Multi-Agent Deep Reinforcement Learning for Slicing and Admission Control in 5G C-RAN

Muhammad Sulaiman*, Arash Moayyedi*, Mohammad A. Salahuddin*, Raouf Boutaba*, and Aladdin Saleh†
*David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada
{m4sulaim, arash.moayyedi, mohammad.salahuddin, rboutaba}@uwaterloo.ca
†Rogers Communications Inc., Ontario, Canada
{aladdin.saleh@rci.rogers.com}

***Abstract*—5G Cloud Radio Access Networks (C-RANs) facilitate new forms of flexible resource management as dynamic RAN function splitting and placement. Virtualized RAN functions can be placed at different sites in the substrate network according to resource availability and slice constraints. Due to limited resource availability in the substrate network, the Infrastructure Provider (InP) must perform network slicing in a strategic manner, and accept or reject slice-requests in order to maximize long-term revenue. In this paper, we propose to use multi-agent Deep Reinforcement Learning (DRL) to jointly solve the problems of network slicing and slice Admission Control (AC). Multi-agent DRL is a promising choice since it is well-suited to problems where multiple distinct tasks have to be performed optimally. The proposed DRL approach can learn the dynamics of slice-request traffic and effectively address these joint problems. We compare multi-agent DRL to approaches that use: (i) simple heuristics to address the problems, and (ii) DRL to address either slicing or AC. Our results show that the proposed approach achieves up to 18% and 3.8% gain in long-term InP revenue when compared to approaches (i) and (ii), respectively. Additionally, we show that multi-agent DRL is preferable to a single-agent DRL approach that addresses the problems jointly. Finally, we evaluate the robustness of the trained model in terms of its ability to generalize to scenarios that deviate from training.**

***Index Terms*—5G, C-RAN, Network Slicing, Admission Control, Multi-agent Reinforcement Learning**

I. Introduction

The Fifth Generation (5G) Radio Access Network (RAN) comprises chains of network functions (NFs) that belong to the New Radio (NR) protocol stack [1]. With the adoption of Cloud RAN (C-RAN) in 5G mobile networks, the substrate network has been re-imagined as a network of interconnected sites, each consisting of a number of commodity servers (nodes). Network Function Virtualization (NFV) allows an infrastructure provider (InP) to virtualize these resources and facilitates flexible and strategic placement of the virtualized network functions (VNFs), at different sites. This can alleviate network bottlenecks, and increase infrastructure utilization and InP revenue.

In a metro 5G C-RAN, the interconnected sites are categorized into tiers [2]. A lower-tier site is in closer proximity to the radio units (RUs), but has less resources, while a higher-tier site is geographically distant from the RUs with more resources. The higher-tier sites allow for centralized placement of resource-hungry VNFs, and lead to higher multiplexing gains via resource sharing among multiple instances of VNFs (*i.e.*, time-multiplexing) [3]. However, the degree of centralization is constrained by the delay tolerance of individual VNFs. With a higher degree of centralization, more unprocessed data has to traverse the inter-site links [4], which leads to a higher bandwidth demand on these links. Therefore, it is imperative that an optimal placement is chosen for the VNFs in 5G C-RAN, such that resource utilization is maximized without creating bandwidth bottlenecks, while also satisfying their latency and throughput requirements.

The 5G mobile networks are poised to support a wide range of services, primarily categorized into enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and massive Machine-Type Communications (mMTC), based on their Quality of Service (QoS) requirements (*e.g.*, bandwidth, latency and mobility). Network slicing is a key enabling technology to offer isolated end-to-end virtual networks (*i.e.*, 5G network slices), that are tailored to satisfy the specific QoS requirements of different services on the same infrastructure. Network slices consist of a chain of the VNFs. While placing these VNFs in 5G C-RAN at different sites (*i.e.*, Virtual Network Embedding (VNE)), it is crucial to consider the service type and its Service-Level Agreements (SLAs). Accepting a slice-request (SR) from a service provider contributes towards the InP's revenue. However, given an InP's limited resources, it is impossible to serve all incoming SRs. Therefore, an Admission Control (AC) decision must be made for each incoming SR, such that it maximizes the InP's long-term revenue.

Recently, Deep Reinforcement Learning (DRL) [5] has shown unprecedented performance in solving problems that were previously too challenging for Artificial Intelligence-based solutions. A DRL agent interacts with an environment and, through trials and corresponding rewards, learns the actions that maximize its cumulative reward without a priori knowledge of the environment or the need for massive datasets. Hence, DRL lends itself well to solving AC and slicing in 5G C-RAN. Numerous works in the literature (*e.g.*, [6, 7]) have proposed either traditional reinforcement learning (RL) or DRL-based solutions to these problems. Though, AC and network slicing are both quintessential to offer differenti-

978-1-6654-0601-7/22/$31.00

ated QoS, most works propose DRL-based solutions to solely one of them. Using DRL to address only one aspect of the network slicing and AC problems and using a naïve approach, such as greedy, for the other one can potentially lead to a loss in the long-term revenue for the InP. For instance, the authors in [6] propose a DRL-based AC solution, however, they assume network slicing does not present a challenge. In this scenario, if the slicing algorithm creates a bandwidth bottleneck in a critical network link, the total number of SRs that can be admitted becomes limited.

Additionally, in practice, the future SRs are not known in advance. Hence, future SRs must be predicted to make intelligent slicing and AC decisions. However, some works in the literature (*e.g.*, [8, 9]) are oblivious to this, limiting the applicability of their solutions. Whereas, a DRL agent can take the future consequences of its actions into account while maximizing its cumulative reward. If a certain slicing decision causes future SRs to be rejected due to a resource bottleneck, the DRL agent anticipates this and avoids that decision.

In the case of joint slicing and AC, the reward formulation for a single DRL agent does not allow to effectively address both problems jointly. For example, if the DRL agent is given a negative reward for a non-optimal AC action, the entire policy (*i.e.*, the neural network used for comparing the optimality of different actions in a given environment state) is impacted. This causes the concurrent slicing action to also be disincentivized, even if it is the optimal action. This concern is alleviated in multi-agent DRL, where the agents can have separate policies for AC and slicing. The reward functions for these policies can be designed such that the two agents learn to work in synergy without negatively impacting one another.

