Generalizable GNN-based 5G RAN/MEC Slicing and Admission Control in Metropolitan Networks

Arash Moayyedi*, Mahdieh Ahmadi*, Mohammad A. Salahuddin*, Raouf Boutaba*, and Aladdin Saleh†
*David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada
{arash.moayyedi, mahdieh.ahmadi, mohammad.salahuddin, rboutaba}@uwaterloo.ca
†Rogers Communications Inc., Ontario, Canada, {aladdin.saleh@rci.rogers.com}

***Abstract*—The 5G RAN functions can be virtualized and distributed across the radio unit (RU), distributed unit (DU), and centralized unit (CU) to facilitate flexible resource management. Complemented by multi-access edge computing (MEC), these components create network slices tailored for applications with diverse quality of service (QoS) requirements. However, as the requests for various slices arrive dynamically over time and the network resources are limited, it is non-trivial for an infrastructure provider (InP) to optimize its long-term revenue from real-time admission and embedding of slice requests. Prior works have leveraged Deep Reinforcement Learning (DRL) to address this problem, however, these solutions either require re-training when facing topology changes or do not consider the slice admission and embedding problems jointly. In this paper, we use multi-agent DRL and Graph Attention Networks (GATs) to address these limitations. Specifically, we propose novel topology-independent admission and slicing agents that are scalable and generalizable to large and different metropolitan networks. Results show that the proposed approach converges faster and achieves up to 35.2% and 20% gain in revenue compared to heuristics and other DRL-based approaches, respectively. Additionally, we demonstrate that our approach is generalizable to scenarios and substrate networks previously unseen during training, as it maintains superior performance without re-training or re-tuning.**

***Index Terms*—5G RAN, MEC, network slicing, Deep Reinforcement Learning, Graph Attention Networks**

I. Introduction

The decomposition of the 5G radio access network (RAN) plays a key role in the 5G New Radio (NR) architecture. The RAN protocol stack, *i.e.*, baseband unit (BBU) functions in 4G, can be split into radio unit (RU), distributed unit (DU), and central unit (CU), which provides flexibility in distributing RAN functions to meet latency and bandwidth requirements of different services. Multi-access edge computing (MEC) can further facilitate stringent quality of service (QoS) guarantees by placing application servers close to end-users. By leveraging Network Function Virtualization (NFV), the 5G infrastructure provider (InP) can cater to a variety of MEC-enabled use-cases with variable QoS requirements, providing end-to-end (E2E) isolated and differentiated networks for each application, *i.e.*, *network slices*. However, to achieve efficiency and revenue gains from network slicing, coordinated and dynamic management of the entire 5G system is imperative, including the RAN, MEC, core, and transport networks connecting them.

Metropolitan 5G networks usually consist of multi-level nodes—access, aggregation, and core—interconnected by mesh-like or multi-ring topologies [1, 2]. Connected to the cell sites are the access nodes, which are linked to the core by aggregation nodes. Centralized sites and links offer more resources, but incur more transport latency and bandwidth as unprocessed data traverses to the higher-layer sites. From the InP's perspective, centralizing RAN and MEC virtual network functions (VNFs) can increase multiplexing gains (*e.g.*, power consumption and maintenance [3]), and prevent computing bottlenecks at the access, provided that the delay constraints and bandwidth requirements of individual VNFs are met. Therefore, to ensure the optimal placement of VNFs, technical and cost-effective trade-offs between throughput, latency, and centralization must be considered.

MEC is best suited to services requiring very high bandwidth and ultra-low latency, such as virtual reality and mission-critical applications. Requests for these services are generated dynamically over time by *slice tenants*, such as service providers. Slices consist of RAN (*i.e.*, RU, DU, CU) and MEC VNFs, along with virtual links (VLs) that connect them, collectively referred to as Virtual Networks (VNs). However, the addition of MEC complicates the RAN slicing problem by imposing the E2E MEC service latency requirements on top of the latency requirements of individual RAN components. InPs generate revenue from accepting and accommodating network slice requests, but not all requests can be met due to resource limitations in the substrate network. Moreover, the revenue earned for each request may vary depending on factors such as the tenant's subscription and QoS requirements. Therefore, to maximize the long-term revenue, the InP should: i) perform admission control (AC) in a way that maximizes the long-term revenue, ii) place the RAN and MEC VNFs according to their delay and processing requirements, and iii) route traffic from the originating cell site to the core, which is the Internet's interface to the mobile infrastructure. We refer to the latter two (*i.e.*, ii and iii) decisions as slicing.

The problem of RAN/MEC slicing has previously been investigated in an offline manner [4, 5, 6]. Recently, deep reinforcement learning (DRL) has shown promising performance in the RAN slicing [7] and AC problems [8, 9] in an online, but disjoint setting. Sulaiman et al. [10, 11] designed a joint AC and RAN slicing solution for a simple multi-tier network using multi-agent DRL. However, they use multi-layer perceptron (MLP) models for both slicing and AC which limits

scalability and generalizability across different networks. With MLP architectures, information about the network state, which includes the topology, and features of all nodes and links, is passed to the model as a whole. Moreover, the formats of these feature vectors are defined during model initialization and training. As a result, with the slightest topological variation, *e.g.*, node/link failure or network expansion, the model becomes obsolete and requires re-training. Training a deep neural network from scratch can be prohibitive, especially in a highly dynamic environment, or under tight service delay requirements.

In this work, we integrate multi-agent DRL with Graph Neural Network (GNN), which provides promising topology-independent feature extraction capabilities [12]. More specifically, we use a recent variant of the popular Graph Attention Networks (GATs) [13], GATv2 [14], which is a spatial-based GNN model (*cf.*, Section IV-A). Spatial-based methods are popular for their efficiency, flexibility, and generalizability and work by propagating node features across edges [12]. In GATs, an attention mechanism is used that effectively exploits the structural characteristics of networks by learning how each node contributes to its neighbours. In addition, different from all other GNNs, they can take the edge features directly into account. Utilizing GATv2 and a novel learning model, we propose a generalizable RAN/MEC slicing DRL agent. For AC, we employ an MLP with an input that consists of features of the nodes and edges where VNFs and VLs are respectively placed, and the slice request information. Such input is independent of the topology, so the MLP-based AC agent can operate in presence of changing network conditions. The main contributions of the paper are:

- The problem of joint AC and MEC/RAN slicing in 5G metropolitan networks under E2E service delay and resource constraints is modeled as an integer linear programming (ILP) problem and proven to be NP-hard.
- A novel solution for the joint online AC and slicing problem is proposed using multi-agent DRL. For slicing, a generalizable GNN-based DRL agent, and for AC, a topology-independent MLP is devised, which allows both of them to operate on arbitrary topologies.
- The proposed approach is evaluated and compared to greedy and state-of-the-art heuristics, and DRL-based solutions. Our model outperforms the baselines by up to 35.2% in the overall revenue gain.
- The robustness and generalization of our slicing and AC agents are evaluated under varying network conditions. The agents outperform other heuristic approaches by up to 25.5% even in previously unseen network topologies.

