Coordinated Slicing and Admission Control using Multi-Agent Deep Reinforcement Learning

Muhammad Sulaiman*, Arash Moayyedi*, Mahdieh Ahmadi*, Mohammad A. Salahuddin*, Raouf Boutaba*, and Aladdin Saleh†
*David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada
{m4sulaim, arash.moayyedi, mahdieh.ahmadi, 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 based on resource availability and slice constraints. Due to limited resources in the substrate network and variability in revenue of slices, the Infrastructure Provider (InP) must perform network slicing in a strategic manner, and accept or reject slice-requests 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 along with reward shaping is a promising choice, which 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 30% and 5.18% 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 for the joint problems in terms of convergence time and InP revenue. Finally, we evaluate the robustness of the trained agents in scenarios that differ from training, such as different arrival rates and real dynamic traffic patterns.**

***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 or 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 meeting 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 in 5G, that are tailored to satisfy the specific QoS requirements of different services on the same infrastructure. Network slices can include chains of RAN and core VNFs. The placement (*i.e.*, Virtual Network Embedding (VNE)) of RAN VNFs in 5G C-RAN should consider the service type and its Service-Level Agreements (SLAs). Accepting a slice-request (SR) contributes to the InP's revenue. However, given an InP's limited resources, it is impossible to serve all incoming SRs. Additionally, the amount of revenue that SRs bring may also vary (*e.g.*, based on their priority). 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]) have proposed either traditional reinforcement learning (RL) or DRL-based solutions to network slicing and AC problems. Though, AC and network slicing are both quintessential to offer differentiated 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 InP revenue. For instance, the authors in [7] 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, the works that consider networking slicing only (*e.g.*, [8]) 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 slicing decision causes future high revenue SRs to be rejected due to a resource bottleneck, the DRL agent anticipates this and avoids that decision.

Reward shaping is an important technique to ensure that an RL agent converges within a reasonable time. However, the restrictions in reward formulation for a single DRL agent does not allow to effectively address both AC and slicing jointly. For example, if the DRL 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 the optimal action. This concern is alleviated in multi-agent DRL (MADRL), 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.

This paper is an extension of our previous work on using MADRL to jointly address slicing and AC in 5G C-RAN, to improve the long-term InP revenue [9]. For this purpose, we design the reward functions such that they lead to convergence and cooperation among the agents. 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. We perform extensive experiments under training and evaluation environments to gauge convergence and robustness of the proposed method, and to provide an exhaustive investigation on how different agents contribute to InP revenue. In this regard, we demonstrate the following:

- We compare the convergence of DRL-based approaches under training environment and showcase the performance of the proposed approach under different configurations of hyper-parameters. We demonstrate the sensitivity to the choice of hyper-parameters and select the best configuration for evaluation.
- We compare the proposed MADRL-based solution against a greedy approach and a node-ranking approach (inspired by [10]), and show that our solution outclasses these heuristic methods 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]). Using extensive evaluations, we show that MADRL with the designed reward for jointly addressing both problems 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 MADRL-based approach outperforms the single-agent counterpart in terms of the achieved long-term InP revenue and convergence time.
- We evaluate the robustness of the trained agents under practical network conditions, such as variable SRs arrival rate, priority and throughput distributions. 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.
- We evaluate the trained agents using real traffic patterns to demonstrate the robustness of the proposed approach and the impact of dynamic arrival rates on InP revenue.

TABLE I: List of frequently used abbreviations and notations.

Abbreviation/Symbol	Meaning
AC	Admission Control
C-RAN	Cloud Radio Access Network
DRL	Deep Reinforcement Learning
eMBB	Enhanced Mobile Broadband
HP	High-Priority
InP	Infrastructure Provider
LP	Low-Priority
MADRL	Multi-agent Deep Reinforcement Learning
mMTC	Massive Machine Type Communication
RAN	Radio Access Network
RL	Reinforcement Learning
RU	Radio Unit
SLA	Service Level Agreement
SR	Slice-Request
URLLC	Ultra Reliable Low-latency Communication
VNE	Virtual Network Embedding
VNF	Virtual Network Function
s_t	SR arriving at time t
α^{s_t}	admission decision of SR s_t
$\boldsymbol{\eta^{s_t}}$	embedding-relationship matrix of SR s_t
$\boldsymbol{B^t}$	bandwidth capacity of substrate links at time t
$\boldsymbol{C^t}$	computation capacity of substrate nodes at time t

Table I facilitates reading by listing the frequently used abbreviations and notations in this paper. 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 provides an overview of DRL while Section V delineates the proposed multi-agent DRL-based slicing and AC solution. After showcasing the results in Section VI, we conclude in Section VII and instigate future research directions.

II. Related Works

There are numerous works in the literature that address AC and network slicing [6, 13]. In this section, we review these works with an emphasis on ML-based approaches.

A. Admission Control

Slice AC pertains to accepting or rejecting a new SR based on factors, such as the SR revenue, priority, resources, and QoS requirement, its impact on the existing and future SRs, and available substrate network resources. The authors in [7, 12] focus on AC with the objective of maximizing long-term InP revenue. Dandachi et al. [12] propose a traditional RL-based approach for 5G slice admission and congestion control. 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. Van Huynh et al. [7] leverage DRL for slice AC and resource allocation, but similar to [12], they model slices and the substrate network in aggregate. In addition, both of these works do not consider the dynamic nature of request arrivals.

Pujol Roig et al. [14] propose a DRL-based approach for dynamic VNF management and orchestration. Requests arrive for a list of individual NFs and are embedded on a pool of homogeneous servers in the Central Unit (CU) or in the remote cloud. Instead of a binary admission decision, when a new request arrives, a DRL agent decides to either scale the corresponding VNF vertically by allocating more resources to it, instantiate a new VNF on a separate server, or offload the VNF to the cloud. Their objective is to minimize the incurred resource and latency costs. Bega et al. [15] propose an RL-based slice admission solution for maximizing InP revenue. They employ two separate RL agents for estimating revenue in the case of accepting and rejecting SRs, respectively. The authors extended their work in [16] using DRL. However, these works only consider radio resources and require knowledge of the arrival process. Sciancalepore et al. [17] propose an online network slice brokering solution to maximize multiplexing gains. The problem is modeled as a budgeted lock-up multi-armed bandit problem, a variation of the well-known multi-armed bandit problem. Nevertheless, similar to [14, 15, 16], 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, computing, and communication resource requirements.

