IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 May 2025, accepted 30 May 2025, date of publication 3 June 2025, date of current version 17 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3576190

RESEARCH ARTICLE

Decentralized and Joint Resource Allocation, Beamforming, and Beamcombining for 5G Networks With Heterogeneous MARL

ALA'A AL-HABASHNA[1,2], (Member, IEEE), JON MENARD[1], GABRIEL WAINER[1], (Senior Member, IEEE), AND GARY BOUDREAU[3], (Senior Member, IEEE)

[1]Systems and Computer Engineering, Carleton University, 1125 Colonel By Dr., Ottawa, ON, K1S 5B6, Canada
[2]School of Computing and Informatics, Al Hussein Technical University, Amman, Jordan
[3]Ericsson Canada, 349 Terry Fox Dr., Kanata, ON, K2K 2V6, Canada

Corresponding author: Ala'a Al-Habashna (alaaalhabashna@cunet.carleton.ca; alaa.alhabashna@htu.edu.jo)

This work was supported in part by Ericsson Canada and in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

ABSTRACT In this paper, we propose a novel Multi-Agent Reinforcement Learning (MARL) -based paradigm for distributed and joint resource allocation, beamforming (BF), and beam combining of uplink transmissions in 5G networks. The proposed paradigm employs two types of heterogenous agents that learn to perform and optimize different tasks in order to achieve the main objective of the system, as well as the objective of the individual agents. In the proposed paradigm, UEs can be multi-agents that optimize their own resource allocation and BF. In addition to these multi agents (i.e., UEs), the BS is a different type of agent that optimizes the combining of UEs' transmissions. We developed three different implementations of our proposal using three different MARL algorithms: Independent Q Learners (IQL), Multi-Agent Deep Deterministic Policy Gradient (MADDPG), and QTRAN. Various experiments were conducted to validate the usability of our proposal. Our results show that the proposed paradigm can successfully optimize the task of joint resource allocation, beamforming, and combining. Furthermore, we provide a comparative analysis of the three different implementations, highlighting noteworthy insights into the strengths and limitations of fully distributed algorithms, such as IQL, in comparison to algorithms employing the Centralized Training with Decentralized Execution (CTDE) framework, exemplified by QTRAN and MADDPG.

INDEX TERMS Deep reinforcement learning, multi-agent reinforcement learning, distributed resource allocation, beamforming, 5G.

I. INTRODUCTION

As the number of mobile devices is exponentially increasing, the Fifth Generation (5G) wireless communication systems are aiming to improve the network capacity by a thousand folds. Furthermore, 5G wireless communication systems are designed to improve spectrum efficiency by up to 15 times [1], [2]. This is to support new applications that require high data rates, high reliability, and low latency in areas such as entertainment, healthcare, agriculture, and industry. An example of these applications in the field of entertainment is ultra-high-definition streaming, which requires data rates around 25 Mbps with less than 100 milliseconds of latency. Other mission-critical applications such as self-driving cars require 50-100 Mbps data rates.

To increase the efficiency of the radio spectrum, which has become a valuable and scarce resource, new technologies have been introduced such as network densification [3], Device to-Device (D2D) communications [4], [5], [6], [7], [8], [9], massive MIMO [10], and other multiple-tiered

The associate editor coordinating the review of this manuscript and approving it for publication was Francisco Rafael Marques Lima.

network architectures with a co-channel deployment [11], [12]. Although such technologies and network architectures provide promising solutions and help meet the increasing demand, they complicate the power and resource allocation problem. Cognitive Radio (CR); where secondary users can sense the spectrum and exploit vacant spectrum bands, is another example where power and resource allocation can be a very challenging task [13], [14], [15], [16]. Other scenarios might involve multiple operators that share the same infrastructure. In such network architectures, the different traffic loads, transmission powers, channel access priorities, and the provision of peer-to-peer communication complicate the dynamics of resource allocation.

Recent studies have shown that optimal resource allocation in multi-tier networks is generally an NP-hard problem, and hence computationally expensive. Traditionally, resource allocation is performed in a centralized manner (e.g., at the Base-Station (BS)). Several attempts have been made to develop centralized algorithms to optimize resource allocation of such networks. For example, in [17], the authors investigate the communication performance of a Space-Air-Ground Integrated Network (SAGIN) that utilizes an active Reconfigurable Intelligent Surface (RIS) and Non-Orthogonal Multiple Access (NOMA) with CR capabilities. The authors aim to maximize the weighted sum mean rate and weighted sum mean energy efficiency for the secondary network. To achieve their objective, they propose an alternating optimization framework based on the Block Coordinate Ascent (BCA) technique to solve this complex problem. In [18], the authors study the problem of efficient resource allocation in opportunistic CR networks, where secondary users opportunistically access the licensed spectrum of a primary network when it is sensed to be idle. The paper proposes a cross-layer resource allocation approach where a fusion center plays a central role in resource allocation and selecting SUs for spectrum sensing. A near-optimal greedy algorithm is proposed to solve the resource allocation problem.

While the proposed centralized methods above might work for small-sized systems, such paradigms that are based on centralized methods for solving resource allocation problems might not be scalable for the aforementioned scenarios. First, the computational complexity grows significantly as the number of nodes in the network increases. Moreover, a great deal of signaling and communication overhead are usually needed to perform the task at a single node, as this requires a single entity to have a global view of the network and its nodes to manage resource allocation. This requires continuous data collection from all nodes or subsystems (sensing information, channel state information, etc.), leading to excessive communication load. This might lead to network congestion or delays in data transmission, and hence, impact optimization efficiency.

Distributed or semi-distributed resource allocation methods can provide more efficient and scalable solutions to solve power and resource allocation problems for complex network scenarios. This is due to the reduced amount of signaling and computational complexity. In such solutions, multiple nodes can perform resource allocation independently. In the extreme case, resource allocation is performed at the edge of the network by the User Equipment (UE) themselves. However, the limited global view in such methods may lead to inefficient or conflicting decisions.

In recent years, Reinforcement Learning (RL) and Deep RL (DRL) [19] have been increasingly utilized to address various problems and challenges in the field of 5G wireless communication systems. This includes problems such as power and resource allocation, handover, and caching and offloading. RL is a learning process where an agent finds an optimal policy by periodically making decisions, observing the results, and adjusting its strategy. This allows agents to learn the characteristics of the environment, avoid the exhaustive search in the Action Space (AS) of the problem, and can provide near-optimal solutions to maximize the end-user performance (e.g., SINR and data rate). This is particularly useful in cases with non-convex optimization problems.

Although RL and DRL are very useful for solving problems with a single agent (e.g., centralized resource allocation), they might not be suitable to develop algorithms for distributed execution. Multi-Agent Reinforcement Learning (MARL), on the other hand, can provide a promising solution for such scenarios with distributed execution, as it involves multiple agents that learn by interacting within a common environment. Each time step, an agent makes a decision to achieve a predetermined goal that maximizes expected future return. The goal in this case would be for agents to learn a policy such that all agents together achieve the goal of the system. This is very suitable for the case of distributed resource allocation, where UEs need to achieve their data rate requirements and also maximize the performance of the network.

In this paper, we propose a novel MARL-based paradigm for joint resource allocation, beamforming (BF), and combining for uplink transmissions in 5G networks. The proposed paradigm employs two types of heterogenous agents that learn to perform and optimize different tasks in order to achieve the main objective of the system, as well as the objective of the individual agents. In the proposed paradigm, UEs can be multi-agents that optimize their own resource allocation and BF. In addition to these multi agents (i.e., UEs), the BS is a different type of agent that optimizes the combining of UEs' transmissions. We developed three different implementations of our proposal using three different MARL algorithms: Independent Q Learners (IQL), Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [20], and QTRAN [21].

A. RELATED WORK

In recent years, RL has rapidly gained attraction for solving diverse problems in 5G networks, such as radio resource management problems, including the intricacy of resource

allocation. Resource allocation, a well-defined problem in wireless communication, encompasses a variety of issues such as resource provisioning, load balancing, scalability, and energy management [22]. In this context, Cong and Lang proposed using a centralized RL algorithm to improve spectrum in dynamic spectrum access environment [23]. In an environment with several secondary users, the authors trained a deep recurrent neural network at a centralized node to allocate the Resource Block Groups (RBGs) used in each secondary user's transmission. The results showed that the proposed RL models significantly increased spectrum utilization and minimized collisions between primary and secondary users when compared to existing centralized algorithms. Likewise, He, et al., used RL to train a centralized scheduler for optimizing TDMA/FDMA scheduling in time-varying interference alignment networks [24]. The centralized RL deep Q-network (DQN) collected channel state information of users requesting service and scheduled the optimal users to actively transmit during the current radio frame. Comparing the RL algorithm to two previously proposed non-RL-based power allocation models [25], [26] showed a considerable performance improvement in both sum rate and energy efficiency.

While centralized RL shows encouraging results, it is unable to address several challenges including scalability, computational complexity, and large signaling overhead. A distributed approach allows computations to be split among multiple nodes, reducing computational complexity, while being more scalable. Furthermore, communication is often limited between nodes reducing the signaling overhead seen when communicating with a central node, e.g., BS. In addition to the above, distributed resource allocation can give the UEs in the network the ability to be autonomous, leveraging scenarios such as CR Networks and multi-operator networks. As such, researchers have been considering new methods that exploit MARL for such problems. Sana et al show the advantages of decentralized MARL over centralized solutions [27]. They compared several centralized approaches against their decentralized MARL algorithm for managing multiple UEs requesting downlink services from nearby BSs. During testing, the authors' solution outperformed the sum-rate performance of centralized baseline methods by almost 40%. In [28], the authors address the requirements of ultra-reliable low-latency communications, and the high throughput demands of mobile broadband users in 5G networks by proposing a distributed MARL-based approach for power and RBG allocation. Their results showed a 21-fold increase in throughput with minimal impact on latency or reliability over a priority-based proportional fairness algorithm for fixed power allocation. Similarly, Simsek et al. proposed using MARL for decentralized power allocation to increase the throughput for individual femtocell UEs [29]. Using a simulation scenario based on the 3GPP Technical Specification Group, and considering each BS as a decentralized agent, the simulation results illustrated MARL's ability in determining the optimal power levels used for each RBG. Thus, reducing interference and increasing throughput for femtocell UEs when compared to traditional algorithms. The work in [30] and [31] further explore the use of MARL for resource allocation in UAV to ground communications and vehicle-to-vehicle communication with both papers showing communication with a centralized system can be eliminated while still satisfying communication requirements.