The paper is organized as follows. Section II reviews the state of the art. Section III provides background on RAN slicing and mathematically formulates the joint AC and MEC/RAN slicing problem. The proposed method is introduced in Section IV and evaluated in Section V. Finally, Section VI concludes the paper.

II. Related Work

A. RAN Slicing

Yu et al. [15] were the first to investigate the 3-layer RAN slicing in the context of metro/aggregation networks. They analyzed the problem of CU/DU placement and routing to minimize the number of central offices (COs) housing the functions under delay and capacity constraints. The authors showed that the increased flexibility of a 3-layer RAN architecture leads to a higher consolidation of COs. Based on the same architecture, Yu et al. [16] investigated isolation-aware slicing and proposed a heuristic for minimizing the number of active COs or wavelengths under isolation and latency constraints. Marotta et al. [17] addressed the same problem but also took into account the reliability requirements of different slices. All of these approaches place VNFs using a heuristic method, and then use variants of the shortest-path algorithm for routing. In addition, they work in an offline setting, *i.e.*, an objective function is optimized over all the requests.

Gao et al. [7] developed a DRL-based method for online RAN function placement and routing with the objective of minimizing the number of active COs, bandwidth, and transport latency. However, they did not consider E2E service latency constraints and slices with finite operation time. The works in [4, 5, 6] investigated function placement in the context of MEC-enabled RAN. They modeled the problem of minimizing the operational cost under delay and capacity constraints as an ILP and solved it using Benders Decomposition [4, 5] and DRL [6]. However, in these methods, the problem is considered in an offline setting and placement is decided for each cell instead of each request. The aforementioned works also do not take AC into consideration. Sulaiman et al. [10, 11] proposed an online joint RAN slicing and AC solution under E2E service delay and resource constraints using multi-agent DRL. They showed that the AC mechanism can lead to higher revenue by preemptively rejecting low-priority slice requests. However, their method is not scalable and generalizable to large and previously unseen substrate networks.

B. Learning using GNN

GNN-based architectures have been employed in networking for applications such as Virtual Network Embedding (VNE) [18], traffic routing [19], and congestion prediction [20], to capture the spatial information hidden in the network topology [18] and benefit from its generalizability to different topologies [19, 20]. Among these works, the problem of RAN slicing shares the most similarities with VNE, which is a resource allocation problem involving mapping a VN onto the substrate network. Therefore, we discuss these works here. Yan et al. [18] were the first to apply DRL with GNNs, specifically Graph Convolutional Network (GCN), to the VNE problem and showed that it can lead to a higher acceptance ratio. The DRL agent places the VNFs one-by-one on substrate nodes with sufficient CPU and bandwidth, and then, the shortest-path algorithm is utilized for routing between the selected nodes.

Zhang et al. [21] investigated the same problem when VNs can dynamically change over time. However, GNN-based

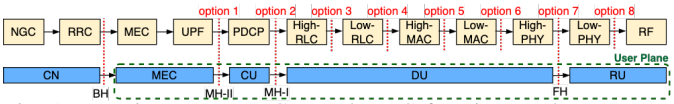


Fig. 1: Mapping RU, DU, CU, and MEC functions and FH, MH-I, MH-II, and BH networks to the split points [24, 25, 26]

DRL is solely used for the initial mapping and a heuristic is leveraged to remap the VN when a change happens. Esteves et al. [22] extended the approach in [18] by considering more resource types for each VNF and using heuristically assisted DRL for faster convergence. However, the use of spectral-based GNN models (*cf.*, Section IV-A) and dense layers in the learning model limit the applicability of these works to static network topologies. Habibi et al. [23] employed a spatial-based graph autoencoder to cluster similar substrate nodes based on their resources and accessibility. However, the final node and link embeddings across clusters are decided using a Breadth-First Search (BFS) algorithm which is not efficient.

In contrast to the aforementioned VNE solutions, in this paper, we use an attention-based GNN, namely GATv2 [14], combined with DRL for slicing. The use of GATv2 and the employed learning architecture allows us to operate on unseen topologies without retraining. Furthermore, we incorporate an AC mechanism and take the E2E service delay constraint into account, which is crucial for delay-sensitive applications.

III. System Model and Problem Statement

A. RAN and MEC Architecture

Fig. 1 shows RAN functions and different split options proposed by 3GPP [24]. Aligned with 3GPP and ITU-T recommendation [25], in this paper, we consider options 2 and 7 for higher layer (CU/DU) and lower layer (DU/RU) splits, respectively, and use the terms midhaul-I (MH-I) and fronthaul (FH) to refer to the corresponding transport network segments. In addition, we assume that the CU includes only the user-plane PDCP function and control-plane functions are placed in the core network (CN). This configuration is particularly useful for applications that do not require rapid call establishment but require a low user-plane delay, *e.g.*, cloud gaming [1]. MEC can be integrated with the 5G network in various ways [26]. We assume that the MEC includes edge application, as well as local user-plane functions (UPF) and is connected through the midhaul-II (MH-II) and backhaul (BH) network to the CU and CN, respectively. Finally, CN is connected to the Internet.

For each network slice, the individual RU, DU, CU, and MEC entities are virtualized and placed at different physical locations according to their latency and resource requirements, and residual capacity in the substrate network. We assume RU is placed on the access node connected directly to the originating cell and CN is placed on the core node, which has abundant capacity, and therefore, we only decide the placement of DU, CU, and MEC, and routing from the access site to the core node. The specifications of MEC depend on the type of service, but we use the following formulations to calculate the computation ($c^{\text{cpu}}_{\text{DU}}$ and $c^{\text{cpu}}_{\text{CU}}$ in Giga Operations Per Second (GOPS)) and bandwidth (λ_l in Mbps) requirements of RAN components and their interconnecting transport segments, respectively.