In this paper, we propose a novel multi-agent DRL-based solution to jointly address slicing and AC in 5G C-RAN, in order to improve the long-term InP revenue. To evaluate the efficacy of our proposed solution, we develop a C-RAN slicing and AC simulation framework that also facilitates the evaluation of other solutions, such as a single-agent DRL-based solution. In this regard, we demonstrate the following:

- We compare the proposed multi-agent DRL-based solution against a greedy approach and a node-ranking approach (inspired by [10]), and show that our solution outclasses these approaches in maximizing the long-term InP revenue.
- We compare the proposed solution against approaches that use DRL to address either slicing or AC (*e.g.*, [11], [12]). We show that using multi-agent DRL to address both of these problems jointly leads to higher long-term InP revenue.
- We compare against a single-agent DRL approach that jointly addresses the slicing and AC problems in 5G C-RAN. We show that multi-agent DRL outperforms the single-agent counterpart in terms of the achieved long-term InP revenue and convergence time.
- We evaluate the robustness of the trained model under practical network conditions, such as variable SR arrival rates. We show that multi-agent DRL is able to generalize to these scenarios and achieve the highest long-term InP revenue when compared to the other approaches.

The rest of the paper is organized as follows. In Section II, we present closely related works, followed by system design for closed-loop orchestration and management of VNFs in 5G C-RAN in Section III. Section IV delineates the proposed multi-agent DRL-based slicing and AC solution, while Section V showcases the evaluation results. We conclude in Section VI and instigate future research directions.

II. Related Works

Several works in the literature address slicing and AC (*i.e.*, admission control) in 5G. Van Huynh et al. [6], Dandachi et al. [12] mostly focus on the AC aspect of 5G slicing and orchestration. The authors in [12] propose a traditional RL-based solution for 5G slice deployment and orchestration. Even though they consider a slice as a set of VNFs, the substrate network is only considered in aggregate. That is, instead of modeling the substrate network as a collection of interconnected sites or nodes, each with its own limited resources, the network is modeled as a single node with a certain amount of resources. This simplifies the slice embedding problem to an unrealistic degree. The authors in [6] propose DRL for slice AC and resource allocation, but similar to [12], they do not model a slice as a collection of VNFs and the substrate network as a set of interconnected nodes.

Pujol Roig et al. [7] propose RL for VNF management and orchestration in an online fashion. When a new VNF placement request arrives, a DRL agent decides to either place that VNF on an already running server by assigning it a part of the server's remaining resources, start up a separate server and allocate its resources to the VNF, or upload the VNF to the cloud, with the objective of minimizing incurred cost. Although the authors deal with the VNF request in an online manner, their model only deals with individual VNFs instead of a network slice. Sciancalepore et al. [13] propose an online network slice brokering solution that maximizes multiplexing gains. The problem is modeled as a budgeted lock-up multi-armed bandit problem, which is a variation of the well-known multi-armed bandit problem. Nevertheless, similar to [7], the authors model a network slice as only requiring a number of Physical Resource Blocks (PRBs), whereas a RAN slice consists of a number of functions each with its own latency and computing resource requirements.

The authors in [8, 9] address the user-centric functional split problem, which deals with splitting the RAN functions between a remote site and a centralized site. They model the problem as an Integer Linear Problem (ILP), and propose solutions based on particle swarm optimization and deep learning, respectively. However, the authors model the substrate network as only having a single remote site and a single distributed site. Additionally, the authors do not factor online AC in their ILP. Murti et al. [14] build on the work of Garcia-Saavedra et al. [15] to address the functional split optimization problem. They form a cost-minimization based objective function, subject to multiple constraints. They address the problem in an offline fashion, where a functional split is decided for each distributed unit in the network, instead of a functional split for each SR

(*i.e.*, slice-request) as it arrives. Murti et al. [16] propose RL as a less complex alternative solution to the problem in [15], but their work suffers from the same drawback.

Wang and Zhang [11] propose RL-based resource allocation for network slicing in 5G C-RAN. Although the authors model a slice as a chain of VNFs and the substrate network as a set of interconnected nodes, they do not address the AC aspect of 5G slicing and orchestration. Liu and Han [17] propose distributed cross-domain resource orchestration (DIRECT) protocol, which optimizes the resources allocated to the slice in multiple domains. However, the proposed algorithm only optimizes resources for in-service slices and it does not perform AC. The work of Raza et al. [18] is the most similar to ours. Authors propose a policy-based RL algorithm for slice AC. Nevertheless, the arriving SRs, in their work, already include the amount of resources required at the remote and central sites. This sidesteps an important aspect of slicing, where all of an SR's functions can be placed at either the remote or the centralized location. Additionally, the selection of the central location (*i.e.*, remote data center) for the SR is done after the AC decision has been made. This precludes the AC agent from knowing the embedding before making the AC decision and can lead to performance degradation in resource-constrained environments. Our work differs from the existing literature in that we jointly address the problem of network slicing and AC in 5G C-RAN.

III. System Design

The system design is based on the MAPE (*i.e.*, monitor, analyze, plan, execute) control loop [19, 20] to facilitate closed-loop, autonomous management and orchestration of VNFs in 5G C-RAN, as depicted in Fig. 1. The data from the substrate network, collected by the monitor module, is forwarded to the analyze module. The analyze module extracts useful knowledge from the raw data, and computes any metrics required for visualization and planning. This processed data and any incoming SR, is received by the plan module.