Despite the promising results achieved by decentralized MARL, it is important to acknowledge its limitation in complex environments. Distributed approaches often have limited access to global information, resulting in partial observability. Furthermore, they can suffer from the non-stationarity of the environment problem, as the actions of the different agents in the environment change unexpectedly. Centralize Training with Decentralized Execution (CTDE) is a notable approach that integrates advantages from both fully centralized paradigms and fully distributed MARL. During training, a centralized network enables information to be shared among all agents, providing stability, and increasing coordination. During execution, each agent operates its fully trained local network without the need to know the global state of the network; deploying the local policy developed during training to still coordinate with unobserved agents. Cao et al. used CTDE MARL to address the task of offloading problem in Mobile-Edge Computing. The authors performed numerous simulations, with several different parameter combinations, which all demonstrated that the MARL-based algorithm could reduce computational delay by 33.38%, increase the channel access success rate by 14.88%, and increase channel utilization by 3.24% with respect to standard centralized RL methods. Motivated by the challenge of managing spectrum resource allocation, Li, Guo, and Xuan proposed using MARL to reduce interference caused by D2D communications. The authors considered an environment with a single BS performing downlink transmission to several D2D pairs. Traditional centralized solutions require exchanging global information, causing significant signal overhead, and additional interference. For this reason, the authors developed a RL algorithm called Neighbor-Agent Actor Critic (NAAC) that takes advantage of CTDE. Simulation results from the paper showed that NAAC reduced D2D outage probability and signaling overhead, improved the sum throughput rate, and outperformed other decentralized approaches including Uncoupled Stochastic Learning Algorithms, Deep Q-learning, and traditional actor-critic.

In [32], we considered the problem of resource allocation for uplink transmission by many UEs in a cell and proposed a MARL-based solution that follows the CTDE paradigm. We showed that a distributed joint policy can be learned by the UEs to manage resource allocation. The trained policy maximized the sum rate of users while achieving the minimum data rate requirements of users during each radio subframe. Additionally, we showed that MARL provides efficient

and near-optimal solutions for computationally infeasible problems [33].

In this paper, we propose a novel MARL-based paradigm for joint resource allocation, BF, and combining for uplink transmissions in 5G networks. The proposed paradigm employs two types of heterogenous agents that learn to perform and optimize different tasks in order to achieve the main objective of the system, as well as the objective of the individual agents. In the proposed paradigm, UEs can be multi-agents that optimize their own resource allocation and BF. In addition to these multi agents (i.e., UEs), the BS is a different type of agent that optimizes the combining of UEs' transmissions. We developed three different implementations of our proposal using three different MARL algorithms. Two of the adopted MARL algorithms, namely the MADDPG [20] and QTRAN [21] frameworks, follow the CTDE approach. The third method, namely IQL, falls under the independent-learners approach. Each algorithm is discussed in Section IV. We ran various simulations to evaluate the performance of the three implementations. The results obtained show that the implementations of the proposed paradigm can successfully learn local policies to maximize the cell data rate and achieve the minimum data rate requirements for each UE. The trained models allow the UEs to achieve the tasks above for any scenario (any distribution of the UEs in the cell).

B. CONTRIBUTIONS

In the following, we summarize the main contributions of this paper,

- A system model, and an optimization model for the problem of joint resource allocation, BF, and combining, to maximize the cell data rate, and achieve a minimum data rate for each UE.
- A novel heterogenous MARL-based paradigm for joint resource allocation, BF, and combining.
- Three different implementations of the proposed paradigm, with three algorithms, namely, IQL, MADDPG, and QTRAN.
- Performance evaluation of the three different implementations.

The rest of this paper is organized as follows: Section II discusses our system model and problem formulation. Section III presents our proposed heterogenous MARL paradigm and provides an overview of MARL and the adopted MARL algorithms for implementing our solution. Section IV presents the simulation environment, a comparative analysis of the MARL-based solutions against two centralized algorithms, and the results for training and testing with the three adopted MARL algorithms. Section V states the conclusion and future work.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a one-cell network, with the BS (also known as gNodeB) at the center of the cell, and multiple UEs distributed in the cell, as can be seen in Fig. 1. The UEs in the cell have data to upload, and they need to share the radio channel. The channel is equally divided into U RBGs. Both the gNodeB and UEs have uniform linear arrays of L_{BS} and L_{UE} antennas, respectively. The received signal at the gNodeB from the m^{th} UE's transmission on the u^{th} RBG, can be written as [34],

$$y_{m,u} = \boldsymbol{f}_{BS,m}^{H} \boldsymbol{H}_{m,u} \boldsymbol{f}_{m} x_{m,u} + \sum_{q \neq m} \boldsymbol{f}_{BS,m}^{H} \boldsymbol{H}_{q,u} \boldsymbol{f}_{q} x_{q,u} + \boldsymbol{f}_{BS,m}^{H} \boldsymbol{n}_{m} \tag{1}$$

where $x_{m,u}$ and $x_{q,u}$ represent the transmitted signals from the m^{th} and q^{th} users, respectively, to the gNodeB on the u^{th} RBG, $\boldsymbol{H}_{m,u}$ represents the channel matrix from the m^{th} UE to gNodeB at the u^{th} RBG, $\boldsymbol{H}_{q,u}$ represents the channel matrix from the q^{th} UE to gNodeB on the u^{th} RBG, $\boldsymbol{f}_m$ and $\boldsymbol{f}_q$ represent the transmitter's (UE) BF vector for the m^{th} and q^{th} users, respectively, $\boldsymbol{f}_{BS,m}$ and $\boldsymbol{f}_{BS,q}$ represent the receiver's combining vector for the transmission of the m^{th} and q^{th} users, respectively, and $\boldsymbol{n}_m \sim N(0, \sigma_n^2 \boldsymbol{I})$ is the received AWGN vector at the gNodeB (sampled from a complex normal distribution with zero mean and σ_n^2 variance). The first term in (1) represents the signal from the intended UE, while the second term represents the interference from other UEs served by the same gNodeB transmitting on the same RBG.

We adopt the geometric channel model in [34], [35], and [36], where the channel model between the m^{th} UE and the gNodeB at the u^{th} RBG can be represented as,

$$\boldsymbol{H}_{m,u} = \frac{\sqrt{L_{UE}}\sqrt{L_{BS}}}{PL_m} \sum_{\rho=1}^{N_m^{\rho}} g_{m,u}^{\rho} \boldsymbol{\alpha}_{BS}\left(\theta_{BS,m}^{\rho}\right) \boldsymbol{\alpha}_{UE}^{H}\left(\theta_{m,BS}^{\rho}\right), \tag{2}$$

where N_m^{ρ} is the number of paths for the transmission from the m^{th} UE to the gNodeB, $g_{m,u}^{\rho}$ is the gain of the p^{th} path from m^{th} UE to the gNodeB at the u^{th} RBG. The path amplitudes are assumed to be Rayleigh distributed, i.e., $g_{m,u}^{\rho} \sim (0, P_R)$, with P_R the average power gain. $\theta_{m,BS}^{\rho}$ and $\theta_{BS,m}^{\rho}$ are the Angle of Departure (AOD) and Angle of Arrival (AoA), respectively, of the p^{th} path from m^{th} UE to the gNodeB, $\boldsymbol{\alpha}_{UE}\left(\theta_{m,BS}^{\rho}\right)$ and $\boldsymbol{\alpha}_{BS}\left(\theta_{BS,m}^{\rho}\right)$ are the array responses of the AOD and AoA, respectively, and PL_m is the average pathloss between the m^{th} UE and gNodeB. It is worth mentioning that in MIMO systems, particularly in uplink transmission, the physical locations of the sender (UE) and the receiver (gNodeB) significantly impact both the AoA and AoD. AoA and AoD are fundamentally geometric concepts determined by the spatial positioning between the transmitting and receiving antennas. Changing the relative positions of the UE and BS directly alters the geometric paths that signals follow, impacting both AoA and AoD. Hence, we propose a system in Section IV that indirectly employs the location of the m^{th} UE within the cell (x_m and y_m) relative to the location of the gNodeB (x_{BS} and y_{BS}) as the state of the UE.

For simplicity, we omitted the time index in the previous equations, even though the variables mentioned above

vary with each Transmission Time Interval (TTI). However, we will explicitly include the time index moving forward. The received SNIR from the m^{th} UE on the u^{th} RGB (assuming that the UE transmits on that RGB) is given by [37],

$$\gamma_{m,u}(t) = \frac{p_{m,u}(t)\left|\boldsymbol{f}_{BS,m}^{H}(t)\,\boldsymbol{H}_{m,u}(t)\boldsymbol{f}_{m}(t)\right|^{2}}{\sum\limits_{q\neq m} p_{q,u}(t)\left|\boldsymbol{f}_{BS,m}^{H}(t)\,\boldsymbol{H}_{q,u}(t)\boldsymbol{f}_{q}(t)\right|^{2}+\sigma_{n}^{2}\left\|\boldsymbol{f}_{BS,m}(t)\right\|^{2}} \tag{3}$$

where $p_{m,u}(t)$ and $p_{q,u}(t)$ are the power of the signal transmitted by the m^{th} and q^{th} users, respectively, to the gNodeB on the u^{th} RBG. Furthermore, the achieved data rate by the m^{th} UE can be calculated as,

$$R_{m}(t)=\sum_{u=1}^{U}\delta_{m,u}(t)\,B\log_{2}\left(1+\gamma_{m,u}(t)\right), \tag{4}$$

where $\delta_{m,u}(t)\epsilon 0, 1\}$ is 1 for that RBG if the m^{th} UE transmits on it at time t, and 0 otherwise, and B is the bandwidth of the RBG. If $R_{m,u}(t)$ is the data rate achieved by the m^{th} UE on the u^{th} RBG at radio subframe t, $R_m(t)$ in (4) is the aggregate data rate of the m^{th} UE on all the RBGs it transmits on at t.