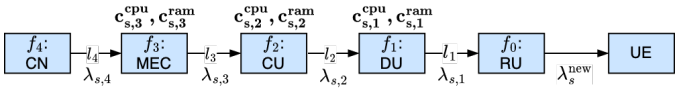


Fig. 2: Virtual network model of service s

$$c^{\text{cpu}}_{\text{DU}} = k^{\text{DU}}_1 BA^2L + k^{\text{DU}}_2 BALM + k^{\text{DU}}_3 A, \quad (1)$$

$$c^{\text{cpu}}_{\text{CU}} = k^{\text{CU}}_1 A, \quad (2)$$

$$\lambda_l = k^l_1 \lambda^{\text{new}} + k^l_2, \forall l \in \{\text{FH}, \text{MH-I}, \text{MH-II}, \text{BH}\}, \quad (3)$$

where k parameters in (3) and (1)-(2) are constant coefficients specific to different RAN functions whose details can be found in [27, 28] and [29], respectively. λ^{new}, B, A, L, and M represent the service traffic, carrier bandwidth, number of antennas, traffic load, and modulation (in bits per symbol). While MH-I (*i.e.*, $\lambda_{\text{MH-I}}$) and MH-II (*i.e.*, $\lambda_{\text{MH-II}}$) bandwidth requirements for each slice request are almost equal to λ^{new}, FH interface (*i.e.*, λ_{FH}) requires considerably higher bandwidth. The bandwidth requirement at the BH network (*i.e.*, λ_{BH}) depends on the amount of Internet traffic of each type of service. Moreover, each RAN component has a specific delay requirement. From the RAN perspective, for DU, depending on the specific vendor implementation of the HARQ loop, it can be up to 2 ms for the case where interleaving is done, and for CU, it could be up to 6 ms [28]. However, since DU and CU are before MEC in the user-plane, their delay requirements depend on both the respective RAN component and the E2E service latency, *i.e.*, the minimum value should be considered.

B. System Model

Substrate network: We consider a network architecture consisting of $\mathcal{N}$ access and aggregation sites equipped with dedicated processing capabilities. Access nodes can be connected to one or multiple cell sites. We also have a core[1] node with abundant resources which can host the CN. These nodes are connected through an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the union of $\mathcal{N}$ and core node (index 0), and $\mathcal{E}$ is the set of all physical links. We denote by C^{cpu}_v and C^{ram}_v the maximum computing and RAM resource capacities of each node $v \in \mathcal{V}$, respectively. Also, each link $e \in \mathcal{E}$ has a certain bandwidth capacity, B_e, and delay, d_e.

Services: We consider a set of $\mathcal{S}$ services, *i.e.*, MEC applications. Throughput and E2E service delay constitute QoS metrics, and are denoted by λ^{new}_s and D^{srv}_s, respectively. RAN and MEC form a VN consisting of a chain of five VNFs, $\mathcal{F} = \{f_0, \cdots, f_4\}$, namely, RU, DU, CU, MEC, and CN, and four VLs $\mathcal{L} = \{l_1, \cdots, l_4\}$, namely, FH, MH-I, MH-II, and BH (*cf.*, Fig. 2). For service s, we can denote the computing and memory resource requirements of VNF f, and bandwidth requirement on VL l by $c^{\text{cpu}}_{s,f}$, $c^{\text{ram}}_{s,f}$, and $\lambda_{s,l}$, respectively. The CPU and bandwidth requirements can be calculated according to equations (1)-(3) for RAN VNFs. We assume that RAM requirements follow the same pattern as

[1]The formulation can trivially be extended to networks with more than one core node.

computation requirements for RAN VNFs. For MEC, these requirements depend on the computational complexity of the specific application. The delay requirement of VNF f is also shown by $D_{s,f}$, which for MEC is equal to D_s^{srv}.

Slice requests (SRs): Requests arrive for different services over time. We characterize each generic SR k by its service type, the access node on which this request was first submitted, n_k^{src}, offered revenue, p_k, and the set of time slots it needs to receive service (*i.e.*, operation time), $\mathcal{T}_k = \{t_k^{\text{arv}}, \cdots, t_k^{\text{arv}} + \tau_k - 1\}$. We analyze the system over period $\mathcal{T}$ and denote the set of all SRs for service s and for all the services that arrive over this period by $\mathcal{K}_s$ and $\mathcal{K}$, respectively.

C. Problem Formulation

Given the knowledge of future SRs, the offline RAN slicing and AC problem should decide about the admission of each SR and the embedding of its corresponding VN.

Decision variables: Let $\alpha = [\alpha_k]_{k\in\mathcal{K}}$ denote admission matrix, where $\alpha_k \in \{0,1\}$ denote whether SR k is admitted ($\alpha_k = 1$) or not ($\alpha_k = 0$). VNF embedding decisions are defined by the matrix $\mathbf{X} = [x_{f,v}^k]_{r\in\mathcal{R},f\in\mathcal{F},v\in\mathcal{V}}$, where $x_{f,v}^k \in \{0,1\}$ indicate whether VNF f of SR k has been placed on physical node v ($x_{f,v}^k = 1$) or not ($x_{f,v}^k = 0$). In this work, we consider single-path routing, *i.e.*, each VL is mapped to one physical path. Let $q \in \mathcal{Q}$ denote a simple path in the physical network. A path is a sequence of links between two nodes $src(q), dst(q) \in \mathcal{V}$. We assume $\emptyset \in \mathcal{Q}$ for which $src(q) = dst(q)$, to consider co-location of VNFs. So, we define matrix $\mathbf{Y} = [y_{l,q}^k]_{k\in\mathcal{K},l\in\mathcal{L},q\in\mathcal{Q}}$ to describe VL embedding decisions, where $y_{l,q}^k \in \{0,1\}$ indicates whether the traffic of VL l in SR k has passed through q ($y_{l,q}^k = 1$) or not ($y_{l,q}^k = 0$).

Admission and embedding constraint: If SR k is admitted, it will remain in the system throughout its operation time, $t \in \mathcal{T}_k$, and the corresponding VN should be embedded into the substrate network. The following constraint ensures the embedding of each VNF and VL of an SR on one physical node and one physical path, respectively, if and only if it is admitted:

$$\alpha_k = \sum_{v\in\mathcal{V}_{k,f}} x_{f,v}^k = \sum_{q\in\mathcal{Q}} y_{l,q}^k, \ \forall k, f, \tag{4}$$

where $\mathcal{V}_{k,f} \subseteq \mathcal{V}$ includes the physical nodes on which VNF f of SR k can be placed. As discussed, the placement is decided for DU, CU, and MEC for which $\mathcal{V}_{k,1} = \mathcal{V}_{k,2} = \mathcal{V}_{k,3} = \mathcal{V}, \forall k$. However, for the sake of formulation, we also define VNF embedding variables for CN and RU and set $\mathcal{V}_{k,4} = \{0\}$ and $\mathcal{V}_{k,0} = \{n_k^{\text{src}}\}$ to impose the placement of CN and RU on the core node and originating access site, respectively.