Raza et al. [18] is the closest to our work, where the authors propose a policy-based RL algorithm for slice AC in 5G C-RAN. However, the arriving SRs in their work, already specify the required computing resources at the remote and central sites based on the latency requirement. This sidesteps an important aspect of slicing, where all of an SR's functions can be placed at either the remote (*e.g.*, for URLLC applications) or the centralized location (*e.g.*, for mMTC applications). Additionally, the selection of the central location (*i.e.*, remote data center) is done using a heuristic after the AC decision has been made. This precludes the AC agent from knowing the embedding before making AC decision and can lead to performance degradation in resource-constrained environments.

B. Network Slicing

The works discussed in this section assume that SRs will be accepted until resources are saturated. Therefore, online proactive AC is not factored into the problem, and the focus is on optimizing the efficiency (*e.g.*, delay, resource cost, utilization) of resource allocation. Koo et al. [19] leverage DRL for network slicing when request are served immediately or in batch mode. They consider multi-dimensional resource allocation (*e.g.*, VMs, bandwidth, memory) with delay requirement that includes the processing delay of SRs. However, the authors consider a slice in aggregate. Wang and Zhang [11] use RL for network slicing in 5G C-RAN. The slice and substrate networks are modeled completely, with the objective of maximizing profit (*i.e.*, difference between revenue and cost). To simplify the problem, the authors divide the problem into function embedding and radio resource allocation, and solve them individually (*i.e.*, using different Q-learning models) rather than jointly. Gao et al. [20] propose a deep double Q-learning for RAN function placement and routing from RU to the data center. Their objective is to minimize delay and resource costs. However, the authors only consider the fronthaul delay constraint and the evaluation focuses on a single service (*i.e.*, slice) type.

In contrast, Solozabal et al. [21] use Neural Combinatorial Optimization paradigm for delay-aware service function chain placement. The authors incorporate resource capacity and delay constraints into the objective using Lagrange relaxation. They employ a DRL model architecture which incorporates an encoder-decoder design based on stacked Long Short-term Memory cells. The model can decide the placement for the whole chain of VNFs, however, to simplify the path selection, the servers are assumed to be connected through a star topology. ML models based on Graph Neural Networks have also been explored in [22] to improve generalization over different network topologies.

Also related are works that consider the functional split problem, which divides the RAN functions between RU and CU while minimizing cost. In [23, 24], authors propose an Integer Linear Problem (ILP) formulation for an offline version, where a functional split is decided for each cell in the network, instead of each user (*i.e.*, SR). The problem is solved optimally using Benders Decomposition in [23] and DRL in [25]. On the other hand, the authors in [8, 26] address the user-centric functional split problem. They model the problem as an 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 RU and a single CU.

Different from the works discussed above, we consider AC and slicing in conjunction by modeling substrate network as

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3222589

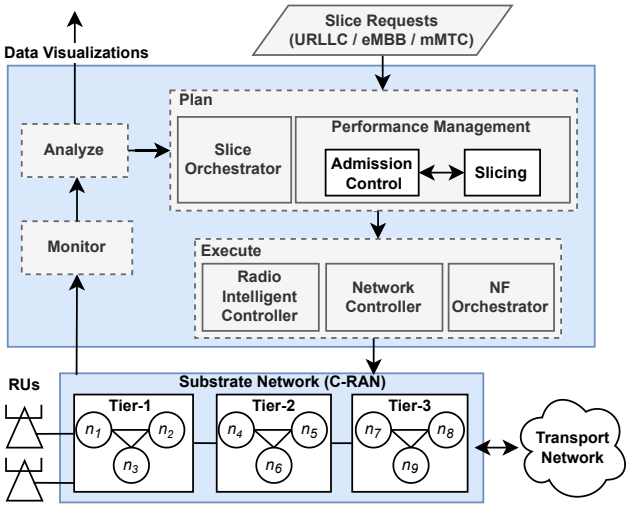


Fig. 1: High-level view of closed-loop, autonomous management and orchestration of VNFs in 5G C-RAN.

a graph with capacity constraints per node and edge, and slice requests as chains of VNFs with delay and throughput requirements, and finite operation time.

III. System Design

The system design is based on the MAPE (*i.e.*, monitor, analyze, plan, execute) control loop [27, 28] to facilitate closed-loop, autonomous management and orchestration of VNFs in 5G C-RAN, as depicted in Fig. 1. The monitor module intelligently collects data from the substrate network and sends it to the analyze module. From raw data, the analyze module extracts useful information and computes various metrics required for visualization and planning (*e.g.*, QoS, network infrastructure state). The processed data and information regarding incoming SRs are received by the plan module.

The plan module performs intelligent slice orchestration and performance management using AI/ML techniques [13]. The proposed intelligent AC and slicing schemes in this paper are sub-components of the performance management component, which are responsible for the admission and embedding of network slices. These sub-components can be employed either concurrently (*i.e.*, both output their decisions independently) or sequentially (*i.e.*, each sub-component can use the output of the other to make its decision). If an SR is admitted, the slice orchestrator passes the appropriate commands (*e.g.*, instantiation of VNFs, links) to the execute module which in turn directs VNF orchestrator, network controller and Radio Intelligent Controller (RIC) components to set up the virtual machines, transport paths, and RAN radio resources in the substrate network, respectively. The RIC, introduced and standardized by the O-RAN Alliance [29], provides advanced control and configuration functionality for efficient management of RAN infrastructure. In this paper, we simulate a substrate network, so the monitor, analyze and execute modules do not present a research challenge, and we only describe the pertinent system design of the AC and slicing components.

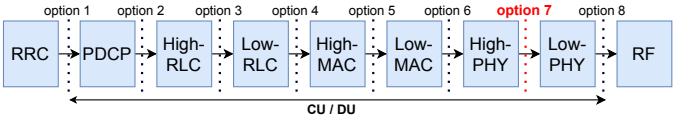


Fig. 2: Functional splits for VNFs in 5G C-RAN [1].

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 $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ represents the set of $|\mathcal{N}| \in \mathbb{N}$ nodes and $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$ represents the set of $|\mathcal{L}| \in \mathbb{N}$ links each with a certain latency. We use $l_{n,n'} \in \mathcal{L}$ to denote the link between the nodes $n, n' \in \mathcal{N}$. At any time t, a node $n \in \mathcal{N}$ has a certain amount of available computing resources, denoted by c_n^t, and a link $l \in \mathcal{L}$ 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 in Tier-1. With more centralized sites (*i.e.*, Tier 2–3), the experienced latency from the RU increases due to the increased path delay. Additionally, nodes at these sites are equipped with higher computing resources to support a larger 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], as 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 CU. With dynamic function splitting, the split for each SR 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 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, 30].

