AN INTELLIGENT NETWORK SLICING FRAMEWORK FOR DYNAMIC RESOURCE SHARING IN MULTI-ACCESS EDGE COMPUTING ENABLED NETWORKS

R. MUNIR†, Y. WEI† and L. TONG‡

† *Beijing University of Posts and Telecommunications, Beijing 100876, China.*
‡ *China United Network Communications Group Co. Ltd Beijing 100044, China*
rizwanmunir@bupt.edu.cn weiyifei@bupt.edu.cn tonglei2@chinaunicom.cn

***Abstract*--Dynamic resource sharing in multi-access edge computing (MEC) enabled networks has gained tremendous interest in the recent past, paving the way for the realization of beyond fifth generation (B5G) communication networks. To enable efficient and dynamic resource sharing, Network Slicing has appeared as a promising solution, virtualizing the network resources in the form of multiple slices employed by the end-users requiring strict latency, proximate computations, and storage demands. In literature, network slicing is primarily studied in the context of communication resource slicing, and little research has been devoted to jointly slicing communication, energy, and MEC resources. In this paper, we, therefore, proposed a joint network-slicing framework that considers 1) communication resources, 2) compute resources, 3) storage resources, and 4) energy resources, and intelligently and dynamically shares the resources between different slices, aiming to improve tenants' overall utility. To this end, we formulated a utility maximization problem as Markov-chain Decision Process. We utilized a tenant's manager that employs a deep reinforcement learning technique named "deep deterministic policy gradient" to enable dynamic resource sharing. Simulation results reveal the effectiveness of the proposed scheme.**

Keywords----Wireless networks, 5G slicing, dynamic resource allocation, Markov decision process, deep reinforcement learning, deep deterministic policy gradient

I. INTRODUCTION

The exponential growth in mobile data traffic along with the emergence of a diverse range of new applications requiring enhanced Quality of Service (QoS) are posing significant challenges—in terms of efficient resource utilization—to modern communication systems, i.e., the fifth generation (5G) and beyond that intend to cope with not only the unparalleled end-users application and services requirements but also support a variety of vertical industries (Afolabi *et al.*, 2018). As reported in the Cisco annual Internet report, the average 5G speed in 2023 will be approximately 575 megabits per second, serving the communication needs of 10.6% of global 5G connections—out of total mobile connections. In Chen *et al*. (2018) with such advanced capabilities, on the one hand, 5G promises to deliver dynamic and flexible communication infrastructure to futuristic technologies, including artificial intelligence (AI), emerging robotic surgeries combing mechanical elements with cutting-edge communication equipment, autonomous driving, immersive video experience using AR/VR technologies, and smart cities, to name a few, on the other hand, confronting various issues especially in guaranteeing the optimal network performance while simultaneously coping the diverse requirements of end-user heterogeneous devices in aforementioned technologies (Deng *et al*. 2019). Hence, providing efficient 5G network resource management solutions to address the challenges discussed above is becoming the need of time. If not effectively handled, it may obstruct the 5G communication from reaching its true potential (Esmat and Lorenzo, 2020).

The challenges mentioned earlier have motivated the researchers to look for potential solutions. Because of various collective research efforts from academia and industry, network slicing has appeared as a promising solution to resolve the network resource management issues and meet the varying end-users' applications and services requirements (Alves Esteves *et al*., 2021). The network slicing concept, to some extent, possesses similarities with monolithic and microservices application architectural styles, in which the single bulky monolithic application is divided into multiple tiny independent components called microservices fulfilling the requirements of the end-user's application (Ferdowsi *et al*., 2021). Likewise, network slicing fundamentally divides the underlying physical network infrastructure resources into multiple tiny components often referred to as network slices that can be managed and controlled independently, permitting the co-existence of various virtual networks in the same physical network space. The realization of network slicing is practically achieved by utilizing network function virtualization (NFV) and software-defined networking (SDN) (Gong *et al*., 2022).

In the early stages of network slicing research, static slicing solutions were proposed formulating the network resource scheduling and allocation as NP-hard problems that can be solved by various heuristic approaches (Hua *et al*., 2020; Hurtado Sánchez *et al.,* 2022). Such solutions, though, provided networking resource-sharing and confronted resource under-utilization or over-utilization issues owing to the fixed amount of resources assigned to a network slice and to dynamically varying request patterns and QoS requirements from mobile devices. To

further understand the issue, let us take an example: the network is divided into three different slices equipped with fixed resources (Korrai *et al*., 2020). Among them, slice A is developed for latency-sensitive applications, slice B is used for bandwidth-hungry applications, whereas slice C is utilized for low-data rate applications. However, the network is deployed in a region where end-users requests for such applications may vary over time. In such situations, it is possible that due to both the excessive workload and limited and fixed resource assignments to slice A, some of the requests may not be handled effectively, which on the contrary, could be managed efficiently if slice A had the flexibility to utilize the resources from slice B or C (Khumalo *et al.*, 2021).

Recently, with the emergence of AI in networking, dynamic networking slicing solutions have been proposed to resolve the static slicing issues as outlined above (Lee *et al.,* 2018). To this end, machine learning (ML) and, specifically deep learning (DL) techniques such as deep Q-learning (DQL) and deep reinforcement learning (DRL) have gained significant interest from the research community. In DRL, an agent such as an SDN controller learns from the environment—i.e., the request patterns—and assigns rewards to the system or function that correctly selects (action) the network resources for each incoming request. The DRL has the potential to handle the scalability of decisions making problems in terms of the high dimensionality of action and state spaces, which was previously nearly impossible by using traditional solutions (Li *et al*., 2018).

Besides, in literature, network slicing is primarily studied in communication resource slicing. Little or no research has been devoted to slicing joint communication and MEC resources slicing. The vision of MEC is to bring the computation and storage resources closer to the end-user, aiming to reduce the latency and backhaul network traffic, thus becoming a core component of Beyond 5G communication systems. Fewer research works consider MEC in their network slicing solution; however, they mainly focus on computing resources and ignore a MEC server's storage/caching capacity (Li *et al.*, 2020; Liu *et al.,* 2021)..

In this paper, therefore, motivated by the success of the DRL technique and going beyond traditional network slicing solutions, we proposed a joint network slicing framework that takes into account the 1) communication resources, 2) compute resources, 3) storage resources, 4) energy resources, and intelligently and dynamically share the resources between different slices using a DRL technique named deep deterministic policy gradient (DDPG), aiming to enhance tenants' overall utility. To this end, we formulated a utility maximization problem as Markov-chain Decision Process and utilized a tenant's manager that employs DDPG to enable dynamic resource sharing.

In summary, the following are the main contributions of this paper.

1. At first, this paper formulates various system models including the business model, network model, wireless communication model, computational model, storage model and energy model considering the use-case of multi-access edge computing enable networks.
2. To share multiple resources intelligently and dynamically between different slices, the proposed scheme utilizes a DRL scheme named DDPG. The proposed joint network-slicing framework takes into account the following four resources: 1) communication resources, 2) compute resources, 3) storage resources, and 4) Energy resources. The DDPG method improves the overall system performance by intelligently and dynamically assigning the aforementioned resources to each slice.
3. Finally, extensive simulations carried out in Python validate the effectiveness of the proposed framework.

