IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 10, 2021, accepted January 4, 2022, date of publication January 11, 2022, date of current version January 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141789

Deep Reinforcement Learning-Based Network Slicing for Beyond 5G

KYUNGJOO SUH, SUNWOO KIM, YONGJUN AHN, SEUNGNYUN KIM, (Graduate Student Member, IEEE), HYUNGYU JU, (Graduate Student Member, IEEE), AND BYONGHYO SHIM, (Senior Member, IEEE)

Department of Electrical and Computer Engineering, Seoul National University, Gwanak-gu, Seoul 08826, South Korea

Corresponding author: Byonghyo Shim (bshim@islab.snu.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government, Ministry of Science and ICT (MSIT), South Korea, under Grant 2020R1A2C2102198; and in part by the MSIT, through the Information Technology Research Center (ITRC) Support Program, supervised by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under Grant IITP-2021-0-02048.

ABSTRACT With the advent of 5G era, network slicing has received a great deal of attention as a means to support a variety of wireless services in a flexible manner. Network slicing is a technique to divide a single physical resource network into multiple slices supporting independent services. In beyond 5G (B5G) systems, the main goal of network slicing is to assign the physical resource blocks (RBs) such that the quality of service (QoS) requirements of eMBB, URLLC, and mMTC services are satisfied. Since the goal of each service category is dearly distinct and the computational burden caused by the increased number of time slots is huge, it is in general very difficult to assign RB properly. In this paper, we propose a deep reinforcement learning (DRL)-based network slicing technique to find out the resource allocation policy maximizing the long-term throughput while satisfying the QoS requirements in the B5G systems. Key ingredient of the proposed technique is to reduce the action space by eliminating undesirable actions that cannot satisfy the QoS requirements. Numerical results demonstrate that the proposed technique is effective in maximizing the long-term throughput and handling the coexistence of use cases in the B5G environments.

INDEX TERMS Network slicing, resource allocation, deep reinforcement learning.

I. INTRODUCTION

5G network is envisioned to be a multi-service network supporting a wide array of services such as enhanced mobile broadband (eMBB), ultra reliable and low latency communications (URLLC), and massive machine-type communications (mMTC) [1], [2]. As a means to accommodate a variety of services in a flexible manner, network slicing, a concept to divide a single physical network into multiple (logically) isolated networks, has received a great deal of attention in recent years [3]. Since a wireless resource block (RB) is divided into multiple slices supporting independent services, a malfunction of a certain slice will not affect other services, thereby ensuring the stability of the entire system. Network slicing can also reduce the maintenance and operation costs significantly since diverse services can be supported by the common physical infrastructure [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Mingjun Dai.

In beyond 5G (B5G) and 6G systems, the main goal of network slicing is to assign the physical resource blocks such that the quality of service (QoS) requirements of eMBB, URLLC, and mMTC services are satisfied simultaneously. To do so, coexistence of eMBB, URLLC, and mMTC needs to be handled properly by the base station (BS) and 5G core network including access and mobility management function (AMF), session management function (SMF), and user plane function (UPF). However, it is in general very difficult to satisfy these diverse service requirements simultaneously since the goal of each service category is dearly distinct. For example, to meet the latency requirement of the URLLC service, the BS should transmit the URLLC packet immediately even in the middle of eMBB/mMTC transmission. In such case, obviously, reception quality of eMBB and mMTC services will be degraded severely due to the abrupt increase in interference.

In order to maximize the system throughput while satisfying the QoS requirements, various network slicing techniques

have been suggested in recent years. In [5], a technique to allocate the equal number of RB to each network slice has been proposed. In [4], a regression tree-based technique to assign different bandwidth to each network slice based on service requirement has been suggested. While these approaches are useful to handle the network slicing in a given time slot, efficacy of this approach would be severely degraded in the dynamically changing wireless environments. This is mainly because to determine the allocation of a RB for a certain service is basically binary integer program [6] so that one has to deal with the exponential increase in computational complexity. For example, when one consider 20 network slices, then we need to check $2^{20} \approx 10^6$ possible resource allocation decisions in each time slot. For this reason, to come up with a network slicing technique that can effectively control the allocation of slices over the long-term period is of great importance for the success of network slicing in B5G and 6G cellular systems.

An aim of this paper is to propose a novel network slicing technique, referred to as DRL-based network slicing (DRL-NS), to improve the system throughput while supporting various services (e.g., eMBB, URLLC, mMTC). In our study, we employ the deep reinforcement learning (DRL), an efficient learning-based technique specialized for solving the sequential decision-making problem as a baseline. In DRL, an agent finds out a series of actions maximizing a long-term cumulative reward among large-scale state-action pairs [7]. In our work, we use DRL to let the gNB (**agent**) observe the channel state, data rate constraints, delay requirements (**states**), and then assign each resource block (RB) to a certain service (**action**) to maximize the overall system throughput (**reward**).

In the network slicing problem, an action space size (i.e., a possible combination of actions) is huge since it is proportional to the number of sequential network slicing decisions. To be specific, when deciding whether the RB is allocated or not for a certain service, the number of possible choices increases exponentially with the number of time slots. Due to this immense action space, gNB is likely to explore undesirable actions (e.g., resource allocation decisions that cannot satisfy the data rate constraints or delay requirements) during the training phase, thereby slowing down the convergence speed and also preventing the DRL from maximizing the reward. To address the problem, we employ an action elimination [8], a technique to exclude undesirable actions among all possible actions to boost up the training speed and quality of trained policy. Due to the elimination of ineffective and meaningless actions, after playing reasonable number of training episodes, the trained gNB generates the network slicing strategies (i.e., resource allocation decisions) improving the system throughput and satisfying the QoS constraints.

The main contributions of this paper are as follows:

- We propose a DRL-based network slicing (DRL-NS) technique that finds out the resource allocation policy maximizing the long-term throughput while satisfying the QoS requirements of eMBB, URLLC, and mMTC services simultaneously in dynamically varying 5G environments.
- We integrate the action elimination to DRL to eliminate undesirable actions that cannot satisfy the QoS requirements. In doing so, exploration of DRL agent is directed toward the desirable actions, improving the chance of making the optimal resource allocation decision and the convergence speed of the DRL training.
- We provide empirical simulation results from which we demonstrate the superiority of DRL-NS over the conventional approaches. For example, DRL-NS achieves about 25% and 15% improvements in the throughput performance over the equal allocation and regression-tree based network slicing techniques, respectively. Even when compared to the vanilla DQN-based network slicing technique, DRL-NS achieves around 10% improvement in the throughput performance.

The rest of this paper is organized as follows. In Section II, we present the related works of network slicing. In Section III, we discuss the system model and explain the network slicing problem. In Section IV, we provide a detailed description of the proposed DRL-NS technique. In Section V, we present the simulation results to verify the performance gain of the proposed technique and conclude the paper in Section VI. Since terminologies and major notations might be unfamiliar to the reader, we summarize the technical terms in Table 1.

TABLE 1. Summary of nomenclature and notations.

