IEEE ComSoc IEEE Open Journal of the Communications Society

Received 17 January 2024; revised 25 February 2024; accepted 17 March 2024. Date of publication 25 March 2024; date of current version 9 April 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3380517

Radio Resource Management for Intelligent Neutral Host (INH) in Multi-Operator Environments

HOJJAT NAVIDAN, MOSTAFA NASERI, INGRID MOERMAN (Senior Member, IEEE), AND ADNAN SHAHID (Senior Member, IEEE)

IDLab, Department of Information Technology, Ghent University-imec, 9052 Ghent, Belgium

CORRESPONDING AUTHOR: H. NAVIDAN (e-mail: hojjat.navidan@ugent.be)

This work was supported by the imec.ICON 5GECO Project, co-financed by imec and received support from Flanders Innovation Entrepreneurship under Project HBC.2021.0673, and in part by the European Union through the Horizon Europe Marie Sklodowska-Curie Staff Exchange Programme "Electric Vehicles Point Location Optimisation via Vehicular Communications (EVOLVE)," under Grant 101086218.

ABSTRACT In the era of fifth-generation (5G) cellular networks and beyond, network sharing has emerged as a promising approach to address the escalating demand for spectrum and infrastructure resources. Intelligent Neutral Host (INH) is an advanced network-sharing method facilitated by Open Radio Access Network (O-RAN) capabilities. This paper addresses the challenge of Radio Resource Management (RRM) in a multi-operator, multi-slice scenario. We propose an algorithm based on Q-learning and deep Q-learning, particularly concerning different Physical Resource Block (PRB) types to cater to diverse operator requirements. Implemented as an xApp on the Colosseum platform, our algorithm introduces a dynamic resource allocation strategy that adheres to Service Level Agreement (SLA) constraints and optimizes real-time Key Performance metrics (KPMs), including throughput, buffer occupancy, and PRB utilization. We assess the performance and efficacy of our algorithm in a complex traffic scenario to demonstrate how it effectively allocates resources among operators' slices to satisfy their respective SLA while ensuring optimal resource utilization. The experimental results show that our proposed algorithm can efficiently allocate resources to individual slices while satisfying the SLA. Compared to traditional algorithms, our approach significantly minimizes SLA violations, reducing them to 2.5% for enhanced Mobile Broadband (eMBB) slices and eliminating them entirely for Ultra-Reliable Low-Latency Communications (URLLC) slices.

INDEX TERMS Intelligent neutral host, multi-operator core network, reinforcement learning, deep Q-learning, Q-learning, service level agreement, radio resource management.

I. INTRODUCTION

THE EMERGENCE of fifth-generation (5G) technology marks a significant leap in wireless communications. Characterized by high data speeds, extremely low latency, and massive device connectivity, 5G is not merely an evolution of its predecessors but a revolutionary framework that promises to support a wide array of applications. The architectural flexibility and scalability make it uniquely suited to cater to the varying demands of different use cases and applications. As 5G networks evolve, they are expected to form the backbone of increasingly interconnected and intelligent digital ecosystems, reshaping industries and user experiences [1].

Network slicing is a key technology in 5G systems that enables dynamic and efficient allocation of network resources to different service types. It allows simultaneous support of multiple, logically separate networks that cater to diverse service needs on a shared infrastructure platform. Network slicing is vital for addressing the wide range of use cases and services in 5G, from enhanced Mobile Broadband (eMBB) applications that demand high data rates to Ultra-Reliable Low-Latency Communications (URLLC) that are crucial for mission-critical communication services requiring minimal latency and tactile Internet [2].

Open Radio Access Network (O-RAN), a pivotal innovation in 5G technologies, is revolutionizing our understanding

and construction of telecommunication infrastructures. It disaggregates and virtualizes the traditional monolithic and proprietary Radio Access Network (RAN), resulting in more flexible, efficient, and open network architectures. O-RAN provides a multi-vendor, interoperable environment expected to disrupt the telecommunications industry, encouraging competition and driving rapid innovation. Furthermore, programmability and extensibility through software-defined networking and network function virtualization make O-RAN an ideal platform to meet the dynamically changing demands of 5G networks [3]. For instance, the RAN Intelligent Controller (RIC) concept within O-RAN enables real-time control and optimization of the RAN through software applications known as xApps [4].

The O-RAN architecture divides the base-station protocol stack into a Centralized Unit (CU), a Distributed Unit (DU), and a Radio Unit (RU), bringing more flexibility into network design. The DU manages high physical layer procedures, while the CU manages less time-sensitive higher-layer procedures. This separation promotes RAN sharing, allowing various operators to share the same hardware resources while maintaining their independent control and configuration of software entities. Network slicing is essential to ensure that each slice meets specific performance metrics without adversely impacting others [5]. However, effectively managing these slices to maintain optimal performance and resource allocation, especially in a dynamic network environment where demands and priorities change, is challenging [6].

In traditional RAN settings, radio resources are typically confined to a single operator, making resource sharing challenging and competitive. The modular and software-defined nature of O-RAN enables operators to share RAN resources based on specific agreements [7]. Operators can even allow a third party, such as a neutral host, to manage these shared resources. This brings us to the innovative concept of an Intelligent Neutral Host (INH), which acts as an impartial entity that manages shared RAN resources on behalf of multiple operators [8]. INH utilizes the O-RAN paradigm to enable control loops for operators using the RIC and applies software-defined networking concept to control potential network congestion in the transport network [8].

INH is particularly advantageous when network coverage is crucial, but infrastructure deployment costs are prohibitive. By leveraging INH, operators can effectively share network infrastructure and associated costs while meeting their Service Level Agreements (SLA). This approach is particularly beneficial in Multi-Operator Core Network (MOCN) scenarios, where several operators share a common RAN but maintain their independent core networks [9]. Equipped with intelligent and adaptive algorithms for resource allocation, INH can dynamically assign resources to different operators based on their real-time traffic demand and SLA requirements [8], [10].

Radio Resource Management (RRM) in MOCN scenarios presents unique challenges, primarily due to the complexity of managing shared resources across multiple operators. In such environments, ensuring fair and efficient allocation of radio resources becomes complicated, as each operator has distinct Quality of Service (QoS) requirements and traffic patterns. This complexity is further amplified in dense urban areas with constant demand for high-speed data and connectivity, leading to intense competition for limited radio resources. Additionally, adapting to real-time changes in network conditions, such as user mobility and fluctuating traffic loads, poses a significant challenge for RRM strategies.

Addressing the challenges of shared resource management in MOCN scenarios, our study introduces a novel approach to RRM within the context of INH. We propose an algorithm based on Reinforcement Learning (RL) to allocate resources dynamically among network slices of multiple operators. The algorithm considers real-time Key Performance Metrics (KPMs) such as throughput, buffer occupancy, and Physical Resource Block (PRB) utilization while adhering to the constraints set by SLA. We deploy this algorithm as an xApp in a real-time O-RAN environment on the Colosseum platform to demonstrate its ability to cater to the dynamic needs of the network. This research illustrates how intelligent resource allocation can optimize network performance, reduce operational costs, and enhance service quality in a multi-slice, multi-operator O-RAN environment. Accordingly, the contributions of this study can be summarized as follows:

1) We introduce a novel approach in RRM for MOCN scenarios, focusing on optimizing network sharing while adhering to operator-specific SLA. By employing a novel function, we quantify the performance of network slices, enabling dynamic resource distribution.
2) Surpassing the conventional focus on a single resource type, our work considers various resource types, including dedicated, prioritized, and shared resources, enhancing the versatility and efficiency of resource allocation within O-RAN settings.
3) Our methodology utilizes Q-learning and deep Q-learning for resource allocation in INH, efficiently allocating PRBs to multiple network slices from different operators while satisfying their individual SLA.
4) Finally, we implement the algorithm as an xApp on the Colosseum platform, providing a realistic evaluation under diverse and dynamic traffic scenarios. The performance of our xApp is compared with widely adapted traditional resource allocation methods in terms of performance metrics and SLA satisfaction per slice and operator.