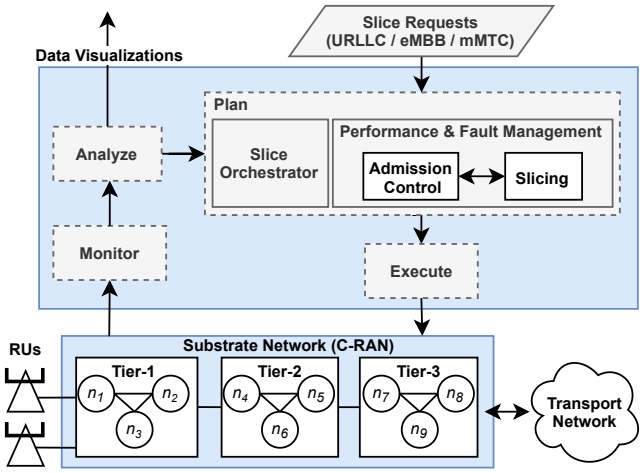

Fig. 1: High-level view of closed-loop, autonomous management and orchestration of VNFs in 5G C-RAN

The plan module performs intelligent slice orchestration and, performance and fault management by leveraging AI/ML techniques [21]. The AC and slicing sub-components, which are a part of the performance & fault management component, are utilized by the plan module to produce intelligent AC and slicing decisions. These can be utilized either concurrently (*i.e.*, both output their decision independently) or sequentially (*i.e.*, one sub-component can utilize the output of the other to make its decision). If the SR is admitted, the slice orchestrator forwards the actions required for its orchestration to the execute module which executes these actions by interacting with the substrate network. Since we simulate a substrate network, the monitor, analyze and execute modules do not present a research challenge. Therefore, in this paper, we only describe the pertinent system design, and propose and evaluate the design of the slicing and AC components.

A. Substrate Network

An example of a multi-tier 5G C-RAN supporting dynamic RAN functional decomposition [2], is shown in Fig. 1. This constitutes the basis of our substrate network design. The substrate network is represented as a graph $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{L})$, where $\boldsymbol{N} = \{n_1, n_2, ..., n_k\}$ represents the set of k nodes and $\boldsymbol{L} \subseteq \boldsymbol{N} \times \boldsymbol{N}$ represents the set of j links each with a certain latency. We use $l_{n,n'} \in L$ to denote the link between the nodes $n, n' \in \boldsymbol{N}$. At any time t, a node $n \in \boldsymbol{N}$ has a certain amount of available computing resources, denoted by c_n^t, and a link $l \in \boldsymbol{L}$ has an available bandwidth, denoted by b_l^t. A set of RUs are connected to Tier-1 site with low-latency and high-bandwidth links. VNFs requiring very low latency can be placed there. As the sites become more centralized (*i.e.*, Tier 2–3 sites), the experienced latency from the RU increases due to the increased path delay. Although nodes at these sites have higher available computing resources, these sites must support a greater number of lower-tier sites.

B. Functional Split Requirements

The NR protocol stack consists of a number of essential functions, namely, RF, Low-PHY, High-PHY, Low-MAC, High-MAC, Low-RLC, High-RLC, PDCP, and RRC. 3GPP enumerates the possible splits for these functions [1], shown in Fig. 2. These splits describe the functions that are to be decentralized at the Distributed Unit (DU) and those that are to be centralized at the Central Unit (CU). With dynamic function splitting, the split for each SR (*i.e.*, slice-request) is not fixed. Instead, it varies for each SR based on its VNFs' placement at different sites. With option 8, all functions except RF signal generation are centralized. This split leads to the highest multiplexing gain, yet has also very strict latency requirements and bandwidth demands. On the other hand, with option 2, only PDCP is centralized at the CU. This leads to high computing resource requirements at the DU but less stringent bandwidth and latency requirements [1, 22].

Based on the functional split options proposed by 3GPP, ITU recommends option 7 as the split between the RU and the DU [23], due to its high bandwidth and strict latency requirements. Therefore, in this paper, we consider Low-PHY

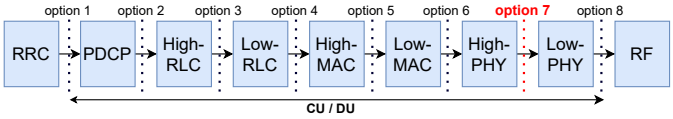

Fig. 2: Functional splits for VNFs in 5G C-RAN [1]

and RF functions to always be placed at the RU. Additionally, since the computing resource modeling between Low-RLC and High-RLC, and Low-MAC and High-MAC is still in progress, we only consider options 2, 4, 6 and 7 as the possible splits. Therefore, we refer to High-PHY, MAC, RLC and PDCP functions as f_1–f_4, respectively.

In 5G C-RAN, these functions are virtualized, *i.e.*, VNFs, and placed at different nodes in the substrate network. Each of these VNFs requires a certain amount of computing resource measured in Giga Operations Per Second (GOPS). The computing resource requirement depends on the type of operations each VNF needs to perform, and can be calculated as [24]:

$$G_1 = G_1^{ref} \cdot \frac{B}{B^{ref}} \cdot (\frac{A}{A^{ref}})^2 \cdot \frac{L}{L^{ref}}, \tag{1}$$

$$G_2 = G_2^{ref} \cdot \frac{B}{B^{ref}} \cdot \frac{A}{A^{ref}} \cdot \frac{L}{L^{ref}} \cdot \frac{M}{M^{ref}}, \tag{2}$$

$$G_3 = G_3^{ref} \cdot \frac{B}{B^{ref}} \cdot \frac{A}{A^{ref}}, \tag{3}$$

$$G_4 = G_4^{ref} \cdot \frac{B}{B^{ref}} \cdot \frac{A}{A^{ref}}, \tag{4}$$

$$G_5 = G_5^{ref} \cdot \frac{B}{B^{ref}} \cdot \frac{A}{A^{ref}}, \tag{5}$$

where G_1–G_2 refer to the computing resource requirements for the sub-functions of f_1, and G_3–G_5 refer to the computing resource requirements for f_2–f_4, respectively. B, A, L, and M refer to the bandwidth, number of MIMO antennas, load, and modulation, respectively. G^{ref}, B^{ref}, A^{ref}, L^{ref}, and M^{ref} are values for the computing resource requirements, bandwidth, MIMO number of antennas, load, and modulation in the reference scenario [24, 25], respectively. The bandwidth requirements for different splits can be modeled as:

$$R(\lambda) = k_1\lambda + k_2, \tag{6}$$

where R is the required link bandwidth, λ is the slice throughput in Mbps, and k_1, k_2 are constants with specific values for the different splits which can be calculated using [26]. The latency requirements depend on the VNF and the end-to-end latency requirement of the SR [27, 28].