Notation	Description
BS	Base station
RB	Resource Block
eMBB	Enhanced mobile broadband
URLLC	Ultra reliable and low latency communications
mTMC	Massive machine-type communications
QoS	Quality of service
RL	Reinforcement learning
DL	Deep learning
DRL	Deep reinforcement learning
DNN	Deep neural network
DQN	Deep Q network
RNN	Recurrent neural network
LSTM	Long short-term memory
UE	User Equipment
RAN	Radio access network
CN	Core network
$(\cdot)^T$	Transpose operator
A	Action space
A^D	Actions that pass the URLLC test
A^F	Actions that pass the URLLC and eMBB tests
x_i	i-th element of a vector $\mathbf{x}$
$x_{i,j}$	(i,j)-th element of a matrix $\mathbf{X}$

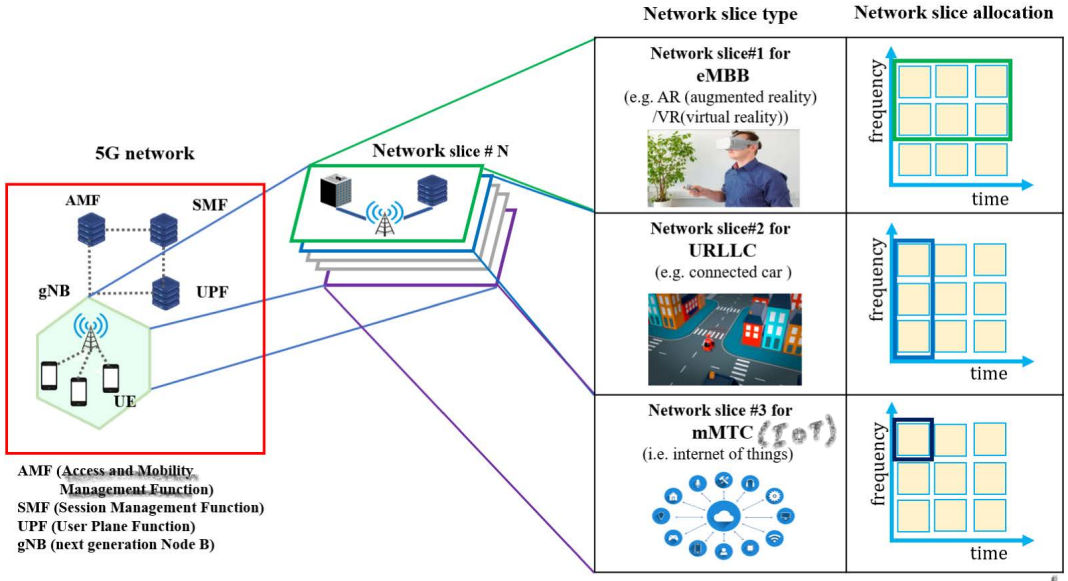

FIGURE 1. Illustration of network slicing in 5G network.

II. RELATED WORKS

In this section, we provide a brief review on the state-of-the-art network slicing techniques. Over the years, various efforts have been made to provide network slicing satisfying the B5G service requirements. For example, in [12], a proportional fair-based network slicing technique pursuing a balance between the throughput and the QoS of user has been proposed.

One popularly used approach to optimally serve multiple network slices over the common physical network is DL, a data-driven learning approach. Due to its ability to provide fast and accurate prediction and decision making, DL has shown great promise in many practical applications [6], [9]–[11]. In fact, since DL is effective in extracting policy from environments, it can be readily used for the decision making problem such as the resource management and scheduling. DL has been also applied in many network slicing problems to come up with a well-informed slicing decision using available physical resources. For instance, in [13], a DL-based technique that predicts the network load on each network slice and then allocates slices based on incoming traffic has been proposed. In [14] and [15], RNN and LSTM-based network slicing techniques that analyze the overall traffic pattern from the sequential traffic data and then allocate slices based on the traffic prediction have been proposed.

Recently, various DRL-based network slicing techniques have been proposed to perform the robust network slicing in the dynamically changing 5G environments [16], [17]. For example, in [17], a DRL-based network slicing technique that controls the large-scale resource allocation has been proposed. To avoid the exploration of undesirable actions that fail to satisfy the QoS requirements, authors in [17] suggested dueling DQN, a DRL technique that identifies the desirable action by explicitly dividing the Q-value function into the state-value and state-dependent action advantage functions.

Our approach is dearly distinct from previous studies in the sense that we integrate the action elimination mechanism to DRL to reduce the action space size. While an agent in the conventional DRL-based network slicing techniques is likely to explore meaningless actions (e.g., actions violating the QoS requirements) due to the immense action space, we avoid exploration of such actions by choosing only the meaningful actions. To be specific, using the action elimination, we identify undesirable actions (set of resource allocation decisions violating rate and delay requirements) and then exclude them from the action space. In doing so, exploration of DRL agent is directed toward desirable actions, improving the chance of making the optimal resource allocation decision maximizing the system throughput and ensuring the QoS requirement satisfaction.

III. SYSTEM MODEL AND NETWORK SLICING PROBLEM

In this section, we explain a network slice model in downlink transmission scenario and specify three types of slices: eMBB, URLLC, mMTC. Also, we formulate the network slicing as a constrained optimization problem.

A. SYSTEM MODEL

In our work, we consider a downlink transmission scenario where M BSs serve K UEs randomly located in the coverage area of the BSs.[1] We denote the set of UEs as $\mathcal{K} = \{1, \cdots, K\}$. The BSs (a.k.a radio unit or gNBs) are connected to a digital unit (DU) to share the channel state information (CSI) between the BSs and UEs (see Fig. 1). The physical

[1] In general, the QoS requirements of downlink communications are far stricter than that of uplink communications since the majority of downlink communications are data transmission while the large part of uplink communications are dedicated to the control signal transmission. For this reason, we put a major emphasis on the network slicing on the downlink.

network is divided into N slices where each RB can be assigned to eMBB, URLLC, or mMTC network slice. The sets of eMBB, URLLC, and mMTC slices are denoted as $\mathcal{I}$, $\mathcal{J}$, and $\mathcal{L}$. The total number of eMBB, URLLC, or mMTC network slices are I, J, and L, respectively ($I + J + L = N$). In order to indicate the allocation of the eMBB, URLLC, and mMTC slices in the resource block grid, we use three binary vectors $\boldsymbol{\alpha}_e \in \mathbb{R}^I$, $\boldsymbol{\alpha}_u \in \mathbb{R}^J$, and $\boldsymbol{\alpha}_m \in \mathbb{R}^L$ where

$$\alpha_e(i) = \begin{cases} 1 & \text{if RB is allocated to the } i\text{-th eMBB slice} \\ 0 & \text{otherwise.} \end{cases}$$

$$\alpha_u(j) = \begin{cases} 1 & \text{if RB is allocated to the } j\text{-th URLLC slice} \\ 0 & \text{otherwise.} \end{cases}$$

$$\alpha_m(l) = \begin{cases} 1 & \text{if RB is allocated to the } l\text{-th mMTC slice} \\ 0 & \text{otherwise.} \end{cases}$$

As a channel model, we consider the Rayleigh fading channel model, one of the most widely used channel model in the wireless communication. We note that the proposed DRL-based network slicing technique is data-driven learning algorithm exploiting the explicit channel coefficient information and thus its performance might not be affected much by the channel model variation. Specifically, the downlink channel coefficient $h_{m,k} \in \mathbb{C}$ between the BS m and the UE k is expressed as

$$h_{m,k} = \sqrt{\beta_{m,k}} g_{m,k}, \tag{1}$$

where $\beta_{m,k}$ is the large-scale fading coefficient and $g_{m,k} \sim \mathcal{CN}(0, 1)$ is the small-scale fading coefficient. In this setup, the data rate R_k of UE k is given by

$$R_k = \log_2 \left(1 + \frac{|\sum_{m=1}^{M} h_{m,k}^* w_{m,k}|^2}{\sigma_k^2} \right), \quad \forall k \in \mathcal{K}, \tag{2}$$

where $w_{m,k}$ is the downlink precoding coefficient from the BS m to the UE k and σ_k^2 is the noise power. In our work, we consider the OFDM systems to avoid the interference caused by the adjacent sub-channels. We also assume that the BSs employ different frequency bands to minimize the inter-cell interference.

B. NETWORK SLICING PROBLEM

Main goal of the network slicing is to maximize the overall system throughput while fulfilling the QoS requirements of various network slices. To achieve the goal, we need to consider three major components in the system throughput: 1) throughput of eMBB slices (T_{eMBB}), 2) throughput of URLLC slices (T_{URLLC}), and 3) throughput of mMTC slices (T_{mMTC}).

1) THROUGHPUT OF eMBB SLICE

When a UE sends a request for the eMBB slice to the mobile network operator, the corresponding throughput $T_{\text{eMBB},k}$ of the UE k is expressed as

$$T_{\text{eMBB},k} = \sum_{i=1}^{I} \alpha_e(i) f_{i,k} R_k, \quad \forall k \in \mathcal{K}, \tag{3}$$

where $f_{i,k}$ is the resource bandwidth allocated to the UE k in the i-th eMBB slice. Note that $T_{\text{eMBB},k}$ should be larger than the rate requirement of UE:

$$T_{\text{eMBB},k} \geq T_{k,\min}, \quad \forall k \in \mathcal{K}. \tag{4}$$

The corresponding sum throughput of eMBB network slice is $T_{\text{eMBB}} = \sum_{k=1}^{K} T_{\text{eMBB},k}$.