Based on the functional split options proposed by 3GPP, ITU recommends option 7 as the split between the RU and the DU [31], due to its high bandwidth and strict latency requirements. Therefore, in this paper, we consider Low-PHY and RF functions to always be placed at the RU. Additionally, since the compute 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 consider only High-PHY, MAC, RLC and PDCP functions, which are denoted by the set $\mathcal{F} = \{f_1, \cdots, f_4\}$.

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 [32]:

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

$$G_2 = G_2^{ref}.\frac{B}{B^{ref}}.\frac{A}{A^{ref}}.\frac{L}{L^{ref}}.\frac{M}{M^{ref}}, \quad (2)$$

$$G_3 = G_3^{ref}.\frac{B}{B^{ref}}.\frac{A}{A^{ref}}, \quad (3)$$

$$G_4 = G_4^{ref}.\frac{B}{B^{ref}}.\frac{A}{A^{ref}}, \quad (4)$$

$$G_5 = G_5^{ref}.\frac{B}{B^{ref}}.\frac{A}{A^{ref}}, \quad (5)$$

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, number of MIMO antennas, load, and modulation in the reference scenario [32, 33], respectively. The output bandwidth requirements of each VNF can be calculated as:

$$R_i(\lambda) = k_{i,1}\lambda + k_{i,2}, \forall f_i \in \mathcal{F}, \quad (6)$$

where λ is the slice throughput in Mbps, and $k_{i,1}$, $k_{i,2}$ are constants with specific values for the different functions (*i.e.*, splits) which can be calculated using [34]. The latency requirements depend on the VNF and the end-to-end latency requirement of the SR [35, 36].

C. Slice-requests

Each incoming SR is assumed to belong to one of the following service types: eMBB, URLLC, or mMTC. SRs arrive one at a time and if accepted, remain operational for a specific duration. An SR arriving at time t, denoted by s_t, is characterized by a service type, a throughput requirement λ^{s_t} that it needs to support, an end-to-end latency requirement, an operation time τ^{s_t}, and offered revenue per unit time r^{s_t}. The admission and slicing decisions must be made when the SR arrives, without any knowledge of future arrivals. 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. It is also assumed that each slice is of either high-priority (HP) or low-priority (LP), and the offered revenue is proportional to the priority.

D. Joint Slicing and AC—Problem Formulation

Let $\alpha^{s_t} \in \{0, 1\}$ denote whether SR s_t is admitted (*i.e.*, $\alpha^{s_t} = 1$) or not (*i.e.*, $\alpha^{s_t} = 0$). The slicing decision for any SR s_t is defined by the following embedding-relationship matrix:

$$\boldsymbol{\eta^{s_t}} = \left[\eta_{i,n}^{s_t} \in \{0, 1\} : 1 \leq i \leq |\mathcal{F}|, n \in \mathcal{N}\right], \quad (7)$$

where $\eta_{i,n}^{s_t} = 1$ if f_i is mapped to substrate network node $n \in \mathcal{N}$, and $\eta_{i,n}^{s_t} = 0$ otherwise. The path selection between 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 nodes n and n', respectively, where $n, n' \in \mathcal{N}$. $\phi_{l,n,n'} = 1$ if link $l \in \mathcal{L}$ is in the shortest path between nodes $n, n' \in \mathcal{N}$. Whereas, $\phi_{l,ru,n} = 1$ if link $l \in \mathcal{L}$ is in the shortest path between RU and the node $n \in \mathcal{N}$. The admission and slicing decisions, for any SR s_t, must satisfy the following constraints:

$$\alpha^{s_t} = \sum_{n \in \mathcal{N}} \eta_{i,n}^{s_t}, \quad \forall 1 \leq i \leq |\mathcal{F}|, \quad (C1)$$

$$\sum_{i=1}^{|\mathcal{F}|} \eta_{i,n}^{s_t} \cdot c_i^{s_t} \leq c_n^t, \quad \forall n \in \mathcal{N}, \quad (C2)$$

$$\sum_{n \in \mathcal{N}} \eta_{1,n}^{s_t} \cdot d_{ru,n} + \sum_{j=2}^{i} \sum_{n \in \mathcal{N}} \sum_{n' \in \mathcal{N}} \eta_{j-1,n}^{s_t} \cdot \eta_{j,n'}^{s_t} \cdot d_{n,n'} \leq d_i^{s_t}, \quad \forall 1 \leq i \leq |\mathcal{F}|, \quad (C3)$$

$$\sum_{n \in \mathcal{N}} \eta_{1,n}^{s_t} \cdot R_1(\lambda^{s_t}) \cdot \phi_{l,ru,n} + \sum_{i=1}^{|\mathcal{F}|-1} \sum_{n \in \mathcal{N}} \sum_{n' \in \mathcal{N}} \eta_{i,n}^{s_t} \cdot \eta_{i+1,n'}^{s_t} \cdot R_i(\lambda^{s_t}) \cdot \phi_{l,n,n'} \leq b_l^t, \quad \forall l \in \mathcal{L}, \quad (C4)$$

where $c_i^{s_t}$ and $R_i(\lambda^{s_t})$ denote the computing resource requirement and output data-rate for VNF f_i of SR s_t, respectively. These are calculated using equations (1)–(5) and equation (6), respectively. $d_i^{s_t}$ is used to denote the latency requirement of the VNF f_i of SR s_t. Constraint (C1) formalizes the relationship between admission and embedding variables for each SR s_t and ensures that all VNFs of an admitted SR are mapped to one substrate network node, which is an assumption commonly made in the literature to simplify the problem. Constraint (C2) makes sure 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, while constraint (C4) makes sure that the link bandwidth limits are not exceeded.

An embedding-relationship matrix $\boldsymbol{\eta^{s_t}}$, for any SR s_t, is valid if it satisfies all of the above constraints assuming the SR is to be admitted, *i.e.*, $\alpha^{s_t} = 1$. An SR s_t is feasible if for the admission, *i.e.*, $\alpha^{s_t} = 1$, there exists a $\boldsymbol{\eta^{s_t}} \in \{0, 1\}^{|\mathcal{F}| \times |\mathcal{N}|}$ that is valid, *i.e.*, meets constraints (C1)–(C4) at time t. The objective of slicing and AC is to maximize the cumulative InP revenue achieved from the admission of SRs which can be expressed as:

$$r_{total} = \sum_{t \in \boldsymbol{T}} \alpha^{s_t} \cdot r^{s_t} \cdot \tau^{s_t}, \quad (8)$$

where $\boldsymbol{T}$ denotes the set of arrival times of SRs dedicated to different service types. Finally, the problem of joint slicing and AC for maximizing InP revenue can be formulated as:

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\eta}} \quad r_{total} \quad \text{s.t.} \quad (C1) - (C4). \quad (9)$$

In a slicing-only problem, when enough resources are available, all SRs are admitted until resources become saturated. A problem formulation in this case will only include embedding variables. On the other hand, in the joint AC and slicing problem, AC is part of the problem formulation. In this case, admission decisions can be used to reject SRs in order to

prevent resource bottlenecks, or to preserve resources for future SRs with potentially higher revenue.