The remainder of the paper is organized as follows: Relevant literature related to our work is reviewed in Section II. Section III outlines our system model and formulates the resource allocation approach. Our proposed RL-based RRM algorithm is detailed in Section IV. Section V discusses the experimental analysis and results. Finally, we conclude the paper in Section VI.

II. RELATED WORK

Recent advancements in RRM focus on addressing the complexities and dynamic nature of next-generation cellular networks [11]. In this landscape, Machine Learning (ML) emerges as a powerful tool to enhance RRM, adapting to varying network conditions [12]. Specifically, applications of ML in managing resources across various RAN-slicing schemes are a focal point of interest. For instance, the work presented in [13] explores the potential of different learning methods to optimize RRM in such environments.

Among other works, [14] addresses resource allocation challenges in virtualized 5G RAN and proposes a dynamic resource provisioning scheme that accounts for varying user requirements in terms of delay and data rate. A shape-based heuristic algorithm complements this scheme to enhance QoS and resource utilization. Widening the scope, Sebakara et al. [15] present an innovative end-to-end approach for resource allocation in virtualized 5G networks that considers both the RAN and core network. They leverage a deep RL-based scheme for joint resource slicing, emphasizing the importance of synchronization for efficient resource allocation. In a similar work, dueling deep Q-network has been suggested for efficient resource slicing and customization, focusing on satisfying QoS requirements and maximizing resource utilization [16].

Within the context of O-RAN, the potential, challenges, and limitations of data-driven optimization approaches in network control have been discussed in [4]. The authors propose a deep RL-based algorithm to optimize KPMs across different network slices and evaluate it against traditional algorithms such as Round-Robin (RR), Proportional Fair (PF), and water-filling. Motalleb et al. [17] address the problem of baseband resource allocation in O-RAN for different 5G service classes. They propose a two-step algorithm using the Lagrangian function and Karush-Kuhn-Tucker conditions to obtain optimal power and resource allocation. The proposed method is validated through simulations showing higher data rates and lower end-to-end delays.

Further enhancing resource allocation in O-RAN environments, Mollahasani et al. [18] propose a dynamic CU-DU selection approach using two actor-critic models. This method allocates resource blocks and distributes the processing load across different layers. The results suggest that dynamically relocating network functions based on service requirements can achieve significant gains in latency and throughput. In another work, Filali et al. [6] propose a deep RL-based xApp to allocate resources among URLLC users, satisfying desired QoS requirements.

Addressing xApp conflicts arising from different vendors, Zhang et al. [19] propose a team learning algorithm based on deep Q-learning. They implement two different xApps for power and radio resource allocation and demonstrate significant performance improvements by eliminating conflicts in O-RAN. The authors extend their work and present a federated deep RL algorithm to coordinate multiple independent xApps for network slicing [20]. In the proposed federated learning approach, each xApp trains a local model, which is then submitted to a coordination model to predict a joint Q-table. In both papers, the authors consider a scenario where operators deploy and manage their xApps independently. However, our study considers a setting where an INH oversees the RAN and its resources, centralizing control and eliminating xApp conflicts.

Existing works mainly address resource allocation for specific network slices or service classes, limiting their adaptability to complex and dynamic RAN-sharing scenarios. While these studies offer valuable insights into resource allocation within O-RAN environments, they predominantly focus on traditional RAN configurations rather than the MOCN scenario. In contrast, our work addresses the challenge of resource allocation within a MOCN scenario where an INH orchestrates resource allocation across a shared infrastructure. This approach not only addresses a previously unexplored aspect of RAN sharing but also proposes a solution specifically designed for this unique scenario, filling a gap in the current literature.

III. SYSTEM MODEL

Our system model, depicted in Fig. 1, represents a MOCN RRM scenario where multiple operators with several slices utilize the same RAN shared and managed by the INH. We assume the RAN is shared among N operators and L slices. Therefore, the sets of all operators and slices available in the system are denoted by $\mathbb{N}$ and $\mathbb{L}$, respectively. Each operator n contains of L_n slices, forming the set $\mathbb{L}_n$. The RAN components, namely the DU, CU, and RU, are shared among all operators' User Equipment (UE). The INH, acting as an impartial entity, oversees the allocation of different resource types among various operators to fulfill their specific QoS requirements.

The RIC is a vital component of the O-RAN architecture, designed to introduce advanced intelligence and programmability into the RAN. The near-real-time RIC (near-RT RIC) operates in a timescale of milliseconds and performs functions like RRM, load balancing, handover, and dynamic spectrum management. It hosts ML models and xApps, which are used to analyze network trends and make high-level decisions regarding network optimization, planning, and policy management [4]. The xApps interact with RIC through the standard E2 interface, facilitating the transfer of RAN telemetry data and real-time KPMs [3]. The primary goal of our RRM xApp is to distribute the corresponding amount of PRBs to each slice. Within each slice, an intra-slice scheduler allocates PRBs to the UEs.

We define three classes of radio resources: *Dedicated*, *Prioritized*, and *Shared*. *Dedicated* resources are exclusively reserved for a specific slice and are allocated to the users within, guaranteeing a minimum level of QoS per their SLA, regardless of the traffic conditions or utilization. *Prioritized* resources are assigned to a specific operator and are not tied to any particular slice. These resources cannot be allocated to slices or users belonging to other operators but can be

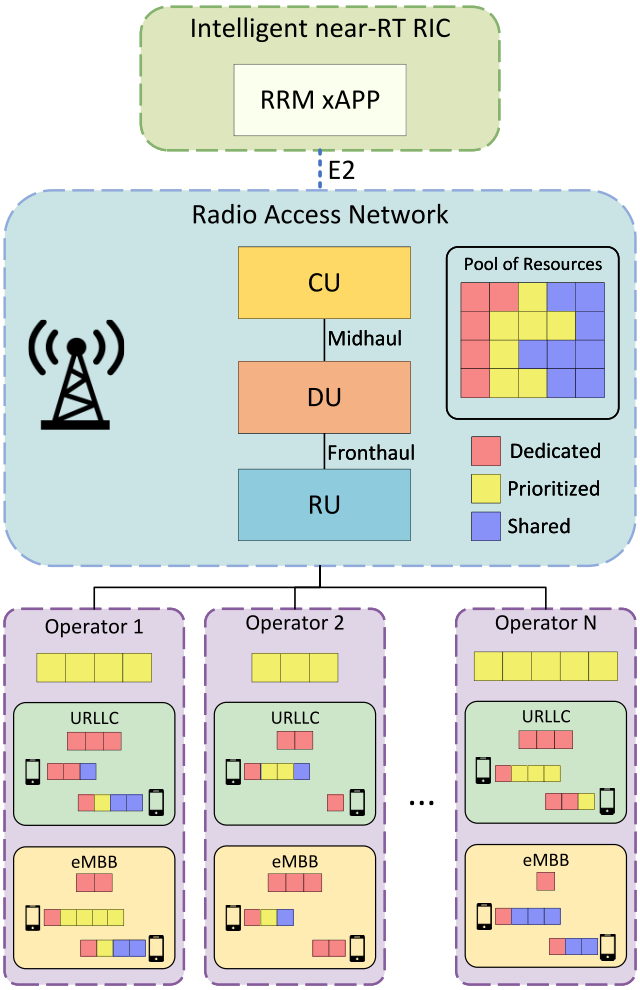


FIGURE 1. Overview of our system model for MOCN scenario with different types of PRBs.

interchanged between the slices or users of the operator as per demand and priority. Finally, *Shared* resources do not belong to any specific slice or operator and can be allocated to any slice on demand. Shared PRBs provide a buffer of resources that can be utilized to meet unexpected demands or peak loads, thus enhancing the overall efficiency of resource utilization.