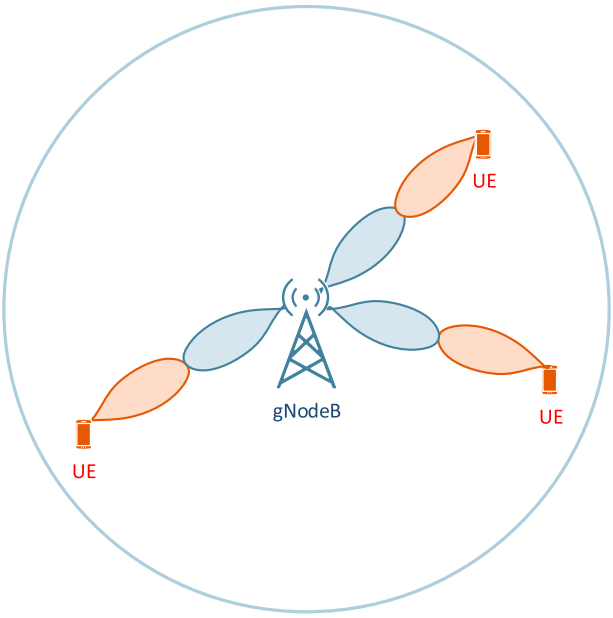

FIGURE 1. Illustration of the system model; UEs performing uplink transmission (withBF) to a BS (using combining).

The goal is to maximize the sum data rate of the UEs in the network at each TTI, while also maintaining a minimum data rate for all UEs. The above problem can be formulated as,

$$\underset{p_{m,u}(t),\boldsymbol{f}_{BS,m}(t),\boldsymbol{f}_{m}(t)}{\operatorname{argmax}}\sum\nolimits_{m\in\{1,2,\ldots M\}}R_{m}(t), \tag{5}$$

Subject to,

$$\boldsymbol{f}_{m}(t)\in\boldsymbol{F}_{UE}, \tag{6}$$

$$\boldsymbol{f}_{BS,m}(t)\in\boldsymbol{F}_{BS}, \tag{7}$$

$$R_{m}(t)\geq R_{min}, \tag{8}$$

where, m is UE's index, $\boldsymbol{F}_{UE}$ is the BF codebook, $\boldsymbol{F}_{BS}$ is the combining codebook, and R_{min} is the minimum data rate that needs to be achieved by each UE every subframe.

Unfortunately, the above problem is NP hard. Hence, we propose a MARL-based solution in this paper.

III. PROPOSED HETEROGENEOUS MARL SOLUTION AND IMPLEMENTATIONS

A. PROPOSED SOLUTION

In this section, we present the proposed heterogeneous-MARL solution. Considering the system model in Section II, we outline the key components and assumption of our approach. As previously mentioned, we consider the UEs as individual agents within the network cell. The gNodeB is assumed to have access to the coordinates of the served UEs, which are used to compute the angles between each UE and the gNodeB (as detailed in Section IV). These angles are utilized to generate an environment status report that captures the relative angular positions of all UEs with respect to the gNodeB. This report is then broadcasted to all UEs either periodically or in an event-driven manner, such as when a UE's location changes. The UEs utilize the status report to retrieve the environment states required to select the next action.

We implement our proposal with three different algorithms. One of them is fully distributed (IQL) and the other two follow the CTDE approach, i.e., MADDPG and QTRAN. In the case of IQL, both training and execution are performed in a decentralized manner at the UE level. On the other hand, with CTDE algorithms, the training process is conducted centrally at the BS, where fully observable state-action information is readily available. Once training is completed, the developed local models are distributed to each UE for local execution. During any subsequent executions, the UEs do not receive any information about the action taken by other UEs. Instead, the UEs rely solely on their local policies to select actions. Hence during execution, the agents are fully decentralized.

In addition to the above, the BS itself acts as another type of agent (and hence, the heterogeneous system) responsible for isolating incoming UE transmissions based on the AOA. For either IQL or the CTDE algorithms, training is completed using the same information available in the environment status report.

B. ADOPTED ALGORITHMS FOR IMPLEMENTATION

1) A QUICK REVIEW OF MARL

RL is a subdomain of machine learning focused on training an agent to find an optimal *policy* through interactions with a mutable environment. At each time step, t, the agent observes the environment states, s_t, and selects an action, a_t, to execute. The executed action transitions the environment into the next state, s_{t+1}. A reward is used to evaluate the environment's

new state. A state-action pair (s_t, a_t) refers to the chosen action at a specific state. The agent uses a *policy* to strategically determine the action to execute at a certain state [38]. The agent's objective is to develop a *policy* to select the best action at each state.

One strategy used by the agent is to use value function (V-function), $V_\pi(s)$, which estimates the utility of being in particular state. However, to evaluate the complete utility of a state, the agent must consider not only the immediate reward received, but also the rewards obtained onward through all future timesteps as well. The cumulative discounted reward, denoted by W, takes into account the immediate reward as well as all future rewards:

$$W_t = w_t + \gamma w_{t+1} + \gamma^2 w_{t+2} + \ldots \gamma^T w_{t+T}. \quad (9)$$

Here gamma, γ, represents the discount factor, where a value between 0 and 1 determines the importance of future rewards [38]. As such, the V-function gives the expected cumulative discounted reward that is obtainable at a certain state,

$$V_\pi(s_t) = \mathrm{E}_\pi[W_t|s_t], \quad (10)$$

where $\mathrm{E}_\pi[.]$ is the expected value.

The Q-function, $Q_\pi(s,a)$, is an extension of $V_\pi(s)$ into an action-value approximation, and it evaluates state-action pairs [39]. Each state-action approximation is known as a Q-value, and is defined by,

$$Q_\pi(s_t, a_t) = \mathrm{E}_\pi[W_t|s_t, a_t]. \quad (11)$$

The Bellman's equation computes the optimal Q function in a recursive manner, instead of summing over the considered time steps, as follows,

$$Q^*_\pi(s_t, a_t) = \mathrm{E}_{\pi,s'}[w_{s,a,s'} + \gamma \max_{a'} Q^*_\pi(s', a')|s_t, a_t]. \quad (12)$$

The Q-values, Q^*, are calculated by summing the reward for the current state-action pair and discounted future reward for the state-action pair in the next timestep. Selecting the maximum Q-value is based on the principle that it corresponds to the state-action pair with the highest expected reward for the subsequent state, which incorporates the maximum Q-value for its own subsequent state. This recursive approach allows the agent to consider all future outcomes for each state-action pair and make decisions that will maximize the long-term reward. The agent's *policy* can utilize the Q-function to determine the best action. Usually, the action with the highest expected Q-value is selected. A DQN is an advanced algorithm utilizing Deep Neural Networks (DNNs) to approximate and map Q-values of actions at each state. With each interaction in the environment, the agent acquires experience, which provides the true action-value for the selected state-action. A DNN is trained by minimizing the loss between the predicted Q-values and the true values received by the environment. Throughout training, DQN continues to execute actions in the environment, providing additional experience to further refine the DNN's approximated values. Once DQN has been successfully trained, it can be utilized by the agent as its Q-function.

MARL is an extension of standard RL to accommodate environments that include two or more agents. Furthermore, the environment's reward is structured depending on the goal of the system. In competitive environments where each agent needs to maximize its individual goal, an individualized reward is assigned to each agent, whereas in cooperative environments, a global reward may be assigned communally to promote collaboration between the agents. In cooperative environments, collaboration between agents is essential in order to achieve the system's goal. The global reward is an evaluation of the group's performance, rather than any agent's individual performance. However, in certain scenarios, individual rewards may also be utilized alongside the global reward to facilitate individuality among agents. As with RL, an agent must interact with the environment to gain experience, which can be used to develop its *policy*. The agent's *local policy* will be used by an agent to take actions to collaborate with or compete against the *local policies* of other agents in the environment. When agents are learning independently and each agent is oblivious to the actions of the other agents during training, this could lead to the non-stationarity of the environment problem; making it challenging to train multiple agents simultaneously in a common environment [20], [40], [41].

We implemented our MARL-based solution with three different MARL algorithms: IQL, MADDPG, and QTRAN. IQL is a simple and scalable decentralized framework. However, its partial observability could lead to the non-stationarity problem in complex environments. QTRAN and MADDPG are CTDE-based algorithms for fully observable centralized networks to alleviate unstable learning and increase convergence speed, while including decentralized networks for distributed execution after training is complete.

2) IQL

Independent learning is a common method used to implement MARL. All agents are unaware of each other in the environment and learn their policies individually. One of the most established independent learning algorithms is IQL, proposed in [42] and [43], where each agent uses its own Q-function to select the best action at each state. Agents can be trained using DQN. IQL is simple and can be trained in a decentralized manner, allowing for parallel training of multiple agents. Training an independent learner is the same as training a single Q-learner, where the agent performs an action, observes updated environment states, obtains a reward, and updates its Q-function to reflect the reward without considering the actions taken by other agents. However, the global reward can be used to reflect the performance of all agents (or the system) caused by all agents' actions. With each agent impacting the state of the environment, estimating the Q-value for a state-action pair can be challenging. Independent learning is appealing as it is simple, scalable, does not

require communication between agents and works for many multi-agent systems [43]. However, IQL's simplicity also exposes some problems. Independent learners' main drawback is their susceptibility to the non-stationarity problem. Agents try to optimize their policies while other agents are doing the same. These simultaneous-policy updates make the environment intractable causing the non-stationarity problem in some cases.