The rest of the paper is organized as follows. Section 2 sheds light on the most recent Literature Review, and the primary motivation behind our work. The system and network models' designs are outlined in Section 3. Problem formulation explained in section 4. Complete detail on the proposed resource scheduling approach for network slicing in multi-access edge computing is described in Section 5. The effectiveness of the proposed scheme was validated through a simulation-based study presented in Section 6. At the end, we finally came to a conclusion.

II. LITERATURE REVIEW

In literature, many research works have been proposed that aim at providing efficient solutions for resource allocation or, in other words, for network slicing. In the following, we shed light on these network-slicing solutions.

A radio resource allocation method to maximize the SLA contract rate and maintain the isolation between the slices is introduced in Nassar and Yilmaz (2019). The problem is solved using the Lagrange dual algorithm. The authors in Nassar and Yilmaz (2022) and Patel *et al*. 2022) introduce the cloud RAN (C-RAN) operator's revenue maximization by correctly accepting the slice requests. Two types of long-term and short-term revenues are considered. The values calculate the long-term revenue in the network slice request, and the short-term revenue is achieved by saving system power consumption in each frame. The optimization problem is formulated as a mixed-integer nonlinear programming (MINP), then to solve the problem, the authors employ a successive convex approximation (SCA) algorithm. The authors formulate a resource allocation problem subject to service isolation, latency, and minimum rate constraints. To maintain the reliability constraint, they use adaptive modulation and coding (Qui *et al.*, 2019). The authors in Tang *et al.* (2019) introduce the Network Slice resource allocation problem in 5G C-RAN to maximize operators' utility. The framework of the problem includes an upper layer that manages the mapping of virtual protocol stack functions; and a lower layer, which controls radio remote

unit association, power, and subchannel allocation. To reduce the complexity of the Q-value table, the authors used the multi-agent Q-learning approach.

Tong *et al.* (2020) investigate a dynamic Network Slice framework for downlink multitenant heterogeneous cloud RAN (H-CRAN) by considering both small-cell and macro-cell tiers. The proposed architecture includes two-level, an upper level for managing admission control, baseband resource allocation, and user association, and a lower level for handling radio resource allocation between users. The objective of the problem is to maximize the throughput of the tenants by considering the constraints of QoS, fronthaul and backhaul capacities, tenants' priorities, baseband resources, and interference. Based on the deep Q-networks (DQN) algorithm, Wang *et al.* (2019) investigates the dynamic resource allocation problem to maximize the E2E access rate for multi-slice and multi-service scenarios. The purpose of Wang and Zhang (2019) is to minimize the E2E delay to guarantee the reliability requirements of Network Slice in the uRLLC application. The authors propose a communication and computation resource allocation algorithm based on subgraph isomorphism. A new dynamic edge/fog Network Slice (EFNwS) scheme to find an optimal slice request admission policy to maximize the InP's long-term revenue is proposed in Wang *et al.* (2022). Tenants can temporarily lease the unused resources to the InP to serve demands exceeding its current resources in stock. A semi-Markov decision process (SMDP) is used to model the arrival of slice requests. A Q-learning algorithm is applied to find the optimal policy under uncertain resource demands.

Moreover, a DRL algorithm and an enhancement based on a deep dueling (Dueling DQEFNwS) algorithm are applied to reduce the computational complexity of Q-learning and improve the convergence time. In this line lies the work of Wei *et al.* (2020a), which proposes a deep Q-learning approach to radio resource slicing and priority-based core network slicing, showing its advantage in addressing demand-aware resource allocation. Wei *et al.* (2020b) formulate the problem of frequency bands allocation and the problem of computation resources orchestration for different slices. These problems are reduced to choosing a particular configuration from a finite set of available configurations by leveraging DDQNs. Similarly, in Yuan *et al.* (2021), a DRL solution based on DDQN is presented for multitenant cross-slice resource orchestration, where again, a discrete number of communication and computation resources have to be allocated to different slice tenants. Finally, a deep deterministic policy gradient (DDPG) with an advantage function is employed in Zhang *et al.* (2022a) to allocate bandwidth resources to different network slices. Compared to the approaches mentioned earlier, the continuous nature of DDPG allows for more fine-grained resource allocation (Zhang *et al*. 2022b; Zhou *et al*., 2020).

How Does the Proposed Scheme is Different?

The resource-sharing scheme discussed above, in general, takes into account communication or computing resources shared individually in cellular or multi-access edge-based networking. To the best of our knowledge, in contrast to the conventional approaches, the proposed scheme is jointly taking into account the communication, such as bandwidth resources, hardware, such as energy resources, and MEC resources, such as computing and storage resources, for network slicing and employs a deep deterministic policy gradient approach to achieve dynamic resource sharing amongst different slices, which is unique compared to works proposed in the literature. The proposed DDPG method improves the overall system performance by intelligently and dynamically assigning the underlying communication, hardware and edge resources to each slice, consequently the improve the end user QoE.

III. SYSTEM MODEL

This section describes the overall system model of our proposed scheme, comprising a business model, network model, communication model, computation model, storage model, and finally, energy model.

A. Business Model

In our business model, we consider three main entities: infrastructure providers denoted as as $I_i^{p'} = \{I_1^{p'}, I_2^{p'}, I_3^{p'} \ldots, I_l^{p'}\}$ where "l" is a total number of infrastructure providers, multiple tenants represented as $T_i' = \{T_1', T_2', T_3' \ldots, T_m'\}$ where "m" is a total number of tenants in the network, and a set of service providers exemplified as $S_i' = \{S_1', S_2', S_3' \ldots, S_n'\}$ where "n" is the total number of service providers in the network. The tenants' T_i' acquire resources from an $I_i^{p'}$ on a contract basis and virtualize them into various slices. A single tenant ($T_{i=k}'$) may own multiple slices. The S_i' provides different services to end-users based on their quality of service (QoS) requirements and demands on the slices as mentioned earlier virtualized by the $T_{i=k}'$, where "k" is a tenant from a set of multiple tenants. Since $T_{i=k}'$ owns multiple slices and has a fixed number of resources, assigning static resources to each slice may lead to inefficient utilization of underlying resources, consequently demeaning the overall business utility of $T_{i=k}'$. To cope with such issues aiming to enhance the $T_{i=k}'$ business utility, this paper proposes dynamic resource allocation to the slices owned by the $T_{i=k}'$, which is ultimately beneficial for $T_{i=k}'$ and improves the QoS requirements of end-users. Figure 1 illustrates the overall business model.

