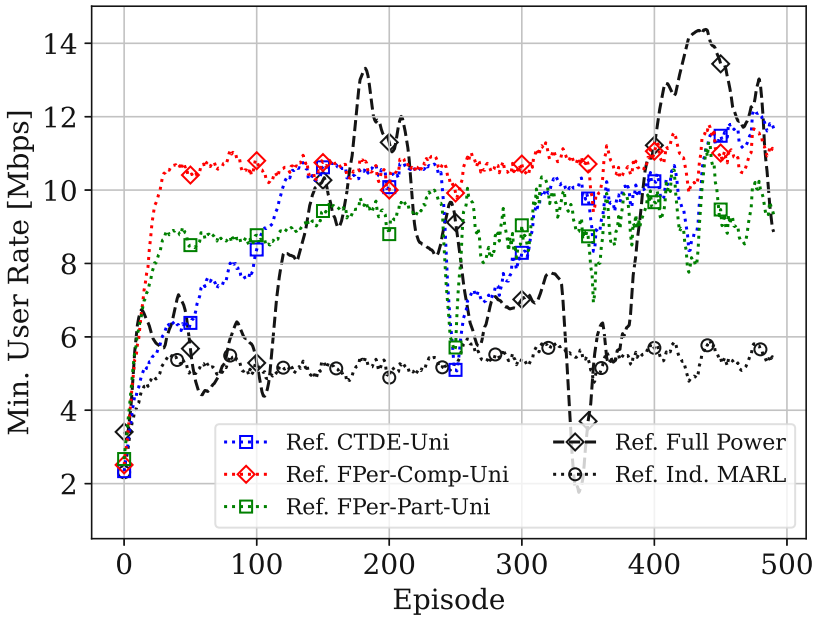


Figure 5.21: Total power consumption for the static, $T_{\text{tog}} = 250$, $K_{\text{tog}} = 0.2K$ scenario with $K = 5$.

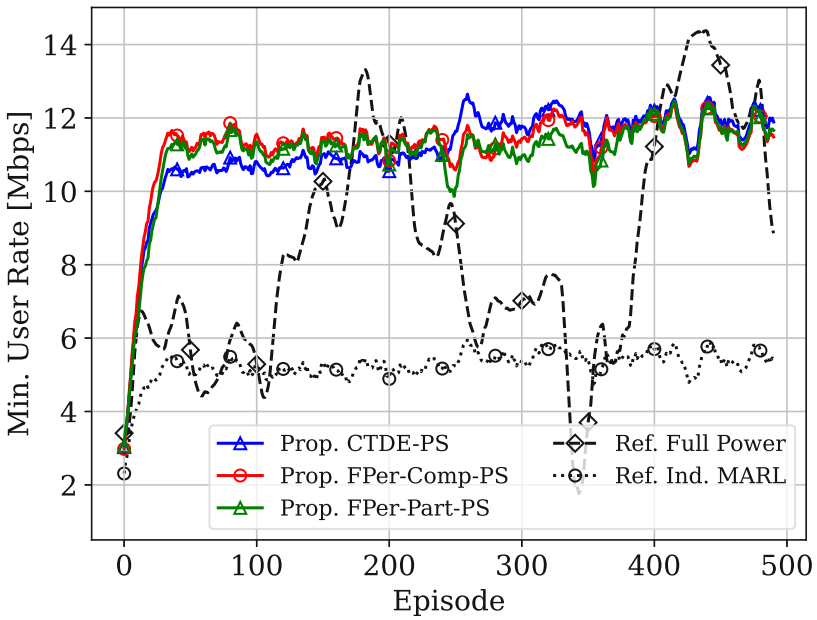
- Mobile users with UE toggling scenario

  Finally, we combine user mobility with device (de-)activation while assuming a larger cell-free network with $M = 30$ single-antenna APs and $K = 10$ single-antenna UEs over a 500×500 m^2 area. Note that we no longer consider the *Ref. Exhaustive* benchmark and SARL-based schemes due to their significantly large action space with $|\mathcal{A}| = 5^{10}$.

  Focusing on Fig. 5.22a and 5.22b, we consistently achieve better minimum user rates with prioritization. Although the CTDE and FPer-Comp methods converge to the same rates, recall in Sec. 5.5.4 that the former incurs a higher communication overhead and is, therefore, inferior to the latter, especially that we only aggregate every 10 time steps. Meanwhile, we configure $|\mathcal{N}_k|$ to $0.5K$ for *Prop. FPer-Part-PS*. Interestingly, we observe that for this sufficiently large neighborhood for each UE, again coupled with prioritization, *Prop. FPer-Part-PS* is able to close the gap with *Prop. FPer-Comp-PS*. This is yet another possibility of reducing overhead, since the overhead is dictated by the size of the neighborhood based on our analysis in Sec. 5.5.4. Thus, by tuning $|\mathcal{N}_k|$ and only tracking a limited number of nearby users that cause the greatest interference, we are able to obtain a good performance while keeping the overhead relatively low. In terms of power consumption in Fig. 5.23, the prioritized sampling-based systems are perceived to be more power-efficient.



(a)



(b)

Figure 5.22: Guaranteed rate for the mobile, $T_{\text{tog}} = 250$, $K_{\text{tog}} = 0.2K$ scenario with $K = 10$.