C. Slice-requests

Each incoming SR is considered to be dedicated to one of the following service types: eMBB, URLLC, or mMTC. A SR arriving at time t, denoted by s_t, consists of its service type, the required throughput λ^{s_t} that it needs to support, its end-to-end latency requirement, its operation time τ^{s_t}, and its offered revenue per unit time r^{s_t}. Multiple SRs cannot arrive at the same instant of time. If the SR is embedded, it consumes the substrate network's computing and bandwidth resources for a duration of τ^{s_t}, after which it departs, freeing up the reserved resources. eMBB slices require the highest bandwidth and can tolerate a moderate amount of latency. On the other hand, URLLC slices require a moderate amount of bandwidth but have very strict latency requirements. Finally, mMTC slices require a moderate amount of bandwidth and can operate under high latency. Additionally, each slice is of either high priority (HP) or low priority (LP), and the offered revenue is proportional to the priority of the slice.

D. Joint Slicing and AC

The slicing decision for any SR s_t is in the form of a $M \times |\boldsymbol{N}|$ embedding-relationship matrix η^{s_t}, where M is the number of VNFs that need to be embedded in the substrate network. $\eta^{s_t}_{f_m,n} = 1$ if f_m is mapped to substrate network node $n \in \boldsymbol{N}$, and $\eta^{s_t}_{f_m,n} = 0$ otherwise. The path selection between the nodes is done based on the shortest-path algorithm. $d_{ru,n}$ and $d_{n,n'}$ are used to denote the shortest-path delay between the RU and node n, and between the nodes n, n', respectively, where $n, n' \in \boldsymbol{N}$. $\phi_{l,n,n'} = 1$ if link $l \in \boldsymbol{L}$ is in the shortest path between nodes $n, n' \in \boldsymbol{N}$. Whereas, $\phi_{l,ru,n} = 1$ if link $l \in \boldsymbol{L}$ in the shortest path between RU and the node $n \in \boldsymbol{N}$. An embedding-relationship matrix η^{s_t}, for any SR s_t, is valid if it satisfies all of the following constraints:

$$\sum_{n \in \boldsymbol{N}} \eta^{s_t}_{f_m,n} = 1 \quad \forall m \in \{1, ..., M\}, \tag{C1}$$

$$\sum_{m=1}^{M} \eta^{s_t}_{f_m,n} \cdot c^{s_t}_{f_m} \leq c^t_n \quad \forall n \in \boldsymbol{N}, \tag{C2}$$

$$d_{ru,n} \cdot \eta^{s_t}_{f_1,n} + \sum_{f_i=f_1}^{f_m} \eta^{s_t}_{f_i,n} \cdot \eta^{s_t}_{f_{i+1},n'} \cdot d_{n,n'} \leq d^{s_t}_{f_m} \quad \forall n, n' \in \boldsymbol{N}, \forall m \in \{1, ..., M\}, \tag{C3}$$

$$\eta^{s_t}_{f_1,n} \cdot R^{s_t}_{f_1}(\lambda^{s_t}) \cdot \phi_{l,ru,n} + \sum_{m=1}^{M-1} \eta^{s_t}_{f_m,n} \cdot \eta^{s_t}_{f_{m+1},n'} \cdot R^{s_t}_{f_m}(\lambda^{s_t}) \cdot \phi_{l,n,n'} \leq b^t_l \quad \forall n, n' \in \boldsymbol{N}, \forall l \in \boldsymbol{L}, \tag{C4}$$

where $c^{s_t}_{f_m}$ and $R^{s_t}_{f_m}(\lambda^{s_t})$ denote the computing resource requirement, and the output data-rate for VNF f_m of SR s_t, respectively. These are calculated using equation (1)–(5) and equation (6), respectively. $d^{s_t}_{f_m}$ is used to denote the latency requirement of the VNF f_m of SR s_t. Constraint (C1) ensures that a VNF is mapped to only one substrate network node, which is an assumption commonly made in the literature to simplify the problem. Constraint (C2) ensures that the substrate nodes have the computing resources available to host the VNFs, constraint (C3) ensures that the path delays meet the latency requirements of each of the VNFs and finally, constraint (C4) ensures that the link bandwidth limits are not exceeded. A SR s_t is feasible if there exists a possible η^{s_t} that is valid.

The objective of slicing and AC (*i.e.*, admission control) is to maximize the cumulative InP revenue achieved (*i.e.*, r_{total}):

$$\begin{aligned} \text{maximize} \quad & r_{total} = \sum_{t \in \boldsymbol{T}} r^{s_t} * \tau^{s_t} \\ \text{s.t.} \quad & \text{(C1)} - \text{(C4)}, \end{aligned} \tag{7}$$

where $\boldsymbol{T}$ denotes the set of arrival times of SRs dedicated to different service types. With a slicing-only problem formulation, when enough resources are available for an SR, the only recourse to reject it preemptively is to deliberately produce an infeasible embedding-relationship matrix, which is antithetical to the idea of slicing. The joint AC and slicing problem, on the other hand, includes the AC as a part of the problem formulation. In this case, AC actions can be used to reject SRs in order to prevent resource bottlenecks, or to preserve resources for future SRs with potentially higher revenue.

IV. Proposed Solution

To jointly address the slicing and AC (*i.e.*, admission control) problems, we propose a multi-agent DRL-based approach which we refer to as M-AC-VNE. In this approach, two separate agents—a slicing agent and an AC agent—are used to produce the embedding-relationship matrix and the admission decisions. The slicing agent is only invoked if an s_t is feasible (*i.e.*, a valid η^{s_t} exists), and the AC agent is invoked if the slicing agent produces a η^{s_t} that is valid.

The agents' observation space includes the incoming SR's (s_t) service type (*i.e.*, URLLC, eMBB, mMTC), its operation time (τ^{s_t}), offered revenue (r^{s_t}) and the current substrate network state ($\boldsymbol{C^t}, \boldsymbol{B^t}$). The AC agent's observation space also includes the resulting substrate network state if the s_t is to be admitted. To train the DRL agents' policy network weights, we utilize the Proximal Policy Optimization (PPO) algorithm [29], which is an on-policy, model-free, policy-based algorithm that outperformed the other DRL algorithms during our trials. Since the objective in equation (7) can only be computed after a training episode ends, it would be too sparse for use as a reward for a DRL agent. Therefore, we use Algorithm 1 and Algorithm 2 to generate the reward for the slicing agent and the AC agent, respectively.