IV. Deep Reinforcement Learning—Primer

In reinforcement learning, an agent interacts with an environment to learn actions that maximize the expected cumulative reward [37]. The interaction between the RL agent and the environment can be formally described using a markov decision process (MDP). The agent-environment interaction in this paper spans an infinite horizon, *i.e.*, the agent acts continuously. Therefore, we consider infinite horizon MDP defined by the tuple (O, A, r, P, ρ_0, γ), where O is the state space, A is the action space, $r : O \times A \times O \to \mathbb{R}$ is the reward function, $P : O \times A \times O \to [0, 1]$ is the state transition probability distribution, ρ is the initial state distribution, and γ is the discount factor. The aim of the RL agent is to learn either a deterministic policy $\pi : O \to A$ or a stochastic policy $\pi : O \times A \to [0, 1]$ that maximizes the expected discounted return given as $G_t = \sum_{k=t+1}^{\infty} \gamma^{k-t-1} r_k$, where r_k is the reward at time step k.

For policy π, the state value function V_π, state-action value function Q_π, and the advantage function A_π are defined as [37]:

$$V_\pi(o) = \mathbb{E}_\pi \left[G_t \mid o_t = o \right], \forall o \in O, \tag{10}$$

$$Q_\pi(o, a) = \mathbb{E}_\pi \left[G_t \mid o_t = o, a_t = a \right], \forall o \in O, \forall a \in A, \tag{11}$$

$$A_\pi(o, a) = Q_\pi(o, a) - V_\pi(o), \forall o \in O, \forall a \in A, \tag{12}$$

i.e., the expected return of starting from state o in the case of $V_\pi(o)$, and starting from state o and taking the action a in the case of $Q_\pi(o, a)$. $A_\pi(o, a)$ measures the relative state-action value of taking action a in state o as compared to the state value of state o. The state-value and state-action value functions are collectively referred to as value functions. When the number of possible states and actions is small, a tabular method can be used to store the value functions and derive an effective policy. However, this method becomes inefficient as the size of the state and action space increases. DRL approximates these tables using deep neural networks, which have the ability to generalize to previously unseen states, while involving a relatively smaller number of learnable parameters. DRL policy optimization algorithms in the literature are based on either learning the policy directly, learning value functions, or learning both. In the latter case, they are called Actor-Critic methods and have been shown to lead to faster empirical convergence [37]. In the following, we briefly discuss the employed policy optimization method.

A. Proximal Policy Optimization

Let π_θ denote the stochastic policy in DRL, *i.e.*, a neural network parameterized by the weights θ. Policy-based methods learn neural network parameters using optimization methods such as policy-gradient. In this paper, we leverage Proximal Policy Optimization (PPO) [38] which is a well-known actor-critic policy-gradient method with monotonic behaviour [39]. In this method, policy network's parameters are updated as:

$$\theta \leftarrow \max_{\theta} \quad \hat{\mathbb{E}}_t \left[\min \left(\frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)} \hat{A}_t \ , \right. \right. \\ \left. \left. \textit{clip} \left(\frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right], \tag{13}$$

where θ, θ_{old} are the parameters before and after the update respectively, $\hat{\mathbb{E}}_t$ is the empirical expectation, $\hat{A}_t$ is the empirical advantage function, and *clip* limits parameter values within the range specified by the PPO hyper-parameter ϵ. We estimate the empirical advantage function using Generalized Advantage Estimator (GAE) [40].

B. Reward Shaping

As equations (10)-(13) suggest, when taking an action, the RL agent not only takes the immediate reward into account but also the discounted rewards it expects to receive in the future. Consequently, an RL agent is able to learn meaningful state-action values even when it receives no (*i.e.*, zero) immediate reward. However, the sparsity of the reward greatly affects the convergence of an RL agent's policy and has made learning from sparse rewards a major challenge in RL [41]. Reward shaping is the process of giving the RL agent additional carefully designed rewards, such that they steer the agent towards the desired behavior faster. Shaped rewards should reflect the contribution each action will have on the long-term reward. Additionally, they should not lead to unintended behavior for the RL agent [42]. We will discuss the reward design for joint slicing and AC problem in Section V-A2.

C. Multi-agent Deep Reinforcement Learning

In multi-agent RL, multiple RL agents act in a shared environment to maximize the long-term return. Agents can be designed to operate cooperatively, competitively, or in a mixed setting. Multi-agent settings often violate the fundamental assumptions underlying the theoretical foundation of single-agent RL that are necessary to guarantee convergence [43]. For example, when multiple RL agents are concurrently learning and acting in a common environment, the environment becomes non-stationary from the perspective of any individual agent. This can prevent the agents' policies from converging towards the optimal even when the goals of different agents are aligned. MADRL refers to a multi-agent RL environment studied through deep learning.

V. Proposed Solution

In this section, we first define the RL environment components for the joint slicing and AC problem and then expose the proposed DRL-based solutions.

A. RL Environment

1) Markov Decision Process

For the joint problem of slicing and AC, the three main components of MDP are as follows:

State (O): The state space includes the information of incoming SR and real-time representation of the substrate network. Specifically, it includes incoming SR's (s_t) service

type (*i.e.*, URLLC, eMBB, mMTC), operation time (τ^{s_t}), and offered revenue (r^{s_t}), and the state of substrate network, ($\boldsymbol{C^t}, \boldsymbol{B^t}$). Note that the type of service determines the specific throughput and end-to-end latency requirements of each SR.

