IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received March 8, 2019, accepted March 31, 2019, date of publication April 10, 2019, date of current version April 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2909670

Dynamic Reservation and Deep Reinforcement Learning Based Autonomous Resource Slicing for Virtualized Radio Access Networks

GUOLIN SUN[1], (Member, IEEE), ZEMUY TESFAY GEBREKIDAN[1], GORDON OWUSU BOATENG[1], DANIEL AYEPAH-MENSAH[1], AND WEI JIANG[2,3], (Member, IEEE)

[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 69731, China
[2]German Research Center for Artificial Intelligence (DFKI GmbH), 67663 Kaiserslautern, Germany
[3]Department of Electrical and Information Technology, Technische University, 67663 Kaiserslautern, Germany

Corresponding author: Guolin Sun (guolin.sun@uestc.edu.cn)

This work was supported in part by the National Natural Science Research Foundation of China under Grant 61771098, in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2018J049, in part by the Fund from the Department of Science and Technology of Sichuan Province under Grant 2017GFW0128, Grant 8ZDYF2268, Grant 2018JY0578, and Grant 2017JY0007, and in part by the ZTE Innovation Research Fund for Universities Program 2016.

ABSTRACT The elastic reconstruction of 5G network services is expected to provide the capability of network slice orchestration to access the network on demand, guarantee service experience on demand, and construct services on demand as well as to construct basic network services with lower costs. It is challenging to have different applications served independently with a proper resource allocation mechanism according to their own requirements. In this paper, we propose a dynamic resource reservation and deep reinforcement learning-based autonomous virtual resource slicing framework for the next generation radio access network. The infrastructure provider periodically reserves the unused resource to the virtual networks based on their ratio of minimum resource requirements. Then, the virtual networks autonomously control their resource amount using deep reinforcement learning based on the average quality of service utility and resource utilization of users. With the defined framework in this paper, virtual operators can customize their own utility function and objective function based on their own requirements. The simulation results show the performances on convergence rate, resource utilization, and satisfaction of the virtual networks.

INDEX TERMS Resource slicing, network virtualization, deep reinforcement learning.

I. INTRODUCTION

With the emergence of fifth generation (5G) mobile cellular networks, a wide variety of innovative services are bound to arise leading to exponential growth of cellular users. 5G networks are envisioned to provide flexible edge-to-core infrastructure to offer diverse applications. The convergence of software-defined networking (SDN) and network virtualization (NV) is a promising approach to support such dynamic networks [1]. To satisfy customers' demands in terms of good quality of service (QoS) is very challenging to service providers (SPs) for such versatile traffics. 5G improves some areas where 4G fails to address properly i.e. higher data rate, very low end-to-end (E2E) latency, massive device connection, reduced operational cost, higher capacity and consistent quality of experience. In terms of 5G services, improvement over 4G can be classified into four main use cases; i) Enhanced mobile broadband (eMBB), which is expected to attain the peak speed up to 20Gbps, with the aid of new radio access technology (RAT), mmWave, massive multiple-input and multiple-output (MIMO). ii) Ultra-reliability and ultra-low-latency communications (URLLC) considering the E2E delivery delay, which is few micro-seconds for mission-critical services, such as smart-grid, automatic drive, motion control and automated factory with remote controlled machine. iii) The scenario of massive connectivity probably introduces hundreds of millions of internet of things (IoT) devices like static sensors, also known as massive IoT (MIoT) scenario. iv) High definition TV (Hdtv) and on-line video streaming with large flows and critical time

The associte editor coordinating the review of this manuscript and approving it for publication was Hamed Ahmadi.

delays such as virtual reality. These scenarios when present in the same one network can be termed as heterogeneous traffics.

Recently, NV has attracted a lot of interests from both academia and industry [2]. NV abstracts resources and services in a way that different services can share the physical hardware. It creates several logical virtual networks (VNs) that work independently on top of a physical network. With NV, physical networks are mostly reduced to packets forwarding. The capability of NV to create virtual networks enables the support of various requirements in a mobile environment where different SPs co-exist. In order to accommodate the significant growth of traffic volume and diverse services in 5G networks, it is natural to extend virtualization from wired networks to wireless networks. The virtualization of wireless network is to realize the process of abstracting, slicing, isolating and sharing radio resources. Moreover, NV is very closely related to recent advances in SDN, which simplifies the management of network services, by decoupling the control plane and data plane on each network node, hiding physical deployments and presenting them as virtualized services. The combination of NV and SDN technologies leads to two complementary goals of 5G, i) a service-oriented network architecture that is programmable and extensible in terms of infrastructure, network services and mobile applications; ii) network slicing that decouples operations of virtual networks on the top of physical infrastructure with slice isolation and customized resource allocation for tenants.

Most of existing works on service oriented networking and virtualization have at least one common limitation i.e. the use of a common objective function in their optimization such as profit, rate and overall quality of experience (QoE) to allocate resource to all slices or to the users of all slices [3]–[5]. This is not efficient because it cannot provide a customized objective function for various applications. Our main contributions in this paper are summarized as follows:

-We propose a two-level framework for radio resource virtualization and allocation in 5G. In the upper level, the infrastructure provider (INP) will dynamically reserve the available unused resource to the appropriate virtual networks. In the lower level, virtual networks can autonomously adjust their resource allocated to their users.

-In dynamic reservation mechanism, the INP collects the unused resources from the slices and reserves them back to the slices that may need extra resource. The unused resources from the slices are reserved back to them to prevent one slice's congestion from affecting the performance of the other slices. For instance, if there is a sudden increase in the number of users, an immediate response in terms of resource allocation to new subscribers may be impossible. Therefore, a portion of the reserved resource is used to serve the new users. In this way, performance degradation of other slices can be avoided. Only the unused resource is reserved to the slices. There is no conflict between any two slices because they already have different reserved resource, as is benefit for slice isolation and slices can independently customize the physical resource allocation to their users.

-In autonomous resource management for multiple slices, we propose a deep reinforcement learning algorithm which autonomously adjusts resource allocated to slices based on the feedback of the average QoS utility and average resource utilization of their users.

-We conduct a comprehensive performance evaluation of our proposed algorithm, comparing it with benchmarks, network virtualization substrate NVS [5] and NetShare [6]. Simulation results verify that the slice satisfaction and resource utilization are improved.

The outline of this paper is as follows: we review the related works in Section II, the system models are introduced in Section III and Section IV provides the problem formulation and algorithm. The evaluation of the proposed method is given in Section V. We conclude this work in Section VI.

II. RELATED WORKS

The idea of service-oriented virtual resource allocation has been explored to tackle the resource allocation issue in sliced network functions. In [7], the authors analyzed the effect of network slicing in 5G radio access network (RAN) and what it offered in the design of RAN. A virtualization framework was proposed to separate mobile virtual network operator (MVNO) from service provider SP in [8]. The problem of virtual resource allocation was modeled as a stochastic game where stockholders compete for network resources in a sequential manner. *Zaki et al.* studied wireless network virtualization in long term evolution (LTE) systems extensively in [9]. *Kokku et al.* designed a two-layer NVS and implemented it, in which slice scheduling and flow scheduling schemes were proposed based on the priority and achievable data rate of a slice [5]. In [10], the authors studied resource reservation in LTE networks where radio resource was reserved for each tenant and an admission control policy was based on resource availability of the network slice. In the proposed scheme, resources were shared among tenants based on their traffic load and priorities. *Panchal et al.* studied various resource sharing techniques and tested them via simulation in LTE network [11]. In [12], a slicing controller was designed to perform resource allocation to balance load among multiple virtual operators. A resource virtualization framework was proposed in [13], where resources are allocated to users from multiple mobile network operators connected together. In [14], authors also developed a resource allocation scheme that dynamically allocated the resources of INP to SP depending on the agreement between them. The aim of this scheme was to achieve the maximum system throughput and fairness among users. In [15], authors formulated an integer programming model for radio resource virtualization by exploring the benefits of device-to-device communication underlaying LTE networks. Authors in [16] used an SDN approach for sharing resources among MVNOs to provide mobile users with more flexibility to access various services in a network. *Caballero et al.* [17] analyzed the so-called 'share-constrained proportional allocation' mechanism and formulated a network slicing game. This mechanism enabled a user to maximize its own utility. *Tseliou et al.* [18] proposed a negotiation-based slicing method for heterogeneous cellular network to deal with the traffic load variations in geographical dimension. In this way, the base station (BS) with fewer radio resources can borrow extra resources from other BSs. Authors combined physical radio resources to create virtual wireless links and allocated services in a form of capacity to tenants with different priorities and service-level agreements (SLAs) in [19]. *Jiang et al.* tried to increase user satisfaction requirements in slices and increased network revenue in cloud RANs using an auction approach [20]. Network slicing management and prioritization in 5G mobile system was presented in [21]. *NetShare*, as a network-wide radio resource management framework, was proposed for fairness sharing in the RAN [6]. In this paper, the resource distribution among the BSs is proportional to the network traffic distribution. *C. Liang* and *F. R. Yu* proposed distributed virtual resource allocation in virtualized wireless cellular networks based on the alternating direction method of multipliers (ADMM), which is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which is then easier to handle [4]. The objective of the optimization was maximizing profit. *Habiba and Hossain* conducted a literature survey on the economic aspects of wireless network virtualization and study auction theory as a fundamental tool to design business models for virtualization of wireless networks, 5G cellular networks in particular [22].

Due to the versatile characteristics of applications, there were various models that estimated the QoE for certain applications and reflected the level of user satisfaction. In the 5G mobile networks, more advanced scenarios envision transfer of high-rate mission-critical traffic flows. Against this background, *Petrov et al.* introduced softwarized 5G architecture for end-to-end reliability of the mission-critical traffic flows [23]. Hap-sliceR, a reinforcement learning technique based network-wide resource slicing was proposed by *Aijaz*, to dynamically respond to the changes in radio environment especially for haptic communications [24]. Because of the definition of state and action space in resource block (RB) level, the problem becomes complicated and not practical to solve.