B. Network Model

In our network model, we consider a cellular network comprising various base stations $ß_i = \{ß_1, ß_2, ß_3 \ldots ß_n\}$ where "n" is the total number of base stations in the (MEC) server providing computation (Č) and storage overall network, equipped with a mobile edge computing (Ș) resources in the proximity of end-users. Moreover, the MEC server also has a limited energy resource (E). The $ß_{i=k}$ (where $k \in [1, n]$) itself is providing band-

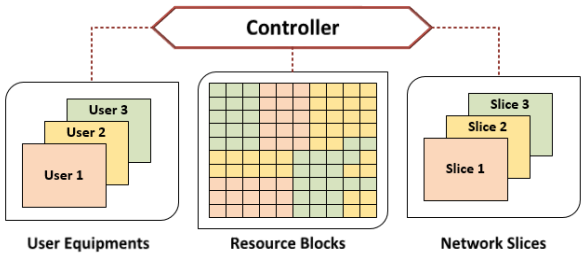

Figure 1 Overall System Model

width (Ɓ) as a resource. As all the ß$_{\text{I}}$ are equipped with MEC, therefore, hereafter, we denoted base stations as ß$_{\text{i}}^{\text{m}}$ providing Č, Ş, and Ɓ. The total computation, storage, and bandwidth resources are denoted as Č$_{\text{total}}$ (cycles), Ş$_{\text{total}}$(bytes), and Ɓ$_{\text{total}}$ (Hz). The resources mentioned earlier at each ß$_{\text{I}}$ are virtualized by different T$_{\text{i}}'$ in the form of a set of slices Ŏ$_{\text{i}}$ = {Ŏ$_1$, Ŏ2, Ŏ$_3$, … Ŏ$_{\text{n}}$} where n is the total number of slices owned by numerous S$_{\text{i}}'$. At a specific time interval, let say ∇$^{\text{td}}$, which is a fixed time duration in a time-slotted system represented as ∇$_{\text{i}}^{\text{td}}$ = {∇$_1^{\text{td}}$, ∇$_2^{\text{td}}$, …, ∇$_{\text{n}}^{\text{td}}$}, the ß$_{\text{i=k}}$ support a set of Ŭ active mobile users, and each slice (ŏ ∈ Ŏ$_{\text{i}}$) serves a specific set of users ŭ$_{\text{i}}$ = {ŭ$_1$, ŭ$_2$, ŭ$_3$, …, ŭ$_{\text{n}}$} where ŭ ⊆ Ŭ represented as ŭ$^{\check{o}}$. Following the notations mentioned earlier, the ŭ$_{\text{i=l}}^{\check{o}}$ (where l ∈ [1,n]) denotes that a user l is being served by slice ŏ and the ŭ$_{\text{i=l}}^{\check{o},ß_{i=k}}$ implies that a user l is being served by slice ŏ at the base station ß$_{\text{i=k}}$. Overall, the total number of users in all the slices satisfies the following equation.

$$\sum_{ŭ\ \in Ŭ,\ ŏ \in Ŏ_i} ŭ_{ŭ_{i=l}}^{ŏ} = |Ŭ| \qquad (1)$$

Moreover, a three-field notation (č$_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ}$, Ş$_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ}$, ƀ$_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ}$) is used to denote the virtualized resources assigned to the slice ŏ at the time slot ∇$_{\text{i=x}}^{\text{td}}$. Among them, the č is a fraction of Č$_{\text{total}}$, ş is a fraction of Ş$_{\text{total}}$, and ƀ is a fraction of Ɓ$_{\text{total}}$ and satisfies the following equation.

$$\sum_{ŏ \in Ŏ_{i \Rightarrow\ i=[1,n]}} č_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ} =$$

$$\sum_{ŏ \in Ŏ_{i \Rightarrow\ i=[1,n]}} Ş_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ} =$$

$$\sum_{ŏ \in Ŏ_{i \Rightarrow\ i=[1,n]}} ƀ_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ} = 1$$

where

$$č_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ}, Ş_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ}, ƀ_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ} \in [0,1] \quad (2)$$

Furthermore, each slice ŏ occupy a dedicated space for a data structure (Ď$_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ}$) inside the ß$_{\text{i=k}}$ for storing the services request received from the end-users at the time interval ∇$_{\text{i=x}\Rightarrow\text{x=[1,n]}}^{\text{td}}$. The average request arrival rate is Ɣ$_{\nabla_{i=x \Rightarrow x=[1,n]}^{td}}^{ŏ}$ following the Poisson process. The slice data structure can be expressed using the following equation (Wang *et al.*, 2022)

$$Ď_{\nabla_{i=x+1}^{td}}^{ŏ} = \max\left\{0, \left(Ř_{\nabla_{i=x}^{td}}^{ŏ} + Ɣ_{\nabla_{i=x}^{td}}^{ŏ}\right)\right\} \qquad (3)$$

where Ř$_{\nabla_{i=x}^{td}}^{ŏ}$ is the remaining services requests in the time slot ∇$_{\text{i=x}}^{\text{td}}$ and can be expressed as follows:

$$Ř_{\nabla_{i=x}^{td}}^{ŏ} = Ď_{\nabla_{i=x}^{td}}^{ŏ} - Ă_{\nabla_{i=x}^{td}}^{ŏ} \qquad (4)$$

where Ă$_{\nabla_{i=x}^{td}}^{ŏ}$ are the services requests successfully served in the time slot ∇$_{\text{i=x}}^{\text{td}}$

C. Wireless Communication Model

In this paper, we modeled the wireless channel between each user ŭ$_i$ and its connected slice ŏ at the base station ß$_{i=k}$ in a specific time slot ∇$_{i=x}^{td}$ as an independent and identically distributed (i.i.d) Rayleigh variable. Moreover, the channel power gained from the user ŭ$_i$ to its slice ŏ at the base station ß$_{i=k}$ in a specific time slot ∇$_{i=x}^{td}$ is denoted as Ĥ$_{ŭ_i=l,ŏ}^{ß_{i=k},\nabla_{i=x}^{td}}$. Furthermore, the transmission power—which is assumed to be constant during the complete transmission duration—required by ŭ$_i$ to forward the packet towards ß$_{i=k}$ can be represented as Ƥ$_{ŭ_i=l,ß_{i=k}}^{tran}$. Then according to the Shannon capacity formula, the maximum achievable data rate in bits/s for each user ŭ$_i$ in an x time slot ∇$_{i=x}^{td}$ can be expressed as follows:

$$Ғ_{ŭ_i=l,ŏ}^{ß_{i=k},\nabla_{i=x}^{td}} = ƀ^{ŏ}\left(1 + \frac{Ƥ_{ŭ_i=l,ß_{i=k}}^{tran} Ĥ_{ŭ_i=l,ŏ}^{ß_{i=k},\nabla_{i=x}^{td}}}{Ϙ^{ß_{i=k}}}\right) \qquad (5)$$

where ƀ^{ŏ} is the maximum bandwidth assigned to the slice ŏ, and Ϙ$^{ß_{i=k}}$ is the noise power at ß$_{\text{i=k}}$.

