Open RAN Slicing for MVNOs With Deep Reinforcement Learning

Abderrahime Filali, *Member, IEEE*, Zoubeir Mlika, *Member, IEEE*, and Soumaya Cherkaoui, *Senior Member, IEEE*

***Abstract*—As 5G networks continue to be deployed and 6G networks begin to be envisioned, mobile network operators (MNOs) are embarking on a revolutionary transformation of the way they manage their networks. Various technology bricks are currently considered paramount in this transformation, including radio access network (RAN) slicing. The concept of an open radio access network (Open RAN) promises to provide more flexibility to support RAN slicing. However, RAN slicing in an O-RAN architecture raises a major challenge in achieving efficient resource sharing among slices, due to the diverse and permanent changes in RAN slices' QoS requirements. To overcome this challenge in a RAN environment involving an MNO and multiple mobile virtual network operators (MVNOs), we propose a two-level RAN slicing mechanism. The first level is executed on a long time-scale to allocate radio resources from the MNO to MVNOs while the second level is executed on a shorter time-scale to allocate MVNO resources to users. This mechanism improves the performance of the RAN slicing operation by enabling users to obtain the required resources as quickly as possible and with a high level of granularity. We formulate the two-level problem as two mathematical optimization problems and we study their NP hardness. To efficiently solve the two-level problem, we first propose a game-theoretic solution to solve the first-level resource allocation problem using a matching algorithm. Next, we propose a deep reinforcement learning (DRL) algorithm that uses the double deep *Q*-network procedure to solve the second-level resource allocation problem. The two proposed algorithms are coupled such that the DRL algorithm uses the solution obtained using the game-theoretic matching algorithm. We show through extensive simulations that the proposed two-level solution outperforms the current state-of-the-art solutions and achieves efficient performance.**

***Index Terms*—5G, deep reinforcement learning (DRL), matching game theory, network slicing (NS), NP-hardness, open radio access network (Open RAN).**

I. Introduction

THE NEW use cases brought by 5G and those envisioned for 6G are expected to open up new revenue streams and business partnerships driven by many verticals. However, these use cases come with high network performance requirements (e.g., number of users, coverage, reliability, etc.), which will only increase in diversity in the future. In order to optimize network operations, maximize coverage, and open up new revenue streams, mobile network operators (MNOs) are adopting different technologies to help make network operation and orchestration more efficient. Among these technologies, there is network slicing (NS) [1], which allows to depart from the classical service differentiation approach.

NS allows MNOs to divide their physical network, from the radio access network (RAN) to the core network, into multiple independent, virtualized, logical networks, called network slices, that operate on a common physical infrastructure. Each MNO then orchestrates logically separated slices across the network's end-to-end infrastructure resources, to strictly conform to the requirements of each slice (e.g., data rate, latency, security, etc.). The core network cannot be the only one involved in the slicing. The RAN also needs to reflect slicing by ensuring radio resource availability, selection, and isolation through adequate radio resource management to conform to each slice service-level agreement (SLA) [2].

The open radio access network (Open RAN) concept, such as the one promoted with the O-RAN architecture [3], promises to pave the way for MNOs toward an open and intelligent RAN environment [4] that allows for greater flexibility in slicing the RAN. An open RAN environment refers to the deployment of new standardized protocols and interfaces as well as interoperable equipment, allowing MNOs to combine and match solutions from different vendors. This prevents vendor lock-in by proprietary systems and increases competition among vendors, which in turn fosters innovation. Although the Open RAN openness provides many benefits, it increases the complexity of controlling the RAN. To overcome this complexity, more intelligence needs to be embedded in the management of the RAN system. To this end, the O-RAN architecture facilitates the exploitation of artificial intelligence (AI)/machine learning (ML) capabilities through RAN intelligent controllers (RICs) [5], [6]. Supported by the intelligence incorporated in the RICs, O-RAN promises to significantly improve the way MNOs can manage their networks toward a zero-touch network system.

NS presents new opportunities not only for MNOs but also for mobile virtual network operators (MVNOs) as the latter can find entirely new ways to differentiate their services.

Manuscript received 21 January 2024; accepted 7 February 2024. Date of publication 13 February 2024; date of current version 9 May 2024. This work was supported by the Natural Sciences and Engineering Research Council of Canada. *(Corresponding author: Abderrahime Filali.)*

Abderrahime Filali and Soumaya Cherkaoui are with the Department of Computer and Software Engineering, Polytechnique Montreal, Montreal, QC H3T 1J4, Canada (e-mail: abderrahime.filali@polymtl.ca; soumaya.cherkaoui@polymtl.ca).

Zoubeir Mlika is with the INTERLAB Research Laboratory, Faculty of Engineering, Department of Electrical and Computer Science Engineering, Université de Sherbrooke, Sherbrooke, QC J1K 2R1, Canada (e-mail: zoubeir.mlika@usherbrooke.ca).

Digital Object Identifier 10.1109/JIOT.2024.3365665

At the RAN level, an MNO can lease its radio resources, e.g., resource blocks (RBs), to several MVNOs. Then, MVNOs can perform a RAN slicing to assign the RBs rented from the MNO to their users [7]. Accordingly, two levels of RAN slicing can be considered, namely, RB allocation to MVNOs and RB allocation to users performed by the MNO and by MVNOs, respectively. In this article, we consider that the RBs allocation to MVNOs is performed by the MNO in a large time-scale and depends on an SLA between the MNO and MVNOs. On the other hand, each MVNO allocates RBs to its associated users in a short time-scale to meet their SLA in terms of data rate and latency.

To achieve highly efficient and cost-effective use of RAN resources, RAN slicing faces major challenges that remain open and deserve further investigation. These challenges include dynamic resource sharing while meeting the diverse and rigorous QoS requirements of RAN slices. A dynamic RAN resource sharing is expected to vary the resource allocated to a slice at different times and with different granularity. Without reliable dynamic resource-sharing mechanisms that support the QoS required by the different slices, resources can be either wasted in some slices or lacking in others. As a result, RAN resources will not be exploited efficiently, and the risk of not satisfying the QoS required by the slices can be significant. In addition, the complexity of the RAN slicing operation can be compounded due to the scarcity of radio resources and permanent changes in the RAN environment, e.g., user density and wireless channel transmission conditions. All this requires a very advanced level of agility and dynamism in the control and orchestration of RAN slicing operations.

To overcome these challenges, recent research efforts on RAN slicing have proposed a large variety of schemes and algorithms [8]–[16]. However, we have identified two major gaps. In fact, these works either: 1) present centralized RAN slicing solutions or 2) fail to consider users' individual resource requirements. In the centralized RAN slicing solutions, a central entity is exclusively responsible for slicing RAN resources, at different levels of granularity, among all the slices required by MVNOs and users. This can significantly impact the time required to create and deliver the desired RAN slices to the users. On the other hand, when the individual requirements of each user are not taken into account, it is generally the overall requirements of a RAN slice or an MVNO that are considered, which include the requirements of several users. In this case, there is a risk of not exactly meeting a user's QoS requirements either by underestimating or overestimating the RAN resources that should be reserved for that user. In addition, our work is also motivated by the fact that these works do not consider the cost of dynamic RAN slicing. In fact, for each MVNO, the RBs allocated to each of its users change each time according to the RBs available and the constraints of its environment. This reconfiguration of an MVNO's RAN slicing policy can increase the QoS of some of its users, but it can also decrease the QoS of others. A degradation in a user's QoS is considered as a RAN slicing reconfiguration cost. Therefore, to further optimize the performance of the RB allocation operation, the RAN slicing reconfiguration cost should be considered.

The novelty of this article lies in proposing a decentralized RAN slicing mechanism while considering the individual QoS requirements of each user. In fact, we propose a two-level RAN slicing (2LRS) mechanism that couples RB allocation to MVNOs in a large time-scale and RB allocation to users in a short time-scale. Allocating RBs at two levels improves the performance of the RAN slicing operation, since it enables users with the desired resources as fast as possible and with a very high level of granularity. In the first level, we leverage game theory to formulate the problem of RB allocation to MVNOs and design a matching algorithm to solve it. In the second level, based on the matching game result of the first RB allocation level, we develop a deep reinforcement learning (DRL)-based algorithm used by each MVNO to assign its RBs to its associated users. Note that we consider, in the second level, the individual requirements of each user in terms of data rate and delay. In addition, the matching game between RBs and MVNOs is based on MVNOs' preferences for RBs, which are related to the data rate achieved when assigning these RBs to their users in the second level. This can improve the QoS provided to users since each RB will be allocated to the MVNO that prefers it.

The proposed 2LRS mechanism is designed in an O-RAN architecture. Indeed, we consider that the matching game algorithm and the DRL-based algorithm can be executed by an O-RAN r-application (rApp) and an O-RAN x-application (xApp), respectively. The rApps and xApps are hosted in the non-real time (non-RT) RIC and near-real time (near-RT) RIC, respectively. They are standalone applications designed to automate the management of RAN functions. In the 2RLS mechanism, an rApp can be used to execute the matching game algorithm since the RB allocation operation to MVNOs can be performed in a large time-scale. In contrast, an xApp can be leveraged to run the DRL-based algorithm due to the fact that RB allocation operation to users should be performed in a short time-scale.

The main contributions of this article can be summarized as follows.

1) We model the two-level RB allocation problem as two mathematical optimization problems and study their computational complexity.
2) We propose a matching game algorithm and a DRL algorithm to solve the problem of allocating RBs to MVNOs and assigning RBs to users, respectively.
3) We design a RAN slicing mechanism that combined both algorithms in an O-RAN architecture.
4) We evaluate, through extensive simulations, the performance of our proposed RAN slicing algorithm and show that it performs benchmark algorithms.

The remainder of this article is organized as follows. In Section II, we present and discuss recent related work. In Section III, we provide the system model. In Section IV, we formulate the optimization models. In Section V, we study and prove the NP-hardness of each optimization model. In Section VI, we present the proposed RAN slicing mechanism. In Section VII, we evaluate the performance of our mechanism and discuss the obtained results. Finally, we conclude this article in Section VIII.

II. Related Work

In a multi-MVNO environment, RAN slicing has been investigated in the literature with many schemes presented to efficiently share the RAN radio resources. Vilà et al. [8] proposed a RAN slicing approach to share radio resources among multiple MVNOs in a multicell scenario. This approach is based on a multiagent RL algorithm where each MVNO has a deep Q-network (DQN) agent. Each agent learns an efficient radio resource allocation policy and then collaborates with other agents to satisfy the SLA between the MVNOs and the infrastructure provider, as well as to improve the exploitation of radio resources. Awada et al. [9] proposed a noncooperative game theory-based algorithm to solve the RAN slicing problem between MVNOs. In this game, MVNOs are competing for a common set of shared RBs in order to maximize their utility related to their users' satisfaction while minimizing the cost of leasing the RBs from the infrastructure provider. However, neither approach considers the individual requirements of each user. Instead, it is the overall requirements of a slice or an MVNO that are considered to ensure their satisfaction.