2) THROUGHPUT OF URLLC SLICE

The throughput of URLLC slice of UE k is

$$T_{\text{URLLC},k} = \sum_{j=1}^{J} \alpha_u(j) f_{j,k} R_k, \quad \forall k \in \mathcal{K}, \tag{5}$$

where $f_{j,k}$ is the resource bandwidth allocated to the UE k in the j-th URLLC slice. In our work, we assume that one data packet should be completely transmitted within one frame in URLLC. That is, the frame duration should be less than or equal to the maximum packet delay D as [18]

$$\frac{F_{j,k}}{T_{\text{URLLC},k}} \leq D_{j,k,\max}, \quad \forall k \in \mathcal{K}, \quad \forall j \in \mathcal{J}. \tag{6}$$

where $F_{j,k}$ is the packet length to UE k in j-th URLLC network slice and $D_{j,k,\max}$ is the maximum packet delay of UE k in j-th URLLC network slice. The corresponding sum throughput of URLLC slice is $T_{\text{URLLC}} = \sum_{k=1}^{K} T_{\text{URLLC},k}$.

3) THROUGHPUT OF mMTC SLICE

Similar to the eMBB and URLLC slices, the throughput of mMTC slice of UE k is given by

$$T_{\text{mMTC},k} = \sum_{l=1}^{L} \alpha_m(l) f_{l,k} R_k, \quad \forall k \in \mathcal{K}, \tag{7}$$

where $f_{l,k}$ is the resource bandwidth allocated to the UE k in the l-th mMTC slice. Note, in contrast to eMBB and URLLC slices, mMTC slice has no rate/latency requirement. The corresponding sum throughput of mMTC slice is $T_{\text{mMTC}} = \sum_{k=1}^{K} T_{\text{mMTC},k}$.

In summary, the total network throughput $T_{\text{total}}^{(t)}$ at the time slot t is given by:

$$T_{\text{total}}^{(t)} = T_{\text{eMBB}}^{(t)} + T_{\text{URLLC}}^{(t)} + T_{\text{mMTC}}^{(t)}. \tag{8}$$

The problem to maximize the overall system throughput over T time slots is formulated as

$$\mathcal{P}: \max_{\{\boldsymbol{\alpha}_e^{(t)}, \boldsymbol{\alpha}_u^{(t)}, \boldsymbol{\alpha}_m^{(t)}\}} \sum_{t=1}^{T} T_{\text{total}}^{(t)} \tag{9a}$$

$$\text{s.t.} \quad T_{\text{eMBB},k}^{(t)} \geq R_{k,\min}^{(t)}, \quad \forall k \in \mathcal{K}, \quad \forall t \in \mathcal{T}, \tag{9b}$$

$$\frac{F_{j,k}}{T^{(t)}_{\mathrm{URLLC},k}} \le D^{(t)}_{j,k,\max}, \quad \forall k \in \mathcal{K}, \quad \forall t \in \mathcal{T}, \quad \forall j \in \mathcal{J} \tag{9c}$$

$$\sum_{i=1}^{I} \alpha_e^{(t)}(i) + \sum_{j=1}^{J} \alpha_u^{(t)}(j) + \sum_{l=1}^{L} \alpha_m^{(t)}(l) \le \tau N. \tag{9d}$$

where $\alpha_e^{(t)}(i)$, $\alpha_u^{(t)}(j)$, and $\alpha_m^{(t)}(l)$ are the elements of binary vectors $\boldsymbol{\alpha}_e^{(t)}$, $\boldsymbol{\alpha}_u^{(t)}$, and $\boldsymbol{\alpha}_m^{(t)}$, respectively. Also, $\mathcal{T} = \{1, \cdots, T\}$, and $0 \le \tau \le 1$ is the ratio of utilized network slices. Note that the remaining $N - \tau N$ slices are reserved for backup.

By plugging (3), (5), and (7), P can be re-expressed as

$$\mathcal{P}': \max_{\{\boldsymbol{\alpha}_e^{(t)}, \boldsymbol{\alpha}_u^{(t)}, \boldsymbol{\alpha}_m^{(t)}\}} \sum_{t=1}^{T} \sum_{k=1}^{K} \left(\sum_{i=1}^{I} \alpha_e^{(t)}(i) f_{i,k} R_k^{(t)} + \sum_{j=1}^{J} \alpha_u^{(t)}(j) f_{j,k} R_k^{(t)} + \sum_{l=1}^{L} \alpha_m^{(t)}(l) f_{l,k} R_k^{(t)} \right) \tag{10a}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \alpha_e^{(t)}(i) f_{i,k} R_k^{(t)} \ge R^{(t)}_{k,\min}, \quad \forall k \in \mathcal{K}, \quad \forall t \in \mathcal{T}, \tag{10b}$$

$$\frac{F_{j,k}}{\sum_{j=1}^{J} \alpha_u^{(t)}(j) f_{j,k} R_k^{(t)}} \le D^{(t)}_{j,k.\max}, \quad \forall k \in \mathcal{K}, \quad \forall t \in \mathcal{T}, \quad \forall j \in \mathcal{J}, \tag{10c}$$

$$\sum_{i=1}^{I} \alpha_e^{(t)}(i) + \sum_{j=1}^{J} \alpha_u^{(t)}(j) + \sum_{l=1}^{L} \alpha_m^{(t)}(l) \le \tau N. \tag{10d}$$

Since $\alpha_e^{(t)}(i)$, $\alpha_u^{(t)}(j)$, and $\alpha_m^{(t)}(l)$ are binary integers, $\mathcal{P}'$ is a mixed-integer programming, which is known as a non-convex NP-hard problem [6]. When we try to solve $\mathcal{P}'$ using the conventional analytic approach (e.g., combinatoric search algorithm), computational burden would be unacceptably high. For instance, if the number of resource blocks and time slots are 10, respectively, then one should search over $2^{10\times 10} \approx 10^{30}$ decision choices to find out the optimal resource allocation decision. To make things worse, this kind of analytic approach has a causality issue since the future-oriented resource allocation decisions require the channel information of future time slots.

IV. DRL-NS

The primary goal of DRL-NS is to learn the proper network slicing strategy maximizing the system throughput. To achieve this goal, the proposed scheme exploits DRL framework in the resource allocation decision. DRL is a DL technique that finds out the optimal policy for the sequential decision making through the interaction with the environment. Specifically, based on the input information (e.g., CSI, the required UE data rates), DNN in the DRL agent (i.e., deep Q-network (DQN)) learns the complicated relationship between the resource allocation decision and the long-term system throughput.

Since the DRL agent learns the policy through trials and errors, performance of DRL depends on the exploration process of action space. In our case, due to the immense action space (e.g., $2^{10} \approx 1000$ resource allocation decisions when we consider 10 network slices), DRL agent needs to explore too many undesirable actions (e.g., resource allocation decision that cannot satisfy the UEs' requirements of eMBB and URLLC slices). This can severely diminish the sample efficiency due to the lack of useful training data, and thus the trained policy might not be optimal in most cases. In fact, in our test experiments, we observe that more than 80% of allocation decisions made by the trained DRL could not satisfy the requirements of eMBB and URLLC slices.

To overcome this problem, we exploit the action elimination mechanism that identifies the resource allocation decisions violating the QoS requirements (i.e., data rate and delay requirements) and then eliminates them from the action space. In doing so, we can reduce the action space and direct the exploration of DRL agent toward the desirable actions, improving the chance of obtaining the optimal resource allocation strategy. Two key ingredients in the proposed action elimination method are 1) URLLC test to check whether each resource allocation decision can meet the delay requirements of URLLC and 2) eMBB test to check whether each resource allocation decision that pass URLLC test can satisfy the data rate requirements of eMBB. By choosing allocation decisions that pass both tests, we can dramatically reduce the action space and also improve the training speed.