Let's assume that each slice has a maximum completion time $\eta_{max}^{ŏ}$ of and minimum throughput $L_{min}^{ŏ}$, then the following minimum throughput requirement should meet by the user ŭ$_i$ in slice ŏ.

$$Ғ_{ŭ_i=l,ŏ}^{ß_{i=k},\nabla_{i=x}^{td}} \geq L_{min}^{ŏ} \qquad (6)$$

D. Computation Model

In our work, we assumed that end-users ŭ$_i$ have a limited computational capacity and, therefore, may offload the computations to the *MEC* server, which in our network model is represented as ß$_i^m$ whose computational resources are virtualized by the slice ŏ. When offloading the computations, the ŭ$_i$ our model utilizes the nested multi-field notation to share the data and corresponding necessary information and can be represented as follows:

$$Ÿ_{ŭ_i=i,ŏ}^{ß_{i=k}^{m}} = \left((\vartheta_{app} D_{ŭ_i=i,ŏ}^{app}, M_{ŭ_i=i,ŏ}^{app}), (\vartheta_{content} D_{ŭ_i=i,ŏ}^{content}, M_{ŭ_i=i,ŏ}^{content}), (C^{app,content}), (L_{max}^{app,content})\right) \qquad (7)$$

where $D_{ŭ_i=i,ŏ}^{app}$ is the application program data and $M_{ŭ_i=i,ŏ}^{app}$ contains the meta information such as length of the application program and name of the application program, among others, $D_{ŭ_i=i,ŏ}^{content}$ is the content on which the computations will be performed and similarly $M_{ŭ_i=i,ŏ}^{content}$ contains the meta information such as length of the content and name of the content, to name a few, $C^{app,content}$ represents the computation cycles required by the corresponding computations. Finally, $L_{max}^{app,content}$ is the minimum latency in which the computation results should arrive back to the user ŭ$_i$. Moreover, the ϑ_{app} and $\vartheta_{content}$ are two

Boolean variables that may take the values either 0 or 1, representing whether the user $ŭ_i$ is offloading the corresponding data to $ß_i^m$.

Since fulfilling the maximum latency requirement is one of the most crucial goals, we consider both transmission and computation latency in our work which can be represented as:

$$L_{ŭ_i=i,ŏ}^{transmission} = \frac{(\vartheta_{app} Size(D_{ŭ_i=i,ŏ}^{app}) + Size(\vartheta_{content} D_{ŭ_i=i,ŏ}^{content}))}{\text{F}_{ŭ_i=l,ŏ}^{ß_{i=k},\nabla_{i=x}^{td}}}$$

$$L_{ŭ_i=i,ŏ}^{computation} = \frac{(C^{app,content})}{ĉ^{ŏ}} \quad (8)$$

where the total latency can be represented as

$$L_{ŭ_i=i,ŏ}^{total} = L_{ŭ_i=i,ŏ}^{transmission} + L_{ŭ_i=i,ŏ}^{computation}$$

where total latency must be less than the maximum latency requirement by the user $ŭ_i$

$$L_{ŭ_i=i,ŏ}^{total} \leq L_{max}^{app,content}$$

Finally, the total latency must also be less than the maximum slice completion time.

$$L_{ŭ_i=i,ŏ}^{total} \leq \eta_{max}^{ŏ}$$

E. Storage Model

In this work, we also assumed that end-users $ŭ_i$ have limited storage capacity and, therefore, may offload the storage of the content to the *MEC* server, which in our network model is represented as $ß_i^m$ whose storage resources are virtualized by the sliceŏ. When offloading the storage, the $ŭ_i$ our model utilizes the nested two-field notation to share the data and corresponding necessary information and can be represented as follows:

$$\mathcal{I}_{ŭ_i=i,ŏ}^{ß_{i=k}^m} = (D_{ŭ_i=i,ŏ}^{content}, S^{content}) \quad (9)$$

where $D_{ŭ_i=i,ŏ}^{content}$ is the data that needs to be stored at $ß_i^m$ and $S^{content}$ is the size of the content in bytes.

Let us assume that slice ŏ has $ş^{ŏ}$ storage capacity out of $Ş_{\text{total}}$ at $ß_i^m$, then $D_{ŭ_i=i,ŏ}^{content}$ must be less than $ş^{ŏ}$, i.e.,

$$D_{ŭ_i=i,ŏ}^{content} \leq ş^{ŏ}$$

F. Energy Model

In this work, we assumed that the tenant $T'_{i=k}$ has limited energy resources, which are virtualized by the slice ŏ. The energy consumption may vary depending on the type of request a tenant receives. For instance, the energy consumption for wireless communication (packet reception and transmission) varies significantly compared to the energy required for computations and storage. Thus $Ĭ_{ŭ_i=i,ŏ}^{ß_{i=k}^m}$ denotes the request that arrives at $ß_{i=k}^m$ from user $ŭ_i = i$ and may take any energy consumption value from the following: wireless communication energy consumption ($Ĕ^{w}$) value, computations energy consumption ($Ĕ^{c}$) value, or storage energy consumption ($Ĕ^{s}$) value, and can be represented as follows.

$$Ĭ_{ŭ_i=i,ŏ}^{ß_{i=k}^m} = [\, Ĕ^{w} \mid Ĕ^{c} \mid Ĕ^{s}] \quad (10)$$

Let us assume that slice ŏ has $ĕ^{ŏ}$ energy capacity out of $Ĕ_{\text{total}}$ at $ß_i^m$, then $Ĭ_{ŭ_i=i,ŏ}^{ß_{i=k}^m}$ must be less than $ĕ^{ŏ}$, i.e.,

$$Ĭ_{ŭ_i=i,ŏ}^{ß_{i=k}^m} \leq ĕ^{ŏ}$$

IV. PROBLEM FORMULATION

To define the problem, first, we would like to present the preliminary details, describing the overall goal of our proposed scheme. In this paper, we aim to increase the utility of the $T'_{i=k}$ who acquire resources from $I_{i=d}^{p'}$ on a contract basis after paying an agreed amount, virtualize them into various slices, and then charge per slice amount from $S'_{i=e}$.

For instance, regarding computations resources, the $T'_{i=k}$ pays € amount to $I_{i=d}^{p'}$ in return for computing resources Č (units/cycle) during the time duration $\nabla_{i=x}^{td}$ and for slice ŏ, denoted as $€_{\nabla_{i=x}^{td}}$, ŏČ, and charges $\wp$ amount from $S'_{i=e}$ during the time duration $\nabla_{i=x}^{td}$ and for slice ŏ, denoted as $\wp_{\nabla_{i=x}^{td},ŏ}^{Č}$. The $T'_{i=k}$ computation utility($\Psi_{\nabla_{i=x}^{td},ŏ}^{Č}$) will be as follows.

$$\Psi_{\nabla_{i=x}^{td},ŏ}^{Č} = \sum_{ŭ_i^{ŏ} \in ŭ^{ŏ}} \wp_{\nabla_{i=x}^{td},ŏ}^{Č} - €_{\nabla_{i=x}^{td},ŏ}^{Č} \quad (11)$$

Similarly, for storage resources, the $T'_{i=k}$ pays $ﬦ$ amount to $I_{i=d}^{p'}$ in return for storage resources Ș (units/byte) during the time duration $\nabla_{i=x}^{td}$ and for slice ŏ, denoted as $ﬦ_{\nabla_{i=x}^{td},ŏ}^{Č}$, and charges ₱ amount from $S'_{i=e}$ during the time duration $\nabla_{i=x}^{td}$ and for slice ŏ, denoted as $₱_{\nabla_{i=x}^{td},ŏ}^{Č}$, then the $T'_{i=k}$ storage utility($₴_{\nabla_{i=x}^{td},ŏ}^{Ș}$) will be as follows

$$₴_{\nabla_{i=x}^{td},ŏ}^{Ș} = \sum_{ŭ_i^{ŏ} \in ŭ^{ŏ}} ₱_{\nabla_{i=x}^{td},ŏ}^{Č} - ﬦ_{\nabla_{i=x}^{td},ŏ}^{Č} \quad (12)$$