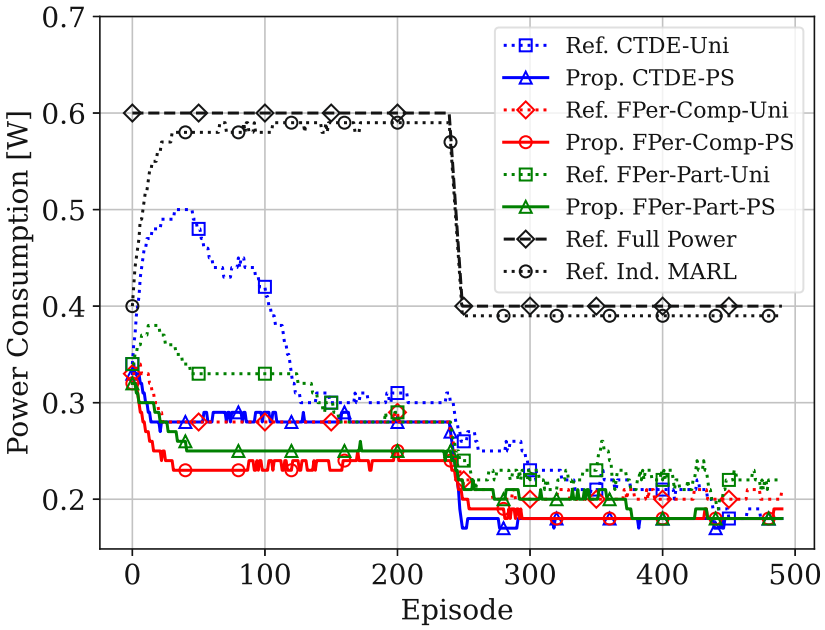


Figure 5.23: Total power consumption for the mobile, $T_{\text{tog}} = 250$, $K_{\text{tog}} = 0.2K$ scenario with $K = 10$.

5.6 Summary

In this chapter, we tackled a classical uplink power control problem that maximizes the minimum user rate of cell-free massive MIMO. We employed DDPG and DDQN, with the latter being more suitable for battery-limited and low-complexity user devices. Our proposed DRL frameworks rely solely on UE rate feedback and are agnostic to the system model. This implies that in a real-world scenario, they would consider the actual achieved user rates and be able to take into account several practical factors that are neglected in existing system models in the literature. The limited information exchange among the network entities is motivated by user privacy preservation and improved scalability. More importantly, we designed our systems such that we also addressed the shortcomings of our previously proposed methods, namely assuming a static RL environment and utilizing a single agent only, as those are prohibitive in the effectiveness of DRL in practical large-scale network deployments.

We focused on the challenging dynamic scenario of combined UE (de-)activation and user mobility, which has not yet been considered in prior works. Traditional optimization techniques require precise knowledge of the activation patterns in advance that is difficult to obtain in practice due to their unpredictability. In contrast, our proposed DRL systems autonomously recognize the patterns on the go. We specifically assumed online learning, where the agent learns continuously until con-

vergence is reached. When a significant change in the environment is detected, for instance, triggered by UE toggling in our case, the system enters a re-training phase to find the optimal power values for the new environment, during which convergence is again achieved. To ensure that our frameworks are able to quickly adapt to such changes in the wireless environment, addressing the slow convergence issue of DRL, we employed prioritized experience replay. We attach TD error-based priority values to the stored experiences that influence which experiences are sampled for updating the DNNs of the DRL algorithm in use. Our simulation results demonstrated that, not only was it able to accelerate convergence, but it also improved the rate and power performance.

We proposed fully centralized SARL and distributed MARL algorithms. Recall that in SARL, a single agent is the lone decision-maker of the system. On the other hand, multiple agents are employed by MARL, resulting in a smaller action space per agent, to enable distributed learning in a more scalable network. We investigated two MARL architectures, namely CTDE and FedPer. The CTDE framework depends on a central node that utilizes the gathered local experiences from the agents for training a single policy. In contrast, while FedPer also relies on a central entity for periodic model aggregation, it keeps the training local at each agent. The FedPer clients only forward the base layer parameters of their local DNNs to the central server. Among the three systems, our analysis showed that FedPer has the least amount of communication overhead. We tested our proposed methods using different combinations of UE toggling and user mobility settings. In our numerical evaluations, FedPer with prioritization boasted a competitive, near-optimal performance with minimal overhead, even while aggregating less frequently. It offers more flexibility by allowing us to tweak the size of neighborhood per user and the model aggregation period. Our experiments also showed that with a sufficiently large neighborhood to monitor the UEs generating the highest interference, prioritized sampling-based FedPer with only partial information in its state vector performed well compared to FedPer with complete information, despite the relatively low overhead incurred by the former setup.

6

Conclusion and Outlook

Cell-free massive MIMO represents a paradigm shift in network architecture, where we transition from the rigid structure of fixed cell boundaries to a seamless network of cooperating APs, enabling a uniformly good performance throughout the coverage area. The ability of DRL to learn online, without requiring a training dataset, sets it apart from other ML techniques and makes it suitable for dynamic wireless environments. Motivated by the potential of these technologies, this dissertation focused on the design of SARL- and MARL-based algorithms for the realization of scalable, self-adapting cell-free massive MIMO networks.

6.1 Summary of Contributions

The first part of the dissertation provided initial insights on the performance comparison between the canonical and user-centric cell-free massive MIMO setups in terms of their fronthaul requirement and guaranteed QoS. We demonstrated that a higher user demand necessitated more serving APs per user-centric cluster, and thus, higher fronthaul capacity utilization. We showed that there exists a cluster size that enables the scalable user-centric variant to perform close to the canonical setup, but with significantly lower fronthaul requirement. This implies that only a subset of APs actually contributes to UE rate performance. While we have successfully dealt with the non-convexity of the formulated optimization problems in this case, the proposed suboptimal algorithms involved relaxations and iterative processes that would be time-consuming to implement in practice.