This classification of resources into dedicated, prioritized, and shared types is not arbitrarily determined but rather is derived from and in accordance with the 3GPP TS 28.541 [21], which provides the management and orchestration specifications for the network resource model. These resource types are integral in managing multi-tenancy in RAN, providing a balanced framework for RRM that focuses on guaranteeing service requirements and efficient resource utilization.

Accordingly, in Fig. 1, each operator serves multiple slices and is allocated some prioritized PRBs. Each slice serves numerous UEs and is allocated some dedicated PRBs, which can only be allocated to UEs corresponding to that slice. Real-time network status and metrics such as the instantaneous throughput, buffer occupancy, and number of used PRBs are provided to the near-RT RIC through the E2 interface.

We define a unique function for each slice called the Slice Performance Index (SPI) that encapsulates specific performance metrics and maps them into a value that reflects the slice performance relative to the SLA. The SPI function allows us to capture diverse SLA requirements and roles of different slices in a quantifiable manner that the decision-making agent can utilize to make informed decisions. It serves a dual purpose - acting as a basis for the learning and decision-making process and as an indicator of SLA adherence for the respective slices. Maintaining high SPI values across all operators and slices ensures SLA compliance, effective resource allocation, and optimal system performance. The SPI function for each slice ($l \in \mathbb{L}$) is defined as

$$S_l = w_1 \frac{T_l}{T_{req}} + w_2 \frac{1}{c + \frac{B_l}{B_{req}}} + w_3 U_l - w_4 \frac{B_l}{B_{max}}, \tag{1}$$

where T_l, B_l, and U_l are the instantaneous throughput, buffer occupancy, and PRB utilization at the slice level, obtained as KPMs from the RAN. T_{req} and B_{req} are the minimum required instantaneous throughput and maximum required buffer occupancy as per SLA, and B_{max} is the maximum size of the downlink buffer. Finally, w_1 to w_4 are weights that can be adjusted based on the slice role, and c is an arbitrary normalization constant.

For eMBB slices, where throughput is crucial, the SPI function is primarily influenced by the instantaneous throughput achieved. Conversely, for URLLC slices, the SPI function is influenced by the current downlink buffer occupancy of the UEs and PRB utilization (ratio of used PRBs to allocated PRBs). Although end-to-end latency might appear to be a key metric for URLLC slices, it can be influenced by external factors that are often beyond the control of the RRM, such as core network delays or processing times at the UE. On the contrary, downlink buffer occupancy is a more direct measure within the scope of the RAN and can be an indirect measure of latency for URLLC applications.

Consequently, the problem of fairly allocating PRBs to the slices at the INH level can be posed as an optimization problem where we aim to maximize the combined SPI for all slices, subject to the constraints of the resource types:

$$\begin{aligned} \underset{\boldsymbol{P}}{\text{maximize}} \quad & \sum_l S_l \\ \text{subject to} \quad & \sum_l p_l \le P_T, \\ & \sum_{l \in \mathbb{L}_n} p_l^{prio} \le p_{Op_n}^{prio}, \quad \forall n \in \mathbb{N} \\ & p_l \ge p_l^{dedi}, \quad \forall l \in \mathbb{L} \\ & P_T = P^{dedi} + P^{prio} + P^{share}. \end{aligned} \tag{2}$$

In (2), p_l, p_l^{prio}, and p_l^{dedi} represent the number of PRBs, prioritized PRBs, and dedicated PRBs allocated to slice l, respectively. P_T is the total number of PRBs available

in the system, Op_n is the n-th operator, and $p^{prio}_{Op_n}$ is the number of prioritized PRBs belonging to operator Op_n. P^{dedi}, P^{prio}, and P^{share} correspond to the total number of dedicated, prioritized, and shared PRBs available in the system, respectively. Finally, $\boldsymbol{P}$ is the PRB allocation vector, defined as $\boldsymbol{P} = [p_1, p_2, \ldots, p_L]$. Each constraint in (2) corresponds to a limitation imposed by the definition of different PRB types.

An essential concept in resource allocation of cellular systems is the Resource Block Group (RBG). While a PRB is the smallest unit of resources that can be allocated to a user in a cell, representing a portion of frequency and time on the radio spectrum, RBG is a bundle of consecutive PRBs treated as a single unit by the scheduling or resource allocation algorithm. The exact size of an RBG (the number of PRBs it contains) can depend on the total system bandwidth and configuration. In resource allocation type 0 [22], RBGs are the minimum allocated resource units and are represented by a bitmap, where each bit represents one RBG. To allocate resources, the xApp has to configure the RBG bitmap per each slice at the RAN level while ensuring efficient utilization of resources and satisfying SLA [23].

IV. METHODOLOGY

In this section, the proposed algorithm for RRM is described. Initially, components of the RL algorithm are introduced. Then, the Q-learning and the more advanced deep Q-learning algorithm are presented.

A. REINFORCEMENT LEARNING

As shown in Fig. 2, the RL framework comprises an agent in the near-RT RIC interacting with the environment (the RAN). Real-time KPMs from the RAN and SLA requirements from a local database serve as the input. At each timestep t, the agent receives the inputs, observes state s_t, and selects an action a_t determined by the state-action function $Q(s_t, a_t)$. This triggers a state transition to s_{t+1} and generates a reward R_{t+1} based on the selected action. To comprehensively understand the RL approach, we must define its components: state space, action space, and reward function. Each component has been designed to encapsulate the essence of our problem space:

States: The states shall represent the current KPMs for each slice derived from the corresponding terms of the SPI function. Specifically, for eMBB and URLLC slices, we consider the throughput and buffer occupancy terms of the (1) as the state, respectively. The state of each slice is mathematically expressed as

$$s_l = \begin{cases} c_1 \frac{T_L}{T_{req}} & \text{if } l \text{ is eMBB} \\ \frac{1}{c_2 + \frac{B_l}{B_{req}}} & \text{if } l \text{ is URLLC} \end{cases} \tag{3}$$

where c_1 and c_2 are normalization constants. Consequently, the state of the agent at each timestep will be

$$s_t = [s_1, s_2, \ldots, s_L]. \tag{4}$$

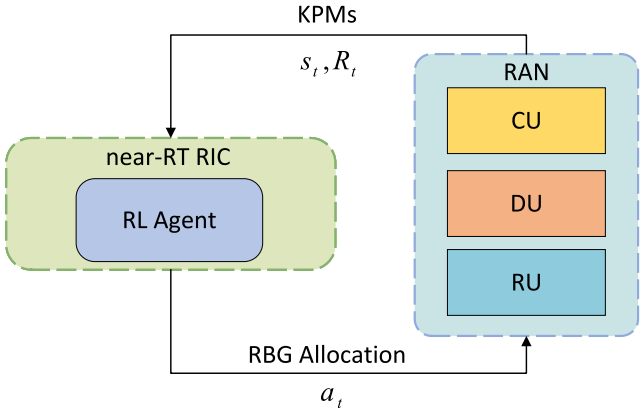


FIGURE 2. Overview of the proposed RL system for RRM.