Routing constraints: The problem of selecting paths for embedding each VL can be framed as the well-known *unsplittable multi-commodity flow* problem [30]. Therefore, routing variables specified by VL embedding matrix, $\mathbf{Y}$, should meet the below flow conservation constraint, where $src(l), dst(l) \in \mathcal{F}$ are source and destination VNFs of VL l:

$$\sum_{\substack{q\in\mathcal{Q}:\\ src(q)=v}} y_{l,q}^k - \sum_{\substack{q\in\mathcal{Q}:\\ dst(q)=v}} y_{l,q}^k = x_{src(l),v}^k - x_{dst(l),v}^k, \ \forall k, v, l. \tag{5}$$

Capacity constraints: The assigned resources to VNFs and VLs should not exceed the capacity of nodes and links in the substrate network. These constraints are expressed by

$$\sum_{s\in\mathcal{S}} \sum_{\substack{k\in\mathcal{K}_s\\ :t\in\mathcal{T}_k}} \sum_{f\in\mathcal{F}} c_{s,f}^{\text{cpu}} x_{f,v}^k \le C_v^{\text{cpu}}, \ \forall v, t \in \mathcal{T}, \tag{6}$$

$$\sum_{s\in\mathcal{S}} \sum_{\substack{k\in\mathcal{K}_s\\ :t\in\mathcal{T}_k}} \sum_{f\in\mathcal{F}} c_{s,f}^{\text{ram}} x_{f,v}^k \le C_v^{\text{ram}}, \ \forall v, t \in \mathcal{T}, \tag{7}$$

$$\sum_{s\in\mathcal{S}} \sum_{\substack{k\in\mathcal{K}_s\\ :t\in\mathcal{T}_k}} \sum_{l\in\mathcal{L}} \sum_{\substack{q\in\mathcal{Q}\\ :e\in q}} \lambda_{s,l} y_{l,q}^k \le B_e, \ \forall e, t \in \mathcal{T}. \tag{8}$$

In the above expressions, the total used resources at time t are computed as the sum of resource requirements of active embedded SRs at that time, *i.e.*, each SR k for which $t \in \mathcal{T}_k$.

Delay constraints: Finally, E2E service delay and individual VNF latency constraints for each SR are expressed by

$$\sum_{\substack{l\in\mathcal{L}:\\ l\le f}} \sum_{q\in\mathcal{Q}} y_{l,q}^k \sum_{e\in q} d_e \le D_{s,f}, \ \forall s, k \in \mathcal{K}_s, 1 \le f \le 3. \tag{9}$$

Objective: The goal of the InP is to grant SRs that lead to the highest long-term revenue. We can formulate the problem of RAN slicing and AC while optimizing the revenue of InP over all the requests (or period $\mathcal{T}$) as

$$\max_{\alpha,\mathbf{X},\mathbf{Y}} \sum_{k\in\mathcal{K}} p_k \tau_k \alpha_k \quad \text{subject to} \quad (4) - (9).$$

The above problem is NP-hard [31] and the information about SRs is not known in advance. To address this, we propose using a DRL-based approach to tackle the problem in an online setting, without requiring prior knowledge of SRs.

IV. GNN-based Multi-agent DRL Framework

The online scenario requires that SRs be handled one-by-one as they arrive, such that the long-term revenue of InP is maximized. When a request arrives, the online DRL-based solution should decide on its admission and VN embedding based on the current network state. To maximize InP revenue, the agent should be rewarded for successful embeddings and revenue generation. However, such reward design may lead to unintended behaviour, as it is not clear whether the lost revenues are a consequence of suboptimal previous admission or embedding actions. To tackle this issue, similar to [10, 11], we use two agents, slicing and AC, in a coordinated manner. In our proposed solution, called GNN-AC-SL, first the slicing agent optimizes the embedding based on the SR's specifications and network state to maximize the number of embedded requests. Note that to reduce action space for the slicing agent, we decompose VN embedding into a sequence of four VNF embeddings. The AC agent then decides whether to accept the SR based on the embedding decision, the system state, and the SR's information, so that long-term revenue is maximized. This section will provide a background on GNNs, followed by a detailed explanation of the two DRL-based agents.

A. Graph Neural Networks

GNNs are neural models designed to operate directly on graph-structured data and have numerous variants. The main goal of a GNN is to learn a low-dimensional vector representation for each node h_v, which can be used for different learning tasks. ConvGNNs are a popular variant of GNNs that extend the convolutional operator to graphs and have two categories: spectral-based and spatial-based [12]. The first category performs graph convolutional operation on the entire graph at once in the Fourier domain and thus suffers from poor scalability and generalizability. Spatial-based GNNs address these limitations by implementing graph convolutions using message-passing between the neighboring nodes of the graph. At each layer, first input graph nodes' features/representations $\left\{h_v \in \mathbb{R}^F \,|\, v \in \mathcal{V}\right\}$ are aggregated with those of their neighbours. The combined representations are then passed through a transformation function g (*e.g.,* a dense layer and a non-linearity) to output new representations $\left\{h'_v \in \mathbb{R}^{F'} \,|\, v \in \mathcal{V}\right\}$. Specifically, for each node, we perform

$$h'_v = g\left(h_v, \texttt{AGGREGATE}\left(\{h_u \,|\, u \in \mathcal{N}_v\}\right)\right), \tag{10}$$

where $\mathcal{N}_v$ is the set of neighbours of node v and `AGGREGATE` can be any permutation invariant function, *e.g.*, mean. The selection of g and `AGGREGATE` in the update process contributes the most variance among different spatial-based models [14].

Many GNN models assume the contributions of the neighbouring nodes on the central node's representation are either identical or pre-determined (*e.g.*, [23]) in the aggregation process. In GAT [13], however, a learned attention layer is used to output a representation based on the weighted average of neighbours' representations. In addition, this method allows us to consider edge features $h_{uv} \in \mathbb{R}^{F_e}$ by including them in the input of the attention layer. In this paper, we use GATv2 which has a simple adjustment to the way attention is calculated in GAT and has been shown to be theoretically and empirically superior [14]. The attention mechanism which is defined by scoring function $e : \mathbb{R}^F \times \mathbb{R}^{F'} \to \mathbb{R}$ in this method calculates the relative importance of the features of the neighbor u to the node v as

$$\text{GATv2} : e(h_v, h_u) = \boldsymbol{a}^\top \text{LeakyReLU}\left(\boldsymbol{W}.\left[h_v \| h_u \| h_{vu}\right]\right), \tag{11}$$

where $\boldsymbol{a}$ and $\boldsymbol{W}$ are learned and $\|$ is the concatenation operator. Using the Softmax function, attention scores are then normalized across all neighbours and used to calculate a new representation for each node, by a weighted average followed by a non-linearity (σ):

$$h'_v = \sigma\left(\sum_{u \in \mathcal{N}_v} \text{softmax}_{\mathcal{N}_v}\left(e(h_v, h_u)\right).\boldsymbol{W} h_u\right). \tag{12}$$

Note that once the parameters of the attention mechanism and linear transformation, *i.e.*, $\boldsymbol{a}$ and $\boldsymbol{W}$, are learned, there is no need for re-training with each topological variation and only (11) and (12) should be recalculated for affected nodes.