Likewise, for bandwidth resources, the $T'_{i=k}$ pays $\Im$ amount to $I_{i=d}^{p'}$ in return for bandwidth resources $\mathcal{B}$ (units/Hz) during the time duration, $\nabla_{i=x}^{td}$ and for slice ŏ denoted as $\Im_{\nabla_{i=x}^{td},ŏ}^{B}$, and charges Ə amount from $S'_{i=e}$ during the time duration $\nabla_{i=x}^{td}$ and for slice ŏ, denoted as $Ə_{\nabla_{i=x}^{td},ŏ}^{Ɓ}$, then the $T'_{i=k}$ bandwidth utility($\mathcal{M}_{\nabla_{i=x}^{td},ŏ}^{B}$) will be as follows

$$\mathcal{M}_{\nabla_{i=x}^{td},ŏ}^{B} = \sum_{ŭ_i^{ŏ} \in ŭ^{ŏ}} Ə_{\nabla_{i=x}^{td},ŏ}^{Ɓ} - \Im_{\nabla_{i=x}^{td},ŏ}^{B} \quad (13)$$

Finally, for energy resources, the $T'_{i=k}$ pays $\mathcal{J}$ amount to $I_{i=d}^{p'}$ in return for energy resources $\underset{\sim}{E}$ (units/Joule) during the time duration $\nabla_{i=x}^{td}$ and for slice ŏ denoted as $\mathcal{L}_{\nabla_{i=x}^{td},ŏ}^{E}$, and charges $\mathfrak{H}$ amount from $S'_{i=e}$ during the time duration $\nabla_{i=x}^{td}$ and for slice ŏ, denoted as $\mathcal{H}_{\nabla_{i=x}^{td},ŏ}^{\underset{\sim}{E}}$, then the $T'_{i=k}$ energy utility($\mathcal{M}_{\nabla_{i=x}^{td},ŏ}^{E}$) will be as follows

$$\mathcal{M}_{\nabla_{i=x}^{td},ŏ}^{E} = \sum_{ŭ_i^{ŏ} \in ŭ^{ŏ}} \mathcal{H}_{\nabla_{i=x}^{td},ŏ}^{\underset{\sim}{E}} - \mathcal{L}_{\nabla_{i=x}^{td},ŏ}^{E} \quad (14)$$

Problem: Now, our goal is to increase the overall profit of $T'_{i=k}$ considering the utilities mentioned earlier and can be defined as follows.

$$\max(\sum_{\nabla_i^{td}} \sum_{Ŏ_i} \Psi_{\nabla_{i=x}^{td},ŏ}^{Č} + ₴_{\nabla_{i=x}^{td},ŏ}^{Ș} + \mathcal{M}_{\nabla_{i=x}^{td},ŏ}^{B} + \mathcal{M}_{\nabla_{i=x}^{td},ŏ}^{E}) \quad (15)$$

V. PROPOSED SOLUTION

The utility optimization problem defined in the previous section is computationally complex owing to the dynamic requirements of slices and their related resources; thus, traditional optimization solutions cannot be employed immediately. To resolve such problems, various machine learning (ML) algorithms such as ML-based resource allocation solutions, including traditional deep reinforcement learning (DRL) algorithms as well as model-free DRL, i.e., deep Q-networks (DQN) and double DQN have appeared as a promising solution. Among the abovementioned approaches, DQN and double DQN are value-based ML approaches utilized to solve problems with continuous action space. However, as our problem include continuous action space, to the obtained optimal solution, we can employ policy gradient-based RL methods such as deterministic policy gradient (DPG) algorithm and deep DPG (DPG) where the former approximate action-value gradient by employing learned approximation of Q function and later off-policy algorithm handle continuous action space by utilizing the actor-critic method. To cope with the slow convergence issue faced in gradient-based RL for large-scale continuous actin spaces, the DDPG combines value and policy-based algorithms to achieve optimal results. Thus, to efficiently resolve our utility optimization problem, we employed DDPG. In the following subsection, we first present a quick primer of DRL fundamentals and then outline the details of the DDPG algorithm to solve our utility optimization problem.

A. A Quick Primer of RL Fundamentals

In the RL algorithm the agent ($T'_{i=k}$) receive the observations by interacting with the environment (E) at each timeslot t and selects an action $A_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}$ from state $S_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}$. It is to note the timeslot $\nabla^{td}_{i=x\Rightarrow x=[1,n]}$ can be set to 1 sec. After selecting the$A_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}$, the A_E executes the action and receives the immediate reward, which can be represented as $\mathrm{R}(A_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}\ S_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}})$. The reward is obtained based on the state transition probability exemplified as $p(S_{\nabla^{td}_{i=x+1}}|S_{\nabla^{td}_{i=x}}, A_{\nabla^{td}_{i=x}})$. An A_E behavior can be indicated by the policy P that maps the state S to the probability distribution of action $P(A)$. The goal of the $T'_{i=k}$ is to learn the policy and maximize the expected reward. We set the following definitions according to our optimization problem using RL.

B. Agent

In the proposed model, as mentioned above, $T'_{i=k}$ receives the network information such as the requirement of computation *(Č)* and storage *(Ș),* bandwidth (*B*), and energy (*E*) resources. The $T'_{i=k}$ then perform suitable action $A_{\nabla^{td}_{i=i}}$ where i is the current time slot according to the set policy P. Afterwards, the network state $S_{\nabla^{td}_{i=x}}$ changes to the next state $S_{\nabla^{td}_{i=x+1}}$ and the agent receives positive rewards ($R^{P'}$) if all the constraints as outlined in Eqs (11) to (14). Otherwise, the agent receives a negative reward if failed to satisfy all the constraints mentioned above.

C. System States

The $T'_{i=k}$ decision-making is based on the system states of various network parameters. In our model, as we considered four essential parameters, i.e., computations resources Č, storage *resources Ș,* bandwidth resources *Ɓ*, and energy resources *E,* the $T'_{i=k}$ decision making solely depends on these parameters. Thus, the system state of $T'_{i=k}$ at time slot t=i can be expressed as follows represents the queue backlog of slices requests where each slice contains the proportion of Č, *Ș, Ɓ, and E.*

$$S_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}} = \left\{Q^{\check{o}[\check{C},\check{S},\check{B},E]}_{1,\nabla^{td}_{i=x\Rightarrow x=[1,n]}}, Q^{\check{o}[\check{C},\check{S},\check{B},E]}_{2,\nabla^{td}_{i=x\Rightarrow x=[1,n]}}(t), \dots, Q^{\check{o}[\check{C},\check{S},\check{B},E]}_{s,\nabla^{td}_{i=x\Rightarrow x=[1,n]}}(t)\right\} \quad (16)$$

D. Action:

As mentioned before, the agent $T'_{i=k}$ receive the observations by interacting with environment E at each timeslot $\nabla^{td}_{i=x\Rightarrow x=[1,n]}$ and selects an action$A_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}$ from state $S_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}$. Likewise, in our proposed scheme the agent does the same. In the proposed scheme the action $A_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}$ of $T'_{i=k}$ can be defined as

$$A_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}} = \{\check{c}^{\check{o}}_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}, \check{s}^{\check{o}}_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}, \check{b}^{\check{o}}_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}, \hat{e}^{\check{o}}_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}\}. \quad (17)$$

