Deep Reinforcement Learning-based Network Slicing for Beyond 5G

KYUNGJOO SUH1, SUNWOO KIM1, YONGJUN AHN1, SEUNGYUN KIM1, HYUNGYU JU1, AND BYONGHYO SHIM1, (Senior Member, IEEE)

1Dept. of Electrical and Computer Engineering, Seoul National Univ., 1 Gwanak-ro, Gwanak-gu, Seoul, 08826 Korea

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

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2020R1A2C102198).

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 each RB to a certain service 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 use action elimination to eliminate 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].

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, the coexistence of eMBB, URLLC, and mMTC needs to be handled properly by the base station (BS) and 5G core network (i.e., 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 at the same time 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 or 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 at every 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 of DRL, thereby slowing down the convergence speed and also preventing the DRL from maximizing the reward. To address the problem, we employ the action elimination [8], an approach to exclude undesirable actions among all possible actions to boost up the training speed and quality of trained policy. Due to the elimination of 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 I.

### 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. Recently, 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. Recently, DL has been 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 distinct from previous studies in the sense that we integrate the action elimination to DRL to reduce the size of action space. 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 $K = \{1, \cdots, K\}$. 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 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 $I$.

\(^1\)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.
**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 the three major components in the system throughput: 1) throughput of eMBB slices ($T_{\text{eMBB}}$), 2) throughput of URLLC slices ($T_{\text{URLLC}}$), and 3) throughput of mMTC slices ($T_{\text{mMTC}}$).

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

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

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:

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

The corresponding sum throughput of eMBB network slice is

$$T_{\text{eMBB}} = \sum_{k=1}^{K} T_{\text{eMBB},k}. \quad (5)$$

2) Throughput of URLLC slice

The throughput of URLLC slice of UE $k$ is

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

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

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

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

$$T_{\text{URLLC}} = \sum_{k=1}^{K} T_{\text{URLLC},k}. \quad (7)$$

3) Throughput of mMTC slice

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

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

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

$$T_{\text{mMTC}} = \sum_{k=1}^{K} T_{\text{mMTC},k}. \quad (8)$$

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

$$T_{(t)} = T_{\text{eMBB}} + T_{\text{URLLC}} + T_{\text{mMTC}}. \quad (9)$$

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

$$\mathcal{P} : \max_{\{\alpha_e^{(t)}, \alpha_u^{(t)}, \alpha_m^{(t)}\}} \sum_{t=1}^{T} T_{(t)} \quad \text{s.t.} \quad T_{(t),k} \geq R_{(t),k,\text{min}}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}, \quad (9a)$$

$$\frac{F_{j,k}}{T_{\text{URLLC},k}} \leq D_{j,k,\text{max}}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}, \quad (9b)$$

$$\alpha_e^{(t)} + \sum_{j=1}^{J} \alpha_u^{(t)}(j) + \sum_{l=1}^{L} \alpha_m^{(t)}(l) \leq \tau N, \quad (9c)$$

where $T_{(t),k}$ is the total throughput of UE $k$ at time slot $t$, $R_{(t),k,\text{min}}$ is the minimum throughput requirement of UE $k$ at time slot $t$, $F_{j,k}$ is the packet length to UE $k$ in $j$-th URLLC network slice, $D_{j,k,\text{max}}$ is the maximum packet delay of UE $k$ in $j$-th URLLC network slice, and $\tau N$ is the total number of RBs in the network slice.
where $\alpha_c(i)$, $\alpha_u(j)$, and $\alpha_m(l)$ are the elements of binary vectors $\alpha_c^t$, $\alpha_u^t$, and $\alpha_m^t$, respectively. Also, $T = \{1, \cdots, T\}$, and $0 \leq \tau \leq 1$ 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

$$\mathcal{P}^t = \max_{\{\alpha_c^t, \alpha_u^t, \alpha_m^t\}} \sum_{t=0}^{\infty} \sum_{i=1}^{K} \sum_{k=1}^{L} \alpha_c^t(i) f_{i,k} R_k^t + \sum_{j=1}^{J} \alpha_u^t(j) f_{j,k} D_{j,k}^t + \sum_{l=1}^{L} \alpha_m^t(l) f_{l,k} P_{l,k}^t$$

s.t. $\sum_{i=1}^{I} \alpha_c^t(i) f_{i,k} R_k^t \geq R_{k,\min}, \forall k \in \mathcal{K}$, $\forall t \in \mathcal{T}$, $\sum_{j=1}^{J} \alpha_u^t(j) f_{j,k} D_{j,k}^t \leq D_{j,k,\max}, \forall k \in \mathcal{K}$, $\forall t \in \mathcal{T}, \forall j \in \mathcal{J}$, $\sum_{i=1}^{I} \alpha_c^t(i) + \sum_{j=1}^{J} \alpha_u^t(j) + \sum_{l=1}^{L} \alpha_m^t(l) \leq \tau N$, $\forall t \in \mathcal{T}$.