In the following subsections, we briefly review the basics of RL and then discuss the state space, action space, and reward function in DRL-NS as well as eMBB and URLLC tests reducing the action space. Finally, we illustrate the training process of DRL-NS and analyze its computational complexity.

A. BASICS OF DEEP REINFORCEMENT LEARNING

In this subsection, we briefly introduce the basics of DRL. Reinforcement learning (RL) is a goal-oriented algorithm that learns how to solve a task using trials and errors. The key ingredients of RL are agent, environment, state, action, and reward [6]. Mission of an agent is to learn the optimal policy through interactions with the environment. In the learning process, an agent observes the current state s_t, takes an action a_t, and then the environment returns the next state s_{t+1} and the immediate reward r_t to the agent as a feedback. The optimal policy π^* maximizing the expectation of cumulative reward is [19],

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | \pi\right] \tag{11}$$

where γ is a discount factor ($0 < \gamma < 1$) to provide less weight to the future reward.

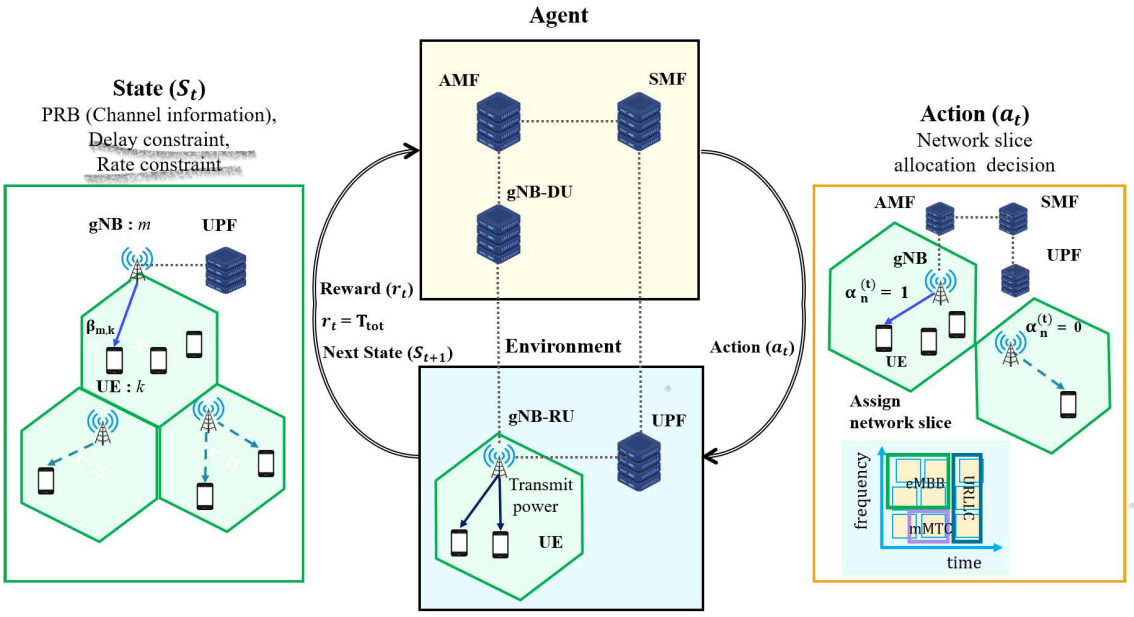

FIGURE 2. Basic structure of DRL-based throughput-maximizing network slicing.

In order to find out π^*, the action-value function $Q^\pi(s, a)$ that represents the expected cumulative reward obtained when carrying out the policy π, is exploited:

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right]. \quad (12)$$

Since $Q^\pi(s, a)$ indicates the expected cumulative reward for taking action a in state s, the optimal policy can be obtained by selecting the action maximizing $Q^\pi(s, a)$. To do so, the action-value function should be available for all possible state-action pairs. To find out the optimal Q-function $Q^*(s, a)$, Bellman equation for $Q^*(s, a)$ is used [19]:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in S} P^a_{ss'} \max_{a' \in A} Q^*(s', a'), \quad (13)$$

where $r(s, a)$ is the reward corresponding to the state-action pair (s, a) and $P^a_{ss'}$ is the transition probability.

To reduce the burden of computing and comparing the Q-value for every state and action, deep Q-network (DQN), a DNN-based function approximator to estimate Q-function (i.e., $Q^*(s, a) \approx Q(s, a, w)$), has been popularly used [6]. Basically, the weight w of DQN is updated to minimize the loss function given by $L(w) = (Y_t^{dqn} - Q(s, a, w))^2$ where $Y_t^{dqn} = r(s, a) + \gamma \max_{a' \in A} Q(s', a', w)$.

B. THE DRL-BASED RESOURCE ALLOCATION MODEL

In this subsection, we discuss the state space, action space, and reward function of the proposed scheme. In our problem setup, a whole network consisting of M BSs and K UEs is considered as an environment and the gNB is serving as an agent (see Fig. 2). Under this setting, we can define the state space, action space, and reward function.

1) STATE SPACE

State contains essential information in the environment used for the policy learning. In the proposed DRL framework, the state of the environment observed by the agent consists of several parts: the minimum rate constraints of UEs $\mathbf{R}_{\min}^{(t)} = [R_{1,\min}^{(t)}, \cdots, R_{K,\min}^{(t)}]^T$ at the time slot t, the slice allocation decision at the previous time slot $\boldsymbol{\alpha}^{(t-1)} = [\alpha_1^{(t-1)}, \cdots, \alpha_N^{(t-1)}]^T$, and the channel matrix $\mathbf{H}^{(t)}$ representing the path loss and the shadowing effect between BSs and UEs at the time slot t. To be specific, $\mathbf{H}^{(t)}$ is expressed as $M \times K$ matrix as

$$\mathbf{H}^{(t)} = \begin{bmatrix} h_{1,1}^{(t)} & \cdots & h_{1,K}^{(t)} \\ \vdots & \ddots & \vdots \\ h_{M,1}^{(t)} & \cdots & h_{M,K}^{(t)} \end{bmatrix}. \quad (14)$$

Also, the maximum delay constraints of UE at the time slot t is expressed as $J \times K$ matrix as

$$\mathbf{D}_{\max}^{(t)} = \begin{bmatrix} D_{1,1,\max}^{(t)} & \cdots & D_{1,K,\max}^{(t)} \\ \vdots & \ddots & \vdots \\ D_{J,1,\max}^{(t)} & \cdots & D_{J,K,\max}^{(t)} \end{bmatrix}. \quad (15)$$

In summary, the state can be expressed as

$$s_t = [\mathbf{H}^{(t)}, \mathbf{H}^{(t-1)}, \mathbf{R}_{\min}^{(t)}, \mathbf{D}_{\max}^{(t)}, \boldsymbol{\alpha}^{(t-1)}]^T. \quad (16)$$

Note that both $\mathbf{H}^{(t)}$ and $\mathbf{H}^{(t-1)}$ are included in s_t so that the DNN in DRL agent can extract the temporally correlated features of the channel information. By exploiting the extracted features among $\mathbf{H}^{(t)}$, $\mathbf{R}^{(t)}$, $\mathbf{D}^{(t)}$, and $\boldsymbol{\alpha}^{(t-1)}$, the DRL agent learns the resource allocation policy maximizing the system throughput while satisfying the QoS requirements.