Action (A): A decision should be made at each step about whether to admit the incoming SR and how the set of its VNFs ($\mathcal{F}$) should be embedded in the substrate network. The admission action has two possible values, *i.e.*, $\alpha^{s_t} \in A^{\text{AC}}$ where $A^{\text{AC}} = \{0, 1\}$. However, the slicing action space includes a set of $|\mathcal{F}|$-dimensional vectors, one entry for each VNF in the chain, representing the mapping to substrate network nodes, *i.e.*, $A^{\text{VNE}} = \{[a_{f_1}, \cdots, a_{f_{|\mathcal{F}|}}] : a_{f_i} \in \mathcal{N}\}$. Note that we denote this action by embedding-relationship matrix $\boldsymbol{\eta^{s_t}}$, defined in (7), which is the binary representation of the slicing action vector $[a_{f_1}, \cdots, a_{f_{|\mathcal{F}|}}]$, *i.e.*, $\eta^{s_t}_{i,n} = \mathbb{1}_{[a_{f_i}=n]}$. Finally, the joint action space is discrete, equalling $A = A^{\text{AC}} \times A^{\text{VNE}}$, having a size of $|A| = 2|\mathcal{N}|^{|\mathcal{F}|}$ which is polynomial in the size of substrate nodes. We will discuss how we deal with this large action space in Section V-D.

Reward (r): The reward function may either be a per-step reward or a per-episode reward. In the former case, it is the revenue offered by an admitted and successfully embedded SR s_t. In the latter case, the reward includes the sum of revenues for all the accepted and successfully embedded SRs, and is given at the end of episode. Both of these rewards will lead the agent towards the same optimal policy. However, they differ in the convergence time as explained in Section IV-B.

Finally, the initial state distribution ρ_0 includes the initial network state ($\boldsymbol{C^0}, \boldsymbol{B^0}$), and the specifications of the initial SR sampled from an SR arrival distribution. The state-transition probabilities P depend on the SR arrival and operation time distributions. Finally, the discount factor $\gamma \to 1$ to reflect the equal importance of future and current revenues. The terminal state is reached after the arrival of all SRs.

2) Reward Design

The long-term revenue maximization objective is realized by rewarding the RL agent with either per-step or per-episode reward from the underlying MDP. Using reward from the underlying MDP, however, comes with a number of problems in our case. First, since the state and action space for the joint problem is quite large and most of the slicing actions are invalid, in practice, the agent has to perform a huge amount of exploration to learn the optimal actions in each state. This problem is exacerbated in per-episode reward from the underlying MDP, since the agent has to learn the optimal action by propagating the reward effect (*i.e.*, advantage function) from the terminal state to the initial states (*cf.*, (13)). Aside from intractably slow learning, another problem that may arise even in the case of per-step reward is that the agent may learn undesired behavior, *i.e.*, instead of using the AC action to reject SRs, it might learn to reject them by producing invalid slicing actions.

In this paper, we use per-step shaped reward which can be used to address these issues, drive the agent to learn faster and produce the desired behavior. For example, since an invalid slicing action violates either one or multiple constraints, the RL agent can be guided to make fewer invalid slicing actions if it receives a negative reward proportional to the number of constraints it violated at each step. However, in this case, the agent may get biased towards the shaped reward instead of the reward produced by the underlying MDP, *i.e.*, the revenue [37]. For example, an agent that receives a negative shaped reward for producing optimal AC action and invalid slicing action might get biased towards producing non-optimal AC actions. We address this problem using a MADRL approach, where distinct agents produce the slicing and AC actions and are rewarded separately. In this case, it is easier to prevent bias as a negative reward for an invalid slicing action will not penalize the corresponding AC action. The proposed MADRL framework is explained in the next subsection.

B. Multi-agent DRL Approach (M-AC-VNE)

To jointly address the slicing and AC problems, we propose a MADRL-based approach, referred to as M-AC-VNE. In this approach, two separate agents—a slicing agent and an AC agent—coordinate with each other in order to maximize the long-term revenue. The slicing agent takes the slicing action $\boldsymbol{\eta^{s_t}}$ and the AC agent takes admission action α^{s_t} for each given SR s_t. The slicing agent is only invoked if the given SR s_t is feasible (*i.e.*, a valid $\boldsymbol{\eta^{s_t}}$ exists), and the AC agent is invoked afterward if the slicing agent produces a $\boldsymbol{\eta^{s_t}}$ that is valid. The feasibility check is done before invoking the slicing agent so that actions produced by the agents always have the potential to be effective. This also helps to discriminate whether an invalid slicing action is due to fundamental resource limitation or the inability of the agent to find a valid solution. The observation space of each agent is the same as the state space described in Section V-A1. To accelerate learning, the AC agent also receives the slicing action $\boldsymbol{\eta^{s_t}}$, decided by the slicing agent, as the state. Note that the coordination and information exchange between agents is confined to data passing within the same machine as shown in Fig. 1 and has no communication constraint.

The reward function of the slicing agent described in Algorithm 1 depends on the number of constraints (C2)–(C4) violated by $\boldsymbol{\eta^{s_t}}$, the number of subsequent SRs (S_{sub}) that remain infeasible and AC agent's admission decision α^{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 gained revenue. As described in Algorithm 2, if the AC agent admits s_t (*i.e.*, $\alpha^{s_t} = 1$), it gets the total offered revenue of that s_t during its lifetime, *i.e.*, $r^{s_t} * \tau^{s_t}$, 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 of rejecting SRs that offer less revenue (*i.e.*, LP SRs) and are likely to cause bottlenecks (*i.e.*, resource intensive SRs). The constants (*i.e.*, +/- 1, 1.5) used in Algorithm 1 to balance between negative and positive rewards are based on trial-and-error and lead to faster learning.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3222589

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

Function REWARDVNE(s_t, α^{s_t}, $\boldsymbol{\eta^{s_t}}$, $\boldsymbol{C^t}, \boldsymbol{B^t}$, subsequent SRs $\boldsymbol{S}_{\text{sub}}$)
Output: slicing agent's $reward$
$reward \leftarrow 0$
foreach *constraint* (C2)–(C4) *violated by* $\boldsymbol{\eta^{s_t}}$ **do**
  $reward \leftarrow reward - 1$
if *(*$\boldsymbol{\eta^{s_t}}$ *is valid)* $\wedge$ *(*$\alpha^{s_t} = 1$*)* **then**
  $reward \leftarrow reward + 1.5$
for $s_{sub} \in \boldsymbol{S}_{sub}$ **do**
  if s_{sub} *not feasible* **then**
    $reward \leftarrow reward - 1.5$
  else
    return $reward$