3) MADDPG

An actor-critic architecture is an RL algorithm consisting of two separate DNNs learning in parallel for an agent. The first DNN, the actor, uses an action-value approximation *policy* to select the best action given the state of the agent. The second DNN, the critic, trains to estimate its agent's value-function considering the current state and the next state resulting from the actor's selected action. The critic is used to guide the actor's action-value. Deep Deterministic Policy Gradient (DDPG) is a popular RL algorithm based on the actor-critic architecture [44]. MADDPG created by Lowe et al. [20] extends DDPG for MARL by adapting the critic network to be centralized. As such MADDPG follows CTDE. In MADDPG, each agent has its own actor and critic networks. The actor is fully decentralized, without any global information or central communication as it only receives states which will be available during execution. However, the critic is centralized with a fully observable set of states and chosen actions for each agent in the environment. The critic is trained to estimate its agent's local value-function considering the current states and actions of all agents, as well as the cumulative value-function for any future states. The critic network is trained by minimizing the Mean Squared Error (MSE) between predicted values, $\hat{y}$, and actual values, y,

$$MSE = \frac{1}{2}\sum \left(y - \hat{y}\right)^2, \tag{13}$$

where $\hat{y}$ is the critic's predicted state-value, and y is calculated as the sum of the immediate reward for the state action reward and the discounted reward for the next timesteps [20], given by,

$$y = w\left(\mathrm{s}_t, \mathrm{a}_t\right) + \gamma V_\pi\left(s_{t+1}\right), \quad \hat{\mathrm{y}} = V_\pi\left(s_t\right). \tag{14}$$

The actor network is trained using the critic's predicted state value as feedback for the chosen action, such that the error is calculated by minimizing the negative mean of the critic's predicted values for a mini batch of training experiences [20],

$$actor\ loss = -mean\left(V_\pi\left(s_t\right)\right). \tag{15}$$

Reducing the negative mean teaches the actor to select actions that will maximize the positive mean, thereby leading to future selected actions producing a higher overall value-function evaluation from the critic. The fully observable critic network is able to optimize its paired actor network to pick the optimal action for the joint *policy*. Additionally, it can facilitate faster convergence toward an effective joint-action policy by leveraging centralized training. Specifically, by observing and incorporating the action tendencies of other agents, it encourages the selection of complementary actions, thereby promoting coordinated behavior among agents. The critic network also helps minimizing unstable learning which can lead to the non-stationarity problem. As an CTDE algorithm, the actor network can be deployed in a completely decentralized manner, while still adhering to the joint-action *policy* even with its partial observability.

4) QTRAN

QTRAN is a proposed CTDE algorithm which uses Value Function Factorization (VFF) to discretize individual agent's contributions toward the join-action function [21]. As each joint-action transitions the environment into the next step, the reward reflects all actions, not explicitly the individual contributions from any single agent. VFF allows an agent to learn their local action-value function from the joint-reward, such that their own optimization leads to the optimization of the joint action-value function. As such, even during execution when the agents are decentralized, the optimal joint-action will be executed simply by each agent following their individual action-value function, without reference to the joint one. QTRAN's VFF employs CTDE for its three types of interconnected DNN estimators:

- Each agent's individual action-value network $f_q : (\tau_i, a_i) \rightarrow Q_i$,
- Centralized joint action-value network $f_r : (\tau, a) \rightarrow Q_{\mathrm{jt}}$, and
- Centralized state-value network $f_v : \tau \rightarrow V_{\mathrm{jt}}$.

where τ_i represents the agent's observation, and Q_{jt} and V_{jt} represent the joint action-value and state-value functions, respectively. An individual action-value network is assigned to each agent in the environment and is used to take actions using the agent's local observations and learned Q-value mapping, unlike the other two networks. The single centralized join-action value network is used to combine the chosen action from each agent's individual network and computes an approximated global Q-values for the joint-action taken. Simultaneously, the state-value network evaluates the combination of each agent's observed states. This eliminates the partial observability of the other two networks and is used to calculate the loss. Furthermore, the loss between the predicted global Q-values and the sum of obtained reward is combined. The combined loss is backpropagated through the centralized networks and into each agent's individual action-value network. The loss calculated by the centralized networks pushes the agent's decentralized networks towards Q-values optimal for the joint-action while maintaining partial observability. After training, each agent's Q-value matches the joint-action value-network, allowing the networks to be fully disconnected from the centralized networks. The individual networks will continue to collaborate with the joint-action, even with completely decentralized execution. QTRAN's advanced VFF enables the efficient discovery

of complex joint optimal actions in MARL environments, even as complexity grows exponentially with each additional agent. A summary of the similarities and differences of the three algorithms is presented in Table 1.

TABLE 1. A summary of the similarities and differences of the three MARL algorithms.

Aspect	IQL [44]	MADDPG [45]	QTRAN [21]
Learning paradigm	Value-based (Q-learning)	Actor-Critic (Policy-based + Value-based)	Value-based (Q-learning with joint-action value decomposition)
Centralized training	No (fully decentralized)	Yes (CTDE)	Yes (CTDE)
Decentralized execution	Yes	Yes	Yes
Cooperation assumption	No; Agents learn independently	Supports cooperative settings	Designed for cooperative settings
Policy type	ε-greedy Q-learning	Deterministic policies (continuous actions)	ε-greedy or greedy policy derived from learned Q-values
Non-stationarity handling	Weak; treats other agents as environment	Improved by centralized critic observing all agents' policies	Stronger; centralized training mitigates non-stationarity
Network architecture	Individual feed-forward Q-networks per agent	Actor and critic networks per agent; central critic has full observability	Individual Q-networks + Centralized joint Q-network
Stability and Convergence in complex interactions	Unstable due to independent updates	Better stability through centralized critic	Improved stability; consistent factorization ensures convergence

IV. RESULTS

We ran various simulations to evaluate the performance of the proposed solutions in multiple scenarios. In this section, we present and discuss the results obtained.

A. ENVIRONMENT SETUP

Our simulation environment is derived from the system model described in Section II, where there is a single cell with a centralized BS serving several UEs. This environment will be considered for model training and testing for each of the MARL solutions proposed in Section IV. L_{UE} and L_{BS} equal 2 and 8 antennas, respectively. Each UE is considered an agent, and the UEs in the cell are required to discover a joint-action policy in order to collectively maximize the goal outlined in Section II. An episode is set to the radio subframe. The beginning of each episode starts with the environment being reset as each UE is placed at a random set of coordinates within the cell's boundaries. This means that UEs are randomly placed within the cell at the start of each episode, enabling the algorithm to learn a policy adaptable to various UE distributions, in contrast to a fixed configuration. The BS is assumed to have the coordinates of the served UEs. This can be done either through uplink Sounding Reference Signals which can be used by the gNodeB to measure the UEs' coordinates or through the Location Management Function (LMF) which can compute the UEs' geographic coordinates. The BS then broadcasts a message that includes, S, the set of environment state variables. We have chosen S to be the set of angles from the base station to each UE in the environment, which can be given as,

$$s_m = tan^{-1}\left(\frac{y_m - y_{BS}}{x_m - x_{BS}}\right), \quad m = 1, 2, \ldots M, \tag{16}$$

where, s_m is the m th variable corresponding to the angle of the mth UE, x_m, y_m, x_{BS}, and y_{BS} are the mth UE coordinates, and the BS coordinates, respectively.

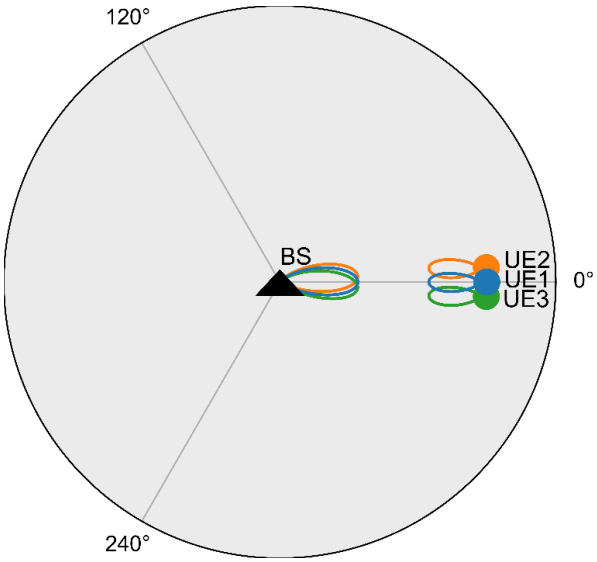

FIGURE 2. Environment configuration for worse case UE distribution.

An agent uses the environment state, S, as an input to its local policy to select an action from the available action set, A. The appropriate combination of selected transmission power on each RBG, and BF vector at the UE is crucial for adequate transmission of the data. Unless otherwise specified, we consider two transmission power options: a power level of 23 dBm (indicating that the UE chooses to transmit on the RBG) or no transmission on that RBG. An agent jointly performs resource allocation (selecting the power level on each RBG) and BF. In the environment considered, a UE will choose from multiple BF angles and RBGs, allowing an AS of A_{UE} for each UE to choose from. A_{UE}, i.e., the AS for each UE, can be calculated as,

$$A_{UE} = 2^{RBG} \times |\boldsymbol{F}_{UE}|, \tag{17}$$

where RBG is the number of RBGs and $|\boldsymbol{F}_{UE}|$ is the number of available BF vectors in the BF codebook.

We adopt a joint reward, w_{UEs}, for all UEs in the environment, which is represented as follows,

$$w_{UEs} = mean\left(R_{UE,RBG}\right) - \eta.std\left(R_{UE}\right), \tag{18}$$

where $R_{UE,RBG}$ is the data rate achieved by a UE on an RBG it transmits on (i.e., $R_{m,u}$, in Section II, when $\delta_{m,u} = 1$), while R_{UE} is the total throughput achieved by a UE over all RBGs (i.e., R_m in Section II). The parameter η is introduced to balance the trade-off between maximizing the aggregate data rate (via the first term) and promoting fairness (via the second term), ensuring that UEs are more likely to achieve their minimum required data rates. $mean\left(R_{UE,RBG}\right)$ is measured by aggregating the data rate achieved by all UEs on all RBGs and dividing it by the number of transmissions. For instance, if a UE transmits on two RBGs, this counts as 2 transmissions. In addition to the UEs, we consider the BS as an additional agent utilizing a separate instance, but same MARL algorithm as the UEs. The BS is responsible for selecting the combining vectors used for receiving the UEs' transmission (i.e., $f_{BS,m}(t) \in F_{BS}$). The BS uses the same state variables, S, to select an action. We have selected a combining codebook with 28 path vectors. The BS selects an action and receives the corresponding reward, w_{BS}, as follows,

$$w_{BS} = mean\left(R_{UE}\right), \tag{19}$$

This means that BS is rewarded for maximizing the aggregate data rate.