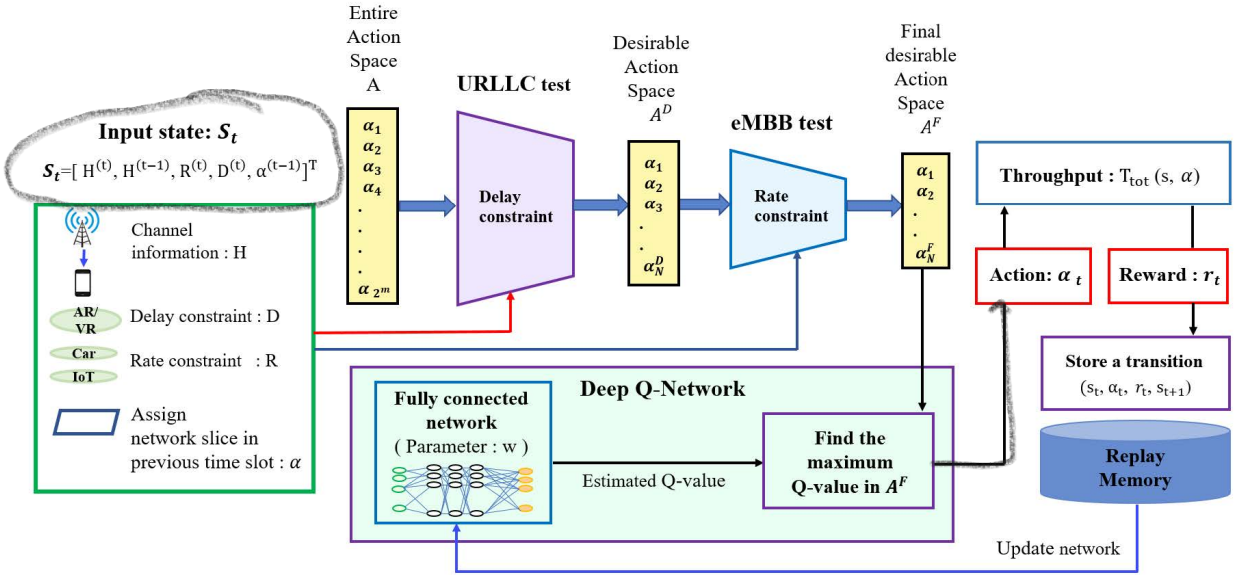

FIGURE 3. Deep Q-network equipped with action elimination for throughput-maximizing network slicing.

2) ACTION SPACE

An action α_t is defined as

$$\alpha_t = \boldsymbol{\alpha}^{(t)} = [\boldsymbol{\alpha}_e^{(t),T}, \boldsymbol{\alpha}_u^{(t),T}, \boldsymbol{\alpha}_m^{(t),T}]^T, \tag{17}$$

where $\boldsymbol{\alpha}_e^{(t)}$, $\boldsymbol{\alpha}_u^{(t)}$, and $\boldsymbol{\alpha}_m^{(t)}$ are the binary vectors that indicate the allocation of eMBB, URLLC, and mMTC slices in the resource block grid, respectively. If we denote the set of possible actions as A, then the size of A (i.e., the number of possible actions) is 2^N. For example, if there are 25 network slices, the size of A is 2^{25}, which is clearly too large to explore. To deal with this problem, we reduce the action space using the URLLC and eMBB tests (we will say more in the next subsection).

3) REWARD

When the action of a time slot is decided, gNB measures the system throughput. Since our goal is to learn the resource allocation policy maximizing the system throughput, we set the reward as the sum of the throughput of all network slices. That is,

$$r_t = T_{\text{total}}^{(t)} \tag{18a}$$

$$= T_{\text{eMBB}}^{(t)} + T_{\text{URLLC}}^{(t)} + T_{\text{mMTC}}^{(t)} \tag{18b}$$

where T_{eMBB}, T_{URLLC}, and T_{mMTC} are the throughputs of eMBB, URLLC, and mMTC slices, respectively. Since all components of T_{total} are the function of slice allocation decisions, the reward maximization problem is equivalent to the problem to determine the allocation of slices maximizing T_{total}.

C. ACTION SPACE REDUCTION VIA ACTION ELIMINATION

The primary purpose of action elimination is to reduce the action space by eliminating the undesirable allocation decisions. As a result, we can improve the chance of making the optimal resource allocation decision maximizing the system throughput and also ensure that the learned resource allocation policy satisfies the QoS requirements of URLLC and eMBB services. One drawback of the proposed method is that computational burden of the training process to obtain the reduced action space is a bit considerable. For example, if we consider 10 network slices, then the action elimination method needs to check $2^{10} \approx 1000$ allocation decisions to identify which decisions violate delay and data rate requirements. To speed up the action elimination process, we can consider the parallel processing. Since parallel processing can simultaneously check whether each individual allocation decision satisfies the QoS requirements or not, we can reduce the computational burden and also speed up the DRL training.

The proposed action elimination method consists of two tests: 1) *URLLC test* to remove the resource allocation decisions that cannot satisfy the delay requirements of URLLC slices and 2) *eMBB test* to remove the infeasible allocation decisions that cannot satisfy the rate requirement of eMBB slices (see Fig. 3).

1) URLLC TEST

In this test, we exclude the infeasible decisions that do not meet the delay requirement of users in URLLC slice from the action space. Let $A = \{\alpha_1, \cdots, \alpha_{2^N}\}$ be the set of all possible resource allocation decisions. To determine whether each allocation decision $\alpha(\in A)$ is infeasible, we measure the throughput of URLLC slice for each α. Recall that the throughput of URLLC slice is expressed as

$$T_{\text{URLLC},k}^{(t)}(s, \alpha) = \sum_{j=1}^{J} \alpha_u^{(t)}(j) f_{j,k} R_k^{(t)}, \quad \forall k \in \mathcal{K}, \tag{19}$$

where $\alpha = \{\alpha_u(j) \in \{0, 1\} | j = 1, \cdots, J\}$ is a single slice allocation decision. When we obtain $T_{\text{URLLC}}^{(t)}(s, \alpha)$ for each α, we eliminate the allocation decision that does not satisfy

Algorithm 1 Training Process of DRL-NS

Input: Channel matrix $\mathbf{H}^{(t)}$, required UE data rates $\mathbf{R}_{\min}^{(t)}$, delay constraints of UEs $\mathbf{D}_{\max}^{(t)}$, resource allocation at previous time slot $\boldsymbol{\alpha}^{(t-1)}$, DRL-based resource allocation decision network h (weight w), total number of time slots T, and learning rate η.

Initialization: $t = 0$ and $w^{(0)} = w^{init}$.

while $w^{(t)}$ does not converge **do**
  for $t = 1, \cdots, T$ **do**
    State $s_t = \{\mathbf{H}^{(t)}, \mathbf{H}^{(t-1)}, \mathbf{R}_{\min}^{(t)}, \mathbf{D}_{\max}^{(t)}, \boldsymbol{\alpha}^{(t-1)}\}$
    Exclude resource allocation decisions through the URLLC test.
    Exclude resource allocation decisions through the eMBB test.
    Obtain $\boldsymbol{\alpha}^{(t)}$ by DQN based on reduced action A^F.
    Compute reward r_t and observe the next state s_{t+1}.
    Store the transition (s_t, a_t, r_t, s_{t+1}) into replay memory $\mathcal{R}$.
  end for
  Randomly sample a mini-batch of the transition (s_i, a_i, r_i, s_{i+1}) with a size of $\mathcal{N}_{\mathcal{R}}$.
  Compute $\nabla_w L(w) = \nabla_w \sum_i (r(s_i, a_i + \gamma \max_{a_{i+1}} Q(s_{i+1}, a_{i+1}, w) - Q(s_{i+1}, a_{i+1}, w))^2$.
  $w^{(t+1)} = w^{(t)} - \eta \nabla_w L(w)$
  $t = t + 1$
end while

the delay requirement of URLLC slice from A to obtain the desirable action space A^D:

$$A^D = \{a \in A | \frac{F_{j,k}}{\sum_{j=1}^{J} \alpha_u^{(t)}(j) f_{j,k} R_k^{(t)}} \leq D_{j,k,\max}, \quad k \in \mathcal{K}, j = 1, \cdots, J\}. \tag{20a}$$

where $F_{j,k}$ is packet length to UE k in slice j.