Since $\alpha_c^t(i)$, $\alpha_u^t(j)$, and $\alpha_m^t(l)$ are binary integers, $\mathcal{P}^t$ is a mixed-integer programming, which is known as a non-convex NP-hard problem [6]. When we try to solve $\mathcal{P}^t$ 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 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 method 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 $\pi^*$ maximizing the expectation of cumulative reward is [19],

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

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

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

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

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 $Q^*(s, a)$. 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]:

$$Q^*(s, a) = r(s, a) + \gamma \max_{a' \in A} Q^*(s', a'),$$ (13)

where $r(s, a)$ is the reward corresponding to the state-action pair $(s, a)$ and $P_{ss'}^{a}$ 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 $L(w) = (Y_{t}^{\text{dqn}} - Q(s, a, w))^2$ where $Y_{t}^{\text{dqn}} = r(s, a) + \gamma \max_{a' \in A} Q(s', a', w)$.

**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 the proposed scheme, 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). In 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 $R_{\text{min}}^{(t)} = [R_{1, \text{min}}^{(t)}, \ldots, R_{K, \text{min}}^{(t)}]^T$ at the time slot $t$, the slice allocation decision at the previous time slot $\alpha^{(t-1)} = [\alpha_{e}^{(t-1)}, \ldots, \alpha_{N}^{(t-1)}]^T$, and the channel matrix $H^{(t)}$ representing the path loss and the shadowing effect between BSs and UEs at the time slot $t$. To be specific, $H^{(t)}$ is expressed as $M \times K$ matrix as

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

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

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

In summary, the state can be expressed as

$$s_t = [H^{(t)}, H^{(t-1)}, R_{\text{min}}^{(t)}, D_{\text{max}}^{(t)}, \alpha^{(t-1)}]^T. \quad (16)$$

Note that both $H^{(t)}$ and $H^{(t-1)}$ 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 $H^{(t)}$, $R_{\text{min}}^{(t)}$, $D_{\text{max}}^{(t)}$, and $\alpha^{(t-1)}$, the DRL agent learns the resource allocation policy maximizing the system throughput while satisfying the QoS requirements.

2) Action space

An action $\alpha_t$ is defined as

$$\alpha_t = \alpha^{(t)} = [\alpha_{e}^{(t)} T, \alpha_{u}^{(t)} T, \alpha_{m}^{(t)} T]^T, \quad (17)$$

where $\alpha_{e}^{(t)}$, $\alpha_{u}^{(t)}$, and $\alpha_{m}^{(t)}$ 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,

$$r_t = T_{\text{total}}(t)$$

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).

C. ACTION SPACE REDUCTION VIA ACTION ELIMINATION
The main purpose of action elimination method 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 first 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 $\alpha$.

$$T_{\text{URLLC}}(s, \alpha)$$

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

C. ACTION SPACE REDUCTION VIA ACTION ELIMINATION
The main purpose of action elimination method 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

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

![Diagram showing the process of deep Q-network equipped with action elimination for network slicing](image-url)
Algorithm 1: Training process of DRL-NS

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

Initialization: $t = 0$ and $w^{(0)} = w^{\text{init}}$.
1: while $w^{(t)}$ does not converge do
2: \hspace{1em} for $t = 1, \ldots, T$ do
3: \hspace{2em} State $s_t = \{H^{(t)}, H^{(t-1)}, R_{\text{min}}, D_{\text{max}}, \alpha^{(t-1)}\}$
4: \hspace{2em} Exclude resource allocation decisions through the URLLC test.
5: \hspace{2em} Exclude resource allocation decisions through the eMBB test.
6: \hspace{2em} Obtain $\alpha^{(t)}$ by DQN based on reduced action $A^F$.
7: \hspace{2em} Compute reward $r_t$ and observe the next state $s_{t+1}$.
8: \hspace{2em} Store the transition $(s_t, a_t, r_t, s_{t+1})$ into replay memory $\mathcal{R}$.
9: \hspace{1em} end for
10: Randomly sample a mini-batch of the transition $(s_t, a_t, r_t, s_{t+1})$ with a size of $N_\mathcal{R}$.
11: Compute $\nabla_w L(w) = \nabla_w \sum_t [r_t(s_t, a_t) + \gamma \max_{a_t+1} Q(s_{t+1}, a_{t+1}, w) - Q(s_t, a_t, w)]^2$.
12: $w^{(t+1)} = w^{(t)} - \eta \nabla_w L(w)$.
13: $t = t + 1$
14: end while