Ravi et al. [10] designed a greedy algorithm to allocate to each user of each MVNO the RBs required to satisfy its data rate requirements. For each user, the proposed algorithm assigns the optimal number of RBs and how much information should be modulated and coded in each RB based on the channel quality indicator reported by this user. The proposed algorithm considers the data rate only as user requirements. To guarantee the data rate and latency requirements of users in a multi-MVNO environment, an adapted version of the genetic algorithm is developed in [11]. This algorithm partitions radio resources among users by taking into account their location and distribution as well as the cell load characteristics. These two algorithms perform the allocation of radio resources to users of all MVNOs in a centralized manner. Ma et al. [12] proposed a hierarchical RAN slicing framework to perform fair resource allocation among MVNOs and between users. In the lower level, an approximation algorithm is used to allocate radio resources to users considering requirements thresholds in terms of data rate and delay. Based on the results obtained at the lowest level, another approximation algorithm is used to derive a fair solution for resource allocation among MVNOs.

Tran and Le [13] investigated the interaction between service providers and users using the Stackelberg game theory. They modeled the competition between service providers for the sale of radio resources to users. Then, a distributed algorithm is designed to find the game equilibrium and achieve efficient payoffs for service providers and users. Stackelberg game theory has also been used in [14], where the infrastructure provider, MVNOs, and users take their own RAN resource allocation strategies to maximize their payoffs. The RAN resource allocation in such a game is performed as follows. The infrastructure provider defines the prices of its RAN resources for the MVNOs, who in turn redefine these prices to sell them to users. Then, each user chooses the amount of resources they want to purchase from the MVNO. Each MVNO collects this information and reports it to the infrastructure provider. The infrastructure provider then creates RAN slices for the MVNOs based on the MVNOs' requirements, which allocate the obtained resources to their users. To solve the formulated game, a heuristic algorithm is proposed. Nevertheless, whenever the infrastructure provider wants to create the slices required by the users, it should wait for the response from the lower level. This can significantly increase the time needed to create and deliver the desired slice.

To deal with the diverse user demands and the dynamics of the RAN environment, Wang et al. [15] developed a DRL-based algorithm that dynamically allocates RAN resources to MVNOs' slices. This algorithm observes the request queues of all MVNOs' slices and assigns to each slice a fraction of RAN resources. The proposed DRL algorithm considers as observation the entire requirements of a slice defined as a queue of incoming service requests, which may not enable a fair allocation of RAN resources among users. In [16], the multitenant radio resource allocation problem is formulated as noncooperative stochastic game. In this game, the competing tenants bid to orchestrate the limited channel access opportunities over their users. In order to approximate the optimal RAN slicing policy of each tenant, a DRL-based algorithm is proposed. In contrast to our RAN slicing mechanism, the channel state between the users and the base station is not considered as part of the observation, which is an important factor that needs to be taken into account in wireless resource allocation problems.

III. System Model

We consider a downlink RAN system with a single gNodeB owned by an MNO. The gNodeB spectrum resources considered in this work are the RBs denoted by $\mathcal{K} = \{1, \ldots, K\}$. The RBs are represented as a time–frequency grid. The MNO leases virtual RAN slices to a set of MVNOs, denoted by $\mathcal{M} = \{1, \ldots, M\}$. Each RAN slice rented to an MVNO is defined by a certain number of RBs. The set of users associated with the gNodeB is denoted by $\mathcal{U} = \{1, \ldots, U\}$, which are randomly located within the coverage area of the gNodeB. Each MVNO m has a subset of users, i.e., subscribers, denoted by $\mathcal{U}_m = \{1, \ldots, U_m\}$, where $\mathcal{U}_m \cap \mathcal{U}_{m'} = \emptyset \ \forall m' \neq m$, and $\cup_{m \in \mathcal{M}} \mathcal{U}_m = \mathcal{U}$. Two types of users are considered, namely, URLLC users and eMBB users. To identify an eMBB user and a URLLC user, we denote the former by $u \in \mathcal{U}_e^m$ and the latter by $u \in \mathcal{U}_u^m$, where $\mathcal{U}_e^m \cap \mathcal{U}_u^m = \emptyset$ and $\mathcal{U}_e^m \cup \mathcal{U}_u^m = \mathcal{U}_m$. eMBB users (resp., URLLC users) have a QoS requirement in terms of data rate (resp., delay).

For a given gNodeB, the RB allocation is performed in two levels. In the first level, the MNO allocates a number of RBs, from the set of RBs $\mathcal{K}$, to each MVNO. The set of RBs allocated to MVNO $m \in \mathcal{M}$ is denoted by $\mathcal{K}_m$, where $\mathcal{K}_m \subseteq \mathcal{K}$. In the second level, each MVNO $m \in \mathcal{M}$ allocates a number of RBs to each of its users, from the RBs preallocated by the MNO, i.e., $\mathcal{K}_m$, to meet their QoS requirements in terms of data rate and delay. In the rest of this article, we refer to the first RB allocation level and the second RB allocation level as the MNO slicing level and the MVNO slicing level, respectively. We assume that time is divided into T slots

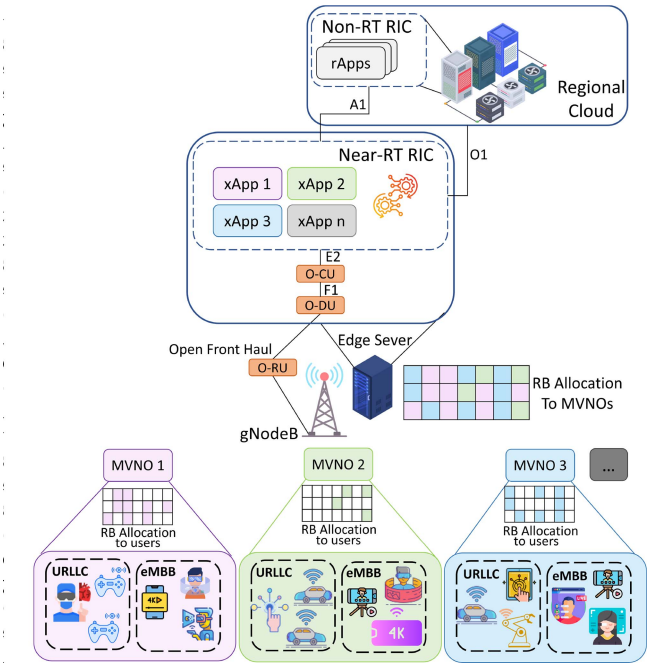

Fig. 1. Overview of the O-RAN architecture.

and the MVNO slicing level is performed in a short time-scale, i.e., at each time slot, whereas the MNO slicing level is performed in a large time-scale, i.e., at each period of T time slots.

A. O-RAN Architecture

Fig. 1 illustrates the O-RAN architecture proposed in this work. The main components of this architecture are the O-RAN radio unit (O-RU), the O-RAN distributed unit (O-DU), the O-RAN central unit (O-CU), the near-RT RIC, and the non-RT RIC. The O-RU, O-DU, and O-CU are responsible for handling the gNodeB functions, where each of them includes a part of these functions depending on the chosen functional split option [17]. In general, the O-RU processes the radio frequency received from the physical layer while the O-DU and O-CU perform the functions related to the upper layers of the RAN. The non-RT RIC and the near-RT RIC present the intelligence of the O-RAN architecture, which automatically controls the RAN operation by leveraging AI/ML capabilities. The non-RT RIC deals with RAN control tasks that tolerate a large time-scale, i.e., with a latency > 1 s, such as lifecycle management for all RAN elements. The near-RT RIC is responsible for handling RAN control tasks that tolerate a short time-scale, i.e., with a latency < 1 s, such as AI/ML inference. The non-RT RIC and near-RT RIC can host standalone applications called rApp and xApp, respectively, to manage and automate different RAN functions.

To deploy O-RAN components in an MNO network, the O-RAN alliance has defined six deployment scenarios [18]. There is no ideal scenario for deploying O-RAN components, since the choice of where to place each component in the network depends on various factors related to the MNO's resource constraints and the services that the MNO intends to provide. In this work, we opt for "Scenario A" where 1) the O-RU is located at the MNO cell site; 2) the O-DU, O-CU, and near-RT RIC are located in an edge server; and 3) the non-RT RIC is located in a regional cloud.

B. MNO Slicing Level

The slicing operation in the MNO slicing level is performed in a large time-scale. Therefore, we consider that the slicing operation is executed by a rApp application since the latter can operates in non-RT.

In the MNO slicing level, each RB $k \in \mathcal{K}$ is exclusively allocated to one MVNO $m \in \mathcal{M}$. Therefore, we define the allocation relation between an RB $k \in \mathcal{K}$ and an MVNO $m \in \mathcal{M}$ by the following binary variable:

$$y_{k,m} = \begin{cases} 1, & \text{if } k \in \mathcal{K} \text{ is assigned to } m \in \mathcal{M} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

And we define the binary variable z_m to be 1 if and only if MVNO m is considered.

The RB allocation between MVNOs should respect the following two constraints. The first constraint, given in (2), states that an RB must be allocated to only one MVNO at a time

$$\sum_{m \in \mathcal{M}} y_{k,m} \leq 1 \ \forall k \in \mathcal{K}. \tag{2}$$

The second constraint, given in (3), defines the number of RBs that should be allocated to an MVNO, which depends on the SLA between the MNO and the MVNOs. Indeed, the MNO must guarantee a minimum number of RBs to each MVNO, which we call the minimum quota of RBs and denoted by $q_m^{\min}$

$$\sum_{k \in \mathcal{K}} y_{k,m} \geq q_m^{\min} \ \forall m \in \mathcal{M}. \tag{3}$$

C. MVNO Slicing Level

The slicing operation in the MVNO slicing level is performed in a short time-scale, where each MVNO $m \in \mathcal{M}$ allocates a number of RBs to each of its users. Thus, for each MVNO, we consider that we have an xApp that performs the RB allocation to its users in a near-RT.

In the MVNO slicing level, we consider the orthogonal frequency-division multiple access (OFDMA) technique. To ensure the orthogonality of downlink transmissions among users of an MVNO $m \in \mathcal{M}$, each RB $k \in \mathcal{K}_m$ is exclusively allocated to one user $u \in \mathcal{U}_m$. Therefore, we define the allocation relation between an RB $k \in \mathcal{K}_m$ and a user $u \in \mathcal{U}_m$ by the following binary variable:

$$x_{u,m}^{k,t} = \begin{cases} 1, & \text{if } k \in \mathcal{K}_m \text{ is assigned to } u \in \mathcal{U}_m \text{ at time-slot } t \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

And we define the binary variable $v_{u,m}^t$ to be 1 if and only if user u is served at time slot t by MVNO m.