For the slicing agent, the reward depends on the number of constraints (C2)–(C4) violated by η^{s_t}, the number of subsequent SRs (S_{sub}) that are infeasible and if AC agent admits s_t. The algorithm is designed to make the slicing agent produce embedding-relationship matrices that are valid, cause the least amount of bottlenecks, and are likely to be accepted by the AC agent. For the AC agent, the reward depends on the r^{s_t}. If the AC agent admits s_t, it gets the total offered revenue by that s_t as the reward. But if subsequent SRs are infeasible, then the AC agent gets a negative reward equal to the potential revenue loss. This reinforces the AC agent's policy to reject SRs that are LP (*i.e.*, low priority) and are likely to cause bottlenecks. The constants (*i.e.*, +/- 1, 1.5) used in Algorithm 1 to balance between the negative and positive rewards are based on trial-and-error and lead to faster learning.

As one of the comparison-cases for the proposed approach, we design the single-agent DRL approach, referred to as S-AC-VNE. In this approach, both the embedding-relationship matrix and admission decision are concurrently produced by a single DRL-agent. The observation space is the same as that of the slicing agent in M-AC-VNE. The reward function for this approach is given in Algorithm 3.

Algorithm 1 Slicing agent's reward in M-AC-VNE

Input : s_t, η^{s_t}, ($\boldsymbol{C^t}, \boldsymbol{B^t}$), subsequent SRs ($\boldsymbol{S_{sub}}$)
Output: slicing agent's reward

$reward \leftarrow 0$
foreach *constraint* (C2)–(C4) *violated by* η^{s_t} **do**
  $reward \leftarrow reward - 1$
end
if *(η^{s_t} is valid) $\wedge$ (SR s_t admitted by AC)* **then**
  $reward \leftarrow reward + 1.5$
end
for $s_{sub} \in \boldsymbol{S_{sub}}$ **do**
  if s_{sub} *not feasible* **then**
    $reward \leftarrow reward - 1.5$
  else
    return *reward*
  end
end

Algorithm 2 AC agent's reward in M-AC-VNE

Input : s_t, η^{s_t}, ($\boldsymbol{C^t}, \boldsymbol{B^t}$), subsequent SRs ($\boldsymbol{S_{sub}}$)
Output: AC agent's reward

$reward \leftarrow 0$
if s_t *is admitted* **then**
  $reward \leftarrow reward + (r^{s_t} * \tau^{s_t})$
end
for $s_{sub} \in \boldsymbol{S_{sub}}$ **do**
  if s_{sub} *not feasible* **then**
    $reward \leftarrow reward - (r^{s_{sub}} * \tau^{s_{sub}})$
  else
    return *reward*
  end
end

Algorithm 3 RL agent's reward in S-AC-VNE

Input : s_t, η^{s_t}, ($\boldsymbol{C^t}, \boldsymbol{B^t}$), max offered revenue by any SR (r_{max})
Output: RL agent's reward

$reward \leftarrow 0$
foreach *Constraint* (C2)–(C4) *violated by* η^{s_t} **do**
  $reward \leftarrow reward - (r_{max} * \tau^{s_t})$
end
if *(η^{s_t} is valid) $\wedge$ (s_t is admitted)* **then**
  $reward \leftarrow reward + (r^{s_t} * \tau^{s_t})$
end
return *reward*

In this case, if s_t is successfully embedded, the reward is proportional to r^{s_t}. Otherwise, a negative reward that is proportional to the number of SR (*i.e.*, slice-request) constraint violations, is given to the agent. It is worth noting that the negative reward in case of a constraint violation by the embedding-relationship matrix is proportional to the maximum offered revenue. This is so that the agent utilizes its AC action to reject SRs, instead of producing an infeasible embedding-relationship matrix for the SRs. In the single-agent case, there

can not be a separate reward for slicing and AC. Due to this, the learning of one aspect can interfere with that of the other. For example, if the agent is given a negative reward for a non-optimal AC action, the entire policy is impacted. This causes the concurrent slicing action to also be disincentivized, even if it is optimal. This leads to a slower and potentially non-optimal learning for S-AC-VNE, as shown in Section V.

V. Evaluation

A. Environment Setup

The simulation is implemented as a custom OpenAI Gym [30] environment, coupled with RLLib [31] on a cluster of 3 servers. Each server includes 16GB of RAM, 8x Intel Xeon 3.30GHz cores and runs Ubuntu 16.04 . Cluster management is done through Ray [32], and DRL algorithms are implemented in RLLib, using Python 3.8.9 and leveraging PyTorch [33]. This setup enables distributed learning to accelerate the training and hyperparameter optimization.

B. Comparative Approaches

During evaluation, we compare to a variety of approaches, aiming to encompass the current baseline and state-of-the-art approaches in 5G C-RAN slicing and AC.

1) Heuristics-based: Greedy & Node-ranking

In the greedy approach, all SRs are accepted and each VNF is placed at the closest node to the RU with enough resources available to host the VNF. In case of a tie, the node with higher available resources is selected. Since the VNFs are placed at the closer nodes first, the delay requirement is readily met and minimum link bandwidth is used. However, this approach is naïve, as once the computing resources at Tier-1 site are exhausted, VNFs that demand a low latency cannot be placed anymore. Consequently, all additional SRs are rejected until resources at Tier-1 site become available again.

To circumvent this bottleneck, we design the Node-ranking approach for slicing (inspired by [10]). For each VNF, starting from the last one in the chain, all the nodes in the substrate network within the VNF's tolerable delay are ranked. This ranking is done based on the tier of the node (*i.e.*, farthest to closest), followed by their available computing resources (*i.e.*, highest to lowest). Finally, the highest-ranked node that does not violate any link bandwidth constraints is selected. This ranking method is used for all SR service types, since it puts the least strain on Tier-1 site's computing resources until the bandwidth to higher-tier sites is constrained.

2) DRL-based: DRL-AC, DRL-VNE & S-AC-VNE

Since a number of works propose DRL for AC without proper modeling of network slicing [6, 12] and for network slicing without modeling AC [11], we deploy two DRL-based benchmark solutions. In the first approach, DRL is used for AC, but slicing is done through the Node-ranking approach described earlier. In the second case, slicing is done by DRL, but all SRs are accepted. The reward formulation for the DRL-agent in both approaches is kept the same as the one for the corresponding DRL-agent in the M-AC-VNE approach. Henceforth, we refer to these approaches as DRL-AC and DRL-VNE, respectively. The single-agent DRL approach (S-AC-VNE) is described in Section IV.