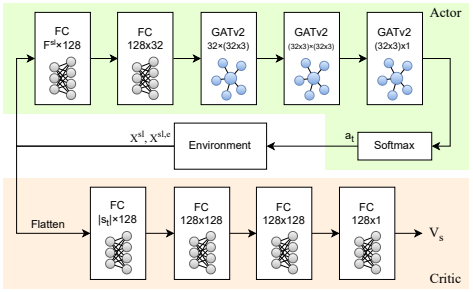


Fig. 3: Slicing agent model architecture

B. RL Environment

State (s_t): For the slicing agent, we represent the state of the whole system as a graph, $s_t = (\mathcal{V}, \mathcal{E})$, with node and edge attributes. Since VNFs will be placed one-by-one, the current VNF should also be considered in the state of the system. We define the node features by $\mathbf{X}^{\text{sl}} = [X_v^{\text{sl}}]_{v \in \mathcal{V}}$, where $X_v^{\text{sl}} \in \mathbb{R}^{F^{\text{sl}}}$ is the feature vector of node v and consists of its maximum and remaining CPU and RAM, its level in the substrate network (*i.e.*, access, aggregate or core), and the delay and remaining bandwidth from the last placed VNF.

X_v^{sl} also includes features of the current SR and its VNFs, namely the operation time, the index and the delay budget of the current VNF, and the required CPU and RAM by the last and current VNFs. Similarly, the matrix $\mathbf{X}^{\text{sl,e}} = [X_{vu}^{\text{sl,e}}]_{(v,u) \in \mathcal{E}}$ represents the edge features, where $X_{vu}^{\text{sl,e}} \in \mathbb{R}^{F^{\text{sl,e}}}$ includes the maximum and remaining bandwidth capacity, and link delay for the edge between nodes v and u.

Different from the slicing agent, the AC agent only considers the hosting nodes and edges of the substrate network based on the given embedding decided by the slicing agent. The input feature of the AC agent is an F^{ac}-dimensional vector $\mathbf{X}^{\text{ac}} \in \mathbb{R}^{F^{\text{ac}}}$ which includes the hosting nodes' maximum and remaining CPU and RAM, level in the substrate network, and the requested CPU and RAM by the embedding along with embedding's sum of requested bandwidth across the substrate network, and the SR's operation time and revenue.

Action (a_t): AC agent's action space is equal to $\mathcal{A}^{\text{ac}} = \{0, 1\}$, corresponding to SR rejection and admission. For each SR, we embed four VNFs in a sequence and VL associated with each VNF is embedded using a shortest-path algorithm. Thus, for the slicing agent, the set of all possible actions is limited to the number of substrate nodes, *i.e.*, $\mathcal{A}^{\text{sl}} = \mathcal{V}$.

Reward (r_t): A total reward of +1 is given to the slicing agent for the successful embedding of all four RAN functions, and partial embedding is not rewarded for the agent to learn how to embed the requests in their complete form. Using this system, a rewarding decision includes up to four steps until an SR embedding succeeds or fails. Likewise, the AC agent receives +1 reward for admission of each SR.

C. Training Algorithm

We train both AC and slicing agents together as a multi-agent RL scenario and utilize the Proximal Policy Optimization (PPO) algorithm [32], which is one of the leading policy-gradient DRL methods. This method has an actor-critic architecture, wherein, the actor generates the probabilities of

Algorithm 1: Slicing and Admission Control

Input: action a_t, current agent δ_{curr}
Output: next state s_{t+1}, reward r_t, next agent δ_{next}

```
1  r_t ← 0
2  if δ_curr ==“Slicing” then
3      if checkActionFeasibility(a_t) then
4          embedding.add(a_t)
5          if len(embedding) == 4 then  // SR embedding is complete
6              r_t ← 1
7              δ_next ←“Admission Control”
8          else
9              moveToNextVNF()
10             δ_next ← “Slicing”
11     else
12         moveToNextSliceRequest()          // infeasible SR
13         embedding ← ∅
14         δ_next ←“Slicing”
15 else if δ_curr ==“Admission Control” then
16     if a_t == 1 then                      // SR is admitted for deployment
17         deploy(embedding)
18         r_t ← revenue
19     δ_next ←“Slicing”, go to Line 12
20 s_{t+1} ← readState()
21 return s_{t+1}, r_t, δ_next
```

selecting different actions in any given state with a neural network (NN) parameterized by θ, *i.e.*, $\pi_\theta(a|s_t)$, and the critic estimates the expected reward of starting in any state s_t with a NN parameterized by ϕ, *i.e.*, $v_\phi(s_t)$. At each training iteration k, first the set of trajectories $\mathcal{D}_k = \{m_i\}$, where $m_i = \{s_0, a_0, r_0, \cdots, s_T, a_T, r_T\}$ are collected for both agents by running action a_t sampled from policy π_{θ_k} at state s_t and receiving reward r_t and next state s_{t+1} according to Alg. 1. In this algorithm, the environment stores the actions until the SR's embedding is complete or the last action violates a constraint. Then, the parameters of the actor and critic networks are updated as

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}} \frac{1}{|\mathcal{D}_k|T} \sum_{m\in\mathcal{D}_k}\sum_{t=0}^{T} \min \left(r_t(\theta)\hat{A}^{\pi_{\theta_k}}(s_t,a_t), clip(\epsilon, r_t(\theta))\hat{A}^{\pi_{\theta_k}}(s_t,a_t)\right), \quad (13)$$

$$\phi_{k+1} = \underset{\phi}{\operatorname{argmin}} \frac{1}{|\mathcal{D}_k|T} \sum_{m\in\mathcal{D}_k}\sum_{t=0}^{T} \left(v_\phi(s_t) - \hat{R}_t\right)^2, \quad (14)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$, $\hat{R}_t$ is the discounted cumulative reward, and $\hat{A}_t$ is the generalized advantage estimate [33].

The NN architecture of actor and critic modules for the slicing agent is shown in Fig. 3. First, the node-wise input is passed through two fully connected (FC) layers to create embeddings used by the GNN layers. Although these layers are FC, they operate on a per-node basis, meaning network information is fed to them as a batch of single-node feature vectors, X_v^{sl}, which has a constant size F^{sl}. Such design is compatible with the dynamicity of our model. Next, three GATv2 layers are used, each with three attention heads of size 32. Finally, the model outputs a single value per node, and a Softmax function is used to determine the node-selection probability distribution. Due to the simpler objective of the critic model and the fact that it is only used during the training phase in which the topology is fixed, we use an MLP model for it which has a faster convergence time. The model is fed with the flattened $|s_t|$-dimensional representation of the state s_t that is stripped of some repetitive features. In the same vein, as explained in Section IV-B, since the input dimensions of the AC agent remain static regardless of the substrate network size and it has a small action space of size 2, we opt for a less complex MLP-based architecture for the AC agent which consists of four fully connected layers.