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

Function REWARDAC(s_t, α^{s_t}, $\boldsymbol{\eta^{s_t}}$, $\boldsymbol{C^t}$, $\boldsymbol{B^t}$, subsequent SRs $\boldsymbol{S}_{\text{sub}}$)
Output: AC agent's $reward$
$reward \leftarrow 0$
if $\alpha^{s_t} = 1$ **then**
  $reward \leftarrow reward + (r^{s_t} * \tau^{s_t})$
for $s_{sub} \in \boldsymbol{S}_{sub}$ **do**
  if s_{sub} *not feasible* **then**
    $reward \leftarrow reward - (r^{s_{\text{sub}}} * \tau^{s_{\text{sub}}})$
  else
    return $reward$

1) Training Algorithm

We optimize the parameters of both AC and slicing agents' policy networks separately by utilizing the PPO algorithm [38], which is an on-policy, model-free, actor-critic algorithm that outperformed other DRL algorithms during our trials. At each training iteration, first current policies of AC (π^{AC}) and slicing (π^{VNE}) agents are run in the simulated environment to gather a set of trajectories for each agent. A trajectory is defined as a sequence of experiences, *i.e.*, $\{o_0, a_0, r_0, o_1, \cdots, o_{T-1}, a_{T-1}, r_{T-1}, o_T\}$, gathered from any interaction between the agent and the environment. Algorithm 3 demonstrates how we generate these trajectories for a single episode. For the slicing agent, an experience is added for each feasible SR in the episode, whereas for the AC agent, an experience is added whenever the slicing agent finds a valid embedding. The size of the trajectory gathered in an episode may therefore differ for AC and slicing agents. Finally, the parameters of the actor and critic networks of both agents are updated simultaneously by adopting the rules explained in Section IV-A. It should be noted that even though there is a non-zero reward at each step of the trajectory (*i.e.*, per-step reward), the actual policy is not updated until a batch of trajectories has been collected.

Algorithm 3 Trajectory generation for one episode in M-AC-VNE.

Input : AC agent policy π^{AC}, Slicing agent policy π^{VNE}, list of generated SRs in one episode $\mathcal{S}$
Output: AC and slicing agents' trajectories, $\mathcal{D}_{AC}$, $\mathcal{D}_{VNE}$
initialize $\mathcal{D}_{AC} \leftarrow \{\}$, $\mathcal{D}_{VNE} \leftarrow \{\}$
for $s_t \in \mathcal{S}$ **do**
  update netw. state $(\boldsymbol{C^t}, \boldsymbol{B^t})$ based on in-service SRs at t
  if s_t *not feasible* **then**
    continue
  else
    update the state of slicing agent $\boldsymbol{o_t^{\text{VNE}}} \leftarrow (\boldsymbol{C^t}, \boldsymbol{B^t}, s_t)$
    sample slicing agent's action $\boldsymbol{\eta^{s_t}} \sim \pi^{\text{VNE}}(a|\boldsymbol{o_t^{\text{VNE}}})$
    if $\boldsymbol{\eta^{s_t}}$ *is valid* **then**
      update the state of AC agent $\boldsymbol{o_t^{\text{AC}}} \leftarrow (\boldsymbol{o_t^{\text{VNE}}}, \boldsymbol{\eta^{s_t}})$
      sample AC agent's action $\alpha^{s_t} \sim \pi^{\text{AC}}(a|\boldsymbol{o_t^{\text{AC}}})$
      $r_t^{\text{AC}} \leftarrow$ REWARDAC(s_t, α^{s_t}, $\boldsymbol{\eta^{s_t}}$, $\boldsymbol{C^t}$, $\boldsymbol{B^t}$, $\boldsymbol{S}_{\text{sub}}$)
      $\mathcal{D}_{AC}$.append($\boldsymbol{o_t^{\text{AC}}}, \alpha^{s_t}, r_t^{AC}$)
    $r_t^{\text{VNE}} \leftarrow$ REWARDVNE(s_t, α^{s_t}, $\boldsymbol{\eta^{s_t}}$, $\boldsymbol{C^t}$, $\boldsymbol{B^t}$, $\boldsymbol{S}_{\text{sub}}$)
    $\mathcal{D}_{VNE}$.append($\boldsymbol{o_t^{\text{VNE}}}, \boldsymbol{\eta^{s_t}}, r_t^{\text{VNE}}$)
    if $\alpha^{s_t} = 1$ **then**
      embed s_t based on $\boldsymbol{\eta^{s_t}}$
return $\mathcal{D}_{AC}$, $\mathcal{D}_{VNE}$

C. Single-agent DRL Approach (S-AC-VNE)

For comparison, we also design the single-agent DRL approach, referred to as S-AC-VNE. In this approach, both slicing and admission actions are concurrently produced by a single DRL agent. The observation and action spaces are similar to those of the MDP described in Section V-A1. The reward function of this approach is defined in Algorithm 4.

Algorithm 4 RL agent's reward in S-AC-VNE.

Input : s_t, α^{s_t}, $\boldsymbol{\eta^{s_t}}$, $(\boldsymbol{C^t}, \boldsymbol{B^t})$, subsequent SRs ($\boldsymbol{S}_{\text{sub}}$), max offered revenue by any SR (r_{max})
Output: RL agent's reward
$reward \leftarrow 0$
foreach *Constraint* (C2)–(C4) *violated by* $\boldsymbol{\eta^{s_t}}$ **do**
  $reward \leftarrow reward - (r_{max} * \tau^{s_t})$
if *(*$\boldsymbol{\eta^{s_t}}$ *is valid)* $\wedge$ *(*$\alpha^{s_t} = 1$*)* **then**
  $reward \leftarrow reward + (r^{s_t} * \tau^{s_t})$
for $s_{sub} \in \boldsymbol{S}_{sub}$ **do**
  if s_{sub} *not feasible* **then**
    $reward \leftarrow reward - (r^{s_{\text{sub}}} * \tau^{s_{\text{sub}}})$
  else
    return *reward*

If s_t is successfully embedded, the reward is proportional to r^{s_t}. Otherwise, a negative reward proportional to the number of violated constraints by $\boldsymbol{\eta^{s_t}}$ and the maximum offered revenue is given to the agent. This will enforce the agent to utilize its AC action and reject SRs, instead of producing an infeasible embedding-relationship matrix for the SRs. In the single-agent case, the slicing and admission actions are awarded together

based on the same reward function. Therefore, 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, and eventually leads to a slower and potentially non-optimal learning for S-AC-VNE, as shown in Section VI. The training process and trajectory generation are similar to the process of M-AC-VNE except that both actions are taken from the same policy network and there will be only one trajectory with both actions combined.