Training of the networks is supported through the use of *experience replay*. At every time step, the agents (UEs and the BS) select and execute their respective actions. The experience-replay buffer logs these experiences (consisting of the state, actions, and rewards) at each time step t. Random samples are then drawn from this buffer to perform minibatch training of the networks. This method enhances training stability and helps prevent convergence to local minima [45]. In IQL, each agent uses an individual feedforward neural network, consisting of an input layer sized to match the agent's observation space, three hidden layers of 128 neurons with ReLU activations, and an output layer representing a Q-value for each possible agent's action. QTRAN adopts a similar structure for each agent's personal Q-network but additionally employs a separate central joint action-value network with four hidden layers of 128 ReLU-activated neurons. MADDPG, on the other hand, assigns each agent both an actor and a critic network; the actor network follows the same structure as IQL, while each critic network has an input layer that includes all the actions and observations, and three hidden layers with 128 neuron each (with ReLU activations).

B. SIMULATION RESULTS

In the following, we discuss the results obtained. We start by comparing the proposed MARL-based solutions with two centralized methods. Afterwards, we compare the training and testing results obtained with our proposal with the three MARL-based algorithms.

1) COMPARATIVE ANALYSIS: CENTRALIZED SEARCH VS MARL

To evaluate the performance of our MARL-based solutions, we compare their performance to that of two centralized search algorithms. The first one is an exhaustive search (brute-force) strategy to systematically evaluate the entire AS, identifying the optimal joint action for the UEs and BS in the cell. The size of the joint AS, denoted by AS_{joint}, is calculated as follows,

$$AS_{joint} = \left(2^{RBG} \times |\boldsymbol{F}_{UE}| \times |\boldsymbol{F}_{BS}|\right)^{M}, \tag{20}$$

where RBG represents the number of RBGs, M corresponds to the number of UEs, $|\boldsymbol{F}_{UE}|$ and $|\boldsymbol{F}_{BS}|$ denote the numbers of available BF and combining vectors, to the UEs and BS, respectively. The second algorithm is a Semi-Greedy (SG) algorithm that makes local sub-optimal choices per device with the hope of approximating a good global solution. The algorithm iterates over all N devices, and for each device it evaluates all possible actions, and selects the best ζ portion of the actions for this device. Then, the algorithm searches over the space of the selected best actions per device to find the best joint action. As such the AS of this algorithm is:

$$AS_{SG} = \left(\zeta \times 2^{RBG} \times |\boldsymbol{F}_{UE}| \times |\boldsymbol{F}_{BS}|\right)^{M}. \tag{21}$$

In this subsection, we compare the results achieved by the centralized algorithms with the MARL-based algorithms. As can be seen in (20), AS_{joint} experiences a polynomial growth with each additional subchannel. In order to reduce the computational requirements for the brute-force approach, we limit the number of RBGs and UEs to three each, for this comparison. The number of BF and combining paths will remain at 4 and 28, respectively, as described in the *Environment Setup*. As such, the complete AS is equal to 719,323,126 combinations. Fig. 2 shows the selected test scenario for this comparison; a worst-case UE distribution, with the UEs positioned at the same location. This configuration needs the most coordination between UEs, while also requiring the optimal BF and combining angles to be chosen. The search algorithm explored the whole action space in 78 hours to find the optimal solution for this test scenario. The solution found resulted in all UEs achieving the same data rate of 86.66 Mbps. This is achieved by having the UEs transmitting on different RBGs. The SG algorithm (with $\zeta = 0.2$) also achieved the same data rate in 39 minutes, achieving a 99.17% reduction in search time.

The MARL algorithms were trained using multiple episodes, each with a random UE distribution. This means that UEs are randomly placed within the cell at the start of each episode, enabling the algorithm to learn a policy adaptable to various UE distributions, in contrast to a fixed configuration. Once the UE data rates stabilized, we concluded training and tested each algorithm on the test scenario.

IQL finished in 380'000 episodes, MADDPG in 370'000 episodes, and QTRAN in 370'000 episodes, each taking less than half an hour. Please note that the trained models can be used to solve any distribution of UEs, not just the test scenario. IQL, MADDPG, and QTRAN each obtained the same data of 86.66 Mbps, matching that achieved by the exhaustive search method. Interestingly, for the test scenario, the MARL algorithms did not select the same joint action. An important point here is that the biggest advantage of the MARL-based algorithms is not only that they were able to train the models in less time, but also the fact that such trained models are able to find plausible joint-actions for any scenario in the execution phase in a very short time (around 2 μs) which is suitable for the Time Transmission Interval (TTI) of 1 ms. On the other hand, the central algorithms, i.e., exhaustive search and semi-greedy algorithms, would still need 78 hours and 39 minutes, each time a joint decision needs to be found, which is not suitable for our time subframe (i.e., TTI).

2) MARL-BASED ALGORITHMS: TRAINING RESULTS

We trained MARL-based models with the three algorithms, i.e., IQL, MADDPG and QTRAN in three scenarios, each with a different number of UEs in the cell. Specifically, the scenarios encompassed 6 to 8 UEs performing uplink transmission on 5 available RBGs. An intentional decision to have more UEs than RBGs was selected to ensure the agents learn to coordinate transmissions on the same RGBs, where UEs might have to share the same RBG. During the training process, the agents were evaluated every 10'000 episodes, where an additional 1000 evaluation episodes were conducted. The evaluation episode is considered successful if all UEs obtain an individual minimum data rate of 15 Mbps. Training was deemed complete once the success percentage during the 1000 evaluation episodes reaches or exceeds 95%.

We utilized the number of training episodes required to reach the 95% success percentage threshold as a metric to determine the performance of each algorithm in terms of convergence speed. The training duration was not considered, as we found that this metric was extremely dependent on the specifications of the computer used to perform training, and the degree of concurrency during training.

Fig. 3 shows the data rate achieved (in the evaluation episodes) during training, for each algorithm. At the start of training, the agents have no prior experience and take seemly random actions in the environment. Consequently, the majority of UEs experience low data rates as their transmissions interfere with each other. However, there are instances where one or two UEs may have slightly better rates when, by chance, they choose favorable actions.

As training progresses, the agents gradually achieve better individual data rates, and consequently better mean data rate. Eventually, the agents of each algorithm learn a near-optimal joint policy, resulting in all UEs converging to a joint policy that maximizes the data rate with all UEs converging to similar data rates. Alongside the maximized mean, at least 95% of the time, all UEs obtain a minimum data rate of 15 Mbps.

Fig. 4 shows the mean data rate achieved by the UEs (for 7 UEs scenario) for each algorithm to compare the mean data rates obtained by the three algorithms. By the end of their training, the three algorithms achieve similar mean data rate values. However, the total number of episodes needed and data rates during evaluation differ between the algorithms. Furthermore, Fig. 5 showcases the success percentage obtained by each algorithm throughout training. Both Fig. 4 and Fig. 5 show that QTRAN is the fastest to converge, followed closely by MADDPG, while IQL requires the most training episodes. This pattern is not unique, as demonstrated in Fig. 6, which displays the number of training episodes for each algorithm across the various scenarios.

In addition to the previously described experiments, we also trained models with the 6-UE scenario, but with three power levels: no transmission, 13 dBm, and 23 dBm. The IQL model was trained for 6 million episodes and achieved a mean data rate of 137.9 Mbps but failed to reach the target success percentage of 95%, achieving a maximum success rate of 89% over the entire training period. In contrast, MADDPG successfully achieved a 95% success rate with an average data rate of 146.8 Mbps after approximately 5.75 million episodes. QTRAN reached the 95% success threshold with an average data rate of 102.7 Mbps in around 2.11 million episodes. However, when QTRAN was trained for 5.75 million episodes (same number of episodes as MADDPG) it achieved a higher mean data rate of 148.3 Mbps. QTRAN achieves superior convergence performance because its design is specifically intended to maintain consistent VFF, which promotes stable and reliable convergence. Furthermore, QTRAN's improved stability and convergence stems from its novel approach to VFF in cooperative MARL [21]. Instead of directly factorizing the joint action-value function under structural constraints, QTRAN aims to transform the original joint action-value function into a new, easily factorizable one that shares the same optimal joint actions. The improvement in the mean data rate achieved by these models can be attributed to two key factors. Firstly, the use of three discrete power levels, as opposed to two, enhances the flexibility for coordination among UEs. This increased flexibility enables more UEs to transmit simultaneously on the same resource block groups (RBGs), thereby improving overall spectrum utilization. Secondly, extended training durations, ranging from 5 to 6 million episodes, allow the models to explore the environment more thoroughly and converge to more effective transmission policies.

With each additional UE added, the environment complexity escalates, demanding the agents to coordinate an increased number of RBG reuse (multiple UEs transmitting on the same RBG). The complexity also increases with increasing the number of power levels, RBGs, or BF or combining vectors (due to increasing the joint action space). With the increased complexity, the advantages of CTDE become more apparent.