TABLE I: MEC applications with characteristics [34]

Application	λ^{new} (Mbps)	MEC CPU (GOPS)	MEC RAM (GiB)	MEC Latency (ms)	Backhaul Traffic (Mbps)	Revenue
Remote Surgery (RS)	20	200	10	1	10	$\mathcal{N}(80, 5)$
Cloud Gaming (CG)	100	1500	30	5	30	$\mathcal{N}(80, 10)$
Virtual Reality (VR)	100	2000	60	10	30	$\mathcal{N}(100, 10)$
Video Streaming (VS)	200	150	10	200	60	$\mathcal{N}(70, 10)$

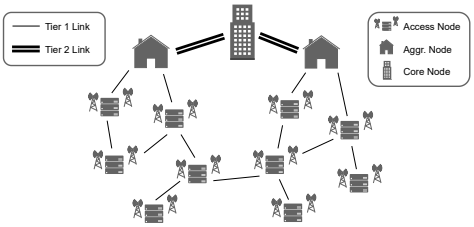


Fig. 4: Baseline metropolitan 5G network used for training

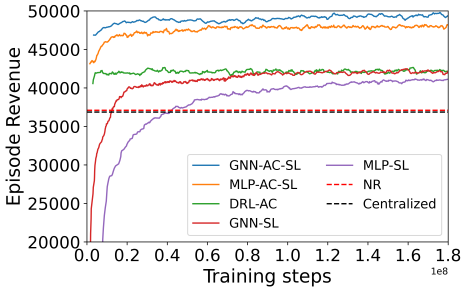
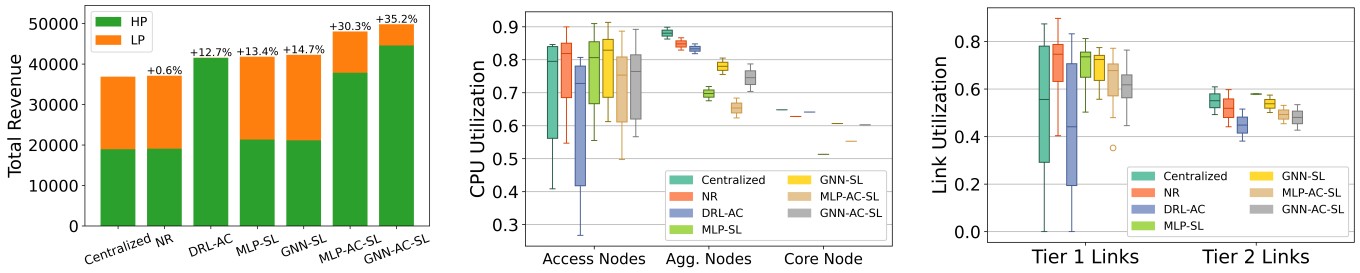


Fig. 5: Episode revenue during training

V. Performance Evaluation

Simulation setup: We consider four delay-sensitive MEC applications shown in Table I with corresponding required throughput, MEC delay, CPU and RAM usage of MEC VNF, and backhaul traffic [34]. Parameters of other VNFs and VLs in the VN are derived as discussed in Section III with RU configuration of 20MHz, 4x4 MIMO, and 64QAM. A simulation is run for 2000 time units with random SRs arriving heterogeneously at different access nodes with a total rate of 1 SR per time unit. Each SR has an operation time following $\mathcal{N}(300, 25)$ and is uniformly assigned to a MEC application and to a priority class (*i.e.*, high-priority (HP) or low-priority (LP)). A HP SR offers twice the revenue of a LP SR. We train our agents on the topology shown in Fig. 4, which comprises 10 access nodes, 2 aggregation nodes, and 1 core node. CPU capacities are equal to 4000, 6000, and 12000 GOPS, and RAM capacities are 100, 150, and 300 GB, for access, aggregation, and core nodes, respectively. Links connecting access nodes to each other and to aggregation nodes are Tier 1 links (*cf.*, Fig. 4) and have a capacity of 2 Gbps and a transmission delay of 1.8 ms. Other links are called Tier 2 links and have a capacity of 3 Gbps and a delay of 4.8 ms [1]. The same network and SR parameters were used



(a) InP revenue with improvement relative to Centralized

(b) Node CPU utilization. Note that there is only one core node.

(c) Link utilization

Fig. 6: Evaluation results on the baseline training network

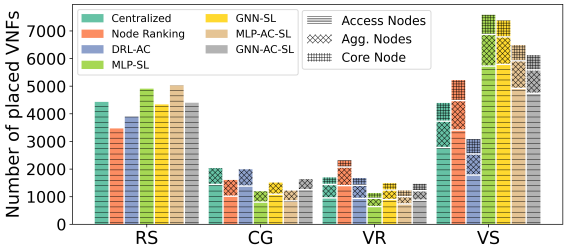


Fig. 7: VNF placement of different methods for each slice type, *i.e.*, MEC application.

TABLE II: GNN-AC-SL training hyper-parameters

Parameter	**AC**	**Slicing**
PPO clipping	0.05	0.2
entropy coefficient	0.1	0.0001
gamma	1	0.99
lambda, learning rate, gradient clipping	1, 1e-05, 10	
SGD iters, batch size	50, 372000	

during training. The training was done on an NVIDIA A100 GPU and took 139 hours. The final results were obtained using the hyperparameters shown in Table II.

Baselines: The following heuristic and DRL-based methods are implemented for comparison. DRL-AC and MLP-AC-SL are the only baselines with an intelligent AC module. Others greedily admit all feasible SRs.

- **Centralized:** a heuristic that places VNFs as close to the core node as allowed by delay and capacity constraints.
- **Node-Ranking (NR):** a heuristic for isolation-aware RAN slicing with delay constraint. We consider the scenario of the highest isolation level in [16], with the objective of minimizing active sites.
- **DRL-AC:** slicing is performed using Centralized algorithm, while the admission is decided by the DRL-based AC agent.
- **[MLP/GNN]-SL:** in these two approaches, slicing is managed by a DRL agent based on MLP/GNN architecture.
- **MLP-AC-SL:** multi-agent DRL-based method that uses MLP architecture for both agents similar to [10, 11].