The achievable data rate of the user $u \in \mathcal{U}_m$ over the RB $k \in \mathcal{K}_m$ can be calculated as follows:

$$r_{u,m}^{k,t} = W \log_2 \left(1 + \frac{P_{u,m} g_{u,m}^{k,t}}{W \sigma^2} \right) \tag{5}$$

where W denotes the bandwidth of an RB, $P_{u,m}$ is the transmission power of the gNodeB to the user $u \in \mathcal{U}_m$, and σ^2 is the power of the additive white Gaussian noise (AWGN). We assume that the bandwidth and the downlink transmission power are the same for all RBs. $g^{k,t}_{u,m}$ is the downlink channel gain between the gNodeB and the user $u \in \mathcal{U}_m$ on RB k at time t, and can be calculated as follows:

$$g^{k,t}_{u,m} = 10^{-\frac{\text{PL}(d^t_{u,m})}{10}} \tag{6}$$

where $\text{PL}(d^t_{u,m})$ is the path loss between the user $u \in \mathcal{U}_m$ and the gNodeB. In this work, the path-loss model considered is conform to 3GPP specifications [19] and can be calculated as follows:

$$\text{PL}(d^t_{u,m}) = 28 + 22\log_{10}(d^t_{u,m}) + 20\log_{10}(f_c) + \sigma^{k,t}_{SF} \tag{7}$$

where $d^t_{u,m}$ is the distance between the user $u \in \mathcal{U}_m$ $\forall m \in \mathcal{M}$ and the gNodeB, f_c is the central frequency of the 5G band, and $\sigma^{k,t}_{SF}$ is the shadow fading over RB k. We consider that σ_{SF} follows a normal distribution.

Since the slicing operation in the MVNO slicing level is performed in a short time-scale, then an MVNO provides a slicing policy among its users in each time slot $t \in T$. For an MVNO $m \in \mathcal{M}$, the total achievable data rate of the user $u \in \mathcal{U}_m$ at $t \in T$ is defined as follows:

$$r^t_{u,m} = \sum_{k \in \mathcal{K}_m} r^{k,t}_{u,m} x^{k,t}_{u,m}. \tag{8}$$

The total achievable data rate of the MVNO $m \in \mathcal{M}$ at $t \in T$ is defined as follows:

$$r^t_m = \sum_{u \in \mathcal{U}_m} r^t_{u,m}. \tag{9}$$

The transmission delay required for a user to download a packet at $t \in T$ can be calculated as follows:

$$d^t_{u,m} = \frac{\xi_u}{r^t_{u,m}} \tag{10}$$

where ξ_u is the packet size of the user $u \in \mathcal{U}_m$.The size of a packet is related to the type of the user, i.e., if the user is an eMBB user or a URLLC user.

In each time slot $t \in T$, each user $u \in \mathcal{U}_m$ achieves a data rate according to the slicing policy of the MVNO $m \in \mathcal{M}$. For a given MVNO $m \in \mathcal{M}$, the slicing policy changes in each time slot depending on its available RBs and the constraints of its environment. This reconfiguration in the slicing policy can either increase the QoS of a user or decrease it. We consider the QoS degradation of an MVNO's users as a slicing reconfiguration cost. The slicing reconfiguration cost is related to the type of users. For a given MVNO $m \in \mathcal{M}$, the reconfiguration cost of an eMBB user $u \in \mathcal{U}^m_e$ in the time slot $t \in T$ is defined as follows:

$$c^t_{u,m} = \begin{cases} C_e, & \text{if } r^t_{u,m} < r^{t-1}_{u,m} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

whereas the reconfiguration cost of a URLLC user $u \in \mathcal{U}^m_u$ in time slot $t \in T$ is defined as follows:

$$c^t_{u,m} = \begin{cases} C_u, & \text{if } d^t_{u,m} > d^{t-1}_{u,m} \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The slicing reconfiguration cost of an eMBB (resp., a URLLC user) is related to the data rate (resp., the delay) achieved by this user. In principal, the costs C_e and C_u given, respectively, in (11) and (12) can be any positive real numbers but, in practice, these costs should be proportional to the QoS variation. In the simulations, we choose $C_e = r^{t-1}_{u,m} - r^t_{u,m}$ and $C_u = d^t_{u,m} - d^{t-1}_{u,m}$.

Finally, the overall slicing reconfiguration cost of the MVNO $m \in \mathcal{M}$ is defined as follows:

$$c^t_m = \sum_{u \in \mathcal{U}^m_e} c^t_{u,m} + \sum_{u \in \mathcal{U}^m_u} c^t_{u,m}. \tag{13}$$

IV. Problem Formulation

In both slicing levels, we aim for an optimal allocation of RBs in terms of achieved data rate and experienced delay. In the MNO slicing level, the objective is to allocate RBs to MVNOs while respecting the SLA between the MVNOs and the MNO. In the MVNO slicing level, the purpose is to reach an optimal allocation of RBs to the eMBB and URLLC users that meets their QoS requirements in terms of data rate and delay.

A. MNO Slicing Level

In order to optimize the RBs allocation operation in the MNO slicing level while satisfying the SLA between the MNO and the MVNOs, we formulate the following optimization problem:

$$\underset{y}{\text{maximize}} \quad \sum_{m \in \mathcal{M}} \sum_{k \in \mathcal{K}} y_{k,m} \tag{14a}$$

$$\text{subject to} \quad y_{k,m} \leq z_m \ \forall m \in \mathcal{M} \ \forall k \in \mathcal{K} \tag{14b}$$

$$\sum_{m \in \mathcal{M}} y_{k,m} \leq 1 \ \forall k \in \mathcal{K} \tag{14c}$$

$$\sum_{k \in \mathcal{K}} y_{k,m} \geq q^{\min}_m z_m \ \forall m \in \mathcal{M} \tag{14d}$$

$$\sum_{k \in \mathcal{K}} y_{k,m} \leq q^{\max}_m z_m \ \forall m \in \mathcal{M} \tag{14e}$$

$$z_m \in \{0,1\}, y_{k,m} \in \{0,1\} \ \forall k \in \mathcal{K} \ \forall m \in \mathcal{M}. \tag{14f}$$

The objective function (14a) maximizes the total number of allocated RBs to MVNOs. Constraint (14b) guarantees that if MVNO m is not considered, i.e., $z_m = 0$, then we should not allocate any RB to it. Constraint (14c) ensures that each RB is allocated to only one MVNO at a time. Constraint (14d) guarantees that the SLA between the MVNOs and the MNO, in terms of the number of RBs, is respected. Constraint (14e) guarantees a fair RBs allocation among the MVNOs by forcing the number of RBs allocated to each MVNO $m \in \mathcal{M}$ to not exceed a maximum number $q^{\max}_m$. Constraint (14f) lists the optimization variables.

B. MVNO Slicing Level

Given the allocated resources for each MVNO $m \in \mathcal{M}$ in the MNO slicing level, the objective in the MVNO slicing level is twofold: 1) find an optimal allocation of RBs to users that improves the users' QoS and 2) decreases the slicing

reconfiguration cost. For this purpose, we transform these two objectives into a single objective function, for each time slot, as follows:

$$\underset{x}{\text{maximize}} \quad \sum_{u\in\mathcal{U}_e^m}\left(\bar{r}_{u,m}^t - c_{u,m}^t\right) - \sum_{u\in\mathcal{U}_u^m}\left(\bar{d}_{u,m}^t + c_{u,m}^t\right) \tag{15a}$$

$$\text{subject to} \quad x_{u,m}^{k,t} \leq v_{u,m}^t \ \forall u \in \mathcal{U} \ \forall k \in \mathcal{K}_m \tag{15b}$$

$$\sum_{u\in\mathcal{U}_m} x_{u,m}^{k,t} \leq 1 \ \forall k \in \mathcal{K}_m \tag{15c}$$

$$\sum_{k\in\mathcal{K}_m} x_{u,m}^{k,t} \leq K_{\max}^u v_{u,m}^t \ \forall u \in \mathcal{U}_m \tag{15d}$$

$$r_{u,m}^t \geq R_{\min}^u v_{u,m}^t \ \forall u \in \mathcal{U}_e^m \tag{15e}$$

$$d_{u,m}^t \leq D_{\max}^u v_{u,m}^t \ \forall u \in \mathcal{U}_u^m \tag{15f}$$

$$v_{u,m}^t \in \{0,1\}, x_{u,m}^{k,t} \in \{0,1\} \ \forall u \in C \ \forall k \in \mathcal{K}_m. \tag{15g}$$

The objective function (15a) defines a joint optimization problem, where the first part maximizes the total sum of the eMBB users' data rates and the second part minimizes the total sum of the URLLC users' delays, at time slot $t \in T$. Constraint (15b) guarantees that if user $u \in \mathcal{U}_m$ is not served, i.e., $v_{u,m}^t = 0$, then we should not allocated any RB to it. Constraint (15c) ensures that each RB is allocated to only one user at a time. Constraint (15d) guarantees a fair RBs allocation among the users by forcing the number of RBs allocated to each user $u \in \mathcal{U}_m$ to not exceed a maximum number $K_{\max}^u$ for each user u. Constraint (15e) ensures that the data rate achieved by an eMBB user must be greater than a minimum threshold $R_{\min}^u$ for each eMBB user u. Constraint (15f) states that the delay of a URLLC user should not exceed a maximum threshold $D_{\max}^u$ for each URLLC user u. Constraint (15g) lists the optimization variables.

V. Computational Complexity Analysis

In the following, we use the participant maximization problem (PMP) [20], which is NP-hard, to prove the NP-hardness of problems (14a) and (15a). The PMP is defined as follows.

Definition 1 (PMP): Instance: Given a number n of students and m of seminars with capacities $B_1, \ldots, B_m \in \mathbb{N}$ and minimum quantities $q_1, \ldots, q_m \in \mathbb{N}$ (where $q_j \leq B_j$ for all $j = 1, \ldots, m$).

Task: Match a subset of the students to the seminars such that the number of students in each seminar j is either zero (if seminar j is not assigned to any student) or at least q_j and at most B_j and the total number of students assigned to seminars is maximized.

A. NP-Hardness of Problem (14a)

Theorem 1: Problem (14a) is NP-hard.

Proof: We reduce the PMP to (14a).

The mathematical statement of (14a) is equivalent to finding an allocation of the RBs set $\mathcal{K}$ to the MVNOs set such that: 1) the number of RBs assigned to the MVNOs is maximized; 2) each RB is allocated to only one MVNO at a time; 3) the maximum number of RBs allocated to an MVNO m (if MVNO m considered) does not exceed $q_m^{\max}$; and 4) the minimum number of RBs allocated to an MVNO m (if MVNO m considered) must be greater than a minimum required threshold $q_m^{\min}$.

To reduce the PMP to (14a), we let 1) the students be the RBs; 2) the seminars be the MVNOs; 3) the minimum quantity q_u be $q_m^{\max}$ for each seminar (MVNO) m; and 4) the maximum capacity B_m be $q_m^{\min}$ for each MVNO m.

The PMP is now clearly reduced to Problem (14a) in polynomial-time. Since the PMP is weakly NP-hard, we conclude that (14a) is also NP-hard, which proves the theorem. ■

B. NP-Hardness of Problem (15a)

Theorem 2: Problem (15a) is NP-hard.

Proof: We prove the theorem by considering a restricted version of (15a). We consider the following restricted version of P.

1) The slicing configuration cost is not considered.
2) There is only the eMBB users.
3) The RBs preallocated by the rApp $\mathcal{K}_m$ for $m \in \mathcal{M}$ are known, i.e., the MNO slicing level is already performed by the rApp.
4) The data rate r_u^k between user u and the RB k is set to 1.