Based on our knowledge and literature review, there is no autonomous resource provisioning based virtualization which allows each slice to independently adjust its resource allocation, although learning based resource embedding for virtualization of nodes and links in wired network and cloud datacenters has been proposed in [25], [26] and [27]. In this paper, we propose a novel dynamic reservation of unused resource for each slice, so that they can independently decide their resource allocation based on reinforcement learning technique.

III. SYSTEM MODEL

A. VIRTUALIZATION MODEL

We refer to different applications in the form of virtual networks which occupy a portion of the network resource as slices. SPs have a contract with the INP for one slice of the network with guaranteed end-to-end requirement in terms of rate, maximum tolerable delay, and reliability. The minimum resource requirements of the slices are made known to the BSs and each BS initially allocates fractional resource to each slice. Then the BSs provide the initial reservation of the INP's resource to the existing slices in proportion to their minimum resource requirements. Autonomous slice resource allocation follows after the initial resource reservation. The slices autonomously update the utilization of their virtualized resource. Then the INP calculates the unused resource and reserves them back to the slices that may be in need of additional resource.

Allocating resources to slices is done by monitoring the status of the slices based on their predefined SLAs. The weight given to the slice is based on the SLA between virtual networks and INP. We assume that, an access controller in the RAN has an overview of the existing slices, their SLAs and which flow belongs to which slice. SLA enforcement takes place by adapting the QoS classes of individual flows. For example, if the SLA of a slice is achieving low latency and high reliability e.g. virtual reality, the service data adaptation protocol (SDAP) sublayer maps any flow of this slice to a corresponding QoS class of index 69 [28]. The SDAP sublayer is configured through radio resource control (RRC) signaling, and the SDAP sublayer is responsible for mapping the QoS flow to the corresponding dedicated radio bearer (DRB). One or more QoS flows can be mapped to the same DRB, and one QoS flow can only be mapped to one DRB. The slice-aware resource allocation is a dynamic process which solves conflicts between slices in a way that all SLAs can be fulfilled [29]. The performance of our slicing scheme is checked by resource utilization and slice QoS satisfaction with a time-varying number of users. It is expected that the performance of one virtual network does not affect the other, therefore performance isolation for QoS is very important [30]. In this paper, resource is isolated in terms of fraction of bandwidth allocated to a specific BS or slice. The overall resource pool is naturally partitioned for BSs in the network. At BS level, the resource of a BS is also partitioned to slices to provide BS level resource isolation. The unused resource still reserved for each slice is allocated to new users when they join the slice. There is a need to efficiently map the virtual resource to physical resources. In our work, a fraction of the system bandwidth is mapped to physical resource to be allocated to users thus; scheduling of physical resources to users as discussed in the form of PRBs in [30]. The different views of the resources used in the dynamic resource management are illustrated in Fig. 1 and defined as follows:

Unused resource (F): The unused resource F on a BS is the fraction of the BS resource that is not allocated to slices.

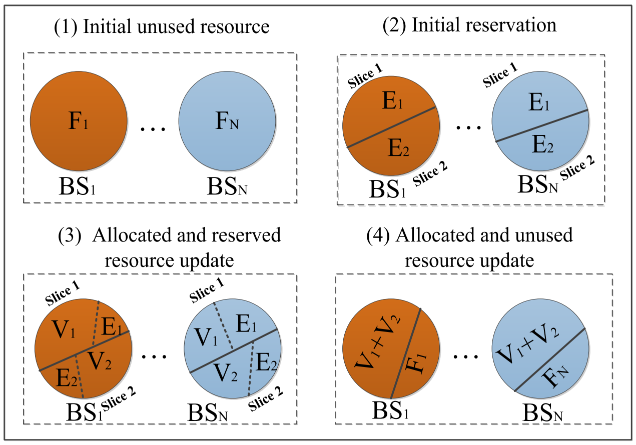
(1) Initial unused resource
F1
FN
BS1
BSN
(2) Initial reservation
Slice 1
E1
E2
Slice 2
(3) Allocated and reserved resource update
V1
V2
(4) Allocated and unused resource update
V1+V2
F1
FN

FIGURE 1. Dynamic resource reservation.

Reserved resource (E): The reserved resource E of a slice at a BS is the fraction of the unused resource of the BS reserved to the slice based on its ratio of resource requirement relative to other slices. The sum of the resource reservations of the slices determines the unused resource of the BS.

Allocated resource (V): The allocated resource V of a slice on a BS is the fraction of the BS resource that is currently allocated and being used by the slice. The sum of the resource allocation fractions of the slices on a BS is less than or equal 1. The sum of the resource allocation and resource reservation fractions of all slices on a BS is equal to 1.

As illustrated in Fig. 1, each BS initially has an amount of unused resource F. When a slice is partitioned, an initial amount of resource E is reserved to it among all of BSs based on the minimum resource requirement of the slice, which is known to the BSs initially. Then, a fraction of the reserved resource, V is allocated to the slice to be shared among its users in the slice leaving a portion of the reserved resource, E for future autonomous resource adjustment. The overall resource of a slice among all of BSs is the sum of the allocated resource V and reserved resource E for the slice after slice resource allocation and reservation update. The unused resource on a BS F is the sum of the reserved resource E for all slices on the BS.

B. NETWORK MODEL

A general single-input-single-output (SISO) downlink cellular network is considered in our system assuming perfect channel estimation and perfect frequency and time synchronization. In the network, a set of small-cell BSs is denoted by $N = \{1, 2, \ldots, |N|\}$. The transmit power of BS n is denoted by P_n. A set of slices is denoted as $M = 1, 2, \ldots, |M|\}$ and a set of total users is denoted as $K = \{1, 2, \ldots, |K|\}$. A set of users of a specific slice m is denoted by K_m, and k_m denotes a single user of the slice. In this paper, we consider bandwidth resource management. Each slice has its own defined QoS requirements. In defining services or slices, we consider the QoS class identifier (QCI) index table in [28]. QCI indexing is a mechanism to ensure that traffics achieve their required QoS. We classify our slices as delay constrained, rate constrained, rate and delay constrained and rate and delay non-constrained slices. We define the delay constrained slice as reliable low latency service e.g. virtual reality and the rate constrained slice as a live streaming service. The delay constrained slice has a requirement of achieving high reliability and low latency by minimizing the sum of delay of users in the slice while the rate constrained slice is concerned with achieving high throughput by maximizing the sum of data rate of the users in the slice. For rate and delay constrained slice, we consider real-time gaming service. Finally, we define a rate and delay non-constrained slice as buffered streaming service. For the rate and delay constrained slice, the multi-objective function has an option to consider sum of data rate and sum of delay. Lastly, the rate and delay non-constrained slice does not have stringent constraints on rate and delay. In such a slice, only the aggregated capacity of the slice is important to the virtual operator. The INP sells the capacity to the virtual operator but is not in charge of its management. The aggregated capacity of the slice is acquired with respect to demand based on SLA between the virtual operator and the INP. In general, the INP is only in charge of this slice and this slice will not be affected by other slices [30]. Without loss of generality, we consider the QoS constraints for delay constrained slice and rate constrained slice in the following sequel.

The minimum data rate requirement of user k in the rate constrained slice m is denoted by $c_{k_m}^{min}$ in terms of packets per second. Assuming user k in slice m occupies the whole resource of BS n, the average achievable data rate of the user k_m from BS n is calculated as follows;

$$c_{k_m,n} = B \cdot log_2 \left(1 + \frac{P_n \left| h_{k_m,n} \right|^2}{\sum_{l \in N, l \neq n} P_l \left| h_{k_m,l} \right|^2 + \sigma^2} \right) \quad (1)$$

where σ^2 denotes thermal noise power at the user, $h_{k_m,n}$ denotes the channel gain and B denotes system bandwidth of BS n in Hz. We assume a user can be associated with multiple BSs. Assuming a fractional resource allocation, the achieved data rate of user k in slice m is calculated as;

$$r_{k_m} = \sum\nolimits_{n \in N} y_{k_m,n} \dot{c}_{k_m,n} \quad (2)$$

where $\dot{c}_{k_m,n} = \frac{C_{k_m,n}}{L_m}$ is the normalized average achievable rate from BS n with respect to the packet size $L_m \cdot y_{k_m,n}$ is the fractional resource allocated to user k of slice m on BS n. In order to set up a delay model for traffics with queuing model, at first, we make the following assumptions: (1) The inter-arrival times between the packets for a user are independent and exponentially distributed with mean $1/\lambda_{k_m}$ seconds, where λ_{k_m} is the packet arriving rate of the user; (2) The lengths of the packets for slices are independent but for user, they are common in slice m as L_m bits. Based on the above assumptions, we can define our delay model for traffics. Clearly, the queuing model of slice for the traffic to user is an M/M/1 queue. Accordingly, the average delay τ_{k_m} experienced by a packet of user k of slice m is

$$\tau_{k_m} = \frac{1}{r_{k_m} - \lambda_{k_m}} \tag{3}$$

where λ_{k_m} is packet arriving rate of the user in packets per second, and r_{k_m} is the normalized user achievable rate with respect to the average packet length.

C. UTILITY MODEL

In this paper, we define the QoS utility model and the resource utilization model to check the utility statistics of slices and users. The QoS utility is used to check whether the QoS of the users is satisfied or not. The resource utilization model is used to check if the data rate capacity or the packet arrival rate of the user is utilized. We define QoS utility as a value in the range of [0, 1] that indicates the satisfaction level of the user. If the QoS constraint is violated, the utility is zero, otherwise, the value is greater than zero.

Definition 1: QoS utility of delay-constrained users

Mathematically, the delay utility of a delay constrained user in the slice is given as [21];

$$U^d_{k_m} = \begin{cases} -b_1 tan^{-1} \\ \left(b_2\left(10^3 \times \tau_{k_m} - b_3\right)\right) + b_4, & for\ 0 \le \tau_{k_m} \le \tau^{max}_{k_m} \\ 0, & otherwise \end{cases} \tag{4}$$

where b_1, b_2, b_3 and b_4 are constants used to customize the shape of the utility curve, $U^d_{k_m}$ denotes the delay utility of user k in the delay constrained slice and $\tau^{max}_{k_m}$ is the upper bound delay of the user expressed in seconds. We can choose the values of the constants by trial and error to find the best utility curve for a given delay range.

The average delay utility of the users of the delay constrained slice is calculated as

$$U^d_m = \frac{1}{K^T_m} \sum\nolimits_{k_m \in K_m} U^d_{k_m} \tag{5}$$

where U^d_m is the average delay utility of the delay constrained slice and K^T_m is the total number of users of slice m.

Definition 2: QoS utility of rate-constrained users.

The rate utility of user k of the rate constrained slice,$U^c_{k_m}$ is calculated as [16];

$$U^c_{k_m} = \begin{cases} \frac{2}{1+e^{-b_5\left(r_{k_m}-b_6\right)}} - 1, & if\ r_{k_m} \ge c^{min}_{k_m} \\ 0, & otherwise \end{cases} \tag{6}$$

where b_5 and b_6 are constants used to customize the utility curve and $c^{min}_{k_m}$ is the minimum data rate constraint of the user in packets per second. The average rate utility of the rate constrained slice is determined by

$$U^c_m = \frac{1}{K^T_m} \sum\nolimits_{k_m \in K_m} U^c_{k_m}. \tag{7}$$

The resource of a slice is a fraction of the bandwidth allocated to it at the BS. The resource allocated to a user also

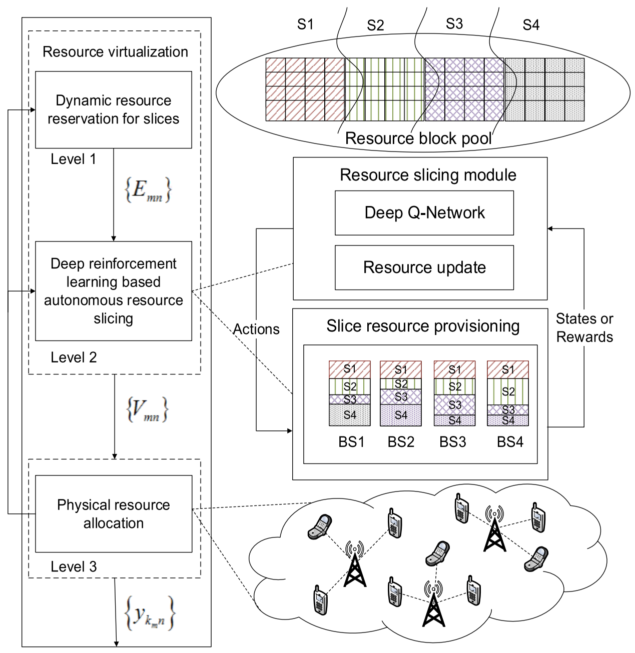

FIGURE 2. System framework.

refers to the data rate capacity of the user. Resource utilization in this paper is defined as the usage of the data rate capacity for the packet arriving rate of the user, i.e. the data rate capacity is equivalent to the packet arrival rate of the user. The resource utilization is high if the allocated resource is closer to the packet arrival rate of the user. The resource utilization decreases as the data rate capacity increases. For the delay constrained user, the resource utilization is determined by dividing the minimum resource required to satisfy the delay constraint of the user by the data rate capacity of the user. The minimum resource required to satisfy the upper bound delay for the delay constrained user can be obtained from equation (3). After rearranging, $r_{k_m} = r^{min}_{k_m} = \lambda_{k_m} + 1/\tau^{max}_{k_m}$. We can generally use a minimum of 1 and the calculated utilization to limit the utilization in the range between [0, 1].

Definition 3: Resource utilization of delay-constrained users.

Mathematically, the resource utilization of user k in the delay constrained slice can be expressed as

$$R^d_{k_m} = \begin{cases} min\left(1, \frac{r^{min}_{k_m}}{r_{k_m}}\right), & \text{if } 0 \le \tau_{k_m} \le \tau^{\max}_{k_m} \\ 1 & otherwise \end{cases}, \tag{8}$$

where $R^d_{k_m}$ is the resource utilization of user k in the delay constrained slice.

The average resource utilization of the delay constrained slice is defined as;

$$R^d_m = \frac{1}{K^T_m} \sum\nolimits_{k_m \in K_m} R^d_{k_m}. \tag{9}$$

Definition 4: Resource utilization of rate-constrained users

The resource utilization of a user in the rate constrained slice is simply calculated as the minimum of 1 and the ratio of the data rate capacity and packet arrival rate as follows;

$$R^c_{k_m} = \begin{cases} min\left(1, \frac{\lambda_{km}}{r_{km}}\right), & \text{if } r_{k_m} \geq c^{\min}_{k_m} \\ 1 & otherwise \end{cases}, \tag{10}$$

where $R^c_{k_m}$ denotes the resource utilization of user k of the rate constrained slice and $c^{min}_{k_m}$ is the minimum data rate requirement of the user in packets per second. The average resource utilization of a rate constrained slice is calculated as

$$R^c_m = \frac{1}{K^T_m} \sum\nolimits_{k_m \in K_m} R^c_{k_m}. \tag{11}$$

IV. PROBLEM FORMULATION

In this section, we perform the entire resource allocation in three stages. In the first stage, the controller reserves unused resources for the slices considering the minimum resource requirement ratios of each slice in the network. In the second stage, radio resources are dynamically allocated to slices according their weights using a deep reinforcement learning based algorithm. The resource update is executed for each slice on each BS according to the resource demand of each slice on each BS. Finally, a flexible intra-slice physical resource allocation is formulated to maximize the rate or minimize delay based on the QoS requirement of the users. The QoS utility and resource utilization are generated based on the reward calculation.

A. DYNAMIC RESOURCE RESERVATION

To mitigate signaling burden when the demand of users at each BS for a slice increases, a resource reservation scheme is adopted, as summarized in Algorithm 1. The controller which has a global view of the network performs the resource reservation. In this paper, slice resource is dynamically reserved based on importance of the BSs for a slice. Based on traffic characteristics and QoS requirements of users at a BS, the controller can determine the minimum resource requirement for a slice. Therefore, at any time, the controller has to determine the estimated minimum resource requirements of the user in order to dynamically reserve resources for the slice. The importance of BS n to user k in slice m is defined as a weight $I^{user}_{k_m,n}$ with the sum of the weights of all BSs to a specific user being equal to one. Mathematically, the weight $I^{user}_{k_m,n}$ of a BS to a user can be expressed as;

$$I^{user}_{k_m,n} = \frac{c_{k_m,n}}{\sum_{n \in N} c_{k_m,n}}, \forall k, m, n. \tag{12}$$

where $c_{k_m,n}$ is the average achievable data rate of the user k_m from BS n.

In order to calculate the estimated minimum resource requirements of the user $c^{min}_{k_m,n}$ on the BSs, we multiply the minimum rate requirement ($r^{min}_{k_m}$) of the user by the weights ($I^{user}_{k_m,n}$) of the BSs to the user,

$$c^{min}_{k_m,n} = I^{user}_{k_m,n} r^{min}_{k_m}, \quad \forall k, m, n, \tag{13}$$

In this paper, we consider two types of users, rate constrained users and delay constrained users. For the rate constrained user, $r^{min}_{k_m}$ is simply $c^{min}_{k_m}$. But for the delay constrained user, $r^{min}_{k_m} = \lambda_{k_m} + 1/\tau^{max}_{k_m}$. Let $I^{BS}_{m,n}$ be the resource reservation weight of slice m at BS n. $I^{BS}_{m,n}$, is defined as the ratio of the minimum resource requirements of the slice m to other slices on a specific BS n. Mathematically, $I^{BS}_{m,n}$ is expressed as:

$$I^{BS}_{m,n} = \frac{\sum_{k_m \in K_m} c^{min}_{k_m,n}}{\sum_{m \in M} \sum_{k_m \in K_m} c^{min}_{k_m,n}}, \forall m, n. \tag{14}$$

The weight of a slice is used for normalization in resource slicing and reservation on BS level and system level, which is different with QoS priority in scheduling. The QoS priority of each slice can be seen on the QCI index table in [30], which does not depend on data rate requirement. Actually, we assume that each slice has a reservation weight based on the minimum resource requirement in SLA. The slice with a higher weight is allocated more unutilized resource of each BS and system instead of a low weight slice. The controller assigns a portion of the unused resource of the network on BS n to the slice m . The reserved resource for a slice m, on BS n $E_{m,n}$ is written as:

$$E_{m,n} = F_n I^{BS}_{m,n}, \quad \forall m, n, \tag{15}$$

where F_n is the total amount of unused resource at BS n.

B. AUTONOMOUS RADIO RESOURCE MANAGEMENT

After the initial resource for the slice has been reserved on each BS by the controller, the slices can autonomously adjust their own resource in order to maximize the QoS utility and resource utilization for their users. The main objective of slice is to ensure that all users are satisfied with limited amount of resource assigned to the slice. The partitioning of resource on the BS ensures fractional bandwidth reservation for slices based on their requirements. The fraction of bandwidth can be mapped to physically isolated RBs [30]. A slice cannot occupy more resources than the total amount of its reserved resource assigned by the slicing controller. This is due to the fact that, mutual interdependence of slices may cause performance degradation.

Reinforcement learning is a class of machine learning techniques where an agent is trained for decision-making mechanism by interacting with an environment through action with the objective of maximizing its cumulative reward. It is defined by the behavior of Markov decision process (MDP), for problems whose probability of the problem entering next state depends only on the current state and action selected. The environment is in a certain state and is changed to next state by an action from the agent, and the agent gains reward for that transition. Q-learning and deep Q-learning are examples of reinforcement learning. The Q-learning is mostly used for problems with discrete state space. Since the virtual resource fraction is a continuous value in the range between [0, 1], we choose deep Q-learning framework and make the decision for any combination of values in this range.

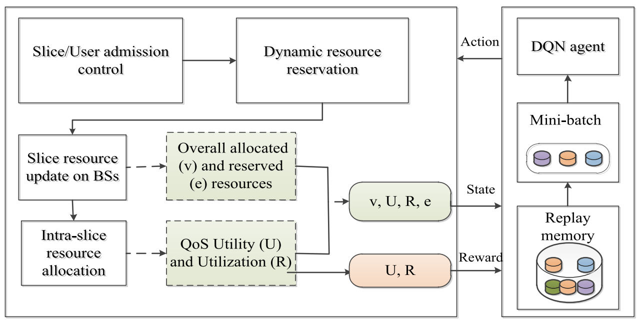
Slice/User admission control
Dynamic resource reservation
Action
DQN agent
Mini-batch
Slice resource update on BSs
Overall allocated (v) and reserved (e) resources
v, U, R, e
State
Replay memory
Intra-slice resource allocation
QoS Utility (U) and Utilization (R)
U, R
Reward

FIGURE 3. **DQN based autonomous radio resource management.**

Deep Q-network (DQN) is formed by substituting the Q-function of the Q-learning by an artificial neural network.

In this section, we formulate the resource slicing problem as an MDP, and then demonstrate how to solve this problem under a deep Q-learning framework. First, we define the state, action, reward, and next state of our DQN. In Figure 3, we illustrate the proposed DQN framework for autonomous resource management.

Deep reinforcement learning: In reinforcement learning, a software agent learns from the environment as interacts with it and receives a reward or a penalty based on the decision it makes. These decisions are known as actions. The agent either receives a reward for making a good decision or a penalty for making a bad one. The goal of the agent is to maximize the cumulative reward through the actions is takes. Reinforcement learning is referred to as deep reinforcement learning when an artificial neural network is used as a function to map discrete states to discrete actions. In this paper, we use a deep Q-learning agent [28].

State(s): The current system state $s(t)$ is determined by the states of the BSs. The system state at time slot t is defined as $s(v, U, R, e)$, where v is the overall allocated virtual resource fraction, U is the average QoS utility, R is the average resource utilization and e is the overall resource reservation for the slice. The proposed DQN model in this paper takes action at the slice level, therefore, we have to aggregate the BS level resource allocations of slices as the overall resource allocation v_m.

Since a specific user on multiple BSs has a weight $I^{user}_{k_m,n}$ to determine the importance of each BS to the user, we aggregate the weights of all BSs to the users to be equal to the weight $I^{slice}_{m,n}$ of the BS to the slice. As stated above, overall resource allocation and reservation are based on the weights of the slices. Hence the weight $I^{slice}_{m,n}$ of a BS to a slice, is calculated as follows;

$$I^{slice}_{m,n} = \frac{\sum_{k_m \in K_m} c^{min}_{k_m,n}}{\sum_{n \in N} \sum_{k_m \in K_m} c^{min}_{k_m,n}}, \forall n, m, \tag{16}$$

In proportion to $I^{slice}_{m,n}$, the overall resource allocation of the slice is calculated as

$$v_m = \sum\nolimits_{n \in N} I^{slice}_{m,n} V_{m,n}, \forall m, \tag{17}$$

Algorithm 1 Dynamic Resource Reservation

```
1   Initialization
2   Set the allocated $\mathbf{V}$ = zeros($m$, $n$) for all
      slices on all BSs, the reserved $\mathbf{E}$ = zeros($m$, $n$) for all
      slices on all BSs, the unused $\mathbf{F}$ =[1,1,...,1] for the
      BSs;
3   Set T ← 0
4   **Iteration**
5   Calculate the weights $I^{slice}_{m,n}$ of BSs to each slice by (16);
6   **If**(T == 0)
7   Collect the minimum resource requirements of each
      user of the slice by (13) and sum up them to find the
      minimum requirement of the slice;
8   Calculate the reservation weight $I^{BS}_{m,n}$ by (14) and the
      initial slice resource reservation $E_{m,n}$ by (15);
9   **Else if**
10    Collect the updated resource allocation $\bar{V}_{m,n}$ from all
      slices by (23);
11    **End if**
12    Update the unused resource $F_n$ on each BS by (24);
13     **If** (traffic statistics changed)
14    Update the reservation weight $I^{BS}_{m,n}$ by (14) and $I^{slice}_{m,n}$
      by (16) for all slices in all BSs;
15    Update the slice resource reservation $E_{m,n}$ by (15);
16    **End**
17    Output $v_m$ and $e_m$ by (17), (18) to DQN;
18    T← T + 1;
19    **End**
```

where $V_{m,n}$ is the resource allocated to the BS and $I^{slice}_{m,n}$ is the weight of the BS n at a slice m. Similarly, the overall resource reservation e_m, is calculated in the same way as

$$e_m = \sum\nolimits_{n \in N} I^{slice}_{m,n} E_{m,n}, \quad \forall\, m \tag{18}$$

Till now, both the overall resource allocated and the overall resource reserved for each slice is normalized in a range between 0 and 1.

Reward (w): The reward w is defined as the sum of average QoS utility and average resource utilization of the slice.

$$w_m = \beta U_m + (1 - \beta) R_m, \forall m, \tag{19}$$

where $\beta \leq 0.5$ denotes the weight of resource utilization. R and U are QoS utility and resource utilization respectively defined in equations (4)-(11). In order to set a limit on the amount of radio resources assigned to a slice, resource utilization is always given a priority higher than that of QoS utility.

Action(a): The actions are a set of discrete percentages A= {-90%...-50%, -40%, -30%, -20%, -10%, 0, 10%, 20%, 30%, 40%, 50%..., 90%}. A negative value indicates a decrease in resources and positive value represents an increase in resource of the slice. A positive action is identified by its index in the neural network. The mapping between the index and a specific action percentage is defined as

$a = index/10 - 1$, where a is the action percentage and $index$ of the action set begins from 1.

Next state(s'): The action will change the state of the slice to a new state. The new QoS utility and the resource utilization of the next state are used to calculate the reward as shown in equation (19). The INP will configure the new resource provisioning of the slices on the BSs and update the unused resource by equation (24) as well as reserve unused resource for each slice according to equation (15).

The Q-value update: Let the set of actions be A and s, a, w, s' be the current state, action, reward and next state. The Q-value of a state-action pair is defined by Bellman equation as:

$$Q(s, a) = w + \gamma_{a' \in A}^{max} \left(Q\left(s', a'\right)\right), \tag{20}$$

where γ is a discount factor. The discount factor, $\gamma \in [0, 1]$, defines the current weight of the current reward on future reward.

ANN Configuration: We configure a feed-forward neural network of 4 inputs and 20 outputs because the state has 4 variables and the actions are 20. There is no known rule for determining the number of hidden layers and neurons. It is better to choose these sizes by trial and error on their performance. We choose two hidden layers with 18 neurons each. We arbitrarily set the sigmoid activation function for the hidden layers but positive linear (poslin) for the output layer because the Q-values are positive.

DQN Algorithm: The proposed DQN algorithm is based on Google's DeepMind in [31]. In the proposed algorithm, testing and training are simultaneous. In the test phase, the agent takes an action for the given state $s(v, U, R, e)$, of the slice either through exploration or exploitation. The agent explores the environment in order to find new actions. At every slicing time, the state, action and reward are stored in replay memory. After the agent makes a decision on the slice, it will update the resource allocation of the slice and calculates new state for the next decision epoch. In the training mode, the agent selects some past experience stored in the experience replay memory. The sample of past experience in which the agent selects from is known as the mini-batch. We use a batch size of 1000 samples at a time. Due to the large state-action pairs of the network, the Q-values are approximated based on the mini-batch samples. Training can occur in background and is independent of the slicing periods. In our simulation, we activate training when five new rows have been added to the replay memory. The proposed algorithm is illustrated in Fig. 3 and summarized in Algorithm 2.

Replay memory (D): The replay memory has fixed size D where entries are stored based on first-in-first-out (FIFO) order. Each row of the replay memory stores the state of the slice, the action enforced for that state and the reward gained from taking the action for that state of the slice. The replay memory has 10,000 rows for such combinations and 6 columns. Four of the columns store the overall resource

Algorithm 2 DQN

```
1  Set a replay memory size D and mini-batch size D';
2  Configure neural network with input to the number of
   variables in the state and output number of actions;
3  **Initialize** Q-network with random weights;
4  Choose epsilon value $\varepsilon$;
5  **While (network active)**
6  Collect state $< v, R, U, e >$;
7  Generate $a$ random number $\pi$;
8  **If** $\pi < \varepsilon$
9  Choose an action randomly;
10   **Else**
11   Input the state to the ANN and choose the action that
     has maximum Q-value;
12   **End If**
13   Update resource by (21) and calculate reward by (19);
14   Update the required allocation $V_{m,n}$ by (23) in all BS;
15   Convert $V$ and $E$ to overall $v_m$ and $e_m$ by (17) and
     (18);
16   Store $< v, R, U, e, a, w >$ in replay memory;
17   **If** (activated by timer)
18   Pick a mini-batch of samples from the replay
     memory;
19   **For** all of samples
20   Calculate output Q-values by ANN;
21   Update target Q-value of the action by (20);
22   Train ANN by a tuple of (input, output Q-value, and
     target Q-value) with a loss function as the mean
     square error of output and target;
23   **End for**
24   **End if**
25   **End while**
```

allocation v, the QoS utility U, the resource utilization R and the overall resource reservation of the state e. The fifth and sixth columns store the action and reward respectively. Initially, the replay memory is empty. Every time the DQN makes a decision, one row is stored in the replay memory. When the memory is full, old rows are overwritten by new data. Due to computational cost, the DQN takes only one of the data samples for training at a time. The sample is of size in mini-batch, set as 1000 in our simulation, which is taken randomly from the replay memory, so that the training is not always subjected to the recent states. The size of the mini-batch is also chosen based on trial and error. If the mini-batch size is very small, the DQN will always be sensitive to the behaviour of the last mini-batch used for training. However, a larger mini-batch size increases the computational cost.

Resource update: The slice updates its resource allocation by the action of the agent. Since the action is decided at the slice level, we need to convert the BS level resource reservation and resource allocation to overall fraction using equation (17) and equation (18). Given $v_m^{(t)}$ in slice level at slicing time t, we update the overall resource $v_m^{(t+1)}$ at slicing time $t+1$ as follows

$$v_m^{(t+1)} = \begin{cases} v_m^{(t)}, & if\ a_m = 0 \\ (1+a_m)\, v_m^{(t)}, & if\ a_m < 0\ , \\ v_m^{(t)} + a_m e_m^{(t)}, & if\ a_m > 0 \end{cases} \tag{21}$$

where a_m is the action for the slice. After resource update at the slice level, the update has to reflect at all the BSs. Hence, slice level resource update is converted back to BS update. If we define $V_{m,n}^{(t)}$ as the resource allocation of slice m at BS n at slicing time t, $V_{m,n}^{(t+1)}$ is the updated resource allocation of slice m at BS n at slicing time $t+1$.

Proposition 1: BS level resource update

Given the maximum attainable resource $\bar{V}_{m,n}^{(t)}$ of slice m on BS n at slicing time t and the overall resource allocation $v_m^{(t+1)}$ of slice m at slicing time $t+1$, the BS level resource allocation update is expressed as $V_{m,n}^{(t+1)} = x_m \bar{V}_{m,n}^{(t)}$, where $x_m = \frac{v_m^{(t+1)}}{\sum_{n\in N} I_{m,n}^{slice} \bar{V}_{m,n}^{(t)}}$.

Proof: To update the BS level resource with the given $v_m^{(t+1)}$ in slice level at slicing time $t+1$, we define the maximum attainable resource $\bar{V}_{m,n}^{(t)}$ of the slice at slicing time t at each BS by adding the allocated resource $V_{m,n}^{(t)}$ and the reserved resource $E_{m,n}^{(t)}$ of the slice m at BS n at slicing time t as follows:

$$\bar{V}_{m,n}^{(t)} = V_{m,n}^{(t)} + E_{m,n}^{(t)} \quad , \forall\, n, \tag{22}$$

As discussed in sub-section IV A, the maximum attainable resource of slice m on BS n at slicing time t, denoted by $\bar{V}_{m,n}^{(t)}$, should match the resource requirement of the slice. We assume the updated overall resource at slicing time $t+1 v_m^{(t+1)}$ from $I_{m,n}^{slice}$ and an unknown coefficient x_m to update the resource allocation at BS level at slicing time $t+1$ as

$$V_{m,n}^{(t+1)} = x_m \bar{V}_{m,n}^{(t)}, \forall n \tag{23}$$

i.e. we take a proportional resource fraction from the maximum attainable resource of the slice at each BS until their sum reaches the overall resource allocation. From equations (17) and (18), the updated overall resource at slicing time $t+1$ can be expressed as;

$$\begin{aligned} v_m^{(t+1)} &= \sum\nolimits_{n\in N} I_{m,n}^{slice} V_{m,n}^{(t+1)} \\ &= (\sum\nolimits_{n\in N} x_m I_{m,n}^{slice} \bar{V}_{m,n}^{(t)}) \\ &= x_m (\sum\nolimits_{n\in N} I_{m,n}^{slice} \bar{V}_{m,n}^{(t)}) \end{aligned} \tag{24}$$

Therefore,

$$x_m = \frac{v_m^{(t+1)}}{\sum_{n\in N} I_{m,n}^{slice} \bar{V}_{m,n}^{(t)}}. \tag{25}$$

In this way, given a known value of x_m, the resource allocation update at BS level is achieved. The formula for finding x_m can be re-written using equation (21) as follows;

If $a_m \le 0$, $x_m = \frac{v_m^{(t+1)}}{\sum_{n\in N} I_{m,n}^{slice} \bar{V}_{m,n}^{(t)}} = \frac{(1+a_m) v_m^{(t)}}{v_m^{(t)} + e_m^{(t)}}, \quad x_m < 1;$

If $a_m > 0$, $x_m = \frac{v_m^{(t+1)}}{\sum_{n\in N} I_{m,n}^{slice} \bar{V}_{m,n}^{(t)}} = \frac{v_m^{(t)} + a_m e_m^{(t)}}{v_m^{(t)} + e_m^{(t)}}, \quad x_m > 1.$

If we assume$F_n^{(t)}$ be the total amount of unused resource of BSn at slicing time t, then the unused resource of the BS at time $t+1$ is updated by

$$F_n^{(t+1)} = F_n^{(t)} - \sum_{m\in M} \left(V_{m,n}^{(t+1)} - V_{m,n}^{(t)} \right), \quad \forall n. \tag{26}$$

The reserved resource $E_{m,n}^{(t+1)}$ of slice m on BS n at slicing time $t+1$ is calculated from Eqn. (15) as;

$$E_{m,n}^{(t+1)} = F_n^{(t+1)} I_{m,n}^{BS}, \forall m, n \tag{27}$$

Finally, the INP configures the updated resource of the slice to the BSs and the slice allocates the resource to its users.

C. CUSTOMIZED PHYSICAL RESOURCE ALLOCATION

In this paper, we consider two types of slices i.e. rate constrained slice and delay constrained slice. The intra-slice physical resource allocation is formulated as an optimization problem by minimizing the sum of user queuing delays for the delay constrained slice or maximizing transmission rate for the rate constrained slice respectively. We formulate the problem as a convex optimization problem and solve it with ADMM algorithm, which is adopted in [32]. In this paper, we assume one user can be associated with multiple BSs. Optimization of the rate or delay performance of all users in different slices is performed in the form of $f_{k_m}(.)$. Therefore, the following customized objective function for the delay constrained slice is given as follows:

$$\begin{aligned} &\underset{\{y_{k_m,n}\}}{minimize} \sum\nolimits_{k_m\in|K_m|} f_{k_m}\left(\{y_{k_m,n}\}_{n\in|N|} \right) \\ &subject\ to\ \tau_{k_m} \le \tau_{k_m}^{max} \forall k, m, \\ &\qquad \sum\nolimits_{k_m=1}^{|K_m|} y_{k_m,n} = \bar{V}_{m,n} \forall n, m, \\ &\qquad y_{k_m,n} \ge 0 \forall k, m, n, \end{aligned} \tag{28}$$

where $V_{m,n}$ is the allocated resource of the slice m on BS n, $y_{k_m,n}$ is the resource allocation of user k of slice m in BS n, $\tau_{k_m}^{max}$ is the maximum delay experienced by a user in the slice.

Physical resource allocation for delay constraint slice: For the delay constrained slice, in order to minimize the average packet delay experienced by the user, we choose $f_{k_m}(.) = \lambda_{k_m} \tau_{k_m}$, in which λ_{k_m} is the packet arrival rate and τ_{k_m} is the delay experienced by a user in the slice.

Physical resource allocation for rate constraint slice:

For the rate constrained slice, resource allocation follows the same approach for delay constrained slice. There is rate constraint for each user to ensure guaranteed QoS i.e. $r_{k_m} \ge c_{k_m}^{min}$. $f_{k_m(.)} = -log\left(\lambda_{k_m} \sum_{n\in|N|} y_{k_m,n} c_{k_m,n} \right)$ is adopted as the objective function. $y_{k_m,n}$ is the fraction of resource assigned to the user and $c_{k_m,n}$ is the rate of the user. The goal is to find the optimal $y_{k_m,n}$ such that the users with a higher packet arrival rate experience minimum delay. This problem (29a) is always convex. Considering scalability of the network, we adopt a distributed approach to solve the problem. To solve the problem in a distributed way, we decouple the problem in (28) as user side and BS side as follows:

$$\underset{\{y_{k_m,n},z_{k_m,n}\}}{minimize} \sum_{k_m=1}^{|K_m|} f_{k_m}\left(y_{k_m}\right)+\sum_{n=1}^{|N|}\Phi\left(z_n\right)$$
$$subject\ to\ y_{k_m,n}-z_{k_m,n}=0\ \forall k,m \quad (29)$$

where $\Phi(z_n) = \mathbb{I}_c(z_n)$ and the global variable z_n represents the resource allocation decision made by the BS n. It can be viewed as local copies of resource allocation decision at each BS.

Algorithm 3 Intra-Slice Physical Resource Allocation

```
Initialization
1  Receive the data rate, virtual resource fraction, delay constraint, and arrival rates of the users from the slice
2  initialize $\rho > 0$ $\theta^{(0)}_{k_m,n}$= 0,$\forall k_m$, n
3  initialize $\left\{y^{(0)}_{k_m}\right\}^{|K_m|}_{k_m=1}$ as $y^{(0)}_{k_m} = [0,0,\ldots,0]^T$
4  initialize $\left\{z^{(0)}_n\right\}^{|N|}_{n=1}$ as $z^{(0)}_n = [0,0,..,0]^T$
5  downscale $c_{k_m,n} = \bar{V}_{m,n}c_{k_m,n}\forall k_m,n$
Iteration
6  Set $t \leftarrow 0$
7  For each user
8  Update the resource allocation vector $y^{(t)}_{k_m}$ in all BSs by (31a) or (33), where $\tau^{(t+1)}_{k_m} := \left(\sum_{n=1}^{|N|} y^{(t)}_{k_m,n}c_{k_m,n} - \lambda_{k_m}\right)^{-1}$ and $\mu_1 \geq 0$ are chosen such that $0<\tau^{(t+1)}_{k_m} \leq \tau^{max}_{k_m}$ and $\mu_1(\tau^{(t+1)}_{k_m} - \tau^{max}_{k_m})=0$.
    End for
9  For each BS
10   Update the resource allocation for all the served users of the slice at BS n by (31b)
11   End for
12   Update $\theta^{(t+1)}_{k_m,n} = \theta^{(t)}_{k_m,n} + \rho\left(y^{(t+1)}_{k_m,n} - z^{(t+1)}_n\right)$ by (31c)
13   $t \leftarrow t + 1$;
14   until termination test satisfied
15   return result
```

The above problem can be termed as a consensus problem and can be solved with ADMM [33]. For the consensus problem in (29a), the augmented lagrangian equation with respect to consensus constraints can be formulated as:

$$\begin{aligned}L_p&\left(y_{k_m,n},z_{k_m,n},\mathrm{O}_{k_m,n}\right)\\ &= \sum\nolimits_{k_m=1}^{|K_m|} f_{k_m}\left(y_{k_m}\right)\\ &\quad+\sum_{n=1}^{|N|}\Phi\left(z_n\right)+\sum\nolimits_{n=1}^{|N|}\sum\nolimits_{k_m=1}^{|K_m|}\mathrm{O}_{k_m,n}(y_{k_m,n}-z_{k_m,n})\\ &\quad+\frac{\rho}{2}\sum\nolimits_{n=1}^{|N|}\sum\nolimits_{k_m=1}^{|K_m|}\left(y_{k_m,n}-z_{k_m,n}\right)^2, \end{aligned} \quad (30)$$

where $\theta_{k_m,n}$ is the Lagrange multiplier and ρ is the penalty parameter. The ADMM iteration process for the t^{th} iteration for primal variables and dual variables updates are:

$$\begin{aligned}y^{(t+1)}_{k_m,n} &= argmin\{f_{k_m}\left(y_{k_m}\right)+\sum\nolimits_{n=1}^{|N|}\theta^{(t)}_{k_m,n}y_{k_m,n}\\ &\quad+\frac{\rho}{2}\sum\nolimits_{n=1}^{|N|}\left(y_{k_m,n}-z^{(t)}_{k_m,n}\right)^2\end{aligned} \quad (31a)$$

$$\begin{aligned}z^{(t+1)}_{k_m,n} &:= argmin\{\Phi\left(z_n\right)-\sum_{k_m=1}^{|K_m|}\theta^{(t)}_{k_m,n}z_{k_m,n}\\ &\quad+\frac{\rho}{2}\sum\nolimits_{k_m=1}^{|K_m|}\left(y^{(t+1)}_{k_m,n}-z_{k_m,n}\right)^2\},\end{aligned} \quad (31b)$$

and

$$\theta^{(t+1)}_{k_m,n} = \theta^{(t)}_{k_m,n}+\rho\left(y^{(t+1)}_{k_m,n}-z^{(t+1)}_{k_m,n}\right) \quad (31c)$$

In solving the problem, for rate constrained users where, $f_{k_m}(.) = -log\left(\lambda_{k_m}\sum_{n\in|N|}y_{k_m,n}c_{k_m,n}\right)$, $y_{k_m,n}$ is as follows:

$$y^{(t+1)}_{k_m,n} = \left[z^{(t)}_{k_m,n}-\frac{\theta^{(t)}_{k_m,n}}{\rho}+\frac{\dot{c}_{k_m,n}}{\rho}\left(\mu_1+\frac{1}{\left(\sum_{n=1}^{|N|}y^{(t+1)}_{k_m,n}\dot{c}_{k_m,n}-\lambda_{k_m}\right)}\right)\right]_+. \quad (32)$$

where μ_1 is chosen such that $\sum_{n=1}^{|N|}y_{k_m,n}\dot{c}_{k_m,n} \geq c^{min}_{k_m}$.

For delay constrained users where $f_{k_m}(.) = \lambda_{k_m}\tau_{k_m}$, $y_{k_m,n}$ as follows:

$$y^{(t+1)}_{k_m,n} = \left[z^{(t)}_{k_m,n}-\frac{\theta^{(t)}_{k_m,n}}{\rho}+\frac{\dot{c}_{k_m,n}}{\rho}\left(\mu_1+\frac{1}{\left(\tau^{(t+1)}_{k_m}\right)^2}\right)\right]_+ \quad (33)$$

where $\mu_1 \geq 0$ is chosen such that $0\leq \tau^{(t+1)}_{k_m} \leq \tau^{max}_{k_m}$. The detailed algorithm of the intra-slice resource allocation for delay constrained slice is summarized in Algorithm 3.

V. SIMULATION RESULTS

A. SCENARIO CONFIGURATION

To evaluate the performance of our proposed scheme, we perform numerical simulations and analysis using MATLAB software. The simulation parameters were chosen based on 5G specifications and standards. In a coverage radius of 500m, a macro BS (MBS) is situated at the center with 4 small-cell BSs (sc-BSs) and users uniformly distributed in the area. Each user is assumed to belong to a specific slice. Without loss of generality, we consider two slices with different rate and delay requirements namely; Slice 1 and Slice 2. Slice 1 is defined as a delay-constrained slice whiles Slice 2 is referred to as a rate-constrained slice. For the delay-constrained slice (Slice 1), the values of variable b_1, b_2, b_3 and b_4 which are used to customize the shape of the delay utility curve are 0.5, 0.06, 70 and 0.5 respectively. In Slice 2, the values of variables in the rate utility function b_5 and b_6 are 0.1 and $c^{min}_{k_m}$-0.01 respectively. These values are constants which are determined by shaping the curve to fit the required QoS of the slice. The penalty parameter ρ used

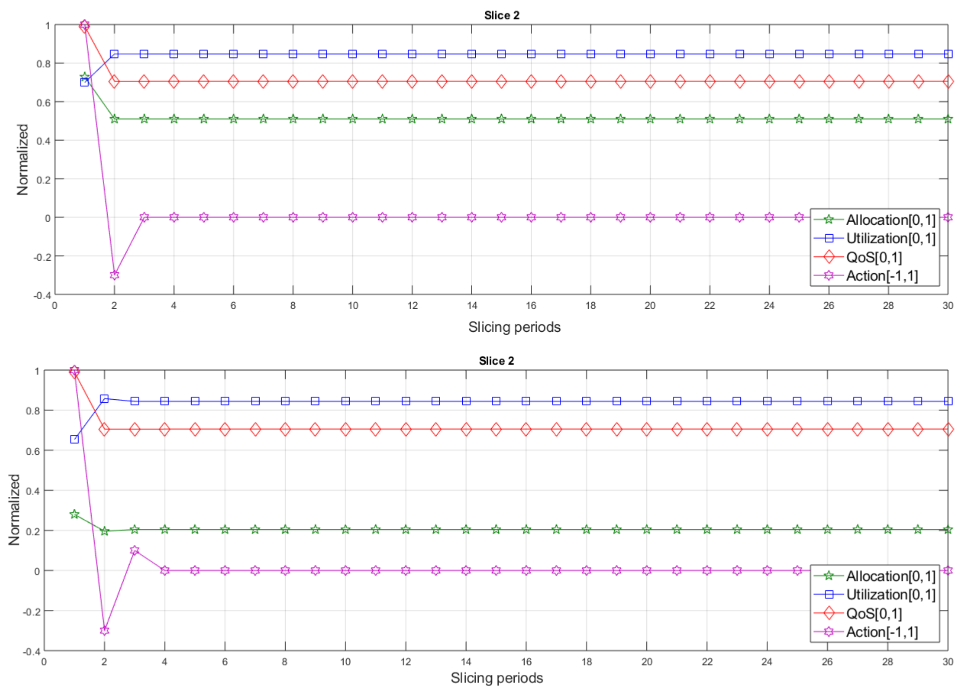

FIGURE 4. Convergence of DQN.

in the ADMM solver is$(max(c_{k_m,n}))^2$. In the DQN algorithm, we set the reward function β to 0.1 and epsilon ε to 0.4. For simplicity, most of the parameters for system and algorithm in the simulation are listed in Table 1. We compare our proposed scheme with two benchmarks, NVS [5] and NetShare [6] on performances of QoS satisfaction and resource utilization for slices. We compare our scheme with NVS because it is a good example of slice-level static resource provisioning. On the other hand, NetShare is a good example of dynamic resource provisioning.

B. CONVERGENCE OF DQN

Unlike supervised learning where labeled data is available, reinforcement learning has no datasets for the agent to learn from at the beginning of the training process. In our proposed DQN algorithm, the Q-values are initialized to zeros at the beginning of the learning process. The network is then trained for a simulation time of one hour until data samples are generated for training and test of convergence. In this simulation, we consider fixed traffic conditions such that, the number of users in Slice 1 and Slice 2 over 30 slicing periods are 197 and 82 respectively. The normalized initial resource allocated to Slice 1 is 0.75 and the remaining 0.25 is allocated to Slice 2. After initial resource allocation, the DQN agent adjusts the allocated resources to the slices by performing a defined action, thus; decreasing the amount of unutilized resource or increasing the amount of over-utilized resource of a slice. From Fig. 4, it can be observed that the overall resource of Slice 1 is decreased from 0.75 to 0.50, approximately 30% decrease at the 2nd slicing period, which corresponds

TABLE 1. Simulation parameters configuration.

Parameters and units	Values
System bandwidth, B	10 MHz
Number of macro cell	1
Number of small cells, N	4
Transmit power, P	30dBm (Small cells); 46dBm (Macro-cell);
Thermal noise spectral density, σ^2	-95dBm
Path loss model	34 + 40 log10*(dBm)*
Number of slices, M	2
Number of users, K	10~400 per slice
Coverage area	1000m x1000m
Average packet length, L	1kb
Packet arrival rates, λ_{k_m}	80-120 *packets per second*
Delay constraint, $\tau_{k_m}^{max}$	100ms with 10^{-7} loss
Rate constraint, $c_{k_m}^{min}$	100 *packets per second*
Slicing period, T	100ms
Size of replay memory, D	10000
Size of mini-batch, D'	1000
Epsilon/ε- greedy	0.4
Learning rate	0.01
Performance goal(MSE)	0.0001
Number of slicing periods	30
Training function	Bayesian

to -30% in the defined action set. Similarly, the QoS utility of Slice 1 decreases at the 2nd slicing period. However, the resource utilization increases at the same slicing period as the resource allocation and QoS utility. After the 2nd slicing period, the DQN algorithm converges to the fixed resource allocation, QoS utility and resource utilization. At convergence, the selected action is 0, meaning there is no decrease or increase in the allocated resource of Slice 1.

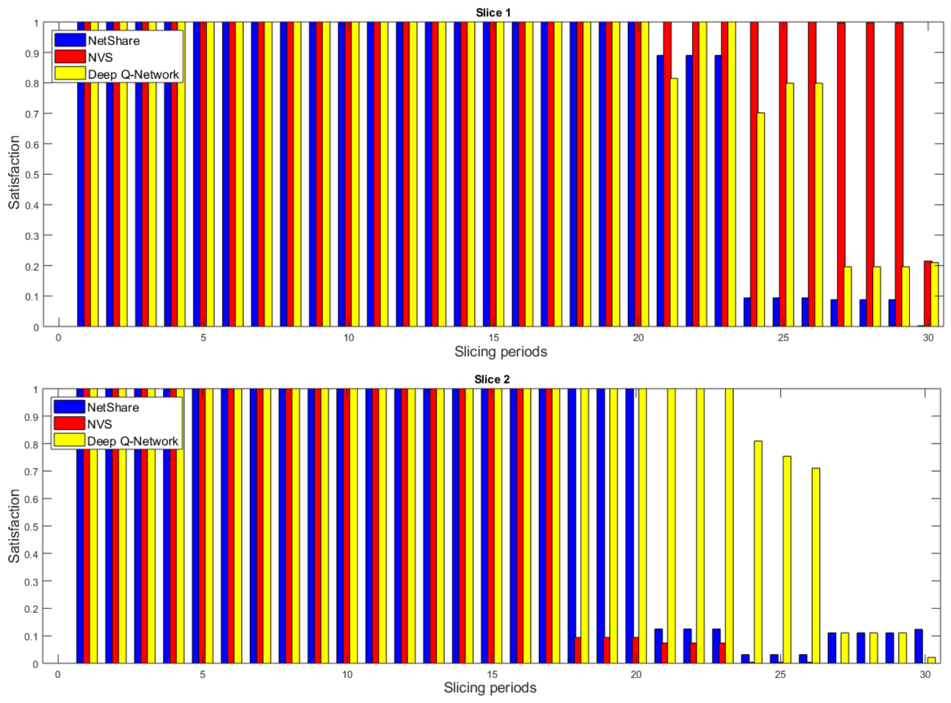

FIGURE 5. Slice satisfaction.

In Slice 2, its initial resource allocated is decreased from 0.25 to 0.20, approximately 20% decrease at the 2nd slicing period, which corresponds to -20% in the defined action set. The QoS utility of Slice 2 decreases at the 2nd slicing period and converges. The resource utilization also increases at the 2nd slicing period before convergence. After the 2nd slicing period, the DQN algorithm converges for fixed resource allocation, QoS utility and resource utilization. It can be concluded that, the DQN algorithm adjusts the resource allocated to the slices during the training phase and finds the optimal action to execute after training for a sufficient period, hence its convergence.

C. PERFORMANCE ON SLICE SATISFACTION

In this simulation, we evaluate the performance of the proposed DQN algorithm in terms of slice satisfaction. To evaluate slice satisfaction, we consider high infeasible load to determine the load limitation of each algorithm in a dynamic environment scenario. We increase the number of users in Slice 1 and Slice 2 by 44 and 36 respectively for every 3 slicing period's interval. Slice satisfaction is defined as the ratio of satisfied users to the total number of users in a slice. Since the users are generated randomly, we run the simulation 20 times to take the worst case scenario for comparing the performance of NVS, NetShare and our proposed DQN based on the responses to changes in the environment conditions and dynamic resource allocation for unbalanced traffic distribution at each BS.

As shown in Fig. 5, NVS satisfied all users in Slice 1 until the 29th slicing period. However, users in Slice 2 cannot be satisfied after the 17th slicing period. This is due to the static resource allocation condition with increasing traffic load in NVS. DQN and NetShare dynamically update their resource allocation based on changes in traffic load from users. Although NetShare supports dynamic network-wide resource allocation, there is no feedback as indicator from the environment to show whether the allocated resource is enough to the slice or not. It can be observed in Fig. 5 that, at the 21st slicing period, NetShare fails to satisfy users of Slice 2. All of users in Slice 1 can be satisfied until the 20th slicing period. DQN satisfies users of Slice 1 and those of Slice 2 fully until the 23rd slicing period. After the 26th slicing period, the satisfaction level of users drops to 0.2 in Slice 1 and 0.1 in Slice 2. The resource utilization feedback that is sent to the DQN agent from the slices enables the adjustment of resource allocated to the slices.

D. PERFORMANCE ON SLICE RESOURCE UTILIZATION

In this simulation, we evaluate the performance of the DQN algorithm based on slice resource utilization. In order to observe the effect of changing traffic load, the same configuration of user population changes is adopted in this simulation. Resource utilization of a slice is the average of the resource utilization of its users as defined in equation (9) and equation (11). As shown in Figure 6, NVS and NetShare demonstrate similar behavior when the number of users increases at every 3 slicing periods. It can be observed that, the DQN algorithm shows an inconsistent behavior at each 3 slicing period intervals due to the increasing number

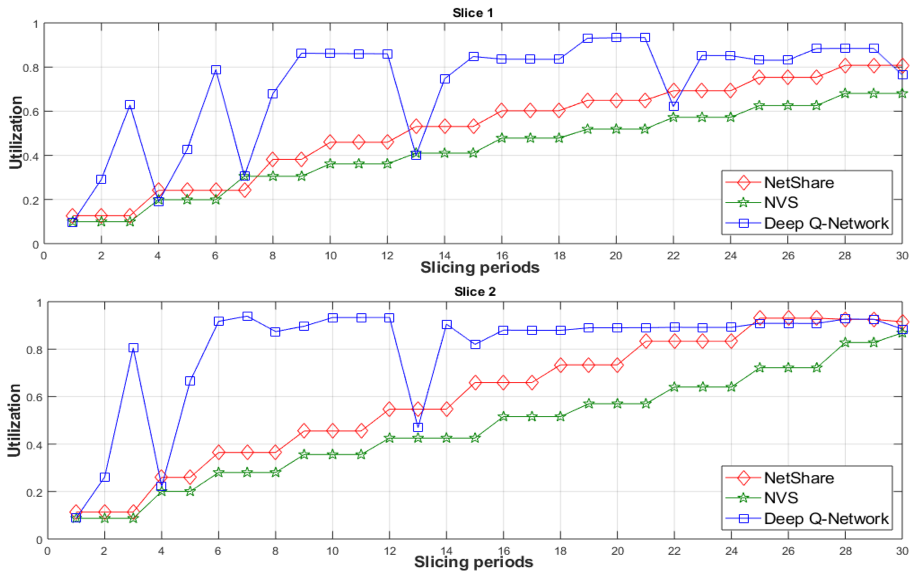

FIGURE 6. Slice resource utilization.

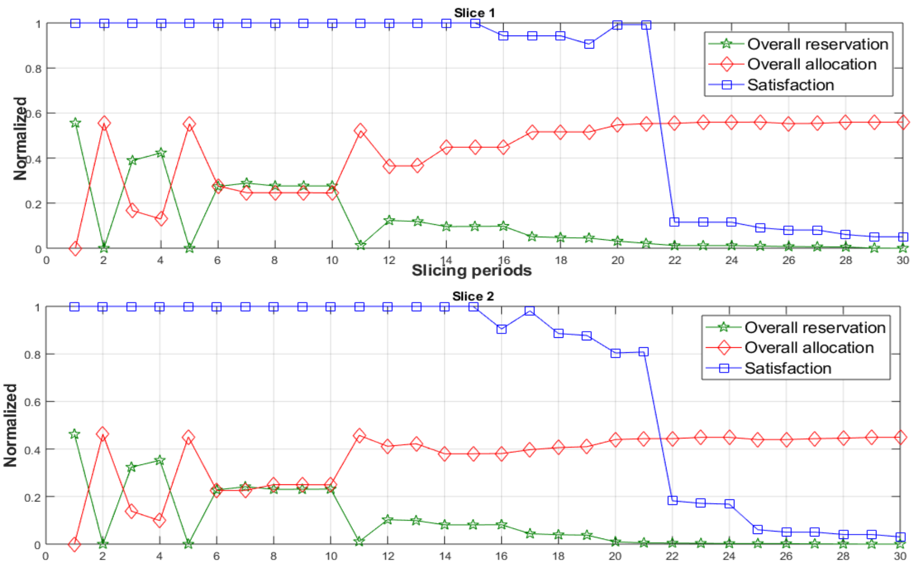

FIGURE 7. Slice satisfaction vs. overall allocation and reservation.

of users at these slicing periods. The DQN algorithm adjusts the resource allocated to the slices by releasing the unused resources based on the QoS utility and resource utilization feedback. With the proposed DQN-based slicing scheme, slices occupy only an essential fraction of system resource that is enough and can be utilized by their users. The fluctuation is introduced at the time we add new users, because the DRL agent does not know the resource requirement of the new users. Thus it does not know how to adjust leaving it with two options, either to allocate resource based on the requirement of the previous users or to use all the reserved resource as well.

E. PERFORMANCE ON OVERALL SLICE RESOURCE RESERVATION AND SATISFACTION

In this simulation, we extend the experiment in sub-section C by plotting slice satisfaction, overall resource allocation in equation (17) and overall resource utilization in equation (18) on the same figure. Figure 7 shows the system-level performance of our proposed DQN algorithm. The sum of the overall resource allocated to and reserved for both slices is always equal to 1. At each 3 slicing period intervals, the user population in slice 1 is increased by 38 whiles that of slice 2 is increased by 35. From Fig. 7, it is shown that the resource allocation increases with increasing number of users while the overall resource reservation for each slice decreases. After the 20th slicing period, the network resource is used up by the slices and as a result, users are no longer satisfied. Since resource allocation and reservation are dynamic, the DQN agent increases the radio resource of the slice with insufficient resource and decreases the resource of the slice with unutilized resource. The effectiveness of our proposed DQN algorithm is again proven, since the reserved resources of both slices were utilized at the same slicing period.

VI. CONCLUSION

In this paper, we proposed a dynamic resource reservation and deep reinforcement learning based autonomous virtual resource slicing framework for the next generation mobile cellular networks. The proposed framework shows the importance of autonomous radio resource management for independent virtual networks by separating the tasks of the INP and virtual network. The INP periodically reserves the unused resource based on the ratios of the minimum resource requirements of the slices. The slices adjust their resource allocation based on their own defined QoS utility functions and resource utilization functions. Deep reinforcement learning was used by the slices to autonomously increase or decrease their resource in proportion to the radio resource reservations in the BSs. Slices were made to choose at which QoS level to get satisfied independently by defining their QoS utility function and the weights whether to focus on resource utilization or QoS utility in the reward function. Performance evaluation showed the proposed independent and autonomous radio resource management improves the resource utilization and satisfaction of the slices. Due to the satisfaction and resource utilization feedback the DQN agent receives from the slices, it is able to adjust the resources of the slices, improving resource utilization and satisfaction. The DQN algorithm converges to the optimal solution and achieves resource utilization level close to 90% which is double the utilization level of NVS and NetShare. At light load, autonomous radio resource management of the DQN algorithm achieved 100% satisfaction up to about 80% saturation which is the best compared with NVS and NetShare.

REFERENCES

[1] M. Derakhshani, S. Parsaeefard, T. Le-Ngoc, and A. Leon-Garcia, "Leveraging synergy of SDWN and multi-layer resource management for 5G networks," *IET Netw.*, vol. 7, no. 5, pp. 336–345, 2018.

[2] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, 1st Quart., 2015.

[3] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing management & prioritization in 5G mobile systems," in *Proc. 22nd Eur. Wireless Conf.*, Oulu, Finland, May 2016, pp. 1–6.

[4] C. Liang and F. R. Yu, "Distributed resource allocation in virtualized wireless cellular networks based on ADMM," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Hong Kong, Apr. 2015, pp. 360–365.

[5] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: A substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1333–1346, Oct. 2012.

[6] R. Mahindra, M. A. Khojastepour, H. Zhang, and S. Rangarajan, "Radio access network sharing in cellular networks," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Goettingen, Germany, Oct. 2012, pp. 1–10.

[7] I. D. Silva *et al.*, "Impact of network slicing on 5G radio access networks," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2016, pp. 153–157.

[8] F. Fu and U. C. Kozat, "Wireless network virtualization as a sequential auction game," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–9.

[9] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "LTE mobile network virtualization," *Mobile Netw. Appl.*, vol. 4, no. 4, pp. 424–432, Aug. 2011.

[10] T. Guo and R. Arnott, "Active LTE RAN sharing with partial resource reservation," in *Proc. IEEE 78th Veh. Technol. Conf. (VTC Fall)*, Sep. 2013, pp. 1–5.

[11] J. S. Panchal, R. D. Yates, and M. M. Buddhikot, "Mobile network resource sharing options: Performance comparisons," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4470–4482, Sep. 2013.

[12] P. C. Garces, X. C. Perez, K. Samdanis, and A. Banchs, "RMSC: A cell slicing controller for virtualized multi-tenant mobile networks," in *Proc. 81st IEEE VTC Spring*, Glasgow, U.K., May 2015, pp. 1–6.

[13] M. Kalil, A. Shami, and Y. Ye, "Wireless resources virtualization in LTE systems," in *Proc. IEEE INFOCOM WKSHPS*, Toronto, Ontario, Apr./May 2014, pp. 363–368.

[14] M. I. Kamel, L. B. Le, and A. Girard, "LTE wireless network virtualization: Dynamic slicing via flexible scheduling," in *Proc. IEEE Veh. Tech. Conf. (VTC)*, Vancouver, British Columbia, Sep. 2014, pp. 1–5.

[15] A. Moubayed, A. Shami, and H. Lutfiyya, "Wireless resource virtualization with device-to-device communication underlaying LTE network," *IEEE Trans. Broadcast.*, vol. 61, no. 4, pp. 734–740, Dec. 2015.

[16] M. Jiang, D. Xenakis, S. Costanzo, N. Oassa, and T. Mahmoodi, "Radio resource sharing as a service in 5G: A software-defined networking approach," *Comput. Commun.*, vol. 107, pp. 13–29, Jul. 2017.

[17] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Perez, "Network slicing games: Enabling customization in multi-tenant mobile networks," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–19.

[18] G. Tseliou, F. Adelantado, and C. Verikoukis, "Scalable RAN virtualization in multitenant LTE-A heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6651–6654, Aug. 2016.

[19] S. Khatibi and L. M. Carreia, "A model for virtual radio resource management in virtual RANs," *EURASIP J. Wireless Commun. Netw.*, vol. 2015, no. 68, pp. 1–12, Mar. 2015.

[20] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: An auction-based model," in *Proc. IEEE ICC*, Paris, France, May 2017, pp. 1–6.

[21] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing management & prioritization in 5G mobile systems," in *Proc. 22nd Eur. Wireless Conf.*, Oulu, Finland, May 2016, pp. 1–6.

[22] U. Habiba and E. Hossain, "Auction mechanisms for virtualization in 5G cellular networks: Basics, trends, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2264–2293, 3rd Quart., 2018.

[23] V. Petrov *et al.*, "Achieving end-to-end reliability of mission-critical traffic in softwarized 5G networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 485–501, Mar. 2018.

[24] A. Aijaz, "Hap-SliceR: A radio resource slicing framework for 5G networks with haptic communications," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2285–2296, Sep. 2018.

[25] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, J. Famaey, and F. De Turck, "Neural network-based autonomous allocation of resources in virtual networks," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Bologna, Italy, Jun. 2014, pp. 1–6.

[26] R. Mijumbi, J. L. Gorricho, J. Serrat, M. Shen, K. Xu, and K. Yang, "A neuro-fuzzy approach to self-management of virtual network resources," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1376–1390, 2015.

[27] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F. D. Turck, and S. Latre, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Krakow, Poland, May 2014, pp. 1–9.

[28] M. Mamman, Z. H. Hanapi, A. Abdullah, and A. Muhammed, "Quality of service class identifier (QCI) radio resource allocation algorithm for LTE downlink," *PLoS One*, vol. 14, no. 1, 2019, Art. no. e0210310.

[29] *ETSI Technical Specification, 5G; NR; Overall Description*, document 3GPP TS 38.300 version 15.3.1 Release 15, Oct. 2018.

[30] A. T. Z. Kasgari and W. Saad, "Stochastic optimization and control framework for 5G network slicing with effective isolation," in *Proc. 52nd Annu. Conf. Inf. Sci. Syst. (CISS)*, Princeton, NJ, USA, Mar. 2018, pp. 1–6.
[31] V. Mnih, "Playing Atari with deep reinforcement learning," in *Proc. NIPS Deep Learn. Workshop*, 2013, pp. 1–20.
[32] X. Luo, "Delay-oriented QoS-aware user association and resource allocation in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1809—1822, Mar. 2017.
[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

GUOLIN SUN received the B.S., M.S., and Ph.D. degrees in communication and information systems from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2000, 2003, and 2005, respectively.

After Ph.D. graduation in 2005, he has got eight years industrial work experience on wireless research and development for LTE, Wi-Fi, the Internet of Things (ZIGBEE and RFID), cognitive radio, localization, and navigation. Before he joined the School of Computer Science and Engineering, UESTC, as an Associate Professor, in 2012, he was with Huawei Technologies Sweden. He has filed over 30 patents, and published over 40 scientific conference and journal papers, acts as a TPC member of conferences. His general research interests include software-defined networks, network function virtualization, and radio resource management.

Dr. Sun serves as a Vice Chair of the 5G oriented cognitive radio SIG of the IEEE [Technical Committee on Cognitive Networks (TCCN)] of the IEEE Communication Society.

ZEMUY TESFAY GEBREKIDAN received the B.Sc. degree in computer science and engineering from the Mekelle Institute of Technology, Mekelle University, Mekelle, Ethiopia, in 2013, and the M.Sc. degree in computer science and technology from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2018.

After completion of his B.Sc. degree, he was an Assistant Lecturer with Bahirdar University and Aksum University. His primary research interests include network virtualization and security, deep learning, big data, and the Internet-of-Things.

GORDON OWUSU BOATENG received the bachelor's degree in telecommunications engineering from the Kwame Nkrumah University of Science and Technology, Kumasi-Ghana, West Africa, in 2014. He is currently pursuing the M.Sc. degree in computer science and technology with the University of Electronic Science and Technology of China (UESTC).

From 2014 to 2016, he worked under subcontracts for Ericsson (Ghana) and TIGO (Ghana). He is also a member of the Mobile Cloud-Net Research Team, UESTC. His research interests include mobile/cloud computing, 5G wireless networks, data mining, D2D communications, and SDN.

DANIEL AYEPAH-MENSAH received the bachelor's degree in computer engineering from the Kwame Nkrumah University of Science and Technology, Kumasi, Ghana, in 2014. He is currently pursuing the M.Sc. degree in computer science and technology with the University of Electronic Science and Technology of China (UESTC).

From 2014 to 2017, he was a Software Developer. He is also a member of the Mobile Cloud-Net Research Team, UESTC. His research interests include generally wireless networks, big data, and cloud computing.

WEI JIANG received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), in 2008.

Since 2008, he worked four years with the Central Research Institute, Huawei Technologies, in the fields of wireless and the 3GPP standardization. In 2012, he joined the Institute of Digital Signal Processing, University of Duisburg-Essen, Germany, where he was a Postdoctoral Researcher and worked for EU FP7 ABSOLUTE project and H2020 5G-PPP COHERENT project. Since 2015, he has been with the Intelligent Networking Group, German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany, as a Senior Researcher, and works for H2020 5G-PPP SELFNET project. Meanwhile, he also works for the Department of Electrical and Information Technology (EIT), Technische University (TU), Kaiserslautern, Germany, as a Senior Lecturer. He has authored more than 30 papers in top international journals and conference proceedings, and has 27 patent applications in wireless communications, most of which have already been authorized in China, Europe, USA, or Japan. He wrote a chapter "From OFDM to FBMC: Principles and Comparisons" for the book *Signal Processing for 5G: Algorithms and Implementations* (Wiley, 2016). His present research interests include digital signal processing, cooperative communications, 5G, and machine learning.

Dr. Jiang served as a Vice Chair of the IEEE TCCN Special Interest Group (SIG) "Cognitive Radio in 5G".