A. Results

1) Training

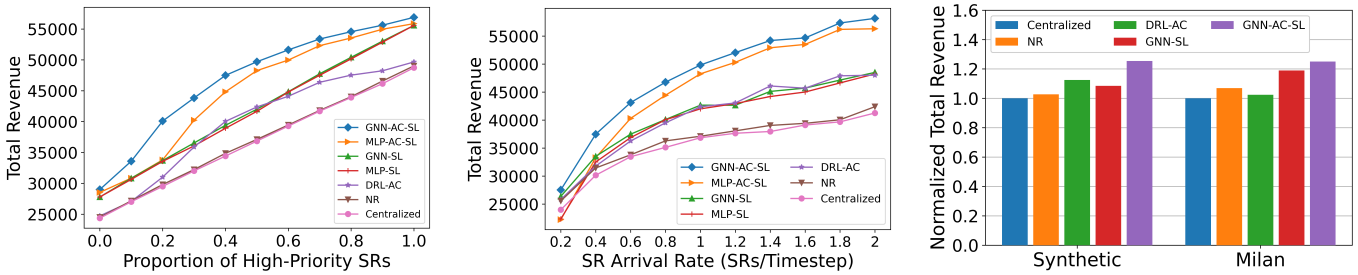
Fig. 5 shows the training performance in terms of the total revenue achieved per episode. Since GNNs are optimized for graph data, GNN-AC-SL and GNN-SL have faster convergence and higher performance compared to their MLP-based counterparts. Moreover, DRL-AC converges the fastest due to small action space, albeit to a local optimum, as the slicing performance of the Centralized approach prevents it from reaching its full potential. However, once the intelligent AC and slicing agents are combined, they can operate in harmony and reach their maximum capacity.

2) Evaluation

Once the training ends, the best-performing model checkpoint is restored to perform evaluations. For evaluation, the exploration is disabled, changing the action selection policy from stochastic sampling to deterministic, thereby improving the revenue margins slightly. Fig. 6a shows the total revenue, the portion achieved by granting HP and LP SRs, and the percentage improvement of all approaches compared to the Centralized baseline. The Centralized approach performs the worst because it greedily places near the core, leading to bandwidth bottlenecks, especially with high-throughput SRs. Next, NR performs relatively similarly because ranking works on a feature weighting basis that requires manual fine-tuning based on SR type and substrate network.

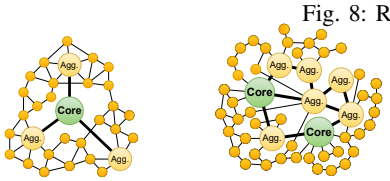
The heuristic approaches are followed by DRL-based methods. Although DRL-AC takes advantage of an intelligent AC, it is stuck on a local optimum, deploying only HP SRs. This is due to the inability of the Centralized approach to avoid bottlenecks and accommodate more SRs. With fewer overall SR embeddings offered to the AC agent, it chooses to skip LP SRs completely. Next, we have the two slicing-only approaches that follow the same pattern as observed in Fig. 5, followed by the two joint slicing and AC solutions. As GNN-SL is shown to be a more efficient slicing solution compared to MLP-SL, the AC agent in GNN-AC-SL is also able to admit more HP SRs in the network without creating bottlenecks. Such advantage makes GNN-AC-SL the highest achieving approach in terms of the overall gained revenue.

Figures 6b and 6c show the distribution of links and nodes utilization across the substrate network. The Centralized approach has the highest utilization of higher-tier nodes, but it falls behind DRL-based slicing algorithms when it comes to access nodes. Centralized and consequently DRL-AC have the highest variation of utilization across access nodes. Furthermore, GNN-based solutions have a higher utilization of higher-tier nodes compared to MLP-based methods without causing link bottlenecks. Finally, the addition of the AC



(a) Revenue vs. proportion of HP SRs (b) Revenue vs. SR arrival rate (c) Revenue under different topologies

Fig. 8: Robustness and generalizability analysis



(a) Synthetic Topology (b) Milan Topology

Fig. 9: New studied topologies: (a) synthetic network with 31 nodes and 54 links, and (b) Telecom Italia metro-regional network with 52 nodes and 101 links [35].

module generally lowers the utilization to make room for potential HP SRs.

Fig. 7 shows an in-depth look at slicing decisions of each method for all SR types, *i.e.*, MEC applications. VS and RS SRs are deployed more than VR and CG in all approaches, as they are less demanding in terms of processing capacity. RS SRs have a restricted delay tolerance which limits their placement to their originating access nodes. On the other hand, VS SRs have the most utilization of the core node due to their relaxed delay requirements. Compared to MLP-based approaches, GNN-based methods aim at increasing the deployment of higher-paying CG and VR SRs at the cost of less SRs of type RS and VS. In this regard, DRL-AC rejects too many VS SRs when compared to MLP-AC-SL and GNN-AC-SL, as only HP SRs are profitable, given the slicing efficiency.

3) *Robustness and Generalizability Analysis*

In this section, we evaluate the robustness and generalizability of trained agents by deviating the simulation conditions from the training scenario. Fig. 8a plots the impact of changing the proportion of HP SRs (from 0.5 used for training) on the total InP revenue. The revenue of slicing-only approaches increase linearly with the proportion of the HP SRs since they are imperceptive to SRs' revenue. In addition, as the proportion of HP SRs approaches 0 or 1, the AC effect diminishes and GNN-AC-SL, MLP-AC-SL, and DRL-AC perform similarly to their slicing-only counterparts. Lastly, with few HP SRs, the difference between MLP-AC-SL and GNN-AC-SL which is due to better slicing performance is more prominent.

Fig. 8b compares algorithms under different network loads resulting from changing SR arrival rate. GNN-AC-SL consistently outperforms the other approaches. Moreover, as the arrival rate decreases, the performance of the MLP-based methods deteriorates significantly. This supports the fact that MLP models can over-fit to the input and are less generalizable in graph applications, when compared to GNNs, to the extent that with an arrival rate of 0.2 SRs/timestep GNN-AC-SL outperforms MLP-AC-SL by 24%.

Finally, to showcase the generalizability of our proposed model on different topologies, we evaluate it on unseen substrate networks, without re-training or re-purposing the model. We consider a moderate-size 31-node synthetic network, and the 52-node real Milan network [35] shown in Fig. 9, which are $2.5\times$ and $4\times$ the size of the training network, respectively, and have similar resource capacities and delays. Fig. 8c only plots the performance of the heuristic and GNN-based approaches because previously trained MLP-based methods cannot operate when the network changes due to their fixed input size. The Centralized approach is set as the baseline for revenue normalization with total revenues of 73016 and 172006 in the Synthetic and Milan topologies, respectively. GNN-AC-SL outperforms the heuristic baseline by 25.5% and 25.1% in the synthetic and Milan topologies, respectively. However, with larger networks, there are less bottlenecks and the impact of the AC approach is less notable, resulting in lower percentage of improvements compared to the Centralized baseline. Overall, these results demonstrate that while trained on a small-scale network, our proposed GNN-AC-SL solution scales well to larger real-world metropolitan networks.