In this case, (15a) becomes equivalent to the following problem at time slot t for MVNO m:

$$\underset{x}{\text{maximize}} \quad \sum_{u\in\mathcal{U}_e}\sum_{k\in\mathcal{K}} x_u^k \tag{16a}$$

$$\text{subject to} \quad x_u^k \leq v_u \ \forall u \in \mathcal{U}_e \ \forall k \in \mathcal{K} \tag{16b}$$

$$\sum_{u\in\mathcal{U}_e} x_u^k \leq 1 \ \forall k \in \mathcal{K} \tag{16c}$$

$$\sum_{k\in\mathcal{K}} x_u^k \leq K_{\max}^u v_u \ \forall u \in \mathcal{U}_e \tag{16d}$$

$$\sum_{k\in\mathcal{K}} x_u^k \geq \mathcal{R}_{\min}^u v_u \ \forall u \in \mathcal{U}_e \tag{16e}$$

$$v_u \in \{0,1\}, x_u^k \in \{0,1\} \ \forall u \in \mathcal{U}_e \ \forall k \in \mathcal{K}. \tag{16f}$$

Note that we omitted the indexes t and m for simplicity.

The mathematical statement of (16a) is equivalent to finding an allocation of the RBs set $\mathcal{K}$ to the eMBB user set such that: 1) the number of RBs assigned to the eMBB users is maximized; 2) each RB is allocated to only one eMBB user at a time (16c); 3) the maximum number of RBs allocated to an eMBB user (if user u is served) does not exceed $\mathcal{K}_{\max}^u$ (16d); and 4) the minimum number of RBs allocated to an eMBB user (if user u is served) must be greater than a minimum required threshold $\mathcal{R}_{\min}^u$ (16e).

To reduce the PMP to (16a), we let 1) the students be the RBs; 2) the seminars be the eMBB users; 3) the minimum quantity q_u be $K_{\max}^u$ for each seminar (user) u; and 4) the maximum capacity B_u be $R_{\min}^u$ for each user u.

Problem (15a) is now clearly reduced to the PMP problem in polynomial-time. Since the PMP is weakly NP-hard, we conclude that the restricted problem formulated in (16a) is also NP-hard, which proves the theorem, i.e., (15a) is NP-hard. ■

VI. Two-Level RAN Slicing Mechanism

The RB allocation problem in the MNO slicing level can be considered as an generalized assignment problem [20] with minimum quotas. In contrast, the RB allocation problem in the MVNO slicing level is more challenging to solve since it depends on the MNO slicing level solution and also includes more challenging objective function and data rate and delay constraints. To obtain an efficient solution to the MNO slicing level RB allocation, we propose a matching game theory approach. On the other hand, to solve the more challenging problem of MVNO slicing level RB allocation, we propose a DRL approach. Our proposed approach is a two-level RAN slicing (2LRS) mechanism that assigns RBs to MVNOs (resp., to users) in a long-term (resp., short-term) procedure. First, we model the MNO slicing problem as a one-to-many matching game. Then, we model the MVNO slicing problem as single-agent MDP. Finally, we combine the two models and we present our 2LRS mechanism.

A. RB Allocation to MVNOs

We have shown that the RB allocation problem in the MNO slicing level is NP-hard. Accordingly, computing an optimal solution requires exponential time in the worst case. Furthermore, in the case of a large-scale RAN system, the problem cannot be approximated properly and will be very difficult to solve in polynomial time. To overcome this challenge, we leveraged game theory to formulate the optimization problem (14a) as a one-to-many matching game and designed a matching algorithm to solve it in a feasible computational time complexity. In fact, formulating the optimization problem (14a) using the matching game allows to find a stable solution while considering multiconstraints, namely, the constraints of the minimum and maximum number of RBs, i.e., (14d) and (14e). In addition, it has been shown that approaches based on matching game algorithms can enable highly efficient solutions to resource allocation problems in general [21], [22], and to RAN slicing problems in particular [23], [24].

1) Matching Game Formulation: To solve the association problem between the MVNOs and the RBs, we formulate the problem as a one-to-many matching game. Two disjoint sets of players are considered, the set of MVNOs $\mathcal{M}$ and the set of RBs $\mathcal{K}$. In such a matching game, each MVNO can have multiple RBs, but each RB can only be assigned to at most one MVNO. For a given MVNO, the constraints of the minimum and maximum number of RBs, i.e., (14d) and (14e), are called minimum quota and maximum quota, respectively. In the following, we define the considered one-to-many matching game and its key concepts, namely, the blocking pairs and the preferences of a player over the other set of players.

Definition 2 (One-to-Many Matching Game): A one-to-many matching μ is a function from $\mathcal{M} \rightarrow \mathcal{K}$ that satisfies:

1) $\forall m \in \mathcal{M},\ \mu(m) \subseteq \mathcal{K}$;
2) $\forall k \in \mathcal{K},\ \mu(k) \in \mathcal{M}$;
3) $\mu(k) = m$ if and only if $k \in \mu(m)$

where $\mu(k) = m$ implies that $y_{k,m} = 1$, otherwise $y_{k,m} = 0$, which shows that the proposed matching game satisfies the constraints (14c). Moreover, μ is a feasible matching if it meets the quotas constrains, where $q_m^{\min} \leq |\mu(m)| \leq q_m^{\max}$, i.e., (14d) and (14e).

The objective of this matching game is to find an appropriate association between the RBs and the MVNOs that meets their preferences and requirements. We use the notation $k \succ_m k'$ to indicate that MVNO m prefers to be allocated RB k rather than k', and the notation $m \succ_k m'$ to indicate that RB k prefers to be allocated to MVNO m rather than m'.

Definition 3 (A Blocking Pair): An RB–MVNO pair (k, m) is said to be a blocking pair of the matching μ, if it satisfies:

1) $m \succ_k \mu(k)$;
2) $k \succ_m k'$, where $\mu(k') = m$.

A matching is said to be stable if there is no incentive for any pair (k, m) for all $k \in \mathcal{K}$ and $m \in \mathcal{M}$ to deviate from μ. In other words, there are no blocking pairs.

The allocation of RBs to the MVNOs is based on a classification (preference-based) procedure as follows. An MVNO classifies the RBs from best to worst based on their weights. The weight of an RB is related to the data rate achieved when allocating this RB to users in the MVNO slicing level. To find out the data rate achieved by users on the RBs, an MVNO can send pilot signals to its associated users. For a given MVNO $m \in \mathcal{M}$, the weight of RB $k \in \mathcal{K}$ is defined as follows:

$$w_{k,m}^{t} = w_{k,m}^{t-1} + \sum_{u \in \mathcal{U}_m} x_{u,m}^{k,t} r_{u,m}^{k,t}. \tag{17}$$

According to (17), the weight of an RB k allocated to MVNO m in time slot t is equal to the sum-rate achieved at t using RB k of all users assigned to MVNO m plus the value of the weight at the previous time slot t. Note that if an RB has never been used by an MVNO, its weight is set to 0.

Definition 4 (MVNO Preference List): The preference list of the MVNO $m \in \mathcal{M}$ is denoted by Γ_m and contains RBs sorted in descending order according to their weight, where

$$k \succ_m k' \Longleftrightarrow w_{k,m} \geq w_{k',m} \tag{18}$$

Definition 5 (RB Preference List): An RB classifies the MVNOs randomly. Thus, the preference list of RB $k \in \mathcal{K}$ is denoted by Γ_k and contains the randomly sorted MVNOs.

2) MSDA Algorithm: To solve standard matching games, Gale and Shapley [25] proposed the deferred acceptance algorithm (DAA), which achieves stable and optimal matching between the players. However, the standard DAA does not support the minimum quota constraint defined in the formulated one-to-many matching game, which is what makes the matching problem with minimum quotas NP-hard [20]. Indeed, the DAA fails to achieve a stable matching when applied with the minimum quota constraint. To satisfy the minimum quotas of the MVNOs, some RBs should be allocated to lower ranked MVNOs in their preference lists. Therefore, an appropriate mechanism is needed to identify which RBs should be allocated to MVNOs with unsatisfied minimum quotas.

The work in [26] proposed an efficient mechanism to solve the minimum quota matching problems called multistage

deferred acceptance (MSDA) algorithm. The main idea of the MSDA algorithm is to reserve enough RBs to meet the minimum quotas of MVNOs. Indeed, some RBs will have the privilege to choose their preferred MVNOs according to their preference lists while others will be reserved to be allocated to the lowest preferred MVNOs. For this purpose, this mechanism introduces the concept of precedence list (PL), which ranks all the RBs based on their priority. In other words, the higher the priority of an RB, the more it will be allocated to a highly ranked MVNO in its preference list. We define the priority of an RB $k \in \mathcal{K}$ at time t based on its weight over MVNOs as follows:

$$\pi_k^t = \sum_{m \in \mathcal{M}} w_{k,m}^t. \tag{19}$$

Equation (19) implies that giving a high priority to an RB according to its weight over MVNOs can improve the QoS provided to users since this RB will be allocated to its preferred MVNO.

Definition 6 (PL): The PL ranks all RBs in a descending order according to their priority as follows:

$$k \succ_{\text{PL}} k' \Longleftrightarrow \pi_k \geq \pi_{k'}. \tag{20}$$

When using PL, a matching game is said to be stable if there are no PL-blocking pairs. Therefore, Definition 3 should be modified to obtain Definition 7.

Definition 7 (A PL-Blocking Pair): An RB–MVNO pair (k, m) is said to be a PL-blocking pair of the matching μ, if it satisfies:

1) $m \succ_k \mu(k)$;
2) $k \succ_m k'$ and $k \succ_{\text{PL}} k'$, where $\mu(k') = m$.

The pseudocode of the proposed MSDA algorithm is presented in Algorithm 1. The MSDA algorithm takes as input the PL list, the preference list of each player, the minimum and the maximum quota of each MVNO $m \in \mathcal{M}$. The output of the MSDA algorithm is a Pareto-optimal matching μ that allocates each RB $k \in \mathcal{K}$ to exactly one MVNO and each MVNO $m \in \mathcal{M}$ to at least $q_m^{\min}$ and at most $q_m^{\max}$ RBs. The MSDA algorithm runs in multiple stages until all RBs are allocated. It starts at stage $s = 0$ by initializing the set of reserved RBs $\mathcal{E}^0$ using the PL, the minimum and the maximum quota of each MVNO $q_m^{0,\min}$, $q_m^{0,\max}$ by $q_m^{\min}$ and $q_m^{\max}$, respectively. In stage s, the algorithm calculates e^s, the number of RBs that should be temporarily reserved to satisfy the minimum quota of the MVNOs (see line 2 in the pseudocode). Then, it builds the set of reserved RBs, denoted by $\mathcal{E}^s$, which contains the e^s least preferred RBs of PL. After reserving some RBs to satisfy the minimum quotas, the MSDA algorithm checks if there are still unreserved RBs. If the remaining set of RBs is not empty, i.e., $\mathcal{E}^{s-1} \backslash \mathcal{E}^s \neq \emptyset$, the DAA is applied to this set with the maximum quota equals to $q_m^{s,\max}$ $\forall m \in \mathcal{M}$. This enables each unreserved RB to eventually be allocated to its preferred MVNO. If the remaining set of RBs is empty, i.e., $\mathcal{E}^{s-1} \backslash \mathcal{E}^s = \emptyset$, it means that the minimum quotas are equal to the reserved RBs. Therefore, the DAA is applied to the set of reserved RBs $\mathcal{E}^s$ with the maximum quota equals to $q_m^{s,\min}$ $\forall m \in \mathcal{M}$.