the delay requirement of URLLC slice from $A$ to obtain the desirable action space $A^D$: $A^D = \{a \in A | \frac{F_{j,k}}{\sum_{j=1}^{J} \alpha_{a}^{(t)}(j)f_{j,k}R^T_{k}} \leq D_{j,k}\text{max}, k \in K, j = 1, \ldots, J\}$.

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 $\alpha$ 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 $a \in A^D$ 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 $A^F = \{a \in A^D | \frac{f_{i,k}}{R^T_{k}} \geq R_{k,\text{min}}, k \in K\}$. 

D. TRAINING OF DRL-NS
An integral part of the proposed DRL-NS is the training process optimizing the set of network parameters $w$. 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 $w^{(t+1)} = w^{(t)} - \eta \nabla_w L(w)$.

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 $s_t = [\text{vec}(\text{vec}(H^{(t)})), \text{vec}(H^{(t-1)}), R_{\text{min}}, \text{vec}(D_{\text{max}}), \alpha^{(t-1)}]^T \in \mathbb{R}^{2M+K+JK+N}$ is fed into the first hidden layer of DQN and then is multiplied by the weight $W_{ij} \in \mathbb{R}^{\alpha \times (2M+K+JK+N)}$ and then the bias $b_{i,1} \in \mathbb{R}^{\alpha}$ is added. Then, for each element, we check whether the value is larger than 0 using the rectified linear unit (ReLU) function ($\alpha$)

| 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 $(\eta)$   | 0.25   |
| Inter-site distance (ISD)       | 200m   |
| Path loss variable $(d_0)$      | 10m    |
| Path loss variable $(d_1)$      | 50m    |
| Shadow fading deviation $(\sigma_{sh})$ | 3dB |
| Number of small cells $(M)$     | 10     |
| Gamma $(\gamma)$               | 0.9    |
| Number of UEs $(K)$             | 4      |
| Mini-batch size                 | 256    |
| Noise power $(\sigma_n^2)$      | -174dBm/Hz |
| Replay memory size $(N_R)$      | 20000  |
| Number of network slices        | 10 EA  |
The action elimination method is expressed as $C_{AE}$.

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

$$C_{DRL-NS} = \epsilon T (C_{AE} + C_{in} + C_{hid} + C_{out})$$

$$= \epsilon T (K(1+J)2^N + 2(h-1)\alpha^2 + h\alpha) + 4(2MK + K + JK + N)\alpha. \quad (25)$$

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 $\epsilon$ and $T$ can be eliminated from (28). Also, the action elimination method is no longer required so that $C_{AE}$ can be eliminated. Therefore, the overall complexity of DRL-NS $C_{DRL-NS}$ in the test phase is given by

$$C_{DRL-NS} = C_{in} + C_{hid} + C_{out}$$

$$= 4(2MK + K + JK + N)\alpha + 2(h-1)\alpha^2 + h\alpha. \quad (26)$$

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 $O((MK + K + JK + N)\alpha)$.

V. SIMULATION

A. SIMULATION SETUP

In this section, we describe the simulation results to demonstrate benefits of DRL-NS in a comprehensive way. In our simulation, we consider a downlink transmission scenario 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 $\nu \in \text{Unif}[v_{min}, v_{max}]$. Note that $v_{max}$ and $v_{min}$ 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. 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 [20], which is expressed as

\[
\beta_{m,k} = P L_{m,k} 10^{\frac{z_{m,k} d_{m,k}}{10}}
\]

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

\[
P L_{m,k} = \begin{cases} 
-L - 35 \log_{10} d_{m,k} & \text{if } d_{m,k} > d_1 \\
-L - 15 \log_{10} d_{m,k} - 20 \log_{10} d_{m,k} & \text{if } d_0 \leq d_{m,k} < d_1 \\
-L - 15 \log_{10} d_{m,k} & \text{if } d_{m,k} \leq d_0 
\end{cases}
\]

where \( d_{m,k} \) is the distance between the gNB \( m \) and the UE \( k \) and

\[
L = 46.3 - 33.9 \log_{10} f - 13.82 \log_{10} h_b \\
- (1.10 \log_{10} f - 0.7) h_u - (1.56 \log_{10} f - 0.8)
\]

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 [21]. The simulation parameters are listed in Table 1. Our approach is implemented based on Tensorflow [22]. 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 maximizing the system throughput [4], 3) proportional fair-based allocation method where RBs are allocated in a way to pursue a balance between the total 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.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
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 throughput performance over the equal allocation and regression tree-based network slicing methods at $D_{\text{max}} = 1$ 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. As shown in Table 1, when SNR = 20dB, it takes about $1.73 \times 10^3$ seconds for the algorithm to converge.