VI. Conclusions

In this work, we addressed the problem of joint RAN/MEC slicing and AC in dynamic metropolitan 5G networks using DRL. Conventional DRL approaches in this area are based on fully connected NN designs that are bound to the input/output data dimensions. Where these models fail to operate on different topologies, our GNN-based solution can effectively scale to large previously-unseen networks without the need for re-training. The results showed that our proposed solution converges faster than the MLP-based methods and achieves as much as 35.2% and 25.5% higher overall revenue, when compared to heuristic baselines, on the training and unseen networks, respectively. In the future, we plan to investigate the effect of training on random topologies and scenarios on generalizability, and the use of attention scores generated by the GAT model for identifying network bottlenecks.

Acknowledgment

This work was supported in part by Rogers Communications Canada Inc. and in part by a Mitacs Accelerate Grant.

REFERENCES

[1] "NGMN overview on 5G RAN functional decomposition," *NGMN Alliance, Final Deliverable, version 1.0*, February 2018.

[2] "Characteristics of transport networks to support IMT-2020/5G," *ITU-T G.8300*, 2020.

[3] M. Shehata, A. Elbanna, F. Musumeci, and M. Tornatore, "Multiplexing gain and processing savings of 5G radio-access-network functional splits," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 982–991, 2018.

[4] A. Garcia-Saavedra, X. Costa-Perez, D. J. Leith, and G. Iosifidis, "FluidRAN: Optimized vRAN/MEC orchestration," in *IEEE Conference on Computer Communications (ICC)*, 2018, pp. 2366–2374.

[5] A. Garcia-Saavedra, G. Iosifidis, X. Costa-Perez, and D. J. Leith, "Joint optimization of edge computing architectures and radio access networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 11, pp. 2433–2443, 2018.

[6] F. W. Murti, S. Ali, and M. Latva-aho, "Deep reinforcement based optimization of function splitting in virtualized radio access networks," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021.

[7] Z. Gao, S. Yan, J. Zhang, B. Han, Y. Wang, Y. Xiao, D. Simeonidou, and Y. Ji, "Deep reinforcement learning-based policy for baseband function placement and routing of RAN in 5G and beyond," *Journal of Lightwave Technology*, vol. 40, no. 2, pp. 470–480, 2021.

[8] V. Sciancalepore, L. Zanzi, X. Costa-Perez, and A. Capone, "ONETS: online network slice broker from theory to practice," *IEEE Transactions on Wireless Communications*, vol. 21, no. 1, pp. 121–134, 2021.

[9] N. Van Huynh, D. Thai Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and fast real-time resource slicing with deep dueling neural networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[10] M. Sulaiman, A. Moayyedi, M. A. Salahuddin, R. Boutaba, and A. Saleh, "Multi-agent deep reinforcement learning for slicing and admission control in 5G C-RAN," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2022, pp. 1–9.

[11] M. Sulaiman, A. Moayyedi, M. Ahmadi, M. A. Salahuddin, R. Boutaba, and A. Saleh, "Coordinated slicing and admission control using multi-agent deep reinforcement learning," *IEEE Transactions on Network and Service Management*, 2022, doi: 10.1109/TNSM.2022.3222589.

[12] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[14] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.

[15] H. Yu, F. Musumeci, J. Zhang, Y. Xiao, M. Tornatore, and Y. Ji, "DU/CU placement for C-RAN over optical metro-aggregation networks," in *International IFIP Conference on Optical Network Design and Modeling (ONDM)*. Springer, 2019, pp. 82–93.

[16] H. Yu, F. Musumeci, J. Zhang, M. Tornatore, and Y. Ji, "Isolation-aware 5G RAN slice mapping over WDM metro-aggregation networks," *Journal of Lightwave Technology*, vol. 38, no. 6, pp. 1125–1137, 2020.

[17] A. Marotta, D. Cassioli, M. Tornatore, Y. Hirota, Y. Awaji, and B. Mukherjee, "Multilayer protection-at-lightpath for reliable slicing with isolation in optical metro-aggregation networks," *Journal of Optical Communications and Networking*, vol. 14, no. 4, pp. 289–302, 2022.

[18] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[19] P. Almasan, J. Suárez-Varela, A. Badia-Sampera, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *arXiv preprint arXiv:1910.07421*, 2019.

[20] K. Poularakis, Q. Qin, F. Le, S. Kompella, and L. Tassiulas, "Generalizable and interpretable deep learning for network congestion prediction," in *IEEE International Conference on Network Protocols (ICNP)*, 2021, pp. 1–10.

[21] P. Zhang, C. Wang, N. Kumar, W. Zhang, and L. Liu, "Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning," *IEEE Internet of Things Journal*, 2021.

[22] J. J. A. Esteves, A. Boubendir, F. Guillemin, and P. Sens, "A heuristically assisted deep reinforcement learning approach for network slice placement," *IEEE Transactions on Network and Service Management*, 2021, doi: 10.1109/TNSM.2021.3132103.

[23] F. Habibi, M. Dolati, A. Khonsari, and M. Ghaderi, "Accelerating virtual network embedding with graph neural networks," in *IEEE International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–9.

[24] "Study on new radio access technology: Radio access architecture and interfaces," *3GPP TR 38.801 V14.0.0 (R14)*, 2017.

[25] "5G wireless fronthaul requirements in a passive optical network context," *ITU-T G Suppl. 66*, 2020.

[26] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "MEC in 5G networks," *ETSI white paper*, vol. 28, 2018.

[27] X. Wang, L. Wang, S. E. Elayoubi, A. Conte, B. Mukherjee, and C. Cavdar, "Centralize or distribute? a techno-economic study to design a low-cost cloud radio access network," in *IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[28] "Small cell virtualization functional splits and use cases Rel. 7.0," *Small Cell Forum*, 2016.

[29] C. Desset, B. Debaillie, V. Giannini, A. Fehske, G. Auer, H. Holtkamp, W. Wajda, D. Sabella, F. Richter, M. J. Gonzalez *et al.*, "Flexible power modeling of LTE base stations," in *IEEE wireless communications and networking conference (WCNC)*, 2012, pp. 2858–2862.

[30] W. Szeto, Y. Iraqi, and R. Boutaba, "A multi-commodity flow based approach to virtual network resource allocation," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 6, 2003, pp. 3004–3008.

[31] M. Rost and S. Schmid, "On the hardness and inapproximability of virtual network embeddings," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 791–803, 2020.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[33] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[34] "Service requirements for the 5g system; stage 1," *3GPP TS 22.261 V18.6.0 (R18)*, 2022.

[35] L. Askari, F. Musumeci, and M. Tornatore, "Latency-aware traffic grooming for dynamic service chaining in metro networks," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.