Overall actions $A_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}}$ of an agent $T'_{i=k}$ for all slices can be defined as follows:

$$A_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}} = \{A_{\nabla^{td}_{i=1}}, A_{\nabla^{td}_{i=2}}, A_{\nabla^{td}_{i=3}} \dots\dots A_{\nabla^{td}_{i=s}}\}$$

E. Reward Function:

In RL-based schemes, the agent receives positive rewards after meeting the required constraints and improving the network performance. In our utility optimization problem, the goal is to increase the utility of the $T'_{i=k}$ by sharing the limited resources received from $I_i^{p'}$ without exceeding the maximum limit and fulling the QoS requirements of the end-users along with the required resources demand from them. Thus, the reward function can be expressed as follows.

$$\begin{aligned}\mathrm{R}_{\nabla^{td}_{i=x\Rightarrow x=[1,n]}} = \mathrm{w}_1 \sum_{\check{o}\in\check{O}_i}\Big(&\Psi^{\check{C}}_{\nabla^{\mathrm{td}}_{\mathrm{i=x}},\check{o}} + \mathfrak{E}^{\check{S}}_{\nabla^{\mathrm{td}}_{\mathrm{i=x}},\check{o}} + \mathcal{M}^{\check{B}}_{\nabla^{\mathrm{td}}_{\mathrm{i=x}},\check{o}} \\ &+ \mathcal{L}^{\mathrm{E}}_{\nabla^{\mathrm{td}}_{\mathrm{i=x}},\check{o}}\Big) + \mathrm{w}_2 \sum_{\check{u}^{\check{o}}_{i=l}\in\check{U},\ \check{o}\in\check{O}_i} (\mathrm{F}^{\beta_{\mathrm{i=k}},\nabla^{\mathrm{td}}_{\mathrm{i=x}}}_{\check{u}_{\mathrm{i}}=\mathrm{l},\check{o}} - \mathcal{L}^{\check{o}}_{min}) \\ &+ \mathrm{w}_3 \sum_{\check{u}^{\check{o}}_{i=l}\in\check{U},\ \check{o}\in\check{O}_i} \left(\eta^{\check{o}}_{max} - L^{total}_{\check{u}_i=i,\check{o}}\right) + \\ &\mathrm{w}_4 \sum_{\check{u}^{\check{o}}_{i=l}\in\check{U},\ \check{o}\in\check{O}_i} (\check{s}^{\check{o}} - D^{content}_{\check{u}_i=i,\check{o}}) + \\ &\mathrm{w}_5 \textstyle\sum_{\check{u}^{\check{o}}_{i=l}\in\check{U},\ \check{o}\in\check{O}_i} (\check{e}^{\check{o}} - \check{M}^{\beta^m_{i=k}}_{\check{u}_i=i,\check{o}})\end{aligned} \quad (18)$$

Table 1: Simulation Parameter Table

Parameter Name	Value
Bandwidth Resources	150MHZ
Computing Resources	225GHZ
Energy Resources	175J
Storage Resources	250GB
Actor Learning Rate	0.001
Critic Learning Rate	0.001
Reward Discount	0.9

In the above equation, owing to the various units of parameters, the normalized waits are employed where $w_1 + w_2 + w_3 + w_4 + w_5 = 1$. Moreover, the first term represents the utility of $T'_{i=k}$, the second term denotes latency constraint, the second represents computational constraint, the third indicates storage constraint, and finally, the fourth term represents energy constraint.

F. Deep Deterministic Policy Gradient Based Resource Allocation

In contrast to the model-based reinforcement algorithms, which require the complete information of the prior environment, transition probabilities, and reward functions, the model-free Q learning algorithm, i.e., the DDPG, is employed to resolve our utility optimization problem. The model-free Q-learning-based algorithms store the Q values in the form of the collection, which can be iterated to achieve the learning in the case of finite state MDP. The Q-value can be updated as follows:

$$\mathrm{Q}_{\nabla_{i=x}^{td}} = \mathrm{R}\left(A_{\nabla_{i=x}^{td}}, S_{\nabla_{i=x}^{td}}\right) + \Upsilon(A \ , S_{\nabla_{i=x+1}^{td}})$$
$$Q\left(A_{\nabla_{i=x}^{td}}, S_{\nabla_{i=x}^{td}}\right) = Q\left(A_{\nabla_{i=x}^{td}}, S_{\nabla_{i=x}^{td}}\right) + \alpha\,(\mathrm{Q}_{\nabla_{i=x}^{td}} - Q\left(A_{\nabla_{i=x}^{td}}, S_{\nabla_{i=x}^{td}}\right)) \quad (19)$$

To update the policy, we employed a parametrized policy-based method in contrast to the value-based approach owing to our problem's ample continuous action space. The policy-based methods efficiently handle high dimensions and large continuous action space (Wang *et al.,* 2022). In parametrized based policy-based approaches at state S, action A follows the probability distribution with parameter θ; thus, the stochastic policy function π_θ at time stamp $\nabla_{i=x}^{td}$ can be represented as follows

$$\pi(A|S,\theta) = P\{A_{\nabla_{i=x}^{td}} = \alpha | S_{\nabla_{i=x}^{td}} = S, \theta = \theta)$$

As mentioned before, DDPG is a model-free reinforcement learning approach, and we utilized a policy-based approach suitable for large continuous spaces. Moreover, the DDPG follows an actor-critic structure in which the training is based on the actor and critic networks. In the proposed approach, the agent, i.e., the $T'_{i=k}$, interacts with the environment, obtains the data tuple containing the information such as state, action, reward, and following state information, and stores them into a replay buffer. The critic and actor then randomly sample the part of the information from the replay buffer and update the policy function and value function parameters. The DDPG utilizes a soft target update to the target network weights.

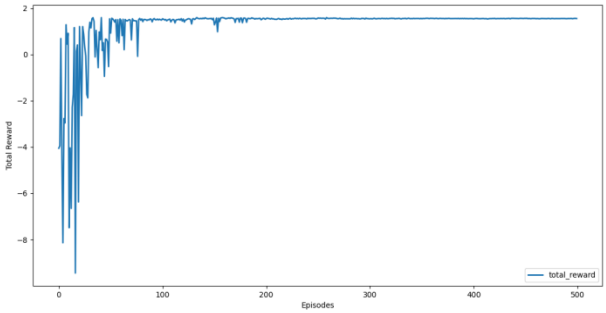

Figure 2: Total Reward as a function of Episodes

VI. EXPERIMENTS AND RESULTS

To measure the effectiveness of the proposed DDPG-based resource allocation algorithm, we implemented the proposed scheme using Python and TensorFlow. We obtained the simulation results as illustrated in the following figures. We selected four different slices where the total resources are as follows: the total amount of bandwidth is 150MHZ, the total computing resources are 225GHZ, the available energy resources are 175J, and the total storage resources are 250GB. Moreover, we varied the resource requirements of each task when requested by the end users. Furthermore, the DDPG setting is as follows. The learning rate for actors in our simulations is 0.001, the learning rate for a critic is also set to 0.001, and the reward discount is 0.9. As mentioned before as the DDPG algorithm follows an actor-critic approach, our implementation, therefore, utilizes two networks i.e., 1) actor and 2) critic in which the actor technically produces the actions to explore whereas the critic decides the accuracy of the actions. Moreover, the actor accuracy is improved based on the information received from the critic network. Besides, DDPG also utilizes a replay buffer to sample experience to revise network parameters. A summary of simulation parameters is presented in Table 1.