Both terms of the (3) are bounded, as both throughput and buffer occupancy of the system have an upper limit. By adjusting the constants c_1 and c_2, the states can be normalized to any desired value.

Actions: The action space consists of potential resource allocation strategies that the RRM agent can execute. In our system, the actions are derived from the concept of RBG. For a given slice, the agent can decrease or increase the allocated resources by one RBG or maintain the current allocation. Therefore, the possible actions for each slice are $[-1, 0, 1]$, and the action at each timestep can be represented as the vector

$$a_t = [a_1, a_2, \ldots, a_L], \ \ a_i \in \{-1, 0, 1\}, \tag{5}$$

where $i = 1, 2, \ldots, L$. This results in an action space size of $|\mathbb{A}| = 3^L$.

Reward: Central to the learning process, the reward function quantifies the success of an action taken in a particular state. It is formulated to capture the efficiency and effectiveness of resource allocation strategy. The agent tries to maximize the minimum SPI across all slices. Actions that result in violations of resource constraints (prioritized or dedicated PRBs) or unfair distribution of resources are penalized, guiding the agent to make different decisions in the future. The reward function is given as

$$R_{t+1} = \beta_1 \min_l S_l - \beta_2 \left(1 - \min_l U_l\right) - \beta_3 \varphi, \tag{6}$$

where φ represents the penalty term imposed for actions that violate the constraints of (2) and β_1 to β_3 are the reward weights. The penalty term can be mathematically expressed as

$$\varphi = \begin{cases} 1 & \text{if } \sum_{l \in \mathbb{L}_n} p_l^{prio} > p^{prio}_{Op_n} \text{ for any } n \in \mathbb{N} \\ 1 & \text{if } p_l < p_l^{dedi} \text{ for any } l \in \mathbb{L} \\ 1 & \text{if } \sum_l p_l > P_T \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The coefficients of the reward function (6) influence the agent's behavior during the decision-making process. β_1 is the weight of the primary objective and encourages the agent

to ensure a minimum level of service across all network slices. β_2 aims to increase PRB utilization by penalizing actions that lead to inefficient PRB usage within any slice. Finally, β_3 deters the agent from taking actions that would violate resource constraints.

B. Q-LEARNING

Q-learning is a model-free RL algorithm that determines the optimal action-selection policy within a finite Markov decision process. It operates by learning the values of state-action pairs without requiring a model of the environment. The algorithm utilizes a Q-table to record the value of each state-action pair, representing the expected future rewards for selecting a specific action in a given state. This Q-table is updated iteratively using the Bellman equation as the agent explores the environment [24]:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left[R_{t+1} + \gamma \max_{a'} Q\left(s_{t+1}, a'\right)\right] \tag{8}$$

In (8), $Q(s_t, a_t)$ is the Q-value of taking action a_t in state s_t at time t, yielding the reward R_{t+1}. The term a' denotes any possible action at the next state (s_{t+1}). The learning rate α adjusts the impact of each update on the Q-value, while the discount factor γ determines the importance of future rewards. As the learning proceeds through sufficient iterations, the values in the Q-table will converge to reflect the significance of each state-action pair, maximizing the expected reward.

During the learning phase, the agent occasionally opts to explore a new action rather than exploiting the best-known action in the current state. This exploration-exploitation trade-off is controlled using the epsilon-greedy strategy, where the agent has a probability ϵ of choosing a random action and a probability $1-\epsilon$ of choosing the best-known action. This strategy ensures convergence of Q-values, preventing the agent from getting trapped in a suboptimal solution. We gradually reduce the value of the ϵ over iterations to shift the balance from exploration to exploitation as the agent learns more about the environment. Initially, a higher epsilon value encourages exploration, preventing the agent from converging to a local optimum. As the learning progresses and the Q-table is updated, the epsilon value decreases, leading the agent to exploit the best-known actions.

The inherent discrete structure of the Q-table necessitates that both the state and action spaces be discrete. This drawback of Q-learning limits its ability to deal with continuous state variables. Given that the state space presented in (4) is continuous, it must be discretized. To address this, we digitize the SPI function for each slice into predefined states, where each state indicates the status of the SLA satisfaction for that slice. The simplest digitization is the binary one, where one indicates that the SLA of that particular slice is satisfied, while zero indicates an unsatisfied SLA. The size of this binary digitized state space is $|\mathbb{S}| = 2^L$.

Finally, the dimensions of the Q-table play a crucial role in determining both the computational feasibility and the performance of the algorithm. A larger Q-table increases the exploration time required for the agent to evaluate each state-action pair, extending the training duration. A strategic adjustment can be made to mitigate this by limiting the agent's decision-making to only one slice per timestep rather than simultaneous decisions across all slices. This approach effectively reduces the action space from $|\mathbb{A}| = 3^L$ to $|\mathbb{A}| = 2L+1$ without severely affecting the response time. Consequently, the size of the Q-table is refined to $|\mathbb{Q}| = 2^L(2L+1)$. The overall procedure of the Q-learning algorithm is presented in Alg. 1.

C. DEEP Q-LEARNING

Deep Q-learning leverages a deep neural network to approximate the Q-value function. In contrast to traditional Q-learning, which relies on a tabular representation of the Q-function, deep Q-learning can generalize over a continuous and ample state space. This feature particularly benefits our setting, where the state space encompasses the SPI function across multiple operators and slices. The input and output dimensions of the deep neural network, which receives the state and calculates the Q-value for every possible action, are determined by the state size and the number of actions, respectively. Specifically, the input dimension is equal to the number of slices (L), and the output dimension corresponds to the size of the action space ($|\mathbb{A}|$).

In traditional Q-learning, the Q-value update is based on the Bellman equation presented in (8), establishing a recursive relationship between the current and subsequent Q-values. However, this recursive relationship can lead to instability in deep Q-learning as the same neural network estimates the current and subsequent Q-values. The instability arises from the moving target problem, where the weight updates change the values it attempts to predict. This issue can be mitigated by using a second neural network, the target network, that decouples the estimation of current and subsequent Q-values [25]. Based on the Bellman equation, the target Q-values can be expressed as

$$y = R_{t+1} + \gamma \max_{a'} Q\left(s_{t+1}, a'; \hat{\theta}\right) \tag{9}$$

where $\hat{\theta}$ represents weights of the target network. These weights are periodically updated with weights of the main network (θ) but are kept constant between updates to provide a stable target for training. The training procedure involves updating the main network weights to minimize the loss between predicted and target Q-values. The loss function, typically a mean squared error, is given by

$$L(\theta) = \mathbb{E}\left[(y - Q(s_t, a_t; \theta))^2\right]. \tag{10}$$

Since consecutive experiences can be correlated, directly updating the neural network at each iteration can be inefficient. This correlation can lead to suboptimal learning and affect the convergence of the neural network. Experience