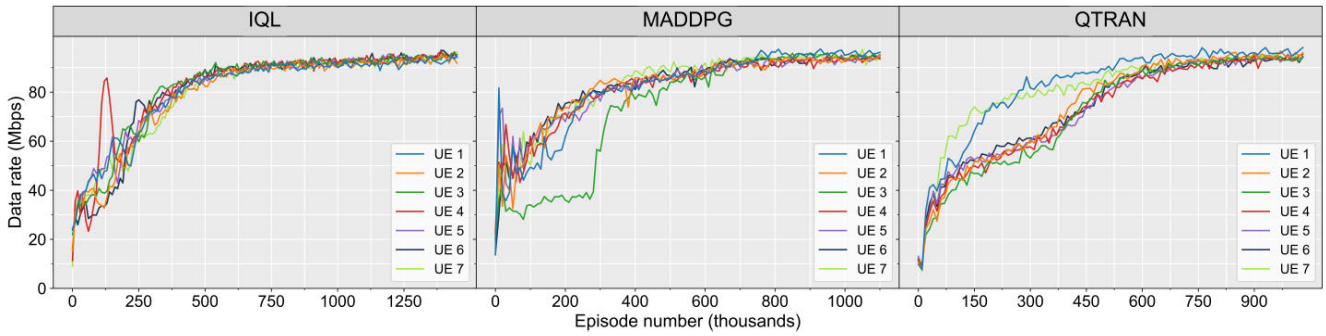

FIGURE 3. Data rates of each UE during training with IQL, MADDPG, and QTRAN (7 UEs).

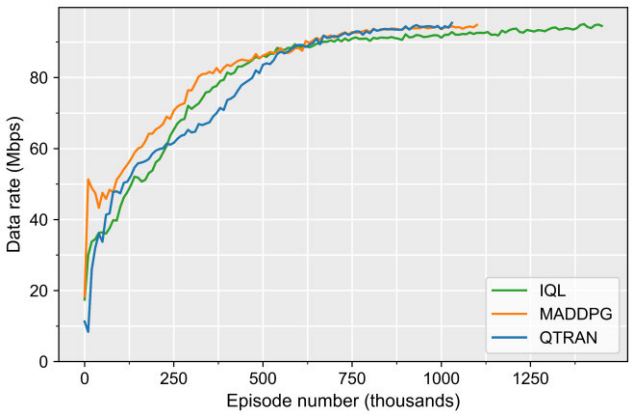

FIGURE 4. Mean data rate of all UEs during training with IQL, MADDPG, andQTRAN (7 UEs).

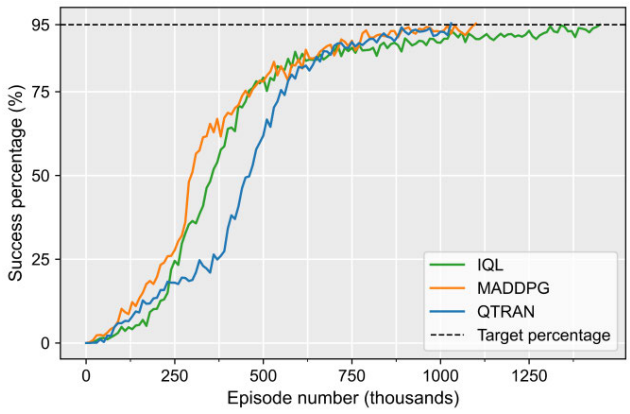

FIGURE 5. Percentage of the obtained successful episodes during training withIQL, MADDPG, and QTRAN (7 UES).

Both QTRAN and MADDPG were able to achieve the target success rate with fewer episodes. However, IQL initially started close to the others but began to require more training episodes with the added complexity. This is likely attributed to the non-stationarity of the environment problem, as the IQL algorithm does not consider or account for the policy of other agents, making coordination a challenging task. As the number of agents (or the AS in general) increases, coupled with the increased requirement to share RBGs, CTDE shows a clear advantage over completely decentralized algorithms when it comes to convergence speed.

3) MARL-BASED ALGORITHMS: TESTING RESULTS

Testing each algorithm consisted of running 1'000'000 tests after training was completed, each with a different distribution of the UEs within the cell. Fig. 7 shows the percentage of UEs that achieved a data rate that equals or exceeds the minimum threshold (15 Mbps). The results show that more than 99% of transmissions achieved data rates that equal or exceeded the threshold, i.e., less than 1% of the transmissions achieved data rates below the threshold. Furthermore, the two CTDE algorithms have a slightly higher success rate, with MADDPG having the highest. Additionally, Fig. 8 shows the minimum and maximum data rates achieved by each algorithm. IQL consistently has the highest maximum data rate, followed by QTRAN, and then MADDPG.

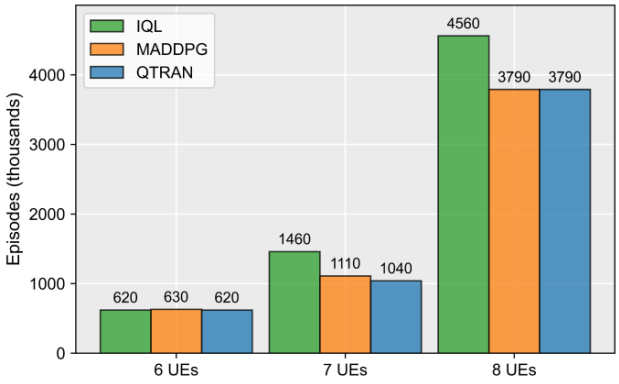

FIGURE 6. Training episodes to reach 95% success percentage.

An extremely high value, i.e., the 500 Mbps data rate seen in IQL, occurs when a greedy agent transmits on all 5 RBGs, thus starving other UEs of any data. This supports the results from Fig. 7, as MADDPG obtains a higher success rate as well as the lowest variance from minimum to maximum, suggesting a fairer policy, with fewer greedy agents.

Fig. 9 focuses on the mean data rate achieved by all UEs during the evaluation episodes. The margin of error values for

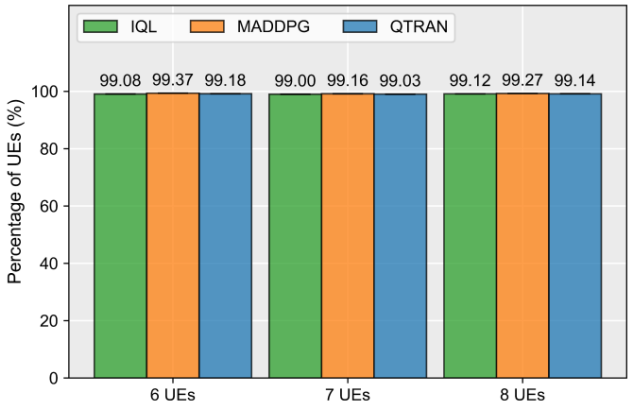

FIGURE 7. Percentage of UEs during testing that achieved data rate equal to or above threshold.

95% confidence interval are shown in the figure, however, they are invisible as these values are relatively too small. This shows that the mean values shown in the figure represent with high confidence the actual mean values. Initially with 6 UEs, QTRAN starts with the highest mean, followed by MADDPG and then IQL with the lowest. With 7 UEs, IQL and MADDPG begin to narrow the difference to QTRAN and by 8 UEs the three algorithms are within 0.03 Mbps of each other. The added complexity of each additional UE requires the algorithms to further refine their joint policy compared to the easier scenarios. In particular, they must increase the coordination between UEs, as greedy behavior will have more impact on the other UEs and cause increased likelihood of their transmissions missing the minimum data rate.

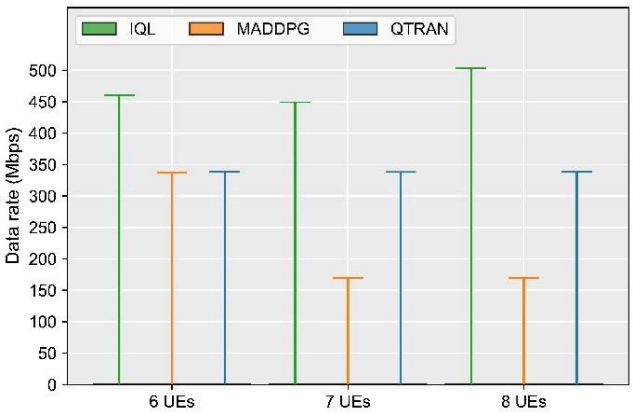

FIGURE 8. Data rate range (from minimum to maximum) from the test results.

As a result, the refined policies achieve a higher average compared to the previous ones, and by the final scenario, all algorithms converge to similar data rates. Recall from Fig. 6, the refined policies needed a higher number of episodes, and that with IQL significantly higher number of episodes were needed to achieve the same target success percentage as MADDPG and QTRAN. From these results, we can see that for the scenarios considered, IQL might be able to achieve similar data rates with significantly higher number of episodes. However, as the number of UEs increases, this might be infeasible, and the superiority of the CTDE algorithms will be more prevalent and much needed.

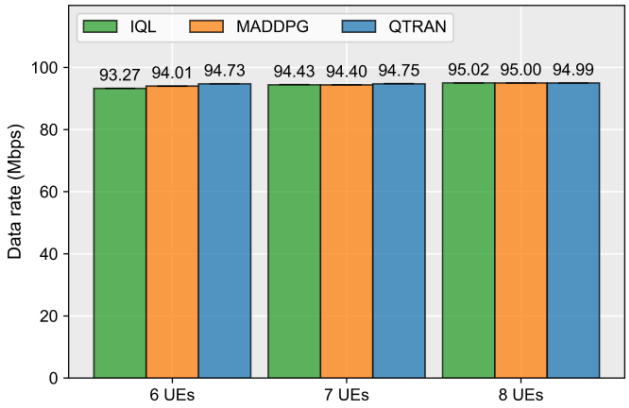

FIGURE 9. Mean data rates of all transmissions from the test results.

Fig. 10 also displays the mean data rate, however, only considering data rate values from the 1^{st} to 99^{th} percentile. This is to remove outliers and analyze the results of successful transmissions, i.e., ones with data rates equal to or above the threshold data rate. As with Fig. 9, QTRAN has the highest mean data rate, with MADDPG and IQL converging closer as the environment complexity increases and they have an increased training duration to refine their policies. These results agree with the findings in Fig. 9. Fig. 11 shows boxplots of the data rate values (1^{st} to 99^{th} percentile). The figure shows all the algorithms reach similar max and median data rates. However, when analyzing the minimum values, MADDPG has the highest values, followed by QTRAN with close values. IQL achieves the lowest minimum data rate of all algorithms. This is because CTDE algorithms can develop a policy resulting in fairer RBG allocation and better coordination between UEs. This also means that they can be used to produce more reliable policies where more UEs achieve their target data rates.