In the second part, we presented our first applications of DRL for network optimization. Our investigations were centered on improving the energy efficiency and scalability of cell-free massive MIMO while maintaining good service in terms of either the guaranteed QoS or the network sum rate. To this end, we formulated various non-convex problems and leveraged SARL to approximately solve them. We developed a framework that derives a near-optimal set of active APs based on the instantaneous user positions, which provided good performance while achieving power savings by turning off underutilized APs. We next designed an algorithm to cluster

the APs in a scalable multi-CPU environment based on the spatial user density information. We showed that by forming a larger AP group to match the anticipated higher concentration of users in a given subregion, we were able to improve the sum rate. We also proposed a SARL system for user-centric clustering or AP-UE association. By optimizing the AP selection for each cluster, we performed close to the upper bound, but with much fewer AP-UE connections. Moreover, the flexible design of our RL reward functions enabled us to easily adapt our frameworks to different operating points, as well as to investigate the interplay of rate performance, power consumption, and fronthaul requirement of the network.

The last part of the dissertation focused on effective SARL- and MARL-based power control strategies in uplink cell-free massive MIMO, aiming to maximize the guaranteed rate. We designed our frameworks to rely solely on UE rate feedback, which comes with several benefits, namely user privacy preservation, reduced overhead for improved scalability, and being agnostic to the system model in use. We assumed a dynamic RL environment, which is characterized by the combination of device (de-)activation and user mobility, allowing us to emulate more realistic scenarios. Compared to traditional optimization methods, our model-free DRL systems operate in an online manner and learn the unpredictable activation patterns on the go. The non-static nature of the wireless environment exposes one weakness of DRL, which is its slow convergence. Thus, we capitalized on TD error-based prioritized experience replay to accelerate learning. We explored both fully centralized SARL and distributed MARL architectures. We further investigated two MARL setups, namely CTDE, which centralizes training, and FedPer, which localizes training and only requires periodic aggregation of the local base layer parameters. Our numerical experiments showed that prioritized sampling-based FedPer achieved near-optimal rate and power performance, with faster convergence, while also outperforming the other algorithms in terms of communication overhead. By properly configuring the FedPer parameter values, such that we aggregated less frequently and only tracked a limited number of users generating non-negligible interference, we obtained good performance without being burdened by the resulting overhead.

6.2 Open Issues and Possible Future Works

- Inter-CPU cooperation

  In Chapters 2 and 4, we defined scalable cell-free massive MIMO as a network implementing user-center clustering and multiple interconnected CPUs. The information exchange among network entities is dictated by the CPU-AP and AP-UE associations. Recall that we group the APs connected to the same CPU in the former, while we determine the subset of APs serving each UE in the latter. Although we tackled the corresponding association problems using DRL, we dealt with them separately in Chapter 4. Specifically, the user-centric clusters were created assuming a single-CPU network in [50]. Meanwhile, multiple CPUs were employed in [67], but the APs forming a user-centric

cluster must all be connected to the same CPU. The ideal case, however, would allow each UE to select its best serving APs, and these APs may be connected to different CPUs. This setup entails inter-CPU cooperation that includes determining the type of information that must be exchanged among them, which has not been covered in this dissertation. User mobility also adds to the complexity, since the APs associated to each mobile UE would change over time. Thus, a possible future work is the design of a unified hierarchical DRL framework that handles both CPU-AP and AP-UE associations in a mobile cell-free network.

- Large discrete action space

  In Chapters 4 and 5, we utilized DDQN in our proposed SARL-based frameworks. The resulting large discrete action spaces, combined with our limited compute capability, led us to simulate small scenarios only and make simplifying assumptions, including the small number of predefined AP groupings in [67]. In the later parts of this dissertation, we dealt with this issue by (1) employing a different DRL algorithm, such as PPO in [50], and (2) transitioning to MARL in [84]. In future work, we may also implement a preprocessing stage to better handle the large action space. For instance, in [127], the discrete actions were first embedded in a continuous space before applying a k-nearest neighbor search to map them back to a discrete set. Such method may help in situations where the problem at hand requires a SARL-based solution or when utilizing SARL as a benchmark to evaluate MARL performance.

- High-mobility scenario

  In Chapter 5, the dynamic cell-free network was characterized by the combination of UE (de-)activation and user mobility. We only considered slow user movements in our simulations. However, high-mobility scenarios also hold practical relevance. In this case, the limited UE rate information in the state vector is likely to be insufficient. This must be extended to include other information, such as CSI and position-dependent ones, to capture the changes in the network for improved performance [80]. Therefore, the design of different state vector definitions, where we identify the necessary information to be tracked in the states, is a logical next step when considering more sophisticated use cases.

- Non-stationary RL environment

  While DRL is an effective tool for optimization, applying it on time-varying environments is non-trivial, and is in fact, an active research field on its own. Deploying RL in a live system and expecting it to continuously adapt to new environments require continual reinforcement learning or lifelong learning [36]. Several challenges arise from this setup, including possible system breakdowns during exploration, as well as catastrophic forgetting [128]. Specifically, whenever a change in the environment is detected, the agent re-enters an exploration

phase that may be time-consuming in practice. We settle for suboptimal solutions during this period. However, in extreme cases, such performance degradation may lead to a system failure. Meanwhile, catastrophic forgetting causes the agent or DNN to forget behaviors learned from past experiences, which may occur given the sequential learning assumed in DRL. To combat these issues, a framework was proposed in [128] that relies on an ensemble of expert policies, with each corresponding to a specific environment, and implements a safety monitor that activates a default policy when needed to avoid system failures. In [129], a change point algorithm was proposed to detect changes in the environment. Although the above-mentioned complexities have not been considered in this dissertation, they present interesting research directions for optimizing non-static wireless networks.