**VI. CONCLUSION**

In this paper, we proposed the DRL-based network slicing framework 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 action elimination so that we could significantly reduce the action space, and therefore improve the chance of making the optimal resource allocation decision. 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 the deep reinforcement learning paradigm to wireless communication systems, we believe that the DRL-NS can be useful tool for the 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 Communications., vol. 25, no. 3, pp. 124-130, Jun. 2018.

[2] S. Kim and B. Shim, “Localizaton of Internet of Things Network via Deep Neural Network Based Matrix Completion,” in Proc. Int. Conf. Commun. Technol. Converg. (ICTC), pp. 1766–1770, Oct. 2020.

[3] R. Peter et al., “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. Salahb, R. Rahim, R. Langar, and R. Boutaba, “Machine learning based resource orchestration for 5G network slices,” in IEEE Global Commun. Conf. (GLOBECOM), Waikoloa, USA, Dec. 2019.

[5] C. Bektas, S. Bockler, F. Kurtz, and C. Wietfeld, “Reliable software-defined ran network slicing for mission-critical 5G communication networks,” in Proc. IEEE Globecom Workshops (CG Wkshps), Dec. 2020.

[6] H. Ju, S. Kim, Y. Kim, H. Lee, and B. Shim, “Energy-efficient ultra-dense network using deep reinforcement learning,” in Proc. IEEE Workshop Signal Process. Adv. Wireless Commun. (SPAWC), pp. 1- 5, 2020.

[7] V. Minh, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “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 Adv. Neural Inform. Process. Syst., pp. 3562-3573, 2018.

[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 GLOBECOM Conf. pp. 1-5, Dec. 2019.

[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., May 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. 3697-3712, May 2019.

[12] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. Kozat, “A resource allocation framework for network slicing,” in Proc. IEEE INFOCOM, Honolulu, HI, USA, Apr. 2018.

[13] A. Thanhtarare, R. Paparokt, V. Walunj, and C. Beard, “DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks,” in Proc. IEEE 10th Ann. Ubiquitous Comput. Electron. Mobile Commun. Conf. (UEMCON), pp. 762-767, Oct. 2019.

[14] S. Xiao and W. Chen, “Dynamic allocation of 5G transport network slice bandwidth based on LSTM traffic prediction,” in Proc. IEEE Int. Conf. Swat. Serv. Sci. (ICISS), pp. 735-739, Nov. 2019.

[15] Y. Cui, X. Huang, D. Wu, H. Zheng, “Machine Learning based resource allocation strategy for network slicing in vehicular networks,” in Proc. IEEE/CIC Int. Conf. Commun. China (ICCC), pp. 454-459, 2020.

[16] M. Yan, G. Feng, J. Zhou, and S. Qin, “Smart multi-RAT access based on multi-agent reinforcement learning,” IEEE Trans. Veh. Technol., vol. 67, no. 5, pp. 4539-4551, May 2018.

[17] Y. Cui, X. Huang, D. Wu, H. Zheng, “Machine Learning based resource allocation for network slicing,” in Proc. IEEE J. Sel. Areas Commun., vol. 37, no. 4, pp. 881–895, Apr. 2019.

[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, A. G. Barto et al., “Introduction to reinforcement learning,” MIT press Cambridge, 1998, vol. 135.

[20] 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, Dec. 2012.

[21] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Represent. (ICLR), pp. 1-15, 2015.

[22] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., “Tensorflow: A system for largescale machine learning,” in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283, 2016.
KYUNGJOO SUH received the B.S. degree from Korea University and M.S. degree from Seoul National University, Korea, in 1993 and 1996, respectively, and the M.S degree in computer and information science and engineering from the University of Florida at Gainesville, FL, USA, in 2001. She is currently working toward the Ph.D. degree in electrical and computer engineering from Seoul National University, Korea. 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 working toward the Ph.D. degree in electrical and computer engineering from Seoul National University, 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 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 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 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 IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE WIRELESS COMMUNICATIONS LETTERS, and Journal of Communications and Networks and the Guest Editor for IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.

***

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/