2) eMBB TEST

In this test, we check whether each resource allocation decision $\alpha \in A^D$ can satisfy the rate requirement for eMBB slice. Specifically, to determine whether $\boldsymbol{\alpha}$ is appropriate to accommodate the eMBB slice, we first measure the throughput of eMBB slice $T_{\text{eMBB}}^{(t)}(s, \alpha)$. Next, similar to the URLLC test we discussed in the previous subsection, we obtain $T_{\text{eMBB}}^{(t)}(s, \alpha)$ for each $\alpha \in A^D$ and then eliminate the allocation decision that does not satisfy the rate requirement of users in eMBB slice. The obtained final desirable action space A^F is

$$A^F = \{\alpha \in A^D | \sum_{i=1}^{I} \alpha_e^{(t)}(i) f_{i,k} R_k^{(t)} \geq R_{k,\min}^{(t)}, k \in \mathcal{K}\}. \tag{21}$$

D. TRAINING OF DRL-NS

An integral part of the proposed DRL-NS is the training process optimizing the set of network parameters w.

TABLE 2. Simulation parameters.

Parameters	Values
Carrier Frequency (f)	2 GHz
Number of time slots (T)	50
BS height (h_B)	15m
Maximum speed of UE (v_{max})	6m/s
Mobile height (h_U)	1.65m
Amplifier efficiency (η)	0.25
Inter-site distance (ISD)	200m
Path loss variable (d_0)	10m
Path loss variable (d_1)	50m
Shadow fading deviation (σ_{sh})	3dB
Number of small cells (M)	10
Gamma (γ)	0.9
Number of UEs (K)	4
Mini-batch size	256
Noise power (σ_n^2)	-174dBm /Hz
Replay memory size ($\mathcal{N}_{\mathcal{R}}$)	20000
Number of network slices	10 EA.

In the training phase, the network parameters are updated to minimize the DQN loss function $L(w) = (r(s, a) + \gamma \max_{a' \in A} Q(s', a', w) - Q(s, a, w))^2$. When the loss function is differentiable, which is true in our case, one can use the stochastic gradient descent (SGD) method to update the parameters. The update operation of SGD is expressed as

$$w^{(t+1)} = w^{(t)} - \eta \nabla_w L(w). \tag{22}$$

where η is the learning rate of DQN.

After obtaining the final desirable action space A^F, the DQN agent calculates the Q-values of all actions in A^F and chooses the action with the maximum Q-value as the output action a_t. Then, the agent computes the immediate reward r_t (see (23)) and the next state s_{t+1} through updated channel matrix $\mathbf{H}$, rate and delay constraints (i.e., $\mathbf{R}$ and $\mathbf{D}$), and the chosen action a_t. In each time slot, the transition tuple (s_t, a_t, r_t, s_{t+1}) observed by the agent is stored to the replay memory. In each iteration of the training phase, a mini-batch data is randomly sampled from the replay memory and the weights of DQN are updated in a direction to minimize the loss value in $L(w)$. The overall training procedure is summarized in Algorithm 1.

E. COMPUTATIONAL COMPLEXITY ANALYSIS

In this subsection, we analyze the computational complexity of the DRL-NS technique in terms of the number of floating point operations (flops). Initially, the state $s_t = [\text{vec}(\mathbf{H}^{(t)}), \text{vec}(\mathbf{H}^{(t-1)}), \mathbf{R}_{\min}^{(t)}, \text{vec}(\mathbf{D}_{\max}^{(t)}), \boldsymbol{\alpha}^{(t-1)}]^T \in \mathbb{R}^{2MK+K+JK+N}$ is fed into the first hidden layer of DQN and then is multiplied by the weight $\mathbf{W}_{i1} \in \mathbb{R}^{\alpha \times (2MK+K+JK+N)}$ and then the bias $\mathbf{b}_{i1} \in \mathbb{R}^{\alpha}$ is added. Then, for each element, we check whether the value is larger than 0 using the rectified

linear unit (ReLU) function (α flops). Thus, the complexity of the initial FC layer $\mathcal{C}_{in}$ is

$$\begin{aligned}\mathcal{C}_{in} &= (2(2MK + K + JK + N) - 1)\alpha + \alpha + \alpha \\ &= 4MK\alpha + 2K\alpha + 2JK\alpha + 2N\alpha + \alpha\end{aligned}$$

Next, by noting that both input and output dimensions of the remaining $h-1$ hidden layers are α, the computational complexity of the remaining hidden layers can be expressed as

$$\mathcal{C}_{hide} = (h-1)((2\alpha - 1)\alpha + \alpha + \alpha) = (h-1)(2\alpha^2 + \alpha). \tag{23}$$

After passing through h hidden layers, the weight multiplication and bias addition are performed in the output FC layer of DQN. Since $\mathbf{W}_{iz} \in \mathbb{R}^{(2MK+K+JK+N)\times\alpha}$ and $\mathbf{b}_{iz} \in \mathbb{R}^{(2MK+K+JK+N)}$, the corresponding complexity $\mathcal{C}_{out}$ of the output layer is

$$\begin{aligned}\mathcal{C}_{out} &= (2\alpha - 1)(2MK + K + JK + N) \\ &\quad + (2MK + K + JK + N) \\ &= 2\alpha(2MK + K + JK + N) \\ &= 4MK\alpha + 2K\alpha + 2JK\alpha + 2N\alpha.\end{aligned}$$

Note that the ReLU function is not applied in the output layer. Lastly, we need to consider the complexity of the action elimination method since it occupies the largest part of the overall complexity in the training process. Basically, in the proposed action elimination method, we need to check 2^N resource allocation decisions for each UE in eMBB slices to identify which decisions violate the data rate requirements. Also, we need to check 2^N allocation decisions for each URLLC slice and UE in URLLC slices to find out which decisions violate the delay requirements. Thus, the complexity of the proposed action elimination method is expressed as

$$\mathcal{C}_{AE} = K(1 + J)2^N. \tag{24}$$

Let ϵ be the maximum training iteration number of DQN, then the overall complexity of DRL-NS $\mathcal{C}_{DRL-NS}$ during the training phase is summarized as

$$\begin{aligned}\mathcal{C}_{DRL-NS} &= \epsilon T(\mathcal{C}_{AE} + \mathcal{C}_{in} + \mathcal{C}_{hide} + \mathcal{C}_{out}) \\ &= \epsilon T(K(1 + J)2^N + 2(h-1)\alpha^2 + h\alpha && (25) \\ &\quad + 4(2MK + K + JK + N)\alpha). && (26)\end{aligned}$$

where T is the total number of time slots. Once DRL-NS is trained, the network parameters (weight and bias) of DQN are no longer updated so that ϵ and T can be eliminated from (28). Also, the action elimination method is no longer required so that $\mathcal{C}_{AE}$ can be eliminated. Therefore, the overall complexity of DRL-NS $\mathcal{C}_{DRL-NS}$ in the test phase is given by

$$\begin{aligned}\mathcal{C}_{DRL-NS} &= \mathcal{C}_{in} + \mathcal{C}_{hide} + \mathcal{C}_{out} \\ &= 4(2MK + K + JK + N)\alpha && (27) \\ &\quad + 2(h-1)\alpha^2 + h\alpha. && (28)\end{aligned}$$

Since $\alpha < (2MK + K + JK + N)$ and h is a small constant, the computational complexity of DRL-NS in the test phase can be expressed as $\mathcal{O}((MK + K + JK + N)\alpha)$.

V. SIMULATION

In this section, we describe the simulation results to demonstrate benefits of DRL-NS in a comprehensive way.