D. Implementation Remarks

The large action space of the joint AC and slicing problem stems from the action space of the $|\mathcal{F}|$-dimensional slicing action, as discussed in Section V-A1. Conventionally the RL-agent's policy network is designed such that the size of the output layer equals that of the action space, and the logits of this policy network are interpreted by a Softmax distribution to produce the action probabilities. This can lead to prohibitively slow learning when dealing with large substrate networks in our case. In this paper, we use the *action branching* neural network architecture [44] for our policy networks in both the single-agent and multi-agent approaches. In this architecture, after some initial common layers, the network is divided into separate branches, one for each action dimension (*i.e.*, VNF). The Softmax layer at each separate branch is then used to produce independent action probabilities for each VNF. The final action probabilities can then be calculated by taking their product. In the case of the slicing agent, this reduces the size of the output layer of the policy network from $\prod_{i=1}^{|\mathcal{F}|} |\mathcal{N}|$ to $\sum_{i=1}^{|\mathcal{F}|} |\mathcal{N}|$ and leads to significantly faster learning.

We use a brute-force approach to check the feasibility of the incoming, or future SRs in Algorithms 1-4 based on the state of the network. Assuming $|\mathcal{N}| >> |\mathcal{F}|$, $O(|\mathcal{N}|^{|\mathcal{F}|})$ operations may be required, for checking the feasibility of an SR at a given network state. Although the number of substrate network nodes in real-life networks is not too large, we make reasonable efforts to optimize the process. These optimizations include limiting the search space for each VNF to those substrate nodes for which the shortest delay between the RUs and nodes is less than the delay requirement of the VNF. Additional optimizations can also be made based on the topology of the substrate network. For the scenario described in Section VI-B, each single experience gathered by simulating the environment requires $2.15 \pm 0.04ms$ with 95% confidence interval and feasibility checking constitutes the majority of this time, requiring $1.2 \pm 0.04ms$. However, this step does not present a significant problem during training (*i.e.*, trajectory generation) since multiple instances of simulation can be run in parallel. Additionally, during evaluation, due to similar implementations, both the heuristics-based approaches detailed in Section VI-C1 have $5.54 \pm 0.033ms$ running time, while the proposed approach requires a relatively constant mean inference time of $1.3ms$.

TABLE II: One-way latency requirement of VNFs for each service type (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

VI. Simulation and Results

We first describe the simulation environment and comparative approaches. Then, we discuss different experiments to evaluate the performance of proposed approaches in the training and evaluation environments.

A. Environment Setup

The simulation is implemented as a custom OpenAI Gym [45] environment, coupled with RLLib [46] on a cluster of 3 servers. Each server has 16GB of RAM, 8x Intel Xeon 3.30GHz cores and runs Ubuntu 16.04. Cluster management is done through Ray [47], and DRL algorithms are implemented in RLLib, using Python 3.8.9 and leveraging PyTorch [48]. This setup enables distributed learning to accelerate the training and hyper-parameter optimization process.

B. Simulation Parameters

In the training phase, the SRs arrive as a Poisson process with an average rate of 5 SRs per hour. The operating hours of each SR come from the normal distribution $\mathcal{N}(6, 0.5)$. Any incoming SR with a probability of 0.5 will be mapped to eMBB, and the rest will be mapped to URLLC and eMBB equally. URLLC, mMTC, and eMBB SRs require 75Mbps, 75Mbps, and 150Mbps of throughput, respectively. For each service type, the maximum one-way latency required for each RAN VNF [34] and the end-to-end latency SLA are given in Table II. These values are used to derive effective latency requirements (*i.e.*, $d_i^{s_t}$) for f_1–f_4. Also, an SR is either HP or LP, with a probability of 0.5. A HP SR offers twice the revenue per hour than a LP SR on average.

For training and evaluating the proposed solution, we utilize a 3-tiered topology, having 9 nodes, 3 in each tier, as shown in Fig. 1. 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 assumed 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.

C. Comparative Approaches

We utilize two heuristic-based and three DRL-based approaches for comparison, aiming to encompass state-of-the-art baselines and approaches in 5G C-RAN slicing and AC.

1) Heuristics-based

(i) Greedy: All SRs are accepted for embedding and each VNF is placed at the closest node to the RU with sufficient resources to host the VNF. In case of a tie, the node with the highest available resources is selected. Since VNFs are first placed at closer nodes, the delay requirement will be met when the SR is feasible and minimum link bandwidth is used. However, this approach is naïve, as all VNFs will be first placed at Tier-1 and once the compute resources at this site are exhausted, low latency VNFs cannot be placed anymore. Consequently, all additional SRs are rejected until resources at Tier-1 site become available again.

(ii) Node-ranking: Inspired by [10], we design the node-ranking approach for slicing. Starting from the last VNF 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 link bandwidth constraints is selected. The ranking method is the same for all service types and it puts the least strain on Tier-1 site's computing resources until the bandwidth to higher-tier sites becomes saturated.

2) DRL-based

A number of works in the literature propose DRL for AC without proper modeling of network slicing [7, 12], and for network slicing without modeling AC [11]. To cover most of these approaches, in addition to the single-agent DRL method (*i.e.*, S-AC-VNE) described in Section V, we consider two additional DRL-based benchmark solutions:

(i) DRL-AC: DRL is used for AC, but slicing is done through the node-ranking approach described earlier.

(ii) DRL-VNE: DRL is used for slicing, but all SRs are accepted until resources are saturated.

For a fair comparison, the reward function for the DRL-agent in both approaches is the same as the corresponding DRL-agent in the M-AC-VNE approach.

D. Training

The performance of DRL-based solutions depends on hyper-parameter choices. Therefore, before evaluating different DRL-based approaches, it is imperative that the best hyper-parameters are identified. For this, we use the grid-search algorithm which consists of dividing the domain of hyper-parameters into a grid of values, training RL agents for each setting, and then selecting the configuration that yields the highest average revenue for evaluation. The candidate parameters for grid-search include the policy network's shapes of $[32, 32, 32]$ and $[128, 128, 128]$, the train batch sizes of 4096 and 16348, and the number of stochastic gradient descent (SGD) steps of 8 and 16 per PPO iteration. This process is applied for M-AC-VNE and S-AC-VNE. The optimal hyper-parameters from the M-AC-VNE approach are employed for DRL-AC and DRL-VNE. Fig. 3 shows the average revenue achieved during training when using different hyper-parameters for the M-AC-VNE approach. The deduced optimal DRL hyper-parameters are listed in Table III, while the corresponding revenue and mean reward achieved during training are shown in Fig. 4 and Fig. 5, respectively.