In conclusion, the testing results show that CTDE achieved higher reliability in meeting and surpassing the minimum data rate threshold, i.e., the percentage of transmissions that satisfied or exceeded the defined 15 Mbps minimum threshold. Also, QTRAN and MADDPG achieved overall similar or higher mean and higher minimum data rates while requiring shorter training durations. In contrast, IQL had the highest variability between minimum and maximum values and consistently had the lowest number of transmissions achieving a data rate equal to or above 15 Mbps. As a result, these findings suggest that CTDE algorithms provide a more reliable joint policy between UEs compared to IQL. This is attributed to the non-stationarity of the environment problem, where IQL agents are trained independently without knowing the actions of other agents in the environment. As previously mentioned,

while IQL might be able to achieve similar data rates with significantly higher number of episodes for the considered scenarios, this might be infeasible as the number of UEs grow, and the superiority of the CTDE algorithms will be more prevalent and much needed.

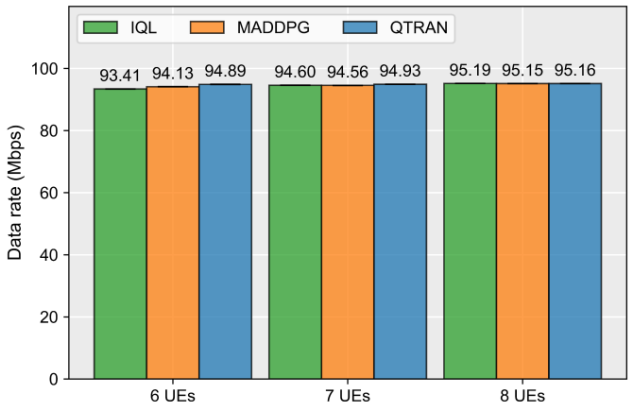

FIGURE 10. Mean data rates of all transmissions from the test results (1st-99th percentile).

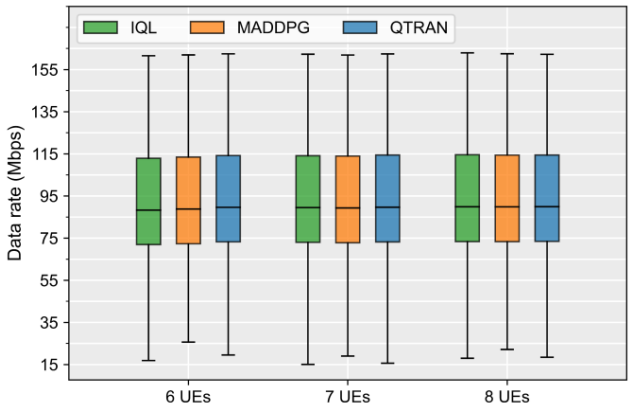

FIGURE 11. Boxplot of data rates of transmissions from the test results (1st-99th percentile).

As previously described, the MARL algorithms were trained over multiple episodes, with each episode featuring a random distribution of UEs. At the beginning of every episode, UEs were randomly positioned within the cell, enabling the algorithms to learn robust policies adaptable to diverse UE distributions, rather than a single fixed configuration. By employing multiple models, each tailored for a specific number of UEs, the proposed framework can effectively adapt to dynamic environments characterized by varying numbers and distributions of UEs within the cell.

4) SENSITIVITY ANALYSIS

In this subsection, we discuss the sensitivity analysis of the algorithms to hyper-parameter values, network architecture, and to the reward function.

Various hyperparameter configurations were evaluated, including learning rate, batch size, and buffer size. As a sample of such results, we present a summary of the results obtained with different hyper-parameter values for MADDPG in Table 2. The other algorithms exhibited similar performance trends. It is important to note that the choice of hyperparameters primarily influenced the convergence behavior of the algorithms; specifically, the number of episodes required to reach satisfactory performance, rather than significantly affecting the mean data rate.

Regarding the learning rate, smaller values typically resulted in slower convergence, whereas larger values accelerated the learning process but sometimes introduced instability, often manifested as oscillatory (ping-pong) behavior due to overshooting. Larger batch sizes, while computationally more expensive, generally led to more stable gradient descent and increased generalization by averaging the loss across more samples. They also allowed for slightly higher learning rates. This effect was especially beneficial in CTDE-based networks (QTRAN and MADDPG) where global observations increase the stability and effectiveness of larger batches during centralized training.

TABLE 2. Achieved results with different hyper-parameter values (MADDPG with 6 UEs).

Learning Rate	0.0001	0.00025	0.0005
Num. of episodes (thousands)	672	630	604
Mean date rate unit (Mbps)	92.74	94.538	92.042
Buffer Size	150'000	250'000	350'000
Num. of episodes (thousands)	804	630	851
Mean date rate unit (Mbps)	94.31	94.538	94.089

Regarding the replay-buffer size, larger buffers offered better performance by providing stability with a large set of past experiences (especially for more complex scenarios). However, overly large buffers slowed down the convergence of the algorithms. This is because a very large buffer retains experiences from outdated policies that no longer reflect the current agents' behaviors. With overly large replay buffer, new relevant experiences are diluted by a vast number of older experiences. This was more apparent in IQL where very large buffer sizes increased non-stationarity.

During our experimentation with varying the neural network architectures of the MARL-based algorithms, we observed that performance was moderately influenced by the network depth. Architectures with two or fewer hidden layers consistently underperformed, highlighting the need for deeper networks to effectively approximate complex functions and policies in high-dimensional state and action spaces. For IQL, a minimum of three hidden layers was necessary

to enable agents to learn accurate value functions from local observations. In QTRAN, incorporating three hidden layers in the joint Q-network significantly enhanced its representational capacity. However, increasing the depth beyond five layers introduced optimization challenges and elevated the risk of violating the network's factorization constraints. Similarly, for the MADDPG critic networks, three hidden layers proved effective in enabling the actor networks to learn more expressive policies. Nevertheless, increasing the number of layers beyond four led to greater sensitivity to non-stationarity and data inefficiency. In terms of width, we experimented with various configurations and found that using 128 nodes per hidden layer provided a favorable balance between learning performance and computational efficiency.

In designing the reward function, we explored the impact of varying the coefficient η associated with the standard deviation of the data rate, which was introduced to promote some degree of fairness among users and help the UEs meet the minimum data rate required. Through empirical evaluation, we found that setting $\eta = 1.5$ provided the best mean data rate while maintaining the target success rate. Lower values of η resulted in a higher mean data rate but at the cost of a reduced success rate, indicating that the algorithm prioritized maximizing throughput. Conversely, increasing η beyond 1.5 (e.g., 2) led to suboptimal outcomes in both mean data rate and success percentage, as the algorithm became overly focused on minimizing data rate variance, thereby compromising the efficient use of the available channel.

5) COMPLEXITY ANALYSIS OF THE CENTRALIZED TRAINING PHASE

In this subsection, we discuss the extra training complexity incurred by the central networks that follow the CTDE approach (i.e., QTRAN and MADDPG). QTRAN uses a centralized joint action-value network that takes the joint actions of all agents as an input and outputs a single scalar value [21]. Training complexity in this case grows linearly with the number of agents, M, primarily due to the increased input size affecting the first layer, i.e., $O(M)$. MADDPG learns a separate centralized critic for each agent [20]. Therefore, if there are M agents, there will be M critic networks. This implies a linear increase in the number of critic networks with the number of agents. Each centralized critic for each agent takes as input the states and actions of all M agents. This means that the input size to each network also grows linearly. As such, the complexity of the central network would exhibit $M \times O(M) = O(M^2)$ growth in complexity with increasing the number of agents.

Although the central networks in QTRAN and MADDPG incur extra training complexity that increases with the number of agents, it is important to note that these CTDE-based algorithms yield more reliable joint policies among UEs than IQL, mainly due to better handling of the environment's non-stationarity. While IQL may eventually achieve comparable data rates, it requires significantly more training episodes and struggles to converge and stabilize as the number of UEs or the action space grows (as can be seen in subsection IV-B.2). Moreover, this centralized training is conducted at the gNodeB, which is equipped with high-performance computational resources, making it well-suited for handling the complexity of multi-agent coordination and model optimization efficiently.

V. CONCLUSION

In this paper, we proposed a novel Multi-Agent Reinforcement Learning (MARL) -based paradigm for distributed and joint resource allocation, BeamForming (BF), and combining for uplink transmissions in 5G wireless networks. Our proposed solution employed two types of heterogenous agents that learn to perform and optimize different tasks in order to achieve the main objective of the system, as well as the objective of the individual agents.

In our solution, UEs can be multi agents that optimize their own resource allocation and BF. We developed three implementations of our proposal; each with a different MARL algorithm. The first implementation employs Independent Q Learners (IQL); an algorithm where both training and resource allocation is done in a distributed manner. The other two implementations are based on two Centralized Training with Distributed Execution (CTDE) -based algorithms, namely, Multi-Agent Deep Deterministic Policy Gradient (MADDPG), and QTRAN.

We executed various simulations to evaluate the performance of our proposals with the three algorithms. First, we compared the three MARL-based solutions with two centralized approaches that finds the optimal solution. All three solutions acquired the same optimal solution as the centralized search algorithms in a significantly reduced number of episodes and execution time. Furthermore, the models generated by the MARL-based solutions are capable of handling any distribution of UEs within the cell, rather than being limited to a single predefined scenario. Additionally, they offer exceptionally fast performance during the execution phase.