A. SIMULATION SETUP

In our simulation, we consider a downlink transmission scenario of 5G where M gNBs simultaneously serve K UEs. The small cells are uniformly distributed in the hexagonal area of inter-site distance (ISD) 200m and the UEs move freely at a constant speed $\upsilon \in \text{Unif}[\upsilon_{min}, \upsilon_{max}]$. Note that υ_{max} and υ_{min} are the max/min speed of UE. A UE changes its velocity when it reaches the edge of service area. In our work, we set the mobile height h_U to the average adult height (165cm). Note that h_U mainly affects the path loss between the BS and the mobile and the variation of the channel due to the height change (say, from 160cm to 190cm) is negligible [20]. For the fading channel model, we apply the small-scale fading coefficient $g_{m,k}$ generated from the complex Gaussian distribution (i.e, $g_{m,k} \sim \mathcal{CN}(0, 1)$) and the large-scale fading coefficient $\beta_{m,k}$ generated based on Hata-COST231 model [21], which is expressed as

$$\beta_{m,k} = PL_{m,k} 10^{\frac{z_{m,k}\sigma_{sh}}{10}} \tag{29}$$

where $PL_{m,k}$ is the path loss and $10^{\frac{z_{m,k}\sigma_{sh}}{10}}$ is the shadow fading ($z_{m,k} \sim \mathcal{N}(0,1)$). In specific, $PL_{m,k}$ is given by

$$PL_{m,k} = \begin{cases} -L - 35\log_{10} d_{m,k} & \text{if } d_{m,k} > d_1 \\ -L - 15\log_{10} d_1 - 20\log_{10} d_{m,k} & \text{if } d_0 \le d_{m,k} < d_1 \\ -L - 15\log_{10} d_1 - 20\log_{10} d_0 & \text{if } d_{m,k} \le d_0 \end{cases} \tag{30}$$

where $d_{m,k}$ is the distance between the gNB m and the UE k and

$$\begin{aligned}L &= 46.3 - 33.9\log_{10} f - 13.82\log_{10} h_b \\ &\quad - (1.1\log_{10} f - 0.7)h_u - (1.56\log_{10} f - 0.8)\end{aligned} \tag{31}$$

where f is the carrier frequency, h_B and h_U are the heights of BS and UE, respectively. Note that the duration of time slot is set to 1 (secs).

The DQN of DRL-NS consists of 5 fully connected layers and the width of a hidden layer is set to 256. For the network parameter training, we use an Adam optimizer, a well-known optimization tool to guarantee the robustness of training process [22]. The simulation parameters are listed in Table 1. Our approach is implemented based on Tensorflow [23]. The DQN is trained on a single NVIDIA GeForce Titan Xp.

We compare the proposed DRL-NS scheme with four baseline network slicing techniques: 1) *equal allocation* method where the equal number of RBs are allocated to each network slice [5], 2) *regression tree-based allocation* method where the number of RBs allocated to each slice is sequentially decided in a way to maximize the system throughput [4], 3) *proportional fair-based allocation* method where RBs are allocated in a way to pursue a balance between the total

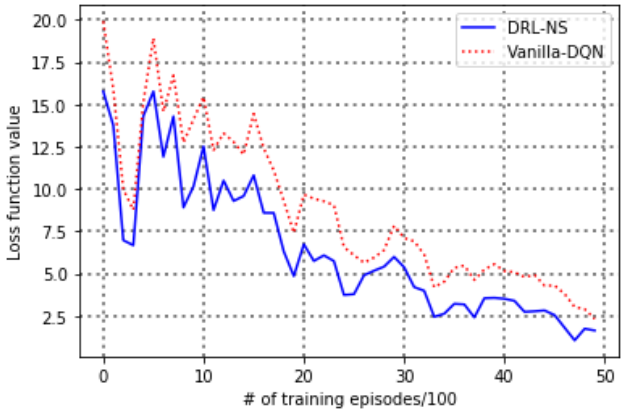

FIGURE 4. DQN loss as a function of the number of training episodes.

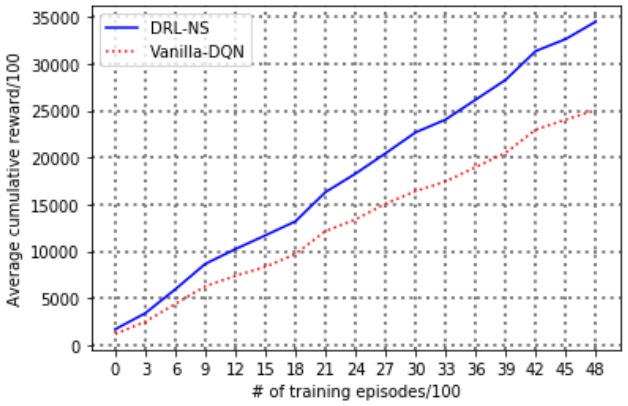

FIGURE 5. Average cumulative reward as a function of the number of training episodes.

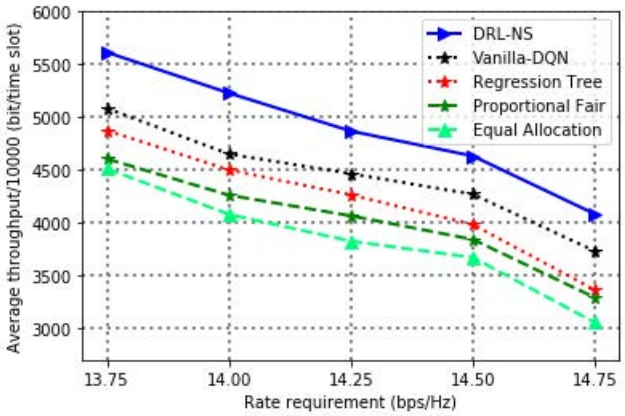

FIGURE 6. Average system throughput as a function of rate requirement R_{min} ($M = 10$, $K = 4$).

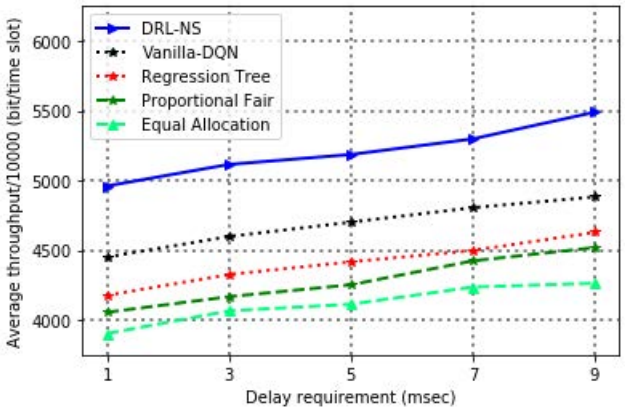

FIGURE 7. Average system throughput as a function of delay requirement D_{max} ($M = 10$, $K = 4$).

throughput and QoS requirement satisfaction [12], and 4) *vanilla DQN-based allocation* method where the allocation of RBs is determined by using basic DQN without special treatment.

B. SIMULATION RESULT

In order to examine the convergence behavior of the proposed DRL-NS, we plot the loss function value as a function of the number of training episodes (see Fig. 4). In this test, we set $R_{min} = 15.0$ bps/Hz. From Fig. 4, we clearly see that the loss function decreases with the number of episodes, which indicates that the training process of DRL-NS is carried out properly in a way to increase the system throughput. Also, since the proposed technique enhances the sample efficiency significantly by exploring the resource allocation decisions in desirable action space, the loss of DRL-NS is smaller than that of the vanilla DQN scheme.

In Fig. 5, we plot the cumulative reward as a function of the number of training episodes in the training phase. We observe that as the number of training episodes increases, the cumulative reward of DRL-NS increases gradually. Since DRL-NS receives fewer penalties by eliminating the infeasible resource allocation decisions, cumulative reward of DRL-NS is much higher than that of the vanilla DQN scheme.

In Fig. 6, we plot the average throughput of DRL-NS as a function of UE's data rate requirement. We observe that DRL-NS outperforms the conventional network slicing methods across the board. For example, when $R_{min} = 13.75$ bps/Hz, DRL-NS achieves about 25% improvement in the average system throughput over the equal allocation network slicing method. This is because while the equal allocation network slicing method allocates the equal number of RBs for each network slice, DRL-NS makes sequential resource allocation decisions in a way to maximize the long-term system throughput. Also, due to the elimination of the infeasible resource allocation decisions incurring sub-optimal allocation policy, DRL-NS achieves 15% improvement in the system throughput over the regression tree-based allocation network slicing method.

We also plot the average throughput of DRL-NS as a function of UE's delay requirement (see Fig. 7). As shown in Fig. 7, we observe that the proposed DRL-NS achieves a significant gain over the conventional methods. For example, DRL-NS achieves around 27% and 19% improvements in the

TABLE 3. DRL-NS average throughput comparison for various resource utilization ratios.

Resource utilization (%)	60	70	80	90
Average throughput (bit/time slot)	1689	2275	2930	3893

TABLE 4. DRL-NS training time comparison for various SNR levels.