C. Simulation Parameters

The maximum latency required for the RAN VNFs' operation [26] and the end-to-end latency SLA for each service type are given in Table I. These values are used to derive effective latency requirements (*i.e.*, $d^{s_t}_{f_m}$) for f_1–f_4.

One-way Latency Requirement (ms)					
Service	$f1$	$f2$	$f3$	$f4$	**End-to-End**
URLLC	0.25	2	6	30	1.5
eMBB	0.25	2	6	30	4
mMTC	0.25	2	6	30	10

TABLE I: VNF latency requirement for each service type

URLLC, mMTC, and eMBB SRs require a throughput of 75Mbps, 75Mbps, and 150Mbps, respectively. For the training scenario, the SRs have a Poisson-distributed arrival rate with an average of 5 SRs per hour. The SR operation time is uniformly distributed with a mean of 6 hours and a standard deviation of 0.5. Any incoming SR has 50% probability of having a service type of eMBB and 50% probability of having the other two service types. Each SR can be either HP (*i.e.*, high priority) or LP with 50% probability. On average, HP SRs offer twice the revenue as compared to LP SRs.

The computing resources available per node are 1500, 2000, and 4000 GOPS at Tier-1, Tier-2, and Tier-3 sites, respectively. The link latencies between RU and Tier-1 site, Tier-1 and Tier-2 sites, and Tier-2 and Tier-3 sites are 0.25ms, 1.2ms, and 4.2ms [2], respectively. The available bandwidth for Tier-1 to Tier-2 site link is 1000Mbps and that of Tier-2 to Tier-3 link is 1500Mbps. Due to the close proximity of nodes, the intra-site links are modeled to have ample bandwidth and negligible latencies. The RUs operate at 20MHz, 2x2 MIMO, and 64QAM. A training episode in DRL consists of 10 days of operation after which the simulation is restarted and a new episode is initiated. After hyperparameter search, we reach those shown in Table II for the PPO algorithm, which are then used for all the DRL-based agents.

DRL Hyperparameter	**Value**
# Hidden-layer neurons	[128, 128, 128]
Learning rate at timesteps [0, 10^7, 10^8]	[3e-3, 5e-4, 1-e4]
Train batch size, Mini-batch size, # SGD iters	16348, 4096, 5
KL coefficient, Lambda	0.4, 0.95

TABLE II: PPO Training Hyperparameters

D. Results

We train each of the DRL-based approaches for 100 million training steps and evaluate their performances using the checkpoint with the highest achieved revenue. Fig. 3 shows the total revenue achieved by the different approaches as the training progresses, and the final InP (*i.e.*, infrastructure provider) revenue achieved during evaluation is shown in Fig. 4. Since the DRL-agents do not perform exploration during evaluation, the average revenue achieved is slightly higher as compared

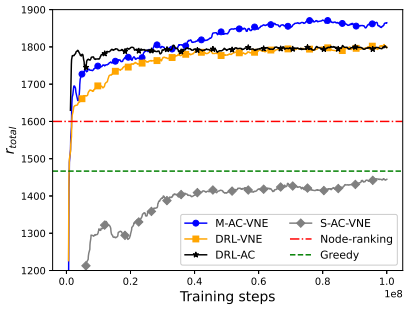

Fig. 3: Episode revenues

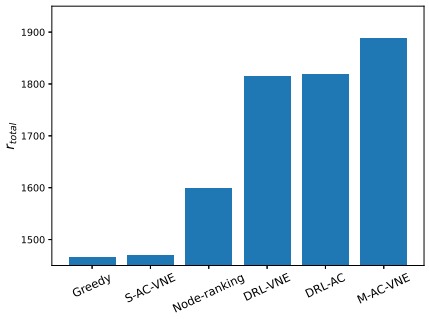

Fig. 4: InP total revenue during evaluation

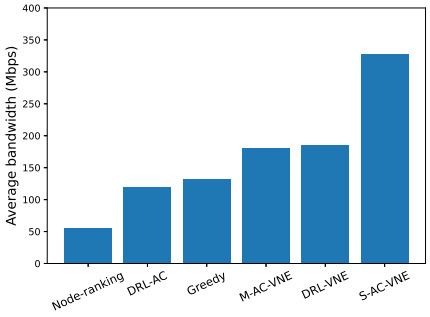

Fig. 5: Average available Tier-1 to Tier-2 link bandwidth during evaluation

to the training phase. Evidently, the greedy approach achieves the lowest revenue. We set this approach as the baseline to compare the rest of the approaches against. S-AC-VNE achieves only 0.3% higher revenue as compared to the greedy approach for reasons explained in Section IV. S-AC-VNE is followed by Node-ranking, DRL-VNE, DRL-AC, and M-AC-VNE approaches, which achieve 9.1%, 23.8%, 24.1%, and 29.5% gain in revenue over the greedy approach, respectively.

The average available bandwidth for Tier-1 to Tier-2 link during evaluation is shown in Fig. 5. The available bandwidth is lowest for the node-ranking approach since this approach places all VNFs at the farthest node, which consumes a higher amount of bandwidth. A similar value can be observed for the DRL-AC approach as it uses the Node-ranking approach for slicing. As opposed to DRL-AC, for M-AC-VNE and DRL-VNE the average available bandwidth is higher. This shows that DRL-based slicing approaches are able to preserve more bandwidth by placing a lesser number of bandwidth-hungry SRs over the Tier-1 to Tier-2 link. This is also corroborated by Fig. 6, which shows that DRL-VNE and M-AC-VNE approaches achieve the highest resource utilization in Tier-3, while still maintaining high resource utilization at Tier-1 and Tier-2. S-AC-VNE approach has the highest average available bandwidth since it has the lowest number of in-service SRs throughout the training, as shown in Fig. 7.