Algorithm 1 Overview of the Q-Learning for RRM

Input: a) Initial Q-table
b) xApp configuration from the database
c) SLA requirements per each operator
Output: RBG allocation bitmap for each slice
for each timestep **do**
  Read the current KPMs from the RAN
  Calculate the SPI function for every slice using (1)
  Update the state of the agent using (4) and (3)
  Calculate the agent reward for the last performed action using (6)
  Update the Q-table using the Bellman Equation
  if a random number between 0 and 1 $\leq \epsilon$ **then**
    Choose a random action (explore)
  else
    Choose best action using $a_t = \arg\max_a (s_t, a_t)$
  end if
  Update ϵ with the decay rate
  Check for illegal actions
  Update the slice allocation configuration and send it to RAN
end for

Algorithm 2 Overview of the Deep Q-Learning for RRM

Input: a) Initial experience replay buffer
b) Initialize θ and $\hat{\theta}$
c) xApp configuration from the database
d) SLA requirements per each operator
Output: RBG allocation bitmap for each slice
for each timestep **do**
  Read the current KPMs from the RAN
  Calculate the SPI function for every slice
  Update the state of the agent using (4) and (3)
  Calculate the agent reward for the last performed action using (6)
  Store the last transition $(s_t, a_t, R_{t+1}, s_{t+1})$ in the replay memory
  Perform a gradient descent step on the DQN model with respect to its loss function
  if a random number between 0 and 1 $\leq \epsilon$ **then**
    Choose a random action (explore)
  else
    Choose best action using $a_t = \arg\max_a (s_t, a_t; \theta)$
  end if
  Sample a mini-batch from the replay memory
  Update θ by minimizing the loss function in (10)
  Periodically update the target network weights $\hat{\theta}$ to match θ
  Update ϵ with the decay rate
  Check for illegal actions
  Update the slice allocation configuration and send it to RAN
end for

replay solves this issue by storing the experiences of the agent at each timestep $(s_t, a_t, R_{t+1}, s_{t+1})$ in a replay buffer. During the training phase, the deep Q-learning algorithm randomly samples a mini-batch of experiences from this replay buffer, breaking the correlation between consecutive learning samples. It also enables the agent to reuse previous experiences, maximizing the learning from each interaction with the environment. The overall procedure of the deep Q-learning algorithm is presented in Alg. 2.

V. EXPERIMENTAL ANALYSIS

This section details experiments conducted to evaluate the proposed framework. First, a brief description of our experimental setup and parameters used within the algorithm is provided. Next, the results of our evaluation are shown. Finally, we discuss the complexity analysis of the algorithm.

A. EXPERIMENT SETUP

We deployed our RRM algorithm in the Colosseum environment, the world's largest wireless network emulator based on Software-Defined Radios (SDRS) [26]. On top of the Colosseum, we utilized SCOPE [23], an open-source framework that builds upon srsRAN [27] capabilities. SCOPE offers advanced data collection tools and APIs for runtime management of cellular stack functionalities [28]. Our experimental setup involves two operators (Op_1 and Op_2), each composed of two slices (URLLC and eMBB) that share the RAN.

Since this work mainly focuses on distributing the resources to slices and mitigating the effect of the intra-slice scheduler on performance, each slice caters to only one UE using all of its available resources. Hence, we require five nodes (SDRs) in the Colosseum system, one for the base station and four for the UEs. Traffic generation is done using the iPerf tool over the srsRAN network. Details of the neural network architecture and parameters of the xApp and the algorithm are outlined in Table 1 and 2, respectively.

The proposed RRM model operates as an xApp within the near-RT RIC, executing every 250ms. During each execution cycle, the xApp collects KPMs from the RAN, makes resource allocation decisions, and communicates the slice allocation configuration back to the RAN over the E2 interface. This cycle constitutes a single iteration of the xApp. While the execution interval is adjustable to meet specific demands, it is vital to consider the trade-offs involved. Decreasing this interval increases the communication frequency between the RAN and the near-RT RIC, potentially overloading the E2 interface. Conversely, increasing it can slow down the agent, hindering its ability to react quickly and consequently affecting the SLA.

The xApp implements the decision-making process as outlined in Algorithms 1 and 2. Upon determining the optimal action, the xApp evaluates if it violates the constraints

TABLE 1. Neural network architecture.

Layer	Input dim.	Output dim.
Input	4	32
Dense, ReLu	32	64
Dense, ReLu	64	32
Output, linear	32	9

TABLE 2. Parameters used throughout the experiment.

Parameter	Value	Parameter	Value
N	2	Decay rate	0.998
L	4	β_1	1
P_T	100	β_2	0.5
RBG size	4	β_3	2
P^{dedi}	16	Mini-batch size	32
P^{prio}	8	T_{req} for eMBB slices	3 Mbps
Bandwidth	20 Mhz	T_{req} for URLLC slices	512 Kbps
Transmit Gain	40 dB	B_{req} for eMBB slices	175 KBytes
α	0.1	B_{req} for URLLC slices	20 KBytes
γ	0.8	B_{max}	200 KBytes

defined in (2). Such illegal actions are not only penalized through the reward function (6) but also prevented from execution, as they could severely impact SLA or other conditions. We employ a neural network architecture with three hidden layers and the ReLU activation function for the deep Q-learning network.

We evaluate the performance of our model in a traffic scenario where, after a period of stability for all slices, the eMBB traffic for Op_2 starts increasing, remains constant for a duration, and then decreases. In contrast, the eMBB traffic for Op_1 demonstrates an opposite pattern: it initially decreases, remains constant for a while, and then increases. URLLC traffic for both operators remains constant throughout the scenario. This scenario is designed to assess the adaptability and resilience of our model in intricate environments, particularly when both operators experience varying and opposing traffic demands.

B. RESULTS

In order to evaluate the performance and efficiency of our deep Q-learning-based RRM xApp, we visualize several metrics after training it for sufficient iterations. These metrics include instantaneous throughput, SPI function, and PRB allocation pattern throughout the traffic scenario. These metrics, illustrated in Fig. 3 collectively provide insights into the xApp's ability to adapt to varying traffic conditions, prioritize resources, and maintain service quality.

At $t = 200$, we observe a sudden increase in eMBB traffic for eMBB/Op_2 while eMBB/Op_1 traffic decreases. This change leads to an immediate drop in the SPI function for both slices, accompanied by a surge in buffer occupancy for the slice experiencing increased traffic as packets accumulate. On the other hand, the buffer occupancy for eMBB/Op_1 declines, reflecting its reduced traffic demand. This brief SLA violation prompts the agent to act, allocating more PRBs to eMBB/Op_2, raising its throughput. Following this adjustment in PRB distribution, a significant reduction in buffer occupancy is evident.

We witness another shift in traffic demand at $t = 400$, with both slices returning to their initial state. Similar to the previous observation, this transition triggers a buffer occupancy spike and a drop in the SPI function for eMBB/Op_1. The agent promptly allocates more PRBs from eMBB/Op_2 to eMBB/Op_1. The PRB allocation for URLLC slices remains consistent throughout the traffic scenario, given the stability in their traffic demands.

In the next stage of our analysis, we compare the performance of our proposed RRM xApp against the widely adopted PF and RR algorithms [29], [30]. The PF algorithm prioritizes a balance between total system throughput and fairness among users [31], while the RR algorithm ensures each user gets an equal share of the available PRBs [32]. Utilizing the identical traffic scenario, we assess the performance of the PF and RR in managing RRM for the INH in two distinct runs. For a detailed analysis, we visualize the empirical Cumulative Distribution Function (CDF) of the KPMs for different slices in Fig. 4. Each point on the CDF curves represents the probability that a specific KPM value for a particular slice will be less than or equal to a certain threshold. The x-axis of the plot shows the range of possible KPM values, while the y-axis depicts the probability of falling within that range.