List of Abbreviations

6G	sixth generation
AP	access point
CoMP	coordinated multipoint
CPU	central processing unit
CRAN	cloud radio access network
CSI	channel state information
CTDE	centralized training, decentralized execution
DDPG	deep deterministic policy gradient
DDQN	double deep Q-network
DNN	deep neural network
DQN	deep Q-network
DRL	deep reinforcement learning
DT	digital twin
ECDF	empirical cumulative distribution function
FDD	frequency-division duplex
FedAvg	federated averaging
FedPer	personalized federated learning
GAE	generalized advantage estimator
i.i.d.	independent and identically distributed
IoT	internet of things
JT	joint transmission
LP	linear programming
MARL	multi-agent reinforcement learning
MILP	mixed-integer linear programming
MIMO	multiple-input multiple-output
ML	machine learning
MMSE	minimum mean-squared error
MRC	maximum ratio combining
MRT	maximum ratio transmission
MSE	mean-squared error

PAE	power allocation error
PPO	proximal policy optimization
QoS	quality of service
ReLU	rectified linear unit
RL	reinforcement learning
SARL	single-agent reinforcement learning
SINR	signal-to-interference-plus-noise ratio
TD	temporal difference
TDD	time-division duplex
UE	user equipment

Notation

The following table describes the notation used throughout this dissertation.

x, $X \in \mathbb{C}$ $(\mathbb{R})$	Complex-valued (real-valued) scalar
$\mathbf{x} \in \mathbb{C}^{N\times 1}$ $(\mathbb{R}^{N\times 1})$	Complex-valued (real-valued) column vector with N elements
$\mathbf{X} \in \mathbb{C}^{N\times M}$ $(\mathbb{R}^{N\times M})$	Complex-valued (real-valued) matrix with N rows and M columns
$\mathcal{X}$	Set of numbers
$\mathbf{I}_N$	Identity matrix of size $N \times N$
$\lvert x\rvert$	Magnitude of a scalar
$\lVert\mathbf{x}\rVert$	L2 norm of a vector
$\lVert\mathbf{x}\rVert_0$	L0 norm of a vector
$\lvert\mathcal{X}\rvert$	Size of a set
$[\mathbf{X}]_{n,m}$	Element at the nth row and mth column of a matrix
$[\mathbf{X}]_{n,:}$	nth row of a matrix
$[\mathbf{X}]_{:,m}$	mth column of a matrix
$(.)^*$	Conjugate operation
$(.)^H$	Conjugate-transpose operation
$\mathbb{E}\{.\}$	Expected value of a random variable
$\mathcal{CN}(\mu, \sigma^2)$	Complex Gaussian distribution with mean μ and variance σ^2
$\Re(.)$	Real part of a complex variable
$\Im(.)$	Imaginary part of a complex variable
$\text{len}(\mathcal{B})$	Length of an object $\mathcal{B}$

Bibliography

[1] Ericsson, "Ericsson Mobility Report," Nov. 2024. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report

[2] R. N. Clarke, "Expanding mobile wireless capacity: The challenges presented by technology and economics," *Telecommunications Policy*, vol. 38, no. 8-9, pp. 693-708, 2014.

[3] T. L. Marzetta, "Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas," *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3590-3600, Nov. 2010.

[4] E. Björnson, J. Hoydis and L. Sanguinetti, "Massive MIMO Networks: Spectral, Energy, and Hardware Efficiency," *Foundations and Trends in Signal Processing*, vol. 11, no. 3-4, pp 154-655, 2017.

[5] J. G. Andrews, X. Zhang, G. D. Durgin and A. K. Gupta, "Are we approaching the fundamental limits of wireless network densification?," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 184-190, Oct. 2016.

[6] M. Thurfjell, M. Ericsson and P. de Bruin, "Network Densification Impact on System Capacity," *IEEE 81st Vehicular Technology Conference (VTC Spring)*, Glasgow, UK, 2015, pp. 1-5.

[7] S. Zhou, M. Zhao, X. Xu, J. Wang and Y. Yao, "Distributed wireless communication system: a new architecture for future public wireless access," *IEEE Communications Magazine*, vol. 41, no. 3, pp. 108-113, Mar. 2003.

[8] S. Venkatesan, A. Lozano and R. Valenzuela, "Network MIMO: Overcoming Intercell Interference in Indoor Wireless Systems," *41st Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California, USA, 2007, pp. 83-87.

[9] R. Irmer et al., "Coordinated multipoint: Concepts, performance, and field trial results," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 102-111, Feb. 2011.

[10] W. Choi and J. G. Andrews, "Downlink performance and capacity of distributed antenna systems in a multicell environment," *IEEE Transactions on Wireless Communications*, vol. 6, no. 1, pp. 69-73, Jan. 2007.

[11] G. Interdonato, P. Frenger and E. G. Larsson, "Scalability Aspects of Cell-Free Massive MIMO," *IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1-6.

[12] A. Lozano, R. W. Heath and J. G. Andrews, "Fundamental Limits of Cooperation," *IEEE Transactions on Information Theory*, vol. 59, no. 9, pp. 5213-5226, Sep. 2013.

[13] E. Nayebi, A. Ashikhmin, T. L. Marzetta and H. Yang, "Cell-Free Massive MIMO systems," *49th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California, USA, 2015, pp. 695-699.

[14] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson and T. L. Marzetta, "Cell-Free Massive MIMO Versus Small Cells," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1834-1850, Mar. 2017.