SNR (dB)	20	25	30
runtime (s)	1.73×10^3	1.75×10^3	2.47×10^3

throughput performance over the equal allocation and regression tree-based network slicing methods at $D_{max} = 1$ msec, respectively. Since the resource allocation decisions that cannot satisfy the delay requirements of UEs are removed, DRL-NS has better chance to find out the optimal resource allocation policy maximizing the system throughput.

In Table 3, we summarize the average throughput of DRL-NS for various resource utilization ratios. We observe that as the ratio of utilized network slices increases, the throughput performance increases as well. This is because throughputs of larger number of eMBB, URLLC, and mMTC network slices add up to the total throughput of the overall system.

Finally, we summarize the training time of DRL-NS for various SNR levels (see Table 4). In this simulation, we declare that DRL-NS converges when the absolute fraction difference of the loss is smaller than the threshold $\varepsilon = 0.001$, and then measure the time to the convergence.[2] As shown in Table 1, when SNR = 20dB, it takes about 1.73×10^3 seconds for the algorithm to converge.

VI. CONCLUSION

In this paper, we proposed the DRL-based network slicing technique called DRL-NS, to improve the system throughput while satisfying all the QoS requirements. In the proposed DRL-NS, undesirable resource allocation decisions violating various QoS requirements are eliminated by specially designed mechanism called action elimination so that we could significantly reduce the action space, and therefore improve the chance of learning the optimal resource allocation policy. Through the simulations on realistic 5G environment, we observed that DRL-NS outperforms conventional schemes by a large margin. In this paper, we restricted our attention on network slicing but we expect that proposed scheme can be extended to many different tasks such as radio link scheduling, beam management, and cognitive radio scheduling. In consideration of a long road ahead to exploit DRL to 6G wireless systems, we believe that the DRL-NS can be served as a useful tool for future wireless applications.

REFERENCES

[1] H. Ji, S. Park, J. Yeo, Y. Kim, J. Lee, and B. Shim, "Ultra-reliable and low-latency communications in 5G downlink: Physical layer aspects," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 124–130, Jun. 2018.

[2] That is, $|\frac{L(w^{(t+1)})-L(w^{(t)})}{L(w^{(t)})}| < \varepsilon$ ($\varepsilon = 0.001$).

[2] S. Kim and B. Shim, "Localization of Internet of Things network via deep neural network based matrix completion," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2020, pp. 1766–1770.
[3] R. Peter, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017.
[4] N. Salhab, R. Rahim, R. Langar, and R. Boutaba, "Machine learning based resource orchestration for 5G network slices," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.
[5] C. Bektas, S. Bocker, F. Kurtz, and C. Wietfeld, "Reliable software-defined RAN network slicing for mission-critical 5G communication networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.
[6] H. Ju, S. Kim, Y. Kim, H. Lee, and B. Shim, "Energy-efficient ultra-dense network using deep reinforcement learning," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–5.
[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, and J. Veness, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[8] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor, "Learn what not to learn: Action elimination with deep reinforcement learning," in *Proc. Adv. Neural Inform. Process. Syst.*, 2018, pp. 3562–3573.
[9] D. Li, H. Zhang, K. Long, W. Huangfu, J. Dong, and A. Nallanathan, "User association and power allocation based on Q-learning in ultra dense heterogeneous networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–5.
[10] Y. Fu, Z. Yang, T. Q. Quek, and H. H. Yang, "Towards cost minimization for wireless caching networks with recommendation and uncharted users' feature information," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6758–6771, Oct. 2021.
[11] W. Xia, G. Zheng, Y. Zhu, J. Zhang, J. Wang, and A. P. Petropulu, "A deep learning framework for optimization of MISO downlink beamforming," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1866–1880, Mar. 2020.
[12] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Honolulu, HI, USA, Apr. 2018, pp. 2177–2185.
[13] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "DeepSlice: A deep learning approach towards an efficient and reliable network slicing in 5G networks," in *Proc. IEEE 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2019, pp. 762–767.
[14] S. Xiao and W. Chen, "Dynamic allocation of 5G transport network slice bandwidth based on LSTM traffic prediction," in *Proc. IEEE 9th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Nov. 2018, pp. 735–739.
[15] Y. Cui, X. Huang, D. Wu, and H. Zheng, "Machine learning based resource allocation strategy for network slicing in vehicular networks," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2020, pp. 454–459.
[16] M. Yan, G. Feng, J. Zhou, and S. Qin, "Smart multi-RAT access based on multiagent reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4539–4551, May 2018.
[17] G. Sun, K. Xiong, and G. O. Boateng, "Resource slicing and customization in RAN with dueling deep Q-network," *J. Netw. Comput. Appl.*, vol. 157, no. 3, pp. 1–13, 2020.
[18] J. Tang, B. Shim, and T. Q. S. Quek, "Service multiplexing and revenue maximization in sliced C-RAN incorporated with URLLC and multicast eMBB," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 881–895, Apr. 2019.
[19] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.
[20] *Study on Channel Model for Frequencies From 0.5 to 100 GHz (Release 14)*, v14.3.0, 3GPP document 38.901, 2018.
[21] Y. Singh, "Comparison of Okumura, Hata and COST-231 models on the basis of path loss and signal strength," *Int. J. Comput. Appl.*, vol. 59, no. 11, pp. 37–41, Dec. 2012.
[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[23] M. Abadi, P. Barham, J. Chen, Z. Chen, and A. Davis, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

KYUNGJOO SUH received the B.S. degree from Korea University, in 1993, the M.S. degree from Seoul National University, South Korea, in 1996, and the M.S. degree in computer and information science and engineering from the University of Florida at Gainesville, Gainesville, FL, USA, in 2001. She is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University. Her research interests include wireless communications, mobile computing, security, and machine learning.

SUNWOO KIM received the B.S. degree in electrical engineering from Stony Brook University, USA, in 2018. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University, South Korea. His research interests include signal processing and deep learning techniques for wireless communications.

YONGJUN AHN received the B.S. degree from the Department of Electrical and Computer Engineering, Seoul National University, South Korea, in 2018, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include 5G and 6G wireless communications and deep learning techniques.

SEUNGNYUN KIM (Graduate Student Member, IEEE) received the B.S. degree from the Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include signal processing and deep learning techniques for the 5G and 6G wireless communications.

HYUNGYU JU (Graduate Student Member, IEEE) received the B.S. degree from the Department of Electrical and Computer Engineering, Seoul National University, South Korea, in 2018, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include 5G and 6G wireless communications and deep reinforcement learning techniques.

BYONGHYO SHIM (Senior Member, IEEE) received the B.S. and M.S. degrees in control and instrumentation engineering from Seoul National University, South Korea, in 1995 and 1997, respectively, and the M.S. degree in mathematics and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign (UIUC), Champaign, IL, USA, in 2004 and 2005, respectively. From 1997 to 2000, he was an Officer (First Lieutenant) and an Academic full-time Instructor with the Department of Electronics Engineering, Korean Air Force Academy. From 2005 to 2007, he was a Staff Engineer with Qualcomm Inc., San Diego, CA, USA. From 2007 to 2014, he was an Associate Professor with the School of Information and Communication, Korea University, Seoul. Since 2014, he has been with Seoul National University (SNU), where he is currently a Professor and the Vice Chair of the Department of Electrical and Computer Engineering. His research interests include wireless communications, statistical signal processing, compressed sensing, and machine learning. He has served as an Elected Member for the Signal Processing for Communications and Networking (SPCOM) Technical Committee of the IEEE Signal Processing Society. He was a recipient of the M. E. Van Valkenburg Research Award from the Department of Electrical and Computer Engineering, University of Illinois, in 2005, the Hadong Young Engineer Award from IEIE in 2010, the Irwin Jacobs Award from Qualcomm and KICS in 2016, the Shinyang Research Award from the Engineering College of SNU in 2017, the Okawa Foundation Research Award in 2020, and the IEEE ComSoc AP Outstanding Paper Award in 2021. He has served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE WIRELESS COMMUNICATIONS LETTERS, and the *Journal of Communications and Networks*, and the Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.