Algorithm 1 MSDA for MNO Slicing Level

Input: $PL, \succ_k \ \forall k \in \mathcal{K}, \succ_m, q_{\min}^m, q_{\max}^m \ \forall m \in \mathcal{M}$;
Output: RB matching μ;
Initialize:
$\mu(m) = \emptyset, q_m^{0,\min} = q_m^{\min}, q_m^{0,\max} = q_m^{\max} \ \forall m \in \mathcal{M}, \mathcal{E}^0 = PL.$

1: $s \leftarrow 1$
2: **while** $PL \neq \emptyset$ **do**
3: Calculate the number of reserved RBs $e^s = \sum_{m \in \mathcal{M}} q_m^{s,\min}$;
4: $\mathcal{E}^s = \{k : k \in \mathcal{K}\}$, is the set of e^s RBs with the lowest priority according to $\succ_{PL}$, where $|\mathcal{E}^s| = e^s$;
5: **if** $\mathcal{E}^{s-1} \backslash \mathcal{E}^s \neq \emptyset$ **then**
6: Run the standard DAA on the RBs in $\mathcal{E}^{s-1} \backslash \mathcal{E}^s$ with maximum quotas for MVNOs equal to $q_m^{s,\max}$;
7: Remove the matched RBs from PL;
8: Add the matched RBs to μ;
9: **else**
10: Run the standard DAA on the RBs in $\mathcal{E}^s$ with maximum quotas for MVNOs equal to $q_m^{s,\min}$;
11: Remove the matched RBs from PL;
12: Add the matched RBs to μ;
13: **end if**
14: **for** *each MVNO* m **do**
15: $q_m^{s+1,\max} \leftarrow q_m^{s,\max} - |\mu^s(m)|$,
16: $q_m^{s+1,\min} \leftarrow \max\{0, q_m^{s,\min} - |\mu^s(m)|\}$;
17: **end for**
18: $s \leftarrow s + 1$
19: **end while**

B. RB Allocation to Users

1) MDP Formulation of the MVNO Slicing Level: For an MVNO $m \in \mathcal{M}$, the MDP formulation is given by the triplet $(\mathcal{S}_m, \mathcal{A}_m, \mathcal{R}_m)$, where $\mathcal{S}_m$ is the state space, $\mathcal{A}_m$ represents the action space, and $\mathcal{R}_m$ is the reward function of agent m.

The State Space: In the MNVO slicing level, the preallocated RBs $\mathcal{K}_m \subseteq \mathcal{K}$ to an MVNO $m \in \mathcal{M}$ is already known. The observation of an agent m, i.e., MVNO, includes the gains obtained between the gNodeB and the users of this MVNO. Thus, the state of agent $m \in \mathcal{M}$ at time slot t is defined as follows:

$$\mathcal{S}_m^t = \mathcal{G}_m^t \tag{21}$$

where $\mathcal{G}_m^t = (g_{u,m}^{k,t} : u \in \mathcal{U}_m, k \in \mathcal{K}_m)$ represents the downlink channel gain between the gNobeB and the users of the MVNO m over RBs $\mathcal{K}_m$ at time slot t. An observation $s_m^t \in \mathcal{S}_m^t$ at time slot t is given by the row vector $[g_{1,m}^{1,t}, \ldots, g_{1,m}^{|\mathcal{K}_m|,t}, \ldots, g_{|\mathcal{U}_m|,m}^{1,t}, \ldots, g_{|\mathcal{U}_m|,m}^{|\mathcal{K}_m|,t}]$. Based on (6) and (7), the dynamic of the MDP model is driven by the dynamic of the channel gain, which relies on user locations and shadow fading over the RBs.

The Action Space: An agent $m \in \mathcal{M}$ needs to decide which RBs from $\mathcal{K}_m$ should be allocated to each user in $\mathcal{U}_m$. Hence, the action space of agent m at time slot t in defined as follows:

$$\mathcal{A}_m^t = \{0, 1\}^{|\mathcal{K}_m| \times |\mathcal{U}_m|} \tag{22}$$

where an action $a_m^t \in \mathcal{A}_m^t$ is given by the row vector $[a_{1,m}^{1,t}, \ldots, a_{1,m}^{|\mathcal{K}_m|,t}, \ldots, a_{|\mathcal{U}_m|,m}^{1,t}, \ldots, a_{|\mathcal{U}_m|,m}^{|\mathcal{K}_m|,t}]$. Note that a_m^t is equivalent to the association matrix $[x_{u,m}^{k,t}]$, i.e., if $a_{u,m}^{k,t} = 1$ then agent m has decided to allocate RB $k \in \mathcal{K}_m$ to user $u \in \mathcal{U}_m$ at time slot t.

In the construction phase of an agent's action space, (15c) and (15d) should be respected. As a result, the size of the action space decreases significantly. This enables to accelerate the learning process of an agent while better exploring the action space.

The Reward Function: At the MVNO slicing level, the objective is to allocate RBs to satisfy the QoS requirements of both eMBB and URLLC users. Therefore, the reward received by an agent when it chooses an action is related to the QoS provided to each of its users in terms of data rate and delay. The reward function must also respect the constraints (15e) and (15f), namely, the minimum data rate $R_u^{\min}$ for each eMBB user u and the maximum delay $D_u^{\max}$ for each URLLC user u. If one of the constraints (15e) and (15f) is not satisfied for at least one user, we consider that the action chosen by an agent as an invalid action. Accordingly, the individual reward function related to the QoS provided to user $u \in \mathcal{U}_m$ at time t is defined as follows:

$$\mathcal{R}_{u,m}^t = \frac{r_{u,m}^t - c_{u,m}^t}{\max\limits_{u \in \mathcal{U}_m^e} \{r_{u,m}^t\}}, \text{ if } u \in \mathcal{U}_m^e \tag{23}$$

$$\mathcal{R}_{u,m}^t = \frac{1}{d_{u,m}^t + c_{u,m}^t}, \text{ if } u \in \mathcal{U}_m^u \tag{24}$$

where, $\mathcal{R}_{u,m}^t$ is normalized to help the agent learn faster.

The total reward received by agent $m \in \mathcal{M}$ at time slot t is defined as follows:

$$\mathcal{R}_m^t = \begin{cases} \sum_{u \in \mathcal{U}_m} \mathcal{R}_{u,m}^t / |\mathcal{U}_m|, & \text{if } a_m^t \text{ is valid} \\ -1, & \text{if } a_m^t \text{ is invalid.} \end{cases} \tag{25}$$

When a_m^t is invalid action, we penalize the agent who chooses it by assigning to it a negative value to prevent it from choosing such actions in the future. Note that the transition function of the MDP model depends on the dynamics of the wireless channels, i.e., an agent moves from one state to the next according to how the channel gains change from one time slot to the next.

2) Deep Reinforcement Learning Algorithm: According to the MDP formulation of the MVNO slicing level problem, the state space and action space are large. To handle such large spaces, a DRL algorithm can be leveraged. DRL is the combination of reinforcement learning and deep learning techniques [27]. It uses a deep neural network (DNN) to approximate an optimal strategy of the agent, where the states are given as inputs and the Q-value of all possible actions is generated as output. This DRL algorithm is known as the DQN algorithm. In DQN, we differentiate between the learning process and inference. In the former, a DNN is trained, in an offline manner, based on the data collected during the interaction with the environment. In the latter, the trained DNN is used to make online decisions.

In the training process, an agent $m \in \mathcal{M}$ uses a DNN to estimate the action-value $\mathcal{Q}(s_m^t, a_m^t; \theta_m)$ of a given state–action pair, where θ_m are the parameters of the DNN network. The objective is to train a DQN to find an optimal policy that maximizes the Q-function value. To do that, the agent uses experiences from the interaction with its environment. At time slot t, an experience is defined by the tuple $exp_m = (s_m^t, a_m^t, \mathcal{R}_m^{t+1}, s_m^{t+1})$, where s_m^t, a_m^t, $\mathcal{R}_m^{t+1}$, and s_m^{t+1} are the current state, the chosen action, the obtained reward, and the next state of the agent $m \in \mathcal{M}$, respectively. Instead of feeding the DNN with sequential experiences, DQN algorithm uses the experience-reply mechanism. In this mechanism, the agent stores its experiences in a replay memory $\mathcal{B}_m$. Then, it samples random batches from the stored experiences. The main reason behind using the experience-reply mechanism is to remove correlations between observations. The DQN of the agent $m \in \mathcal{M}$ is trained by minimizing the following loss function:

$$\mathcal{L}_m(\theta_m) = \mathbb{E}\left[\left(\mathcal{Q}_m^{DQN} - \mathcal{Q}(s_m^t, a_m^t; \theta_m)\right)^2\right] \tag{26}$$

with

$$\mathcal{Q}_m^{DQN} = \mathcal{R}_m^{t+1} + \gamma \max_{a_m} \left\{\mathcal{Q}\left(s_m^{t+1}, a_m; \theta_m'\right)\right\} \tag{27}$$

which denotes the target value of the Q-function, and $0 \leq \gamma \leq 1$ is called the discount factor. The Q-network and the target Q-network are the same except that the parameters of the target network are copied once in a while from the parameters of the original network, i.e., $\theta_m' = \theta_m$ every τ steps. The loss function is minimized using the stochastic gradient descent algorithm, which is applied to update the DNN parameters θ_m in each learning step.

In DQN, the same values are used by the max operator in (27) to select and evaluate an action. As a result, the performance of standard DQN may be low due to the overestimation of the Q-values. To deal with this issue and thus to decouple actions selection from actions evaluation, Van Hasselt et al. [28] introduced the double DQN (DDQN) algorithm. In DDQN, two different DQNs are used, namely, a Q-network $\mathcal{Q}$ and a target-network $\mathcal{Q}'$ with different parameters. The former is used to select an action and the latter is used to evaluate it. Indeed, DDQN avoids the overestimation problem by decomposing the max operation in the target value in the DQN loss function (27) into action selection and action evaluation as follows:

$$\mathcal{Q}_m^{DDQN} = \mathcal{R}_m^{t+1} + \gamma \mathcal{Q}\left(s_m^{t+1}, \arg\max_{a_m} \left\{\mathcal{Q}\left(s_m^{t+1}, a_m; \theta_m\right)\right\}; \theta_m'\right) \tag{28}$$

where, θ_m' are the parameters of the target Q-network $\mathcal{Q}(\cdot, \cdot; \theta_m')$. We can clearly see from (28) that the actions selection is performed using the Q-network $\mathcal{Q}(\cdot, \cdot; \theta_m)$ but the actions evaluation is performed using the target network $\mathcal{Q}(\cdot, \cdot; \theta_m')$. In (26), we replace $\mathcal{Q}_m^{DQN}$ with $\mathcal{Q}_m^{DDQN}$ and the DDQN algorithm aims to minimize the corresponding loss using stochastic gradient descent. The training process of the DDQN algorithm is presented in Algorithm 2.