[15] Z. Chen and E. Björnson, "Channel Hardening and Favorable Propagation in Cell-Free Massive MIMO With Stochastic Geometry," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5205-5219, Nov. 2018.

[16] S. Shamai and B. M. Zaidel, "Enhancing the cellular downlink capacity via co-processing at the transmitting end," *IEEE 53rd Vehicular Technology Conference (VTC Spring)*, Rhodes, Greece, 2001, pp. 1745-1749.

[17] Ö. T. Demir, M. Masoudi, E. Björnson and C. Cavdar, "Cell-Free Massive MIMO in Virtualized CRAN: How to Minimize the Total Network Power?," *IEEE International Conference on Communications (ICC)*, Seoul, Republic of Korea, 2022, pp. 159-164.

[18] J. Zheng et al., "Mobile Cell-Free Massive MIMO: Challenges, Solutions, and Future Directions," *IEEE Wireless Communications*, vol. 31, no. 3, pp. 140-147, Jun. 2024.

[19] H. A. Ammar, R. Adve, S. Shahbazpanahi, G. Boudreau and K. V. Srinivas, "User-Centric Cell-Free Massive MIMO Networks: A Survey of Opportunities, Challenges and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 611-652, 2022.

[20] G. Interdonato, E. Björnson, H. Q. Ngo, P. Frenger and E. G. Larsson, "Ubiquitous cell-free Massive MIMO communications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 197, 2019.

[21] Ö. T. Demir, E. Björnson and L. Sanguinetti, "Foundations of User-Centric Cell-Free Massive MIMO," *Foundations and Trends in Signal Processing*, vol. 14, no. 3-4, pp. 162-472, 2021.

[22] W. Zeng, Y. He, B. Li and S. Wang, "Pilot Assignment for Cell Free Massive MIMO Systems Using a Weighted Graphic Framework," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 6190-6194, Jun. 2021.

[23] E. Nayebi, A. Ashikhmin, T. L. Marzetta, H. Yang and B. D. Rao, "Precoding and Power Optimization in Cell-Free Massive MIMO Systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 7, pp. 4445-4459, Jul. 2017.

[24] E. Björnson and L. Sanguinetti, "Making Cell-Free Massive MIMO Competitive With MMSE Processing and Centralized Implementation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 77-90, Jan. 2020.

[25] H. Q. Ngo, G. Interdonato, E. G. Larsson, G. Caire and J. G. Andrews, "Ultradense Cell-Free Massive MIMO for 6G: Technical Overview and Open Questions," *Proceedings of the IEEE*, vol. 112, no. 7, pp. 805-831, Jul. 2024.

[26] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson and T. L. Marzetta, "Cell-Free Massive MIMO: Uniformly great service for everyone," *IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Stockholm, Sweden, 2015, pp. 201-205.

[27] S. Buzzi and C. D'Andrea, "Cell-Free Massive MIMO: User-Centric Approach," *IEEE Wireless Communications Letters*, vol. 6, no. 6, pp. 706-709, Dec. 2017.

[28] E. Björnson and E. Jorswieck, "Optimal Resource Allocation in Coordinated Multi-Cell Systems," *Foundations and Trends in Communications and Information Theory*, vol. 9, no. 2–3, pp 113-381, 2013.

[29] N. Kato, B. Mao, F. Tang, Y. Kawamoto and J. Liu, "Ten Challenges in Advancing Machine Learning Technologies toward 6G," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 96-103, Jun. 2020.

[30] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255-260, 2015.

[31] M. Zaher, Ö. T. Demir, E. Björnson and M. Petrova, "Learning-Based Downlink Power Allocation in Cell-Free Massive MIMO Systems," *IEEE Transactions on Wireless Communications*, vol. 22, no. 1, pp. 174-188, Jan. 2023.

[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[33] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," *NIPS Deep Learning Workshop*, Lake Tahoe, USA, 2013.

[34] W. Zhao, J. P. Queralta and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey," *IEEE Symposium Series on Computational Intelligence (SSCI)*, Canberra, Australia, 2020, pp. 737-744.

[35] X. Lin et al., "6G Digital Twin Networks: From Theory to Practice," *IEEE Communications Magazine*, vol. 61, no. 11, pp. 72-78, Nov. 2023.

[36] K. Khetarpal, M. Riemer, I. Rish and D. Precup, "Towards Continual Reinforcement Learning: A Review and Perspectives," *Journal of Artificial Intelligence Research*, vol. 75, pp. 1401-1476, 2022.

[37] N. C. Luong et al., "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133-3174, 2019.

[38] C. -X. Wang et al., "On the Road to 6G: Visions, Requirements, Key Technologies, and Testbeds," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 905-974, 2023.

[39] Y. S. Nasir and D. Guo, "Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239-2250, Oct. 2019.

[40] M. Botvinick et al., "Reinforcement Learning, Fast and Slow," *Trends in Cognitive Sciences*, vol. 23, no. 5, pp. 408-422, 2019.

[41] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. Agüera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, USA, 2017.

[42] E. Björnson and L. Sanguinetti, "Scalable Cell-Free Massive MIMO Systems," *IEEE Transactions on Communications*, vol. 68, no. 7, pp. 4247-4261, Jul. 2020.

[43] C. D'Andrea and E. G. Larsson, "User Association in Scalable Cell-Free Massive MIMO Systems," *54th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, USA, 2020, pp. 826-830.

[44] H. Q. Ngo, L. -N. Tran, T. Q. Duong, M. Matthaiou and E. G. Larsson, "On the Total Energy Efficiency of Cell-Free Massive MIMO," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 1, pp. 25-39, Mar. 2018.