A detailed examination shows that our RRM xApp (deep Q-learning) significantly surpasses RR and PF algorithms in terms of URLLC slice buffer occupancy. A similar pattern is observed in the eMBB slices throughput, where the gap between the two slices reflects their differing traffic demands. Regarding throughput for eMBB/Op_1, RR performs closely to our xApp and better than PF. This is because RR distributes PRBs equally among the slices, resulting in eMBB/Op_1 receiving more PRBs than needed. However, this advantage diminishes for eMBB/Op_2 during peak traffic demands, causing the performance of RR to fall behind both PF and our xApp.

While at first glance, PF and RR might appear superior in PRB utilization, especially for the eMBB slices, their inadequacy in maintaining SLA becomes noticeable. Conversely, RR outperforms PF in keeping the SLA satisfied for the URLLC slices, as it ensures equal PRB distribution between all slices. This fairness guarantees no SLA violation for URLLC slices, albeit at the trade-off of poor PRB utilization. Remarkably, our xApp manages to reduce the SLA violation rate to a mere 2.5% for eMBB slices compared to PF's 36.1% and RR's 17.9%. Moreover, it completely eliminates the SLA violations for URLLC slices, highlighting the ability

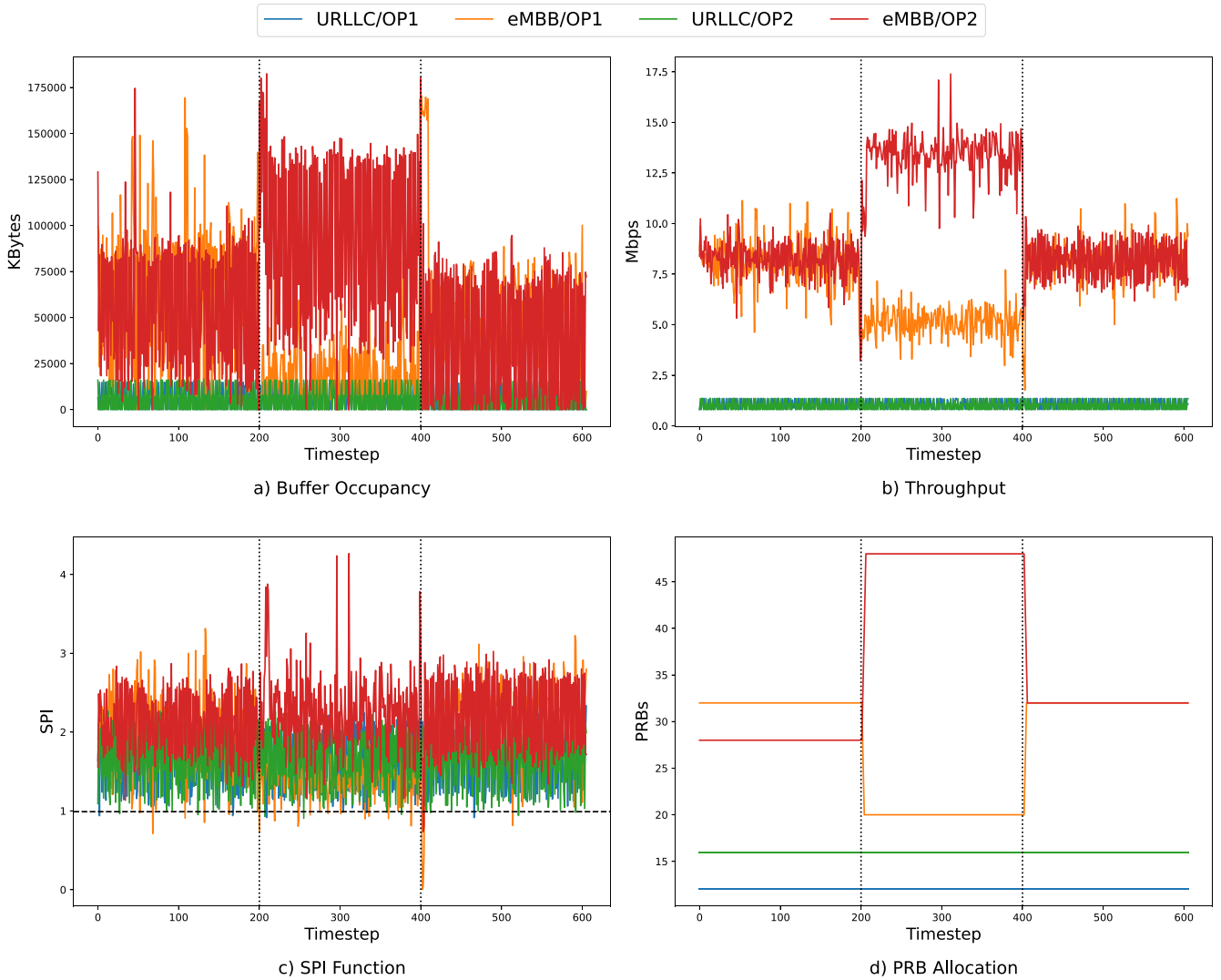


FIGURE 3. Metrics for each slice during the traffic scenario obtained using the deep Q-learning xApp. The x-axis represents timesteps after starting the traffic scenario, where each timestep corresponds to 250ms. The vertical dotted lines at $t = 200$ and $t = 400$ indicate the changes in the traffic scenario and metrics, while the horizontal dotted line in (c) represents the SLA violation threshold.

of our xApp to cater to SLA demands consistently across various operators and slices.

Despite the advanced capabilities of deep Q-learning, such as handling high-dimensional state spaces and leveraging deep learning for function approximation, the performance results were closely aligned with those of Q-learning. Since the network size (slices and operators) is chosen to be small due to implementation limitations, the xApp may not fully exploit the complex pattern recognition and generalization capabilities of deep Q-learning. In other words, the effectiveness and simplicity of Q-learning in this setting suggest that small networks may not require the additional complexity and computational resources that deep Q-learning entails. However, it is clear that as network size increases, the scalability of Q-learning becomes questionable.

C. COMPLEXITY ANALYSIS

The complexity of the Q-learning algorithm primarily stems from determining and updating the Q-value approximations, with the complexity of each update being $O(1)$. Therefore, the time complexity of the optimal allocation strategy across multiple iterations becomes $O(mt)$, where m represents the number of iterations required for convergence and t denotes the operation interval of the algorithm.

On the other hand, the complexity of deep Q-learning is influenced by the architecture of the underlying neural network and the dimensions of the state and action spaces. For a fully connected network with J layers, the operation time of each update is given by $\sum_{j=0}^{J-1} u_j u_{j+1}$, where u_j denotes the number of neurons in layer j. Therefore, the time complexity for deep Q-learning is expressed by $O(mt \sum_{j=0}^{J-1} u_j u_{j+1})$ [33].