Due to a limited amount of resources available in the substrate network, there can only be a limited number of SRs (*i.e.*, slice-requests) in-service at any given time. The number of in-service SRs depends on different factors, such as the availability of computing resources at lower-tier sites and available link bandwidths. Fig. 7 shows the average number of in-service SRs during each episode as the DRL-based approaches are trained. It can be observed that as the approaches with DRL-based slicing train, the average number of in-service SRs increases. This is because the slicing agent learns to produce embedding-relationship matrices, such that there is a lesser number of bottlenecks, resulting in more SRs to be accepted. Additionally, we can see that the greedy approach has the lowest number of in-service SRs after the S-AC-VNE approach due to the bottleneck described in Section V-B1. The Node-ranking approach circumvents this bottleneck, which leads to an increase in the average number of in-service SRs over the duration of the episode. But with this approach the link bandwidth can become a bottleneck. We can see that the DRL-VNE approach achieves the highest average number of in-service SRs by optimally balancing the computing and bandwidth resource utilization.

To maximize the long-term revenue, some LP SRs should be preemptively rejected to keep the resources available for the HP (*i.e.*, high-priority) SRs. Fig. 8 shows the percentage of HP embedded SRs during an episode as the training progresses. As expected, for the greedy and Node-ranking approaches, it is 50%, whereas, for the approaches with DRL-based AC (*i.e.*, admission control), it is higher. Indeed, DRL-AC leads to accepting the highest proportion of HP SRs. This percentage is highest for the DRL-AC approach, even though it does not have the highest average number of in-service SRs. This leads to a comparatively higher revenue in the case of DRL-AC when compared to approaches with a higher average number of in-service SRs. Finally, we can see that M-AC-VNE achieves the highest average revenue (*cf.*, Fig. 3) by optimally balancing between the average number of in-service SRs (*cf.*, Fig. 7) and the average number of embedded SRs that are HP (*cf.*, Fig. 8).

The episode reward for DRL-AC is the earliest to reach a plateau during its training, followed by DRL-VNE, as shown in Fig. 9. This is because these approaches are using DRL for either slicing or AC, but not for both. Additionally, we see that although the episode reward for DRL-VNE decreases after 25M training steps, the revenue (*cf.*, Fig. 3) does not follow the same trend. This is because, in addition to the number of embedded SRs, the reward for the DRL-agent also depends on the number of SR constraints' violations.

Although the training is stopped at $100M$ training steps, the episode rewards for M-AC-VNE and S-AC-VNE still show a rising trend. Moreover, M-AC-VNE converges to a higher episode reward compared to S-AC-VNE. It is possible for the latter to achieve the same long-term reward as the former, since the policy represented by the two policy networks of the M-AC-VNE can also be represented by a single policy network of S-AC-VNE. In practice, however, we see that S-AC-VNE achieves much lower revenue in $100M$ training steps.

Finally, to test the robustness of the trained model under

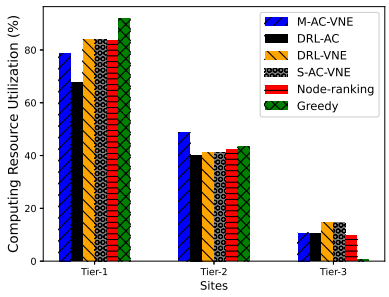

Fig. 6: Computing resource utilization during evaluation

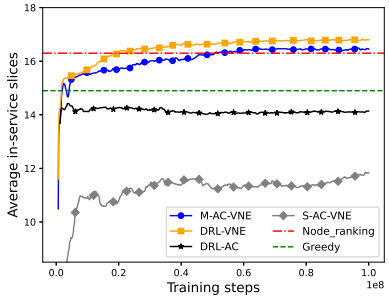

Fig. 7: Average in-service SRs during training

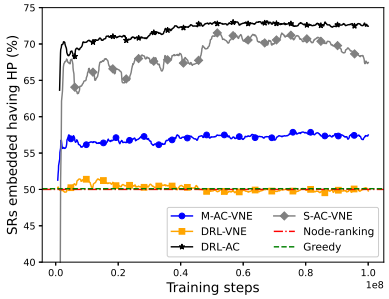

Fig. 8: Percent of embedded SRs that are HP

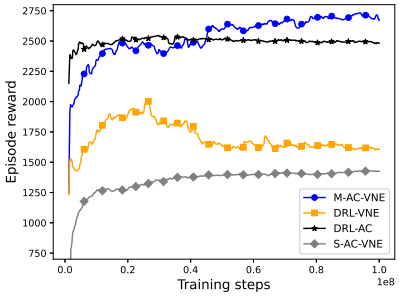

Fig. 9: Episode reward for DRL-based approaches

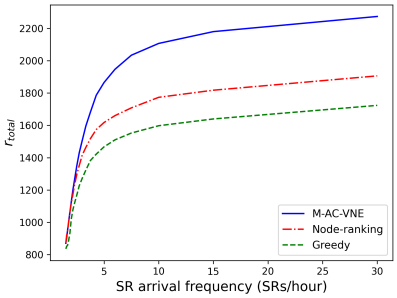

Fig. 10: Evaluation revenue for different SR arrival rates

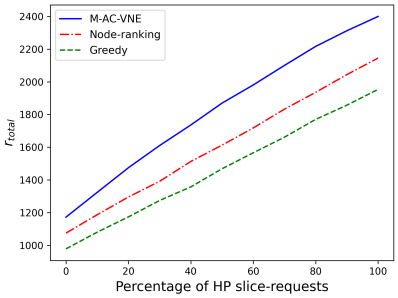

Fig. 11: Evaluation revenue for different HP SR percentage

diversified conditions, we vary the arrival rate and priority of SRs. The M-AC-VNE approach is trained with an average SR arrival rate of 5 SRs/hour and with 50% probability of any SR being HP. First, we introduce previously unseen load conditions to the agents, by deviating the SR arrival rate from the one used during training. Fig. 10 displays the robustness of the approach against such loads. In more saturated network conditions (*i.e.*, rapid arrival of SRs), not only the proposed approach maintains its performance, but the revenue gap between it and the baseline approaches also increases. At the highest point, the total revenue difference between our approach, and the Node-ranking and greedy approaches reaches 19.3% and 31.9%, respectively. Whereas, under less demanding conditions (*i.e.*, sporadic arrival of SRs), this difference is smaller. We speculate that in such situations, there is a lesser need for more intelligent decision-making, as there is a lesser chance of creating bottlenecks.