TABLE III: PPO training hyper-parameters.

DRL Hyper-parameter	M-AC-VNE	S-AC-VNE
# Hidden-layer neurons	[32, 32, 32]	[32, 32, 32]
LR at timestep [0, 10^7, 10^8]	[3e-3, 5e-4, 1-e4]	[3e-3, 5e-4, 1-e4]
Train batch size	16348	16348
Mini-batch size	4096	4096
# SGD iters	16	16
KL coefficient, Lambda	0.4, 0.95	0.4, 0.95

As discussed in Section IV-C, achieving policy equilibrium and agent cooperation are two challenges of multi-agent RL that are relevant to the joint AC and slicing problem. Fig. 4 shows that the proposed solution achieves policy equilibrium as the revenue converges without oscillations. In addition, the fact that M-AC-VNE achieves higher revenue than single-agent RL-based approaches demonstrates cooperation between the agents. We can see that M-AC-VNE learns to achieve a high revenue quite early in its training. This is because initially, when ample resources are available, it can place incoming SRs without difficulty. But as the resources become scarce, it has to optimize the balance between the compute resource at lower tiered sites, and inter-tier bandwidth required to utilize higher tiered sites. This requires more complex decision making which causes the learning to slow down.

We can observe that the episode reward for DRL-AC is the earliest to reach a plateau during its training, followed by DRL-VNE. This is because of their smaller action space, *i.e.*, these approaches require DRL to learn either slicing or AC actions, but not both. Additionally, we can see that although the episode reward for DRL-VNE decreases after 25M training steps, the revenue (*cf.*, Fig. 4) does not follow the same trend. This is because, in addition to the number of embedded SRs, the shaped reward for the DRL-agent also depends on the number of SR constraints' violations. From Fig. 5, we can see that, 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 due to their larger action space. Moreover, M-AC-VNE converges to a higher episode revenue compared to S-AC-VNE. It is possible for the latter to achieve the same long-term revenue 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. However, in practice, using the shaped rewards, S-AC-VNE achieves much lower revenue within $100M$ training steps.

E. Evaluation

1) InP Revenue

The final InP (*i.e.*, infrastructure provider) revenue achieved during evaluation is shown in Fig. 6. Since DRL-agents do not perform exploration during evaluation, the average revenue achieved is slightly higher as compared to the training phase. Evidently, the greedy approach achieves the lowest revenue.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3222589

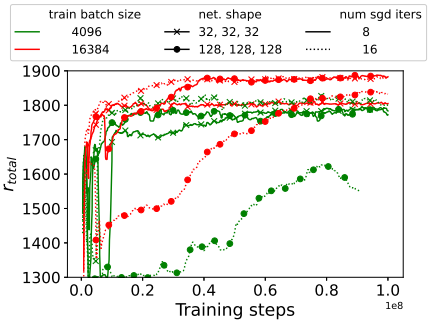


Fig. 3: Episode revenues of M-AC-VNE for different hyper-parameters.

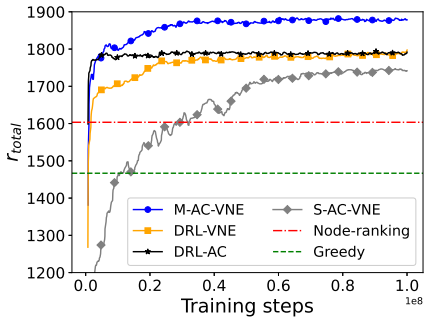


Fig. 4: Episode revenues for DRL-based approaches. Revenue gained by heuristics shown as dotted lines for reference.

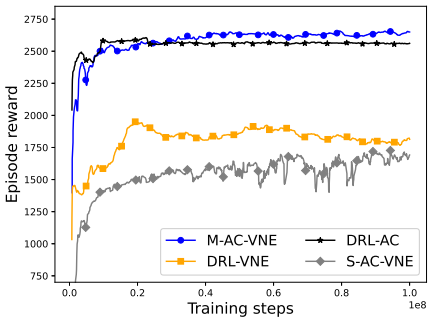


Fig. 5: Episode rewards for DRL-based approaches.

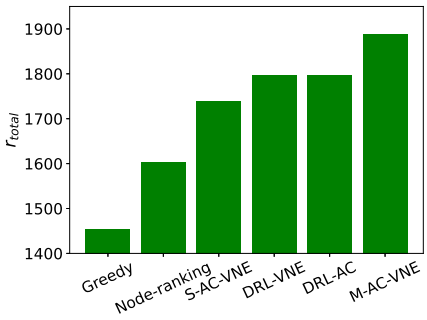


Fig. 6: InP total revenue.

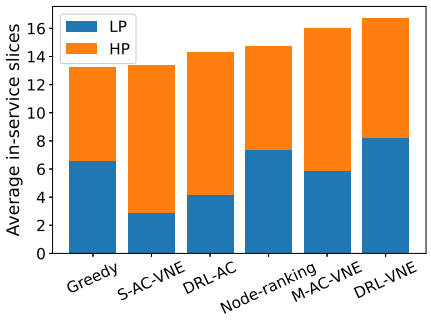


Fig. 7: Average in-service SRs.

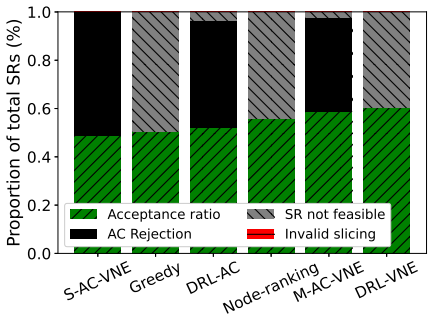
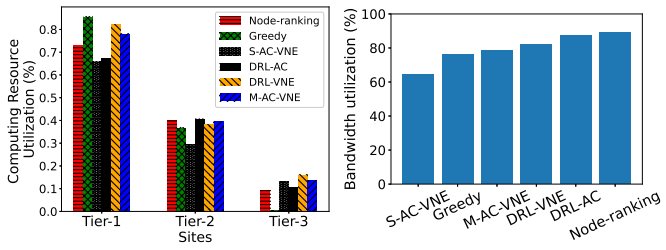


Fig. 8: AC and slicing statistics.



(a) Compute resources of all tiers.

(b) Tier-1 to Tier-2 link bandwidth.

Fig. 9: Average Resource utilization.

We set this approach as the baseline to compare the rest of the approaches against. The node-ranking approach achieves only 10.3% higher revenue as compared to the greedy approach. It is followed by