Algorithm 2 DDQN Algorithm Training Process for Agent m

Initialize: θ, θ', $\mathcal{B}_m$;

1: **for** *each episode* **do**
2: Reset the environment of the MVNO $m \in \mathcal{M}$;
3: **for** *each step* **do**
4: Get observation $\mathcal{S}_m^t$ as in (21);
5: Choose an action a_m^t according the current policy (22);
6: Calculate reward $\mathcal{R}_m^t$ (25);
7: Obtain the next observation $\mathcal{S}_m^{t+1}$ according to channel dynamics;
8: Store the experience $exp_m = (s_m^t, a_m^t, \mathcal{R}_m^{t+1}, s_m^{t+1})$ in replay buffer $\mathcal{B}_m$;
9: **if** *batch size* **then**
10: Randomly sample a mini-batch from $\mathcal{B}_m$;
11: Calculate target Q-value according to (28);
12: Calculate the loss and update θ_m according to (26) replacing $\mathcal{Q}_m^{DQN}$ with $\mathcal{Q}_m^{DDQN}$;
13: **end if**
14: **if** *target step* **then**
15: Update the target parameters θ'_m;
16: **end if**
17: **end for**
18: **end for**

C. Two-Level RAN Slicing Mechanism

The MNO slicing level and the MVNO slicing level are performed in different time scales. The former is performed in a large time-scale, and the latter in a short time-scale. In the MVNO slicing level, each agent, i.e., each MVNO xApp, performs inference by allocating RBs to its users using the trained DDQN algorithm as shown in Fig. 2. Note that the training process of each MVNO's DNN is performed in a regional cloud using the data collected from the environment through the O1 interface of the O-RAN architecture. In the MNO slicing level, the rApp uses the MSDA algorithm, i.e., Algorithm 1, to assign RBs to MVNOs in a large time-scale, i.e., at each period of $\mathcal{T}$ time slots. To do that, Algorithm 1 requires the preference lists of all MVNOs Γ_m $\forall m \in \mathcal{M}$. The preference list of an MVNO $m \in \mathcal{M}$ is built using Algorithm 3. The construction phase of a preference list of an MVNO $m \in \mathcal{M}$ lasts $\mathcal{T}$ time slots. During this period, the allocated RBs $\mathcal{K}_m \subseteq \mathcal{K}$ to each MVNO $m \in \mathcal{M}$ is fixed. For a given MVNO $m \in \mathcal{M}$, Algorithm 3 takes as input the set of RBs $\mathcal{K}$ and outputs its preference list Γ_m. At the beginning, at $t = 0$, the weights of RBs $\mathcal{K}_m$ are all set to 0. For each time slot $t \geq 1$, the MVNO gets the data rate of each RB $k \in \mathcal{K}_m$ resulting from the inference phase of the DDQN algorithm. Then, the weight $w_{k,m}^t$ of each RB $k \in \mathcal{K}_m$ is calculated using (17). Note that if an RB is not allocated to a user during the period $\mathcal{T}$, its weight is set also to 0.

After $\mathcal{T}$ time slots, each MVNO sends to the rApp, through the A2 interface of the O-RAN architecture, its preference list and the calculated weights of its RBs. The rApp uses the received weights to build the PL list using (19). Then, it allocates RBs to the MVNOs using Algorithm 1. Once the RBs are allocated to the MVNOs, the DDQN algorithm is applied in a shorter time-scale, i.e., at each time slot to obtain the allocated RBs to the users.

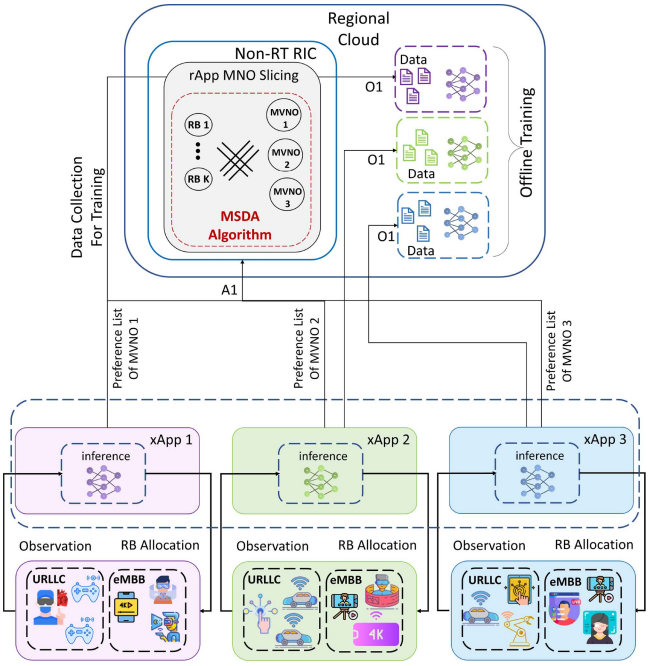

Fig. 2. Proposed 2LRS mechanism.

Algorithm 3 Preference List Construction Algorithm

Input: $\mathcal{K}$;
Output: Γ_m;
Initialize: $w_{k,m}^{t=0} = 0$ $\forall k \in \mathcal{K}$,

1: **for** *each* $t \leftarrow 1$ *to* $\mathcal{T}$ **do**
2: **for** *each* $k \in \mathcal{K}_m$ **do**
3: Get the data rate obtained by allocating k to a user $u \in \mathcal{U}_m$;
4: Update $w_{k,m}^t$ using (17);
5: **end for**
6: **end for**
7: Build the preference list Γ_m according to the calculated weights of RBs.

D. Complexity of 2LRS Mechanism

The MSDA algorithm iterates over multiple stages until the set PL of RBs is empty. In every stage the number of RBs is decreased by 1 in the worst case. That is, in the worst case, MSDA will require at most K stages to break the *while* loop. Most stages of MSDA will go into the *if* condition on line 5 in Algorithm 1, except for the last stage (if any). That is, the DAA algorithm will be executed $K-1$ times on $\mathcal{E}^{s-1} \backslash \mathcal{E}^s$ and once on $\mathcal{E}^s$. In the worst case, one MVNO will be assigned to one RB and thus DAA will requires $\mathcal{O}(KM)$ iterations for $K \gg M$. Since, the *while* loop requires at most K iterations, then MSDA will require $\mathcal{O}(K^2M)$ iterations in the worst case.

The time complexity of the training of the DDQN algorithm for agent m mainly depends on the number of iterations

TABLE I
RAN PARAMETERS

Parameter	Value
Total number of users	15
Number of MVNOs	3
Bandwidth of an RB	180 KHz
f_c	3 GHz
σ_{SF}	$\mathcal{N}(0, 4^2)$
Transmit power of gNodeB	30 dBm
Power of AWGN, σ^2	-174 dBm/Hz
Packet size for an eMBB & URLLC user	400 & 120 bits
Minimum data rate for eMBB user, $\mathcal{R}_{min}$	100 kbps
Maxim delay for URLLC user, $\mathcal{D}_{max}$	10 ms

TABLE II
DDQN HYPERPARAMETERS

Hyper-parameter	Value
Learning rate	0.001
Epsilon/ϵ-greedy	1
Discount factor	0.996
ϵ-min	0.01
Size of replay memory	100000
Size of mini-batch	64
Target network update interval	1000 steps
Loss function	Mean squared error
Optimizer	Adam
Activation function	ReLu

and the batch size [29], [30]. The DDQN iterates over E episodes and for each episode it iterates over T steps. In each step, DDQN computes the state, the action, the next state, and the reward. This computation requires $\mathcal{O}(KU)$ iterations in the worst case. The remaining computations are related to sampling the mini-batch of size B, and performing the gradient descent. Therefore, at the end, the DDQN of agent m requires at most $\mathcal{O}(ET(KU + BL))$, where L is the worst case computation complexity of backpropagation.

VII. SIMULATION RESULTS

A. Simulation Environment and Parameters

We consider a single-cell RAN, where the gNodeB is located at the center of a square of area (500×500) m^2. The gNodeB serves the users of three MVNOs. The users are randomly distributed inside the coverage area of the gNodeB. Each user is assumed to be either an eMBB user or a URLLC user and is subscribed to only one MVNO. For the sake of simplicity, the transmission power of the gNodeB is uniformly allocated to all users (equal power allocation is assumed). The channel gains between the users end the gNodeB are generated based on 3GPP specifications [19]. The parameters of the simulations are chosen based on the settings in [31]. Unless otherwise specified, the parameter values used in the simulations are summarized in Table I.

Each MVNO trains a DDQN model where the Q-network and the target network consist of two fully connected hidden layers, each with 256 neurons. Rectified linear unit (ReLU) is used as an activation function to avoid the vanishing gradient problem in the backpropagation, which accelerates the learning process [32]. The Adam optimizer is used with a learning rate of 0.001 since it is computationally efficient [33]. The main DDQN hyperparameters are summarized in Table II.

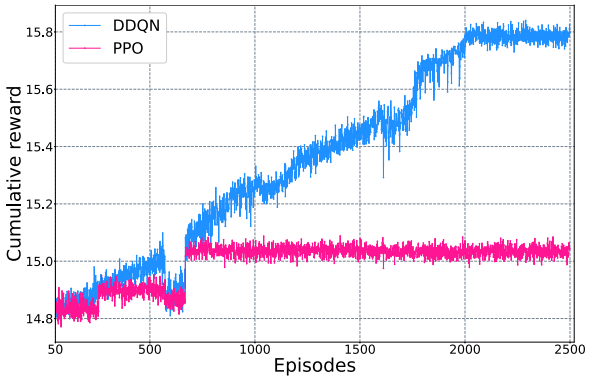

Fig. 3. Training performance.

To implement the DDQN algorithm, i.e., training phase and inference, we used PyTorch [34], [35].

The simulations are conducted on a laptop with a 2.2-GHz Intel Core i7-8750H processor, 16 GB of RAM and NVIDIA GeForce GTX 1070 graphic card.

B. Training Results

To evaluate the training performance of the DDQN algorithm, we compare its learning efficiency with an advanced DRL algorithm called proximal policy optimization (PPO). This choice enables a comparison between DDQN, which is an off-policy RL algorithm, and PPO, which belongs to the class of on-policy RL algorithms. In fact, the PPO algorithm uses the same policy to make decisions and evaluate actions, whereas DDQN uses different policies to select and evaluate actions.

Fig. 3 illustrates the average reward behavior of agents during training episodes. It can be observed that the learning performance of both algorithms is approximatively the same in the early stage of training, episode < 500. Then, the learning speed of PPO decreases and stops evolving after episode 700, while DDQN maintains a high speed and starts to converge at episode 2000. This indicates that the DDQN agent achieves a good learning performance after an acceptable number of training episodes. In both algorithms, the cumulative rewards present some small fluctuations. They are mainly due to the change in channel gains and the users' location, which are reset at each step and every 50 episodes, respectively.