[45] F. Riera-Palou, G. Femenias, A. G. Armada and A. Pérez-Neira, "Clustered Cell-Free Massive MIMO," *IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

[46] O. Zhou, J. Wang and F. Liu, "Average Downlink Rate Analysis for Clustered Cell-Free Networks with Access Point Selection," *IEEE International Symposium on Information Theory (ISIT)*, Espoo, Finland, 2022, pp. 742-747.

[47] M. Guenach, A. A. Gorji and A. Bourdoux, "Joint Power Control and Access Point Scheduling in Fronthaul-Constrained Uplink Cell-Free Massive MIMO Systems," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2709-2722, Apr. 2021.

[48] H. V. Nguyen et al., "On the Spectral and Energy Efficiencies of Full-Duplex Cell-Free Massive MIMO," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 8, pp. 1698-1718, Aug. 2020.

[49] C. F. Mendoza, S. Schwarz and M. Rupp, "Cluster Formation in Scalable Cell-free Massive MIMO Networks," *16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Thessaloniki, Greece, 2020, pp. 62-67.

[50] C. F. Mendoza, S. Schwarz and M. Rupp, "User-Centric Clustering in Cell-Free MIMO Networks using Deep Reinforcement Learning," *IEEE International Conference on Communications (ICC)*, Rome, Italy, 2023, pp. 1036-1041.

[51] C. Han et al., "Green radio: radio techniques to enable energy-efficient wireless networks," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 46-54, Jun. 2011.

[52] J. Wu, Y. Zhang, M. Zukerman and E. K. -N. Yung, "Energy-Efficient Base-Stations Sleep-Mode Techniques in Green Cellular Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 803-826, 2015.

[53] F. Han, S. Zhao, L. Zhang and J. Wu, "Survey of Strategies for Switching Off Base Stations in Heterogeneous Networks for Greener 5G Systems," *IEEE Access*, vol. 4, pp. 4959-4973, 2016.

[54] T. V. Chien, E. Björnson and E. G. Larsson, "Optimal Design of Energy-Efficient Cell-Free Massive Mimo: Joint Power Allocation and Load Balancing," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 5145-5149.

[55] T. X. Vu, S. Chatzinotas, S. ShahbazPanahi and B. Ottersten, "Joint Power Allocation and Access Point Selection for Cell-free Massive MIMO," *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020, pp. 1-6.

[56] G. Femenias, N. Lassoued and F. Riera-Palou, "Access Point Switch ON/OFF Strategies for Green Cell-Free Massive MIMO Networking," *IEEE Access*, vol. 8, pp. 21788-21803, 2020.

[57] J. García-Morales, G. Femenias and F. Riera-Palou, "Energy-Efficient Access-Point Sleep-Mode Techniques for Cell-Free mmWave Massive MIMO Networks With Non-Uniform Spatial Traffic Density," *IEEE Access*, vol. 8, pp. 137587-137605, 2020.

[58] S. Jung and S. -E. Hong, "Performance analysis of Access Point Switch ON/OFF schemes for Cell-free mmWave massive MIMO UDN systems," *International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Republic of Korea, 2021, pp. 644-647.

[59] C. F. Mendoza, S. Schwarz and M. Rupp, "Deep Reinforcement Learning for Dynamic Access Point Activation in Cell-Free MIMO Networks," *25th International ITG Workshop on Smart Antennas (WSA)*, French Riviera, France, 2021, pp. 1-6.

[60] H. A. Ammar, R. Adve, S. Shahbazpanahi, G. Boudreau and K. V. Srinivas, "Distributed Resource Allocation Optimization for User-Centric Cell-Free MIMO Networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 5, pp. 3099-3115, May 2022.

[61] F. Riera-Palou and G. Femenias, "Decentralization Issues in Cell-free Massive MIMO Networks with Zero-Forcing Precoding," *57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, Illinois, USA, 2019, pp. 521-527.

[62] S. Kim et al., "Revisiting the Coverage Boundary of Multi-CPU Cell-Free Massive MIMO: CPU Cooperation Aspect," *IEEE International Conference on Communications (ICC)*, Rome, Italy, 2023, pp. 1022-1028.

[63] S. Kim et al., "CPU-Cooperative Power Control Scheme for Scalable Cell-Free Massive MIMO Systems," *IEEE Transactions on Wireless Communications*, vol. 23, no. 10, pp. 13904-13919, Oct. 2024.

[64] P. Biswas, R. K. Mallik and K. B. Letaief, "Optimal Access Point Centric Clustering for Cell-Free Massive MIMO Using Gaussian Mixture Model Clustering," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 675-687, 2024.

[65] J. Wang, L. Dai, L. Yang and B. Bai, "Rate-Constrained Network Decomposition for Clustered Cell-Free Networking," *IEEE International Conference on Communications (ICC)*, Seoul, Republic of Korea, 2022, pp. 2549-2554.

[66] J. Wang, L. Dai, L. Yang and B. Bai, "Clustered Cell-Free Networking: A Graph Partitioning Approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 8, pp. 5349-5364, Aug. 2023.

[67] C. F. Mendoza, S. Schwarz and M. Rupp, "Deep Reinforcement Learning for Spatial User Density-based AP Clustering," *IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, Oulu, Finland, 2022, pp. 1-5.

[68] T. C. Mai, H. Q. Ngo and L. -N. Tran, "Energy-efficient power allocation in cell-free massive MIMO with zero-forcing: First order methods," *Physical Communication*, vol. 51, 2022.