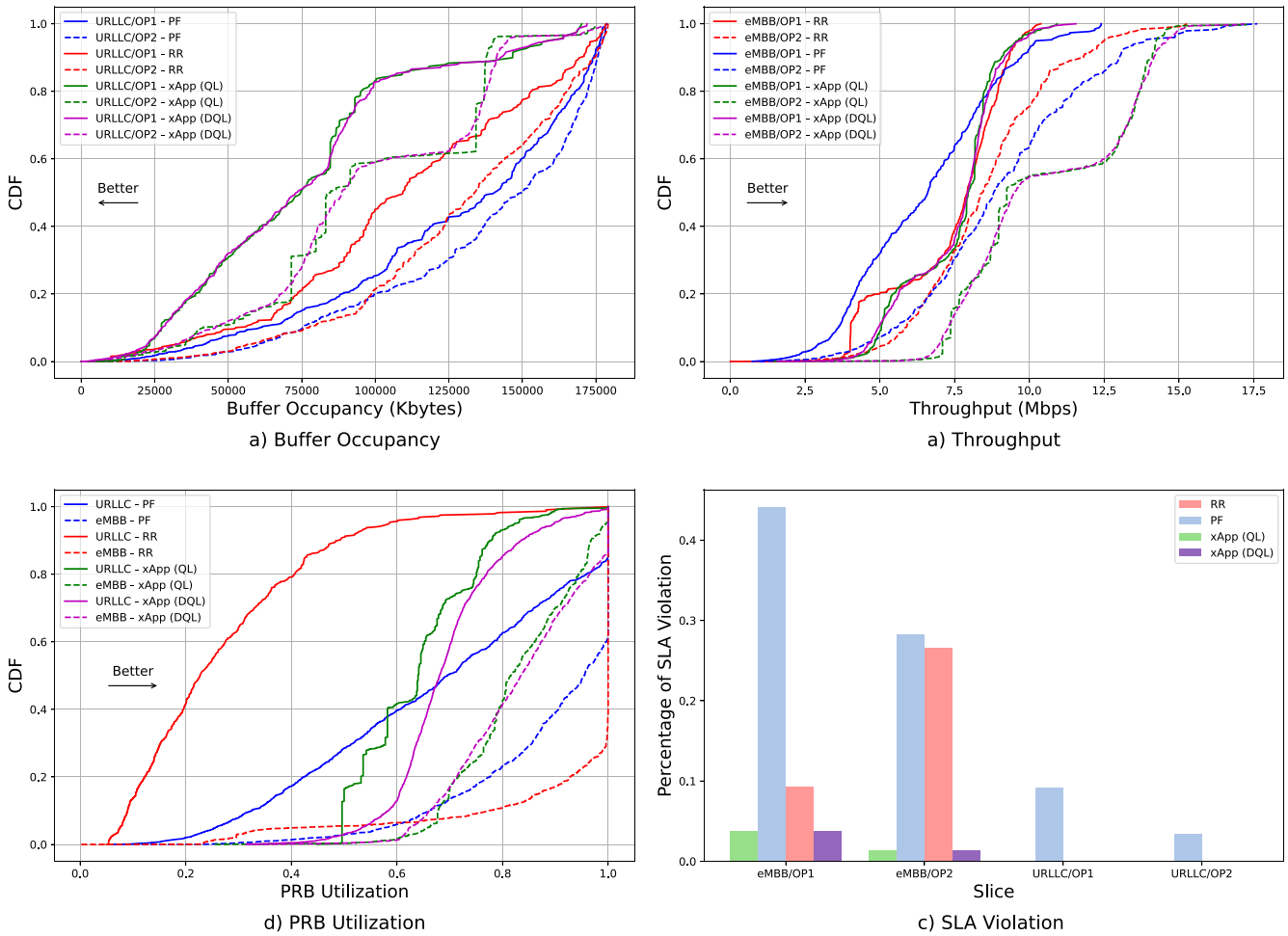


FIGURE 4. **Comparison of the empirical CDF of buffer occupancy (a), throughput (b), PRB utilization (c), and the percentage of SLA violation (d) between our RRM xApp and two traditional RRM algorithms. xApp (QL) stands for the xApp using Q-learning, and xApp (DQL) stands for the xApp using deep Q-learning.**

Training time for Q-learning is relatively shorter due to the more straightforward structure of updating the Q-table, which does not require the extensive computational resources that deep neural networks demand. However, as the size of the state and action spaces increases, the Q-table grows exponentially, diminishing its efficiency. Hence, traditional Q-learning is better suited for scenarios with a few slices served by the INH, while deep Q-learning becomes advantageous for more extensive networks.

VI. CONCLUSION

In this paper, we explored the evolving dynamics of RAN sharing, highlighting the pivotal role of the INH in shaping the future of telecommunications, particularly for 5G and beyond. Our study addresses the challenge of managing resources efficiently and fairly for multiple operators within the INH. By accommodating different PRB types and proposing an RRM algorithm based on Q-learning and deep Q-learning, we present a comprehensive solution tailored explicitly for multi-operator, multi-slice environments. This approach underscores the potential of the INH to ensure equitable and efficient resource allocation while meeting real-time KPMs and SLA constraints. Our Experimental evaluations conducted in the Colosseum environment demonstrate the superiority of our approach over traditional methods, particularly in its ability to reduce SLA violations drastically.

Although our current model employs deep Q-learning, we recognize the potential of alternative methods in this domain, such as actor-critic networks. Furthermore, the scalability of our proposed solution remains to be further explored through simulations. We plan to integrate these algorithms, which are inherently more scalable and stable, to enhance further the efficiency and adaptability of our RRM xApp. Furthermore, we plan to test and refine our xApp in more intricate traffic conditions, ensuring it remains robust and efficient in real-world deployments.

REFERENCES

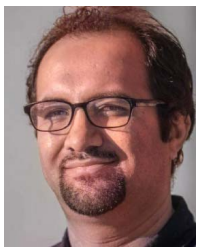
[1] A. Gupta and R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2015.
[2] S. E. Elayoubi, S. B. Jemaa, Z. Altman, and A. Galindo-Serrano, "5G RAN slicing for verticals: Enablers and challenges," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 28–34, Jan. 2019.