C. 2LRS Performance Evaluation

To evaluate the proposed 2LRS-DDQN mechanism, we compared its performance against three benchmark mechanisms, which we called 2LRS-PPO, 2LRS-random, and SLRS for single-level RAN slicing.

2LRS-PPO and 2LRS-random mechanisms are similar to our 2LRS-DDQN mechanisms where the RAN slicing operation is performed in two levels. In the 2LRS-PPO mechanism, the MNO allocates RBs to MVNOs based on the proposed MSDA algorithm. Then, in the MVNO slicing level, each MVNO allocates RBs to its associated users based on PPO algorithm. In the 2LRS-random, the MNO slicing level is performed randomly and the MVNO slicing level is performed

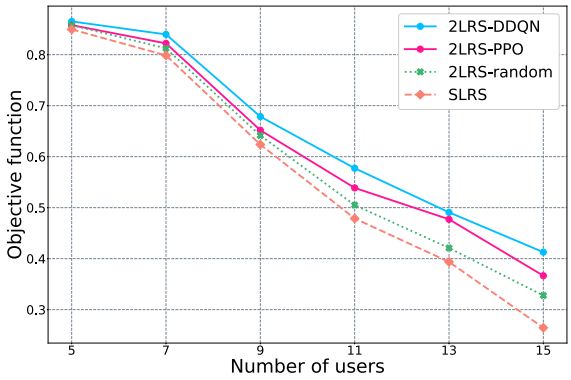

Fig. 4. Objective function (15a) versus the number of users.

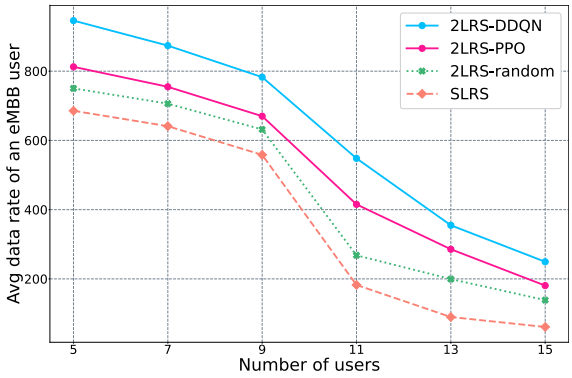

Fig. 5. Impact of the number of users on the average data rate of an eMBB user.

based on DDQN algorithm. Precisely, first, in the MNO slicing level, the allocation of RBs to MVNOs is performed in a random manner by respecting only the SLA between the MNO and the MVNOs in terms of the minimum number of RBs that each MVNO should be allocated. Second, in the MVNO slicing level, each MVNO allocates the RBs to its users based on the proposed DDQN algorithm.

In the SLRS [8] mechanism, the RAN slicing operation is performed in a single level. SLRS is a multicell RB sharing mechanism enabling MVNOs to meet the requirements of their users. This mechanism is based on a multiagent DQN algorithm. A DQN agent is associated with each MVNO that learns the appropriate policy to allocate the required RBs in each RAN cell. In each RAN cell, there is a maximum data rate threshold that can be provided to an MVNO to prevent excessive resource use by the MVNO. SLRS is an ideal benchmark for our 2LRS-DDQN mechanism since it investigates the RAN slicing challenge in a scenario similar to our case, which consists of an MNO providing radio resources to multiple MVNOs seeking to satisfy their users' requirements. Indeed, SLRS: 1) is a competitive mechanism that performs the RAN slicing operation in a multi-MVNO environment; 2) considers an SLA between the MNO and the MVNOs; and 3) is based on a DQN algorithm that allocates the RBs according to the requirements of the users.

To make the comparison fair, we adapt SLRS to meet the requirements of our system model as follows: 1) we consider only a single-cell RAN environment; 2) the total cell capacity and the maximum data rate that can be provided to an MVNO are adapted according to the number of RBs in the simulation parameters; 3) the data rates requested by the three MVNOs vary according to the patterns provided in [8]; and 4) to calculate the objective function (15a), the obtained data rate by an MVNO is uniformly distributed between its users.

In Fig. 4, we illustrate the performance of all mechanism in terms of user QoS and slicing reconfiguration cost, i.e., objective function (15a), when varying the number of users. From this figure, we make the following observations.

1) For all mechanisms, we observe that the objective function value decreases as the number of users increases. This is due to two main factors. When the number of users increases: a) the competitiveness among them increases to obtain sufficient number of RBs that meets their requirements in terms of data rate and delay and b) the slicing reconfiguration cost increases. Indeed, when the number of users is low, the three mechanisms behave generously with users by allocating several RBs to each of them. In contrast, when the number of users becomes large, each mechanism assigns to each user the minimum number of RBs needed to meet its QoS requirements in order to satisfy all users. In this case, users who had high QoS will experience a reduction in QoS to the required minimum level, which increases the slicing reconfiguration cost.
2) The performance of 2LRS-DDQN is better than that of all benchmark mechanisms in terms of objective function value (15a). Indeed, 2LRS-DDQN achieves a higher performance than 2LRS-PPO, as the PPO algorithm used in the MVNO slicing level can be trapped in a local optimum. This is due to the fact that on-policy algorithms can fail to find an adequate tradeoff between the exploration process and the exploitation process, since the same policy is used in both processes. 2LRS-DDQN and 2LRS-PPO are more efficient than 2LRS-random and SLRS because, in the 2LRS-DDQN and 2LRS-PPO mechanisms, each MVNO obtains its preferred RBs from the MNO using the MSDA algorithm. In fact, the RBs allocated to each MVNO are those that provide high data rates when assigned to its associated users. In the 2LRS-random mechanism, the RBs are allocated to the MVNOs randomly, respecting only the SLA between the MNO and the MVNOs in terms of the minimum number of RBs that each MVNO should have. As a result, the data rate achieved by an MVNO's users will be lower than the data rate provided by the RBs preferred by this MVNO. On the other hand, SLRS mechanism assigns RBs to an MVNO based on the total data rate requested by this MVNO instead of on the data rate requested by each user individually.

Figs. 5 and 6 depict the average data rate of an eMBB user and the average delay of a URLLC user as the number of users varies, respectively. To assess the average data rate achieved by an eMBB user, we generated a distribution of user QoS requirements such that there are more eMBB requirements than URLLC requirements. To do so, the probability

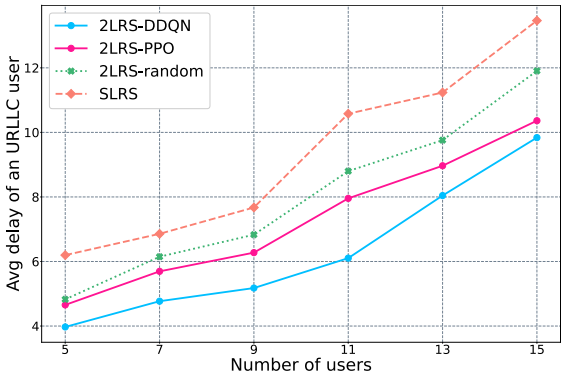

Fig. 6. Impact of the number of users on the average delay of a URLLC user.

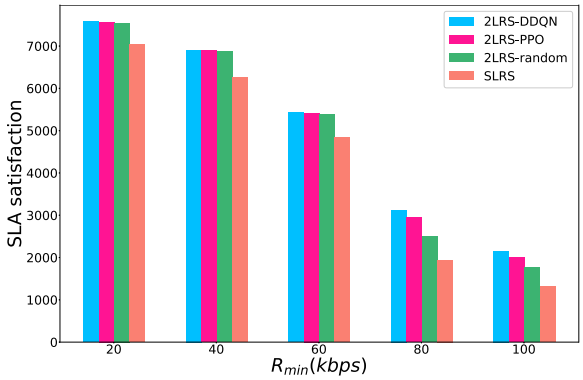

Fig. 7. Impact of the minimum data rate threshold ($\mathcal{R}_{\min}$) on the number of valid actions.

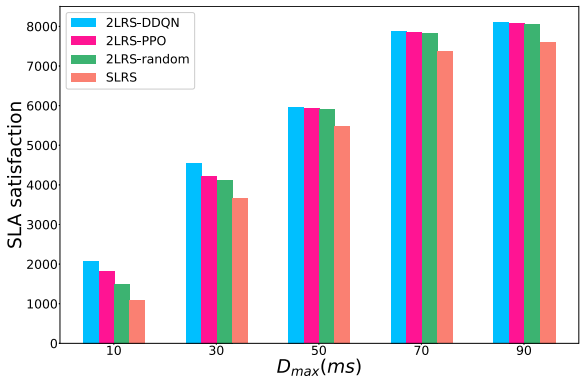

Fig. 8. Impact of the maximum delay threshold ($\mathcal{D}_{\max}$) on the number of valid actions.

distribution of eMBB users and URLLC users are 75% and 25% for each MVNO. Similarly, to evaluate the average transmission delay experienced by a URLLC user, the probability distribution of eMBB users and URLLC users are 25% and 75% for each MVNO.

From Figs. 5 and 6, we have the following observations.

1) For all mechanisms, the average data rate of an eMBB user decreases and the average delay of a URLLC user increases as the number of users increases since the number of RBs allocated to each user becomes smaller to satisfy all users.
2) 2LRS-DDQN and 2LRS-PPO always outperforms the 2LRS-random and SLRS mechanisms. This is due to the RBs allocated to each MVNO in the MNO slicing level, which are those that provide high data rates when assigned to its associated uses. On the other hand, when compared to the 2LRS-PPO mechanism, the 2LRS-DDQN mechanism has the highest average data rate for an eMBB user and the lowest average transmission delay for a URLLC user, because the 2LRS-PPO mechanism has reached a suboptimal performance level.
3) The minimum data rate threshold $\mathcal{R}_{\min} = 100\,\text{kb/s}$ and the maximum transmission delay $\mathcal{D}_{\max} = 10\,\text{ms}$ are always satisfied by 2LRS-DDQN, 2LRS-PPO, and SLRS, even when the number of users becomes large. This is due to the definition of the reward, which is related to the satisfaction of these two constraints. Indeed, in the training phase, the invalid actions are punished.

Figs. 7 and 8 show the cumulative number of times where the four mechanisms choose valid actions under different user's SLA requirements, i.e., minimum data rate and maximum transmission delay thresholds, respectively. Note that an action chosen by an agent is considered valid if it meets the user's SLA requirements, i.e., (15e) and (15f). From Figs. 7 and 8, we conclude the following.