[69] S. Chakraborty, Ö. T. Demir, E. Björnson and P. Giselsson, "Efficient Downlink Power Allocation Algorithms for Cell-Free Massive MIMO Systems," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 168-186, 2021.

[70] S. Mosleh, H. Almosa, E. Perrins and L. Liu, "Downlink Resource Allocation in Cell-Free Massive MIMO Systems," *International Conference on Computing, Networking and Communications (ICNC)*, Honolulu, Hawaii, USA, 2019, pp. 883-887.

[71] S. Chakraborty, E. Björnson and L. Sanguinetti, "Centralized and Distributed Power Allocation for Max-Min Fairness in Cell-Free Massive MIMO," *53rd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, USA, 2019, pp. 576-580.

[72] Y. Zhao, I. G. Niemegeers and S. H. De Groot, "Power Allocation in Cell-Free Massive MIMO: A Deep Learning Method," *IEEE Access*, vol. 8, pp. 87185-87200, 2020.

[73] C. D'Andrea, A. Zappone, S. Buzzi and M. Debbah, "Uplink Power Control in Cell-Free Massive MIMO via Deep Learning," *IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Le gosier, Guadeloupe, 2019, pp. 554-558.

[74] Y. Zhang, J. Zhang, Y. Jin, S. Buzzi and B. Ai, "Deep Learning-Based Power Control for Uplink Cell-Free Massive MIMO Systems," *IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, 2021, pp. 1-6.

[75] N. Rajapaksha, K. B. Shashika Manosha, N. Rajatheva and M. Latva-Aho, "Deep Learning-based Power Control for Cell-Free Massive MIMO Networks," *IEEE International Conference on Communications (ICC)*, Montreal, Québec, Canada, 2021, pp. 1-7.

[76] W. Li, W. Ni, H. Tian and M. Hua, "Deep Reinforcement Learning for Energy-Efficient Beamforming Design in Cell-Free Networks," *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Nanjing, China, 2021, pp. 1-6.

[77] Y. Al-Eryani, M. Akrout and E. Hossain, "Multiple Access in Cell-Free Networks: Outage Performance, Dynamic Clustering, and Deep Reinforcement Learning-Based Design," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 4, pp. 1028-1042, Apr. 2021.

[78] L. Luo et al., "Downlink Power Control for Cell-Free Massive MIMO With Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 6772-6777, Jun. 2022.

[79] M. Rahmani et al., "Deep Reinforcement Learning-based Power Allocation in Uplink Cell-Free Massive MIMO," *IEEE Wireless Communications and Networking Conference (WCNC)*, Austin, Texas, USA, 2022, pp. 459-464.

[80] X. Zhang, M. Kaneko, V. An Le and Y. Ji, "Deep Reinforcement Learning-based Uplink Power Control in Cell-Free Massive MIMO," *IEEE 20th Consumer Communications & Networking Conference (CCNC)*, Las Vegas, Nevada, USA, 2023, pp. 567-572.

[81] T. Schaul, J. Quan, I. Antonoglou and D. Silver, "Prioritized Experience Replay," *4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

[82] S. Zhou, Y. Cheng, X. Lei and H. Duan, "Deep Deterministic Policy Gradient With Prioritized Sampling for Power Control," *IEEE Access*, vol. 8, pp. 194240-194250, 2020.

[83] C. F. Mendoza, M. Kaneko, M. Rupp and S. Schwarz, "Accelerated Deep Reinforcement Learning for Uplink Power Control in a Dynamic Cell-Free Massive MIMO Network," *IEEE Wireless Communications Letters*, vol. 13, no. 6, pp. 1710-1714, Jun. 2024.

[84] C. F. Mendoza, M. Kaneko, M. Rupp and S. Schwarz, "Enhancing the Uplink of Cell-Free Massive MIMO through Prioritized Sampling and Personalized Federated Deep Reinforcement Learning," accepted in *IEEE Transactions on Cognitive Communications and Networking*.

[85] Y. Zhao, I. G. Niemegeers and S. M. H. De Groot, "Dynamic Power Allocation for Cell-Free Massive MIMO: Deep Reinforcement Learning Methods," *IEEE Access*, vol. 9, pp. 102953-102965, 2021.

[86] I. M. Braga, R. P. Antonioli, G. Fodor, Y. C. B. Silva and W. C. Freitas, "Decentralized Joint Pilot and Data Power Control Based on Deep Reinforcement Learning for the Uplink of Cell-Free Systems," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 957-972, Jan. 2023.

[87] M. G. Arivazhagan, V. Aggarwal, A. K. Singh and S. Choudhary, "Federated Learning with Personalization Layers," *arXiv preprint arXiv:1912.00818*, 2019.

[88] S. Eddine Hajri, J. Denis and M. Assaad, "Enhancing Favorable Propagation in Cell-Free Massive MIMO Through Spatial User Grouping," *IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Kalamata, Greece, 2018, pp. 1-5.

[89] R. A. Howard, *Dynamic Programming and Markov Processes*. MIT Press, 1960.

[90] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[91] F. AlMahamid and K. Grolinger, "Reinforcement Learning Algorithms: An Overview and Classification," *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Ontario, Canada, 2021, pp. 1-7.

[92] H. v. Hasselt, A. Guez and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *30th AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, USA, 2016.

[93] H. Hasselt, "Double Q-learning," *24th Annual Conference on Neural Information Processing Systems (NIPS)*, 2010.

[94] L. Lin, "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching," *Machine Learning*, vol. 8, pp. 293–321, 1992.

[95] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap and G. Wayne, "Experience Replay for Continual Learning," *33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[96] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 2015.