We also conducted a comparative analysis of the training and testing results between the three MARL-based solutions. During training, we found that CTDE algorithms completed training in similar times, while the completely distributed solution (IQL) initially showed similar training requirements but needed significantly higher number of episodes as the action space increased. This is attributed to the non-stationarity of the environment. Testing results demonstrated that CTDE-based algorithms (QTRAN and MADDPG) offered higher reliability in meeting or exceeding the minimum data rate threshold, as well as achieving comparable or superior mean and minimum data rates with shorter training durations. In contrast, IQL exhibited the highest variability in data rate performance, reflecting its limited ability to learn stable joint policies in non-stationary environments where agents are trained independently. While IQL may achieve similar performance with significantly more training

episodes, its scalability becomes impractical as the number of UEs or action space increases. These findings highlight the advantage of CTDE frameworks in enabling more reliable coordination among agents.

REFERENCES

[1] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren, "Scenarios for 5G mobile and wireless communications: The vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014.

[2] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[3] Y. Teng, M. Liu, F. R. Yu, V. C. M. Leung, M. Song, and Y. Zhang, "Resource allocation for ultra-dense networks: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2134–2168, 3rd Quart., 2019.

[4] A. Al-Habashna and G. Wainer, "QoE awareness in progressive caching and DASH-based D2D video streaming in cellular networks," *Wireless Netw.*, vol. 26, no. 3, pp. 2051–2073, Apr. 2020.

[5] A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, "Improving wireless video transmission in cellular networks using D2D communication," Canada Provisional Patent, 47 111, May 25, 2015.

[6] A. Al-Habashna, G. Wainer, G. Boudreau, and R. Casselman, "Cached and segmented video download for wireless video transmission," in *Proc. 49th Annu. Simulation Symp.*, 2016, pp. 1–8.

[7] A. Al-Habashna, G. Wainer, and S. Fernandes, "Improving video streaming over cellular networks with DASH-based device-to-device streaming," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. (SPECTS)*, Jul. 2017, pp. 1–8.

[8] A. Al-Habashna and G. Wainer, "Improving video transmission in cellular networks with cached and segmented video download algorithms," *Mobile Netw. Appl.*, vol. 23, no. 3, pp. 543–559, Jun. 2018.

[9] A. Al-Habashna, S. Fernandes, and G. Wainer, "DASH-based peer-to-peer video streaming in cellular networks," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. (SPECTS)*, Montreal, QC, Canada, Jul. 2016, pp. 1–8.

[10] F. Jameel, M. A. A. Haider, and A. A. Butt, "Massive MIMO: A survey of recent advances, research issues and future directions," in *Proc. Int. Symp. Recent Adv. Electr. Eng. (RAEE)*, Oct. 2017, pp. 1–6.

[11] E. Hossain, M. Rasti, H. Tabassum, and A. Abdelnasser, "Evolution toward 5G multi-tier cellular wireless networks: An interference management perspective," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 118–127, Jun. 2014.

[12] W. H. Chin, Z. Fan, and R. Haines, "Emerging technologies and research challenges for 5G wireless networks," *IEEE Wireless Commun.*, vol. 21, no. 2, pp. 106–112, Apr. 2014.

[13] A. Al-Habashna, O. A. Dobre, R. Venkatesan, and D. C. Popescu, "Cyclostationarity-based detection of LTE OFDM signals for cognitive radio systems," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010, pp. 1–6.

[14] A. Al-Habashna, O. A. Dobre, R. Venkatesan, and D. C. Popescu, "Joint signal detection and classification of mobile Wimax and LTE OFDM signals for cognitive radio," in *Proc. Conf. Rec. 44th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2010, pp. 160–164.

[15] A. Al-Habashna, O. A. Dobre, R. Venkatesan, and D. C. Popescu, "Joint cyclostationarity-based detection and classification of mobile Wimax and LTE OFDM signals," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kyoto, Japan, Jun. 2011, pp. 1–6.

[16] A. Al-Habashna, O. A. Dobre, R. Venkatesan, and D. C. Popescu, "Wi$_{MAX}$ signal detection algorithm based on preamble-induced second-order cyclostationarity," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010, pp. 1–5.

[17] J. Li, L. Yang, Q. Wu, X. Lei, F. Zhou, F. Shu, X. Mu, Y. Liu, and P. Fan, "Active RIS-aided NOMA-enabled space-air-ground integrated networks with cognitive radio," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 1, pp. 314–333, Jan. 2025.

[18] N. Biswas, G. Das, and P. Ray, "Buffer-aware user selection and resource allocation for an opportunistic cognitive radio network: A cross-layer approach," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 1940–1954, Oct. 2022.

[19] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[20] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2017, pp. 6379–6390.

[21] K. Son, D. W. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. ICML 36th Int. Conf. Mach. Learn.*, Jan. 2019, pp. 5887–5896.

[22] S. T. Selvi, C. Valliyammai, and V. N. Dhatchayani, "Resource allocation issues and challenges in cloud computing," in *Proc. Int. Conf. Recent Trends Inf. Technol.*, Apr. 2014, pp. 1–6.

[23] Q. Cong and W. Lang, "Deep multi-user reinforcement learning for centralized dynamic multichannel access," in *Proc. 6th Int. Conf. Intell. Comput. Signal Process. (ICSP)*, Apr. 2021, pp. 824–827.

[24] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10433–10445, Nov. 2017.

[25] N. Zhao, F. R. Yu, H. Sun, and M. Li, "Adaptive power allocation schemes for spectrum sharing in interference-alignment-based cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 5, pp. 3700–3714, May 2016.

[26] M. Deghel, E. Bastug, M. Assaad, and M. Debbah, "On the benefits of edge caching for MIMO interference alignment," in *Proc. IEEE 16th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2015, pp. 655–659.

[27] M. Sana, A. De Domenico, W. Yu, Y. Lostanlen, and E. C. Strinati, "Multi-agent reinforcement learning for adaptive user association in dynamic mmWave networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6520–6534, Oct. 2020.

[28] M. Elsayed and M. Erol-Kantarci, "AI-enabled radio resource allocation in 5G for URLLC and eMBB users," in *Proc. IEEE 2nd 5G World Forum (5GWF)*, Sep. 2019, pp. 590–595.

[29] M. Simsek, A. Czylwik, A. Galindo-Serrano, and L. Giupponi, "Improved decentralized Q-learning algorithm for interference reduction in LTE-femtocells," in *Proc. Wireless Adv.*, Jun. 2011, pp. 138–143.

[30] H. Ye, G. Y. Li, and B. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[31] J. Cui, Y. Liu, and A. Nallanathan, "The application of multi-agent reinforcement learning in UAV networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2019, pp. 1–6.

[32] J. Menard, A. Al-Habashna, G. Wainer, and G. Boudreau, "Distributed resource allocation in 5G networks with multi-agent reinforcement learning," in *Proc. Annu. Modeling Simulation Conf. (ANNSIM)*, Jul. 2022, pp. 802–813.

[33] J. Menard, "Decentralized resource allocation in 5G networks with heterogeneous multi-agent reinforcement learning," Ph.D. thesis, Dept. Syst. Comput. Eng., Carleton Univ., OT, Canada, 2023.

[34] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 831–846, Oct. 2014.

[35] R. W. Heath, N. González-Prelcic, S. Rangan, W. Roh, and A. M. Sayeed, "An overview of signal processing techniques for millimeter wave MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 3, pp. 436–453, Apr. 2016.

[36] P. Schniter and A. Sayeed, "Channel estimation and precoder design for millimeter-wave communications: The sparse way," in *Proc. 48th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2014, pp. 273–277.

[37] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[38] L. Graesser and W. Keng, *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. Reading, MA, USA: Addison-Wesley, 2019.

[39] R. S. Sutton and A. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 285–286, Jan. 2005.

[40] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Appl. Sci.*, vol. 11, no. 11, p. 4948, May 2021.

[41] W. Lei, *A Study of Wireless Communications With*. Stockholm, Sweden: KTH Royal Institute of Technology, 2022.
[42] C. Watkins, "Learning from delayed rewards," Ph.D. thesis, King's College, Cambridge, U.K., 1989.
[43] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Internation Conf. Mach. Learn.*, Jan. 1993, pp. 330–337.
[44] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019, *arXiv:1509.02971*.
[45] V. Mnih, K. Kavukcuoglu, D. Silver, and A. Graves, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[46] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1801–1819, 4th Quart., 2014.

ALA'A AL-HABASHNA (Member, IEEE) received the Master of Engineering degree from the Memorial University of Newfoundland, Canada, in 2010, and the Ph.D. degree in electrical and computer engineering from Carleton University, Canada, in 2018. Currently, he is an Assistant Professor with Al Hussein Technical University, Amman, Jordan; and an Adjunct Research Professor with Carleton University, Ottawa, Canada. His current research interests include 5G wireless networks, machine learning, computer vision, IoT applications, localization, multimedia communication over wireless networks, signal detection and classification, cognitive radio systems, and discrete-event modeling and simulation.

JON MENARD received the bachelor's degree in software engineering and the M.A.Sc. degree in electrical and computer engineering from Carleton University, Ottawa, Canada. His research interests include machine learning, reinforcement learning, and their applications in wireless communications.

GABRIEL WAINER (Senior Member, IEEE) is a Full Professor with Carleton University, Ottawa, ON, Canada. His research interests include modeling and simulation methodologies, discrete-event modeling, cyber-physical systems, and wireless networks. He is a fellow of SCS and a Distinguished Speaker of ACM. He is the Editor-in-Chief of *SIMULATION: Transactions of the Society for Modeling and Simulation International* (SCS).

GARY BOUDREAU (Senior Member, IEEE) received the B.A.Sc. degree in electrical engineering from the University of Ottawa, in 1983, the M.A.Sc. degree in electrical engineering from Queens University, in 1984, and the Ph.D. degree in electrical engineering from Carleton University, in 1989. From 1984 to 1989, he was a Communications Systems Engineer with Canadian Astronautics Ltd. From 1990 to 1993, he was a Satellite Systems Engineer with MPR Teltech Ltd. From 1993 to 2009, he was with Nortel Networks in a variety of wireless systems, architecture, and management roles within the CDMA and LTE basestation product groups. Since 2010, he has been an Employee with Ericsson Canada, where he is currently a Senior System Developer for 5G and 6G cellular networks. His research interests include digital and wireless communications, signal processing, and machine learning.