1) For all mechanisms, we noticed that the number of valid actions becomes large under soft SLA constraints, i.e., a low minimum data rate threshold and a high transmission delay threshold, since it becomes easier to meet such SLA constrains.
2) 2LRS-DDQN, 2LRS-PPO, and 2LRS-random have almost the same performance under soft SLA constraints, i.e., $\mathcal{R}_{\min} \leq 60\,\text{kb/s}$ and $\mathcal{D}_{\max} \geq 50\,\text{ms}$. However, in the case of a more stringent SLA, the superiority of our 2LRS-DDQN mechanism in terms of performance becomes more significant than that of the other mechanisms, i.e., $\mathcal{R}_{\min} \geq 80\,\text{kb/s}$ and $\mathcal{D}_{\max} \leq 30\,\text{ms}$. The two-level mechanisms, i.e., 2LRS-DDQN, 2LRS-PPO, and 2LRS-random, largely outperform SLRS for all minimum data rate thresholds and maximum transmission delay thresholds. This can be explained by the fact that SLA constraints are considered in the reward function when model training is performed in the MVNO slicing level.

VIII. Conclusion and Future Work

In this article, we proposed a two-level RAN slicing mechanism to allocate RBs to eMBB and URLLC users. In the first level, MNO performs RAN slicing to allocate RBs to MVNOs in a large time-scale. In the second level, each MVNOs allocates the obtained RBs from the MNO to its associated users in a short time-scale. We formulated the two-level RB allocation problem as two mathematical optimization problems and proved that they are NP-hard. To solve the first level RB allocation problem, we proposed a matching game-based algorithm. Then, we modeled the RB allocation problem in the second level as a single MDP and developed a DRL algorithm

to solve it. We designed the proposed two-level RAN slicing mechanism in an O-RAN architecture. On the one hand, the RAN function of allocating RBs to MVNOs is performed by a rApps since it can be executed on a large time-scale. On the other hand, the RBs allocation function of each MVNO to its users is performed by an xApp as it has to be executed in a short time-scale. Simulation results have shown that our mechanism can achieve robust and efficient performance in meeting the QoS requirements of URLLC and eMBB users. Moreover, our mechanism outperforms benchmark algorithms.

Although the O-RAN architecture's native virtualization and embedded intelligence facilitate leveraging the proposed RAN slicing mechanism, there are still some challenges related to O-RAN slicing that may hamper the expected performance. In the following, we discuss some interesting future research directions that are worth investigating.

1) *Security and privacy* are the main repercussions of deploying O-RAN architecture. This is due to the openness feature of the O-RAN architecture, which advocates the use of open and interoperable interfaces to ensure communication between RAN functions. Indeed, these open interfaces can be responsible for the emergence of several threats and vulnerabilities. In addition, threats and vulnerabilities may appear in some O-RAN elements, such as rApp or xApp, as they may have been developed by untrusted sources. As a result, the security of RAN slices and the privacy of their data can be threatened. To protect RAN slices from the security and privacy threats related to O-RAN, it is necessary to design and implement effective security and privacy-preserving mechanisms.
2) *Interslice Isolation:* Isolation among RAN slices is a major challenge within the O-RAN architecture. In fact, O-RAN enables dynamic sharing of RAN resources among slices through the intelligence integrated into the RICs. The RICs automate the reconfiguration of RAN slices, i.e., scale down/up RAN slice resources according to the QoS requirements of each RAN slice. However, the dynamic reconfiguration of RAN slices experiencing significant traffic fluctuations should not compromise the isolation of the remaining RAN slices. Therefore, the tradeoff between efficient RAN resource sharing among slices and appropriate interslice isolation must be considered.
3) *Mobility Management:* User mobility management in a sliced RAN network presents complex challenges in an O-RAN architecture. In the RAN, resource allocation decisions are mainly based on spatiotemporal variations in users' traffic due to their mobility. Since resource management in O-RAN is intelligence-led, i.e., based on AI/ML techniques, RAN resource scheduling and allocation tasks are primarily driven by data collected on user mobility. Accordingly, the O-RAN should introduce features that facilitate the collection of user mobility data, which will be exploited by AI/ML algorithms to meet users' stringent QoS requirements. For instance, blockchain can serve as a platform of trust to establish a reliable relationship between multiple entities, which can secure and improve the RAN slice orchestration.

REFERENCES

[1] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong, "Network slicing: Recent advances, taxonomy, requirements, and open research challenges," *IEEE Access*, vol. 8, pp. 36009–36028, 2020.

[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] "O-RAN: Towards an open and smart RAN," O-RAN Alliance e.V., Alfter, Germany, White Paper, Oct. 2018.

[4] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," 2022, *arXiv:2202.01032*.

[5] O-RAN Working Group 2, *Non-RT RIC and A1 Interface WG: Non-RT RIC Architecture*, O-RAN Alliance e.V., Alfter, Germany, Oct. 2022.

[6] O-RAN Working Group 3, *Near-Real-Time RAN Intelligent Controller E2 Service Model (E2SM)*, O-RAN Alliance e.V., Alfter, Germany, Mar. 2022.

[7] L. Guijarro, J.-R. Vidal, and V. Pla, "Competition between service providers with strategic resource allocation: Application to network slicing," *IEEE Access*, vol. 9, pp. 76503–76517, 2021.

[8] I. Vilà, J. Pérez-Romero, O. Sallent, and A. Umbert, "A multi-agent reinforcement learning approach for capacity sharing in multi-tenant scenarios," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9450–9465, Sep. 2021.

[9] Z. Awada, K. Boulos, M. El-Helou, K. Khawam, and S. Lahoud, "Distributed multi-tenant RAN slicing in 5G networks," *Wireless Netw.*, vol. 28, no. 7, pp. 3185–3198, 2022.

[10] D. A. Ravi, V. K. Shah, C. Li, Y. T. Hou, and J. H. Reed, "RAN slicing in multi-MVNO environment under dynamic channel conditions," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4748–4757, Mar. 2022.

[11] S. O. Oladejo and O. E. Falowo, "Latency-aware dynamic resource allocation scheme for multi-tier 5G network: A network slicing-multitenancy scenario," *IEEE Access*, vol. 8, pp. 74834–74852, 2020.

[12] T. Ma, Y. Zhang, Z. Han, and C. Li, "Heterogeneous RAN slicing resource allocation using mathematical program with equilibrium constraints," *IET Commun.*, vol. 16, pp. 1772–1786, Sep. 2022.

[13] T. D. Tran and L. B. Le, "Resource allocation for multi-tenant network slicing: A multi-leader multi-follower Stackelberg game approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8886–8899, Aug. 2020.

[14] J. Hu, Z. Zheng, B. Di, and L. Song, "Multi-layer radio network slicing for heterogeneous communication systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2378–2391, Oct.–Dec. 2020.

[15] Z. Wang, Y. Wei, F. R. Yu, and Z. Han, "Utility optimization for resource allocation in edge network slicing using DRL," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[16] X. Chen et al., "Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2377–2392, Oct. 2019.

[17] "Study on new radio access technology: Radio access architecture and interfaces; (Release 14), Version 14.0.0," 3GPP, Sophia Antipolis, France, Rep. TR 38.801, Mar. 2017.

[18] "O-RAN use cases and deployment scenarios white paper," O.-R. Alliance, Alfter, Germany, White Paper, 2020.

[19] "5G Study on channel model for frequencies from 0.5 to 100 GHz; (Release 15), Version 15.0.0," 3GPP, Sophia Antipolis, France, Rep. TR 38.901, Jul. 2018.

[20] S. O. Krumke and C. Thielen, "The generalized assignment problem with minimum quantities," *Eur. J. Oper. Res.*, vol. 228, no. 1, pp. 46–55, 2013.

[21] Y. Meng, Z. Zhang, Y. Huang, and P. Zhang, "Resource allocation for energy harvesting-aided device-to-device communications: A matching game approach," *IEEE Access*, vol. 7, pp. 175594–175605, 2019.

[22] A. Pratap, "Cyclic stable matching inspired resource provisioning for IoT-enabled 5G networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12195–12205, Nov. 2022.

[23] S. F. Abedin, A. Mahmood, N. H. Tran, Z. Han, and M. Gidlund, "Elastic O-RAN slicing for industrial monitoring and control: A distributed matching game and deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 71, no. 10, pp. 10808–10822, Oct. 2022.

[24] F. Debbabi, R. L. Aguiar, R. Jmal, and L. Chaari Fourati, "Game theory for B5G upper-tier resource allocation using network slicing," *Wireless Netw.*, vol. 29, pp. 2047–2059, Feb. 2023.
[25] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Am. Math. Monthly*, vol. 69, no. 1, pp. 9–15, 1962.
[26] D. Fragiadakis, A. Iwasaki, P. Troyan, S. Ueda, and M. Yokoo, "Strategyproof matching with minimum quotas," *ACM Trans. Econom. Comput. (TEAC)*, vol. 4, no. 1, pp. 1–40, 2016.
[27] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[28] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
[29] J. Chen, C. Guo, R. Lin, and C. Feng, "Tasks-oriented joint resource allocation scheme for the Internet of Vehicles with sensing, communication and computing integration," *China Commun.*, vol. 20, no. 3, pp. 27–42, Mar. 2023.
[30] T. Liu, Y. Zhang, Y. Zhu, W. Tong, and Y. Yang, "Online computation offloading and resource scheduling in mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6649–6664, Apr. 2021.
[31] 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.
[32] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y.-C. Liang, "Network slice reconfiguration by exploiting deep reinforcement learning with large action space," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2197–2211, Dec. 2020.
[33] J. Mei, X. Wang, K. Zheng, G. Boudreau, A. B. Sediq, and H. Abou-Zeid, "Intelligent radio access network slicing for service provisioning in 6G: A hierarchical deep reinforcement learning approach," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6063–6078, Sep. 2021.
[34] A. Ta$\acute{}$ık, Z. Mlika, and S. Cherkaoui, "Clustered vehicular federated learning: Process and optimization," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 25371–25383, Dec. 2022.
[35] N. Ketkar and J. Moolayil, "Introduction to PyTorch," in *Deep Learning With Python*. Berkeley, CA, USA: Apress, 2021, pp. 27–91.

Abderrahime Filali (Member, IEEE) received the Ph.D. degree in electrical engineering from the Université de Sherbrooke, Sherbrooke, QC, Canada, in 2021.

He is a Postdoctoral Research Fellow with the Department of Computer and Software Engineering, Polytechnique Montreal, Montreal, QC, Canada. His current research interests include resource management in next-generation networks.

Dr. Filali served as a reviewer for several international conferences and journals.

Zoubeir Mlika (Member, IEEE) received the M.Sc. and Ph.D. degrees from the University of Quebec at Montreal, Montreal, QC, Canada, in 2014 and 2019, respectively.

He is a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Sherbrooke, Sherbrooke, QC, Canada. His current research interests include vehicular communications, resource allocation, design and analysis of algorithms, and computational complexity.

Soumaya Cherkaoui (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the Université de Sherbrooke, Sherbrooke, QC, Canada.

She is a Full Professor with the Department of Computer and Software Engineering, Polytechnique Montreal, Montreal, QC, Canada. She authored numerous conference and journal papers.

Prof. Cherkaoui was awarded with recognitions for her work including several best paper awards at IEEE conferences. She is an IEEE ComSoc Distinguished Lecturer and a Professional Engineer in Canada, and has served as the Chair of the IEEE ComSoc IoT-Ad Hoc and Sensor Networks Technical Committee. She has been on the editorial board of several IEEE journals.