[97] B. T. Polyak and A. B. Juditsky, "Acceleration of Stochastic Approximation by Averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838-855, Jul. 1992.

[98] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," *4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

[99] V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," *13th Annual Conference on Neural Information Processing Systems (NIPS)*, Denver, Colorado, USA, 1999.

[100] G. E. Uhlenbeck and L. S. Ornstein, "On the Theory of the Brownian Motion," *Physical Review*, vol. 36, pp. 823-841, 1930.

[101] J. Hollenstein, S. Auddy, M. Saveriano, E. Renaudo and J. Piater, "Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance," *Transactions on Machine Learning Research*, 2022.

[102] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[103] J. Schulman, P. Moritz, S. Levine, M. I. Jordan and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

[104] W. Ejaz et al., "A comprehensive survey on resource allocation for CRAN in 5G and beyond networks," *Journal of Network and Computer Applications*, vol. 160, Jun. 2020.

[105] S. Schwarz, "Remote Radio Head Assignment and Beamforming in Dynamic Distributed Antenna Systems," *IEEE International Conference on Communications (ICC)*, Kansas City, Missouri, USA, 2018, pp. 1-6.

[106] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming, version 2.2," Jan. 2020. [Online]. Available: https://cvxr.com/cvx

[107] M. ApS, "The MOSEK optimization toolbox for MATLAB manual, version 8.1.0.82," Oct. 2019. [Online]. Available: https://docs.mosek.com/8.1/toolbox/index.html

[108] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[109] Z. -Q. Luo and S. Zhang, "Dynamic Spectrum Management: Complexity and Duality," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 1, pp. 57-73, Feb. 2008.

[110] Q. Shi, M. Razaviyayn, Z. -Q. Luo and C. He, "An Iteratively Weighted MMSE Approach to Distributed Sum-Utility Maximization for a MIMO Interfering Broadcast Channel," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331-4340, Sep. 2011.

[111] M. Razaviyayn, M. S. Boroujeni and Z. -Q. Luo, "A stochastic weighted MMSE approach to sum rate maximization for a MIMO interference channel," *IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Darmstadt, Germany, 2013, pp. 325-329.

[112] D. Lee et al., "Spatial modeling of the traffic density in cellular networks," *IEEE Wireless Communications*, vol. 21, no. 1, pp. 80-88, Feb. 2014.

[113] Claussen, "Efficient modelling of channel maps with correlated shadow fading in mobile radio systems," *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications*, Berlin, Germany, 2005, pp. 512-516.

[114] A. V. Kini, M. Hosseinian, M. -i. Lee and J. Stern-Berkowitz, "Reevaluating cell wraparound techniques for 3D channel model based system-level simulations," *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, Louisiana, USA, 2015, pp. 171-176.

[115] M. Bashar, K. Cumanan, A. G. Burr, M. Debbah and H. Q. Ngo, "On the Uplink Max–Min SINR of Cell-Free Massive MIMO Systems," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2021-2036, Apr. 2019.

[116] T. H. L. Dinh et al., "Towards an Energy-Efficient DQN-based User Association in Sub6GHz/mmWave Integrated Networks," *17th International Conference on Mobility, Sensing and Networking (MSN)*, Exeter, United Kingdom, 2021, pp. 645-652.

[117] C. Tsinos, S. Spantideas, A. Giannopoulos and P. Trakadas, "Over-the-Air Computation with Quantized CSI and Discrete Power Control Levels," *Wireless Communications and Mobile Computing*, vol. 2023, 2023.

[118] R. Lowe et al., "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," *31st Annual Conference on Neural Information Processing Systems (NIPS)*, Long Beach, California, USA, 2017.

[119] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, vol. 55, pp. 895–943, 2022.

[120] J. K. Gupta, M. Egorov and M. Kochenderfer, "Cooperative Multi-agent Control Using Deep Reinforcement Learning," *16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, São Paulo, Brazil, 2017.

[121] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli and S. Whiteson, "Counterfactual Multi-Agent Policy Gradients," *32nd AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA, 2018.

[122] D. Horgan et al., "Distributed Prioritized Experience Replay," *6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[123] H. De Oliveira, M. Kaneko and L. Boukhatem, "Federated Multi-Agent Deep Reinforcement Learning for Intelligent IoT Wireless Communications," *IEEE Vehicular Technology Magazine*, in press, Aug. 2024.

[124] H. De Oliveira, M. Kaneko and L. Boukhatem, "Smart Band Association for Wireless IoT Networks: a Personalized Federated Multi-Agent Deep Reinforcement Learning Approach," *IEEE 100th Vehicular Technology Conference (VTC Fall)*, Washington DC, USA, 2024.

[125] Y. Zhang et al., "Towards Cost-Efficient Federated Multi-agent RL with Learnable Aggregation," *28th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Taipei, Taiwan, 2024.

[126] X. Huang, S. Leng, S. Maharjan and Y. Zhang, "Multi-Agent Deep Reinforcement Learning for Computation Offloading and Interference Coordination

in Small Cell Networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9282-9293, Sep. 2021.

[127] G. Dulac-Arnold et al., "Deep Reinforcement Learning in Large Discrete Action Spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[128] P. Hamadanian, M. Schwarzkopf, S. Sen and M. Alizadeh, "Demystifying Reinforcement Learning in Time-Varying Systems," *arXiv preprint arXiv:2201.05560*, 2022.

[129] S. Padakandla, P. K. J. and S. Bhatnagar, "Reinforcement learning algorithm for non-stationary environments," *Applied Intelligence*, vol. 50, pp. 3590–3606, 2020.