In the following, we discuss the results obtained through the simulations.

Figure 2 illustrates the variations of total reward according to the number of episodes. We can see that the proposed DDPG-based resource allocation algorithm converges approximately after 200 episodes. We plot the 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. Therefore, as depicted in Fig. 2, the total reward increases with each episode and stabilizes after approximately 200 episodes. Moreover, Fig. 3 shows that the overall utility of the tenant is increasing, which is the primary goal of our proposed scheme. The utility is basically the profit that a tenant may earn, and the proposed efficient dynamic resource sharing scheme increases the overall tenant's profit by intelligently assigning the resources among slices. Figures 4 and 5 shows the bandwidth resources allocation and computation resources allocations concerning the number of episodes. The end-users requests in slice 1 are

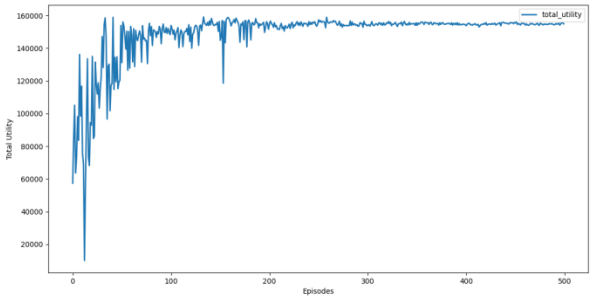

Figure 3: Total Utility as a function of Episodes

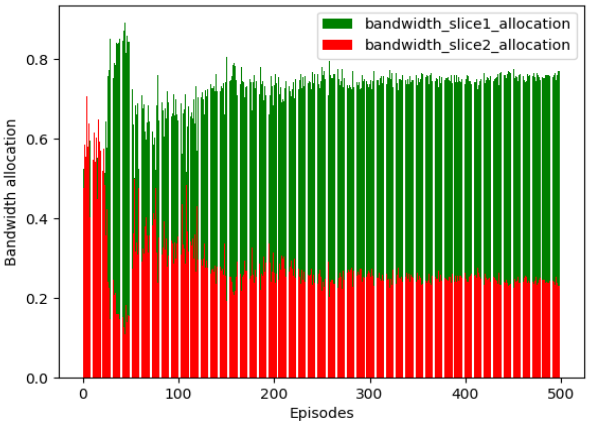

Figure 4: Bandwidth resource allocation as a function of several episodes

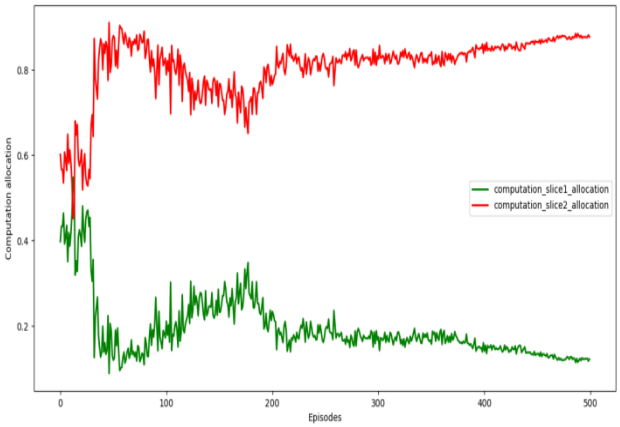

Figure 5 Computations resource allocation as a function of several episodes

bandwidth hungry and thus require high bandwidth compared to the end-users requests in slice 2. Whereas the end-users requests in slice 2 are compute-intensive requests compared to slice 1 end-users requests. Thus, the proposed DDPG dynamic resource allocation algorithm assigned high bandwidth resources to slice 1, whereas the high computation resources are assigned to slice 2. Figure 4 shows that the proposed scheme assigned 70-80% bandwidth resources and approximately 20-30% computation resources to slice 1. Similar trends can be seen in Fig. 5 for compute-intensive resource allocations to slices. The effect of variations in computation capability and bandwidth allocation on the utility gained by the tenant. The size of the computation capability and efficient bandwidth assignment affects the revenue/profit that the tenant obtains from the allocation of computing and bandwidth resources.

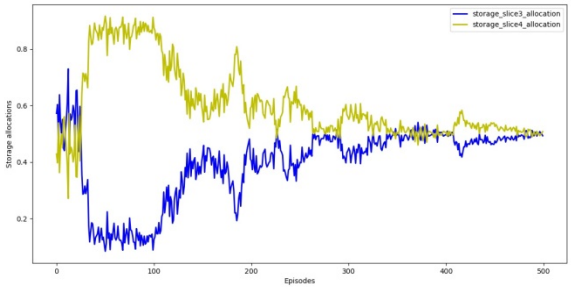

Figure 6: Storage resources allocation for the number of episodes

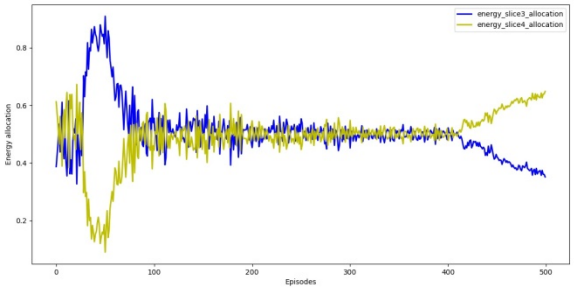

Figure 7: Energy resource allocations to the number of episodes.

Figures 6 and 7 shows the energy resources allocation and storage resources allocations to the number of episodes. The end-users requests in slices 3 and 4 are somewhat following a normal distribution. Approximately half of the requests are energy-intensive requests, whereas the remaining half require storage space. Thus, the proposed DDPG dynamic resource allocation algorithm assigned a relatively equal amount of energy resources and storage resources to slice 3 and 4 respectively. From Figs. 6 and 7, it can observe that approximately 50% of energy and storage resources are assigned to slice 3 and 4. It is to note these resource assignments are purely dynamic and follow the DDPG algorithm. An exception can be seen in Fig. 7, where resource assignment varies 400 episodes which might be owing to the varying request patterns from end-users. Again, the size of the storage capacity and sufficient energy provisioning affects the revenue/profit that the tenant obtains from the allocation of storage and energy resources.

In contrast to the model-based reinforcement algorithms, which require the complete information of the prior environment, transition probabilities, and reward functions, the model-free Q learning algorithm, i.e., the DDPG, is employed to resolve our utility optimization problem. The model-free Q-learning-based algorithms store the Q values in the form of the collection, which can iterate to achieve the learning in the case of finite state MDP.

VII. CONCLUSION

To increase system throughput while meeting all QoS requirements this paper proposes a deep deterministic policy gradient-based dynamic resource allocation framework for multi-access edge computing networking. This paper has several contributions. First, we present a quick primer of fundamental concepts and the related research effort. Second, we outline the system and network model

formulations. Third, complete detail of the proposed DDPG-based dynamic resource allocation is presented. Finally, through simulations in a 5G environment, it is evident that DDPG network slicing that intelligently and dynamically assigns the underlying network and edge resources to slices works significantly better than traditional techniques. We believe that the DDPG network slicing can be used as a helpful tool for upcoming wireless applications, taking into account the long road ahead

REFERENCES

Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A. and Flinck, H. (2018) Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions. *IEEE Communications Surveys and Tutorials*. **20**, 2429–2453.