Furthermore, we evaluate the proposed approach under varying percentages of HP SRs. Fig. 11 shows the revenue achieved by the proposed approach and the baseline approaches in this scenario. It is evident that M-AC-VNE is able to generalize to this scenario as well and maintain its lead in the achieved long-term InP revenue. In cases where there are no HP SRs, or when all of the SRs are HP, the M-AC-VNE is still able to achieve a higher revenue when compared to baseline approaches as it is able to avoid resource bottlenecks. Additionally, DRL also makes use of the higher revenues offered by the HP SRs by admitting a higher percentage of corresponding SRs. As a result, as the percentage of HP SRs increases, the M-AC-VNE approach shows a higher rate of increase in revenue. However, as the HP SRs become oversaturated, the rate of increase decreases.

VI. Conclusion

In this paper, we studied the problem of joint slicing and AC in 5G C-RAN. Maximizing an InP's long-term revenue involves a number of factors which make the joint slicing and AC problem quite complex. This complexity calls for a close collaboration between the slicing and AC modules. We propose multi-agent DRL as an effective approach to address slicing and AC problems jointly and show that DRL-based approaches that address only one aspect of the problem lead to a loss in potential InP revenue. Our results show that the proposed approach achieves as much as 29.5% higher revenue when compared to approaches based on simple heuristics and DRL approaches that address the two problems separately. Our results also show that multi-agent DRL achieves 29.1% higher long-term InP revenue and leads to faster convergence, when compared to a single-agent DRL approach that jointly address the slicing and AC problems. In the future, we will evaluate our approach in a more complex metro 5G RAN environment. Further investigation is also needed to make the proposed approach more robust by training the agent under more diverse and varying network conditions.

Acknowledgement

This work was supported in part by Rogers Communications Canada Inc. and in part by a Mitacs Accelerate Grant. We thank Massimo Tornatore and Nashid Shahriar for their constructive feedback on the project and the manuscript.

REFERENCES

[1] 3GPP, "Study on new radio access technology: Radio access architecture and interfaces," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.801, Apr. 2017.

[2] "NGMN Overview on 5G RAN Functional Decomposition," NGMN Alliance, Final Deliverable, 2 2018, version 1.0.

[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] U. Dötsch, M. Doll, H.-P. Mayer, F. Schaich, J. Segel, and P. Sehier, "Quantitative analysis of split base station processing and determination of advantageous architectures for LTE," *Bell Labs Technical Journal*, vol. 18, no. 1, pp. 105–128, 2013.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] 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.

[7] J. S. Pujol Roig, D. M. Gutierrez-Estevez, and D. Gündüz, "Management and Orchestration of Virtual Network Functions via Deep Reinforcement Learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 304–317, 2020.

[8] S. Matoussi, I. Fajjari, S. Costanzo, N. Aitsaadi, and R. Langar, "5G RAN: Functional Split Orchestration Optimization," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1448–1463, 2020.

[9] S. Matoussi, I. Fajjari, N. Aitsaadi, and R. Langar, "Deep Learning based User Slice Allocation in 5G Radio Access Networks," in *IEEE Conference on Local Computer Networks (LCN)*, 2020, pp. 286–296.

[10] 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.

[11] X. Wang and T. Zhang, "Reinforcement Learning Based Resource Allocation for Network Slicing in 5G C-RAN," in *IEEE Computing, Communications and IoT Applications (ComComAp)*, Shenzhen, China, 2019, pp. 106–111.

[12] G. Dandachi, A. De Domenico, D. T. Hoang, and D. Niyato, "An Artificial Intelligence Framework for Slice Deployment and Orchestration in 5G Networks," *IEEE Transactions on Cognitive Comm. and Networking*, vol. 6, no. 2, pp. 858–871, Jun. 2020.

[13] V. Sciancalepore, L. Zanzi, X. Costa-Perez, and A. Capone, "ONETS: Online Network Slice Broker From Theory to Practice," *arXiv preprint arXiv:1801.03484*, 2020.

[14] F. W. Murti, J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Pérez, and G. Iosifidis, "An Optimal Deployment Framework for Multi-Cloud Virtualized Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2251–2265, 2021.

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

[16] 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.

[17] Q. Liu and T. Han, "DIRECT: Distributed Cross-Domain Resource Orchestration in Cellular Edge Computing," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 181–190.

[18] M. R. Raza, C. Natalino, P. Ohlen, L. Wosinska, and P. Monti, "Reinforcement Learning for Slicing in a 5G Flexible RAN," *Journal of Lightwave Technology*, vol. 37, no. 20, pp. 5161–5169, Oct. 2019.

[19] R. Boutaba, N. Shahriar, M. A. Salahuddin, S. R. Chowdhury, N. Saha, and A. James, "AI-Driven Closed-Loop Automation in 5G and beyond Mobile Networks," in *ACM Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility (FlexNets)*, 2021, p. 1–6.

[20] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine Learning for Cognitive Network Management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018.

[21] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.

[22] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 146–172, 2019.

[23] G.sup.5GP, "5G Wireless Fronthaul Requirements in a PON Context," ITU-T, G.Sup66 Supplement, Jul. 2019.

[24] C. Desset, B. Debaillie, V. Giannini, A. Fehske, G. Auer, H. Holtkamp, W. Wajda, D. Sabella, F. Richter, M. J. Gonzalez, H. Klessig, I. Gódor, M. Olsson, M. A. Imran, A. Ambrosy, and O. Blume, "Flexible power modeling of LTE base stations," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2012, pp. 2858–2862.

[25] D. Szczesny, A. Showk, S. Hessel, A. Bilgic, U. Hildebrand, and V. Frascolla, "Performance analysis of LTE protocol processing on an ARM based mobile platform," in *International Symposium on System-on-Chip*, 2009, pp. 056–063.

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

[27] 3GPP, "Transport requirement for CU&DU functional splits options," 3rd Generation Partnership Project (3GPP), discussion R3-162005, Aug. 2016.

[28] A. Maeder, M. Lalam, A. De Domenico, E. Pateromichelakis, D. Wübben, J. Bartelt, R. Fritzsche, and P. Rost, "Towards a flexible functional split for cloud-RAN networks," in *European Conf. on Networks and Comm. (EuCNC)*, 2014, pp. 1–5.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[31] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3053–3062.

[32] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A Research Platform for Distributed Model Selection and Training," *arXiv preprint arXiv:1807.05118*, 2018.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035.