[3] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1376–1411, 2nd Quart., 2023.
[4] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in O-RAN for data-driven nextG cellular networks," *IEEE Commun. Mag.*, vol. 59, no. 10, pp. 21–27, Oct. 2021.
[5] C.-L. I, S. Kuklinskí, and T. Chen, "A perspective of O-RAN integration with MEC, SON, and network slicing in the 5G era," *IEEE Netw.*, vol. 34, no. 6, pp. 3–4, Nov./Dec. 2020.
[6] A. Filali, B. Nour, S. Cherkaoui, and A. Kobbane, "Communication and computation O-RAN resource slicing for URLLC services using deep reinforcement learning," *IEEE Commun. Stand. Mag.*, vol. 7, no. 1, pp. 66–73, Mar. 2023.
[7] L. Giupponi and F. Wilhelmi, "Blockchain-enabled network sharing for O-RAN in 5G and beyond," *IEEE Netw.*, vol. 36, no. 4, pp. 218–225, Jul./Aug. 2022.
[8] J. F. Nunes Pinheiro et al., "5GECO: A cross-domain intelligent neutral host architecture for 5G and beyond," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2023, pp. 1–6.
[9] M. Kassis, S. Costanzo, and M. Yassin, "Flexible multi-operator RAN sharing: Experimentation and validation using open source 4G/5G prototype," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, Jun. 2021, pp. 205–210.
[10] R. Bajracharya, R. Shrestha, H. Jung, and H. Shin, "Neutral host technology: The future of mobile network operators," *IEEE Access*, vol. 10, pp. 99221–99234, 2022.
[11] B. Agarwal, M. A. Togou, M. Marco, and G.-M. Muntean, "A comprehensive survey on radio resource management in 5G HetNets: Current solutions, future trends and open issues," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2495–2534, 4th Quart., 2022.
[12] I. A. Bartsiokas, P. K. Gkonis, D. I. Kaklamani, and I. S. Venieris, "ML-based radio resource management in 5G and beyond networks: A survey," *IEEE Access*, vol. 10, pp. 83507–83528, 2022.
[13] Y. Azimi, S. Yousefi, H. Kalbkhani, and T. Kunz, "Applications of machine learning in resource management for RAN-slicing in 5G and beyond networks: A survey," *IEEE Access*, vol. 10, pp. 106581–106612, 2022.
[14] K. Xiong, S. S. Rene Adolphe, G. O. Boateng, G. Liu, and G. Sun, "Dynamic resource provisioning and resource customization for mixed traffics in virtualized radio access network," *IEEE Access*, vol. 7, pp. 115440–115453, 2019.
[15] S. R. A. Sebakara, G. Sun, G. O. Boateng, B. Mareri, R. Ou, and G. Liu, "SNAF: DRL-based interdependent E2E resource slicing scheme for a virtualized network," *IEEE Trans. Veh. Technol.*, vol. 72, no. 7, pp. 9069–9084, Jul. 2023.
[16] G. Sun, K. Xiong, G. O. Boateng, G. Liu, and W. Jiang, "Resource slicing and customization in RAN with dueling deep Q-Network," *J. Netw. Comput. Appl.*, vol. 157, May 2020, Art. no. 102573.
[17] M. K. Motalleb, V. Shah-Mansouri, S. Parsaeefard, and O. L. Alcaraz López, "Resource allocation in an open RAN system using network slicing," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 1, pp. 471–485, Mar. 2023.
[18] S. Mollahasani, M. Erol-Kantarci, and R. Wilson, "Dynamic CU-DU selection for resource allocation in O-RAN using actor-critic learning," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.
[19] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Team learning-based resource allocation for open radio access network (O-RAN)," in *Proc. IEEE Int. Conf. Commun.*, May 2022, pp. 4938–4943.
[20] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Federated deep reinforcement learning for resource allocation in O-RAN slicing," in *Proc. IEEE Global Commun. Conf.*, Dec. 2022, pp. 958–963.
[21] *Management and Orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3*, 3GPP, Sophia Antipolis, France, TS28.541, Rel-16 v16.6.0, Sep. 2020. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/28541.htm
[22] *NR; Physical Layer Procedures for Data*, 3GPP, Sophia Antipolis, France, TS38.214, Rel-16 v16.14.0, Jun. 2023. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/38214.htm
[23] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: An open and softwarized prototyping platform for NextG systems," in *Proc. ACM Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, Jun. 2021, pp. 415–426.
[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, (Adaptive Computation and Machine Learning Series), 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[25] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[26] L. Bonati et al., "Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*. Los Angeles, CA, USA, Dec. 2021, pp. 105–113.
[27] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: An open-source platform for LTE evolution and experimentation," in *Proc. 10th ACM Int. Workshop Wireless Netw. Testbeds, Exp. Evaluat. Characterization*, New York, NY, USA, Oct. 2016, pp. 25–32.
[28] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "OpenRAN Gym: AI/ML development, data collection, and testing for O-RAN on PAWR platforms," *Comput. Netw.*, vol. 220, Jan. 2023, Art. no. 109502.
[29] K. Ashfaq, G. A. Safdar, and M. Ur-Rehman, "Comparative analysis of scheduling algorithms for radio resource allocation in future communication networks," *PeerJ Comput. Sci.*, vol. 7, May 2021, Art. no. e546.
[30] A. Mamane, M. E. Ghazi, G.-R. Barb, and M. Oteşteanu, "5G heterogeneous networks: An overview on radio resource management scheduling schemes," in *Proc. 7th Mediterr. Congr. Telecommun. (CMT)*, Oct. 2019, pp. 1–5.
[31] P. K. Taksande, P. Chaporkar, P. Jha, and A. Karandikar, "Proportional fairness through dual connectivity in heterogeneous networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.
[32] M. H. Habaebi, J. Chebil, A. Al-Sakkaf, and T. Dahawi, "Comparison between scheduling techniques in long term evolution," *IIUM Eng. J.*, vol. 14, no. 1, pp. 67–76, 2013.
[33] J. Tan and W. Guan, "Resource allocation of fog radio access network based on deep reinforcement learning," *Eng. Rep.*, vol. 4, no. 5, 2022, Art. no. e12497.

HOJJAT NAVIDAN received the B.Sc. degree in electrical engineering from Shahid Beheshti University in 2019 and the M.Sc. degree in communication networks from the University of Tehran, Iran, in 2022. He is currently pursuing the Ph.D. degree in electrical engineering with the Internet Technology and Data Science Lab in collaboration with imec, Ghent University. His research interests include the applications of artificial intelligence and machine learning in wireless communications and networks, radio resource management, and 5G/6G networks.

MOSTAFA NASERI received the B.Sc. degree in electrical engineering from the Shiraz University of Technology and the M.Sc. degree in electronics and communication engineering from the Beijing University of Posts and Telecommunications in 2019. Prior to his master's, he worked with the University of Tehran. He is currently the Ph.D. Candidate with IDLab, Ghent University, Belgium. His research focuses on the generalization of machine learning algorithms in adaptive environments. He has been involved in projects related to reinforcement learning in wireless communication and deep learning applications in the ultrawideband angle of arrival estimation.

INGRID MOERMAN (Senior Member, IEEE) received the degree in electrical engineering and the Ph.D. degree from the Ghent University in 1987 and 1992, respectively, where she became a Part-Time Professor in 2000. She is currently combines a Full Professor position with part-time allocation with Ghent University and a Staff Member at IDLab, a Core Research Group of imec with research activities embedded with Ghent University and University of Antwerp. She is a Program Manager of the "Deterministic Networking" track, part of the connectivity program at imec, and in this role, she coordinates research activities on end-to-end wired/wireless networking solutions driven by professional and mission-critical applications that have to meet strict quality of service requirements in terms of throughput, bounded latency and reliability in challenging application areas like industrial automation, vehicular networks, safety-critical operations, and professional entertainment. She has a longstanding experience in running and coordinating national and EU research-funded projects.



ADNAN SHAHID (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2006 and 2010, respectively, and the Ph.D. degree in information and communication engineering from Sejong University, South Korea, in 2015. From March 2015 to August 2015, he worked as a Postdoctoral Researcher with Yonsei University, South Korea. Following that, from September 2015 to June 2016, he worked as an Assistant Professor with Taif University, Saudi Arabia. He is currently working as a Professor with the Internet Technology and Data Science Lab, Ghent University and imec. He is the Secretary and a Voting Member of IEEE P1900.8 (active PAR) Standard for Training, Testing, and Evaluating Machine-Learned Spectrum Awareness Models. He has been involved in several projects such as DARPA Spectrum Collaboration Challenge, European H2020 (eWINE, WiSHFUL), and ESA (CODYSUN, MRC100). His research interests include machine learning and artificial intelligence for wireless communications and networks, decentralized learning, radio resource management, the Internet of Things, 5G/6G networks, localization, and connected healthcare.