Alves Esteves, J.J., Boubendir, A., Guillemin, F. and Sens, P. (2021) Controlled Deep Reinforcement Learning for Optimized Slice Placement. *IEEE International Mediterranean Conference on Communications and Networking, MeditCom 2021*. 20–22.

Chen, X., Zhao, Z., Wu, C., Bennis, M., Liu, H., Ji, Y. and Zhang, H. (2018*) Multi-Tenant Cross-Slice Resource Orchestration: A Deep Reinforcement Learning Approach*, 1–30. arxiv.org/abs/1807.09350.

Deng, Z., Du, Q., Li, N. and Zhang, Y. (2019) RL-Based Radio Resource Slicing Strategy for Software-Defined Satellite Networks. *International Conference on Communication Technology Proceedings, ICCT*, 897–901.

Esmat, H.H. and Lorenzo, B. (2020) Deep Reinforcement Learning Based Dynamic Edge/Fog Network Slicing. *IEEE Global Communications Conference, GLOBECOM 2020*. 8–13.

Ferdowsi, A., Abd-Elmagid, M.A., Saad, W. and Dhillon, H.S. (2021) Neural combinatorial deep reinforcement learning for age-optimal joint trajectory and scheduling design in UAV-assisted networks. *IEEE Journal on Selected Areas in Communications,* **39**, 1250-1265.

Gong, S., Zou, Y., Xu, J., Hoang, D. T., Lyu, B. and Niyato, D. (2022) Optimization-driven hierarchical learning framework for wireless powered backscatter-aided relay communications. *IEEE Transactions on Wireless Communications*. **21**, 1378-1391.

Hua, Y., Li, R., Zhao, Z., Chen, X. and Zhang, H. (2020) GAN-powered deep distributional reinforcement learning for resource management in network slicing. *IEEE Journal on Selected Areas in Communications*. **38**, 334-349.

Hurtado Sánchez, J. A., Casilimas, K. and Caicedo Rendon, O.M. (2022) Deep Reinforcement Learning for Resource Management on Network Slicing: A Survey. *Sensors*. **22**, 3031

Khumalo, N.N., Oyerinde, O.O. and Mfupe, L. (2021) Reinforcement learning-based resource management model for fog radio access network architectures in 5G. *IEEE Access*. **9**, 12706-12716.

Korrai, P., Lagunas, E., Sharma, S. K., Chatzinotas, S., Bandi, A. and Ottersten, B. (2020) A RAN resource slicing mechanism for multiplexing of eMBB and URLLC services in OFDMA based 5G wireless networks. *IEEE Access*. **8**, 45674-45688.

Lee, Y.L., Loo, J., Chuah, T.C. and Wang, L.C. (2018) Dynamic network slicing for multitenant heterogeneous cloud radio access networks. *IEEE Transactions on Wireless Communications*. **17**, 2146-2161.

Li, R., Zhao, Z., Sun, Q., Chih-Lin, I., Yang, C., Chen, X., Zhao, M. and Zhang, H. (2018) Deep reinforcement learning for resource management in network slicing. *IEEE Access*. **6**, 74429-74441.

Li, T., Zhu, X. and Liu, X. (2020) An end-to-end network slicing algorithm based on deep Q-learning for 5G network. *IEEE Access.* **8**, 122229-122240.

Liu, Y., Ding, J. and Liu, X. (2021) Resource allocation method for network slicing using constrained reinforcement learning. *2021 IFIP Networking Conference (IFIP Networking)*.

Nassar, A. and Yilmaz, Y. (2019). Reinforcement learning for adaptive resource allocation in fog RAN for IoT with heterogeneous latency requirements. *EEE Access*. **7**, 128014-128025.

Nassar, A. and Yilmaz, Y. (2022) Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities. *IEEE Internet of Things Journal*. **9**, 222-235.

Patel, V., Chesmore, A., Legner, C. M. and Pandey, S. (2022) Trends in Workplace Wearable Technologies and Connected-Worker Solutions for Next-Generation Occupational Safety, Health, and Productivity. *Advanced Intelligent Systems*. **4**, 2100099.

Qi, C., Hua, Y., Li, R., Zhao, Z. and Zhang, H. (2019) Deep reinforcement learning with discrete normalized advantage functions for resource management in network slicing. *IEEE Communications Letters*. **23**, 1337-1341.

Tang, J., Shim, B. and Quek, T.Q. (2019) Service multiplexing and revenue maximization in sliced C-RAN incorporated with URLLC and multicast eMBB. *IEEE Journal on Selected Areas in Communications*. **37**, 881-895.

Tong, Z., Zhang, T., Zhu, Y. and Huang, R. (2020) Communication and computation resource allocation for end-to-end slicing in mobile networks. *2020 IEEE/CIC International Conference on Communications in China (ICCC)*. 1286-1291.

Wang, H., Wu, Y., Min, G., Xu, J. and Tang, P. (2019) Data-driven dynamic resource scheduling for net-

work slicing: A deep reinforcement learning approach. *Information Sciences*. **498**, 106-116.

Wang, X. and Zhang, T. (2019) Reinforcement learning based resource allocation for network slicing in 5g c-ran. *2019 Computing, Communications and IoT Applications (ComComAp)*. 106-111.

Wang, Z., Wei, Y., Yu, F.R. and Han, Z. (2022) Utility Optimization for Resource Allocation in Multi-Access Edge Network Slicing: A Twin-Actor Deep Deterministic Policy Gradient Approach. *IEEE Transactions on Wireless Communications*. **21**, 5842 – 5856.

Wei, F., Feng, G., Sun, Y., Wang, Y., Qin, S. and Liang, Y.C. (2020a) Network slice reconfiguration by exploiting deep reinforcement learning with large action space. *IEEE Transactions on Network and Service Management*. **17**, 2197-2211.

Wei, F., Feng, G., Sun, Y., Wang, Y. and Liang, Y.C. (2020b) Dynamic Network Slice Reconfiguration by Exploiting Deep Reinforcement Learning. *IEEE International Conference on Communications.* 7–11.

Yuan, Y., Zheng, G., Wong, K.K. and Letaief, K.B. (2021) Meta-reinforcement learning based resource allocation for dynamic V2X communications. *IEEE Transactions on Vehicular Technology*. **70**, 8964-8977.

Zhang, H., Xu, S., Zhang, S. and Jiang, Z. (2022a) Slicing Framework for Service Level Agreement Guarantee in Heterogeneous Networks—A Deep Reinforcement Learning Approach. *IEEE Wireless Communications Letters*. **11**, 193-197

Zhang, T., Wu, C., He, Y., Zou, Y. and Liao, C. (2022b) Gain parameters optimization strategy of cross-coupled controller based on deep reinforcement learning. *Engineering Optimization*. **54**, 727-742.

Zhou, L., Zhang, T., Li, J. and Zhu, Y. (2020) Radio resource allocation for RAN slicing in mobile networks. *2020 IEEE/CIC International Conference on Communications in China (ICCC).* 1280-1285.

Received: June 10, 2022
Sent to Subject Editor: July 14, 2022
Accepted: August 24, 2022
Recommended by Subject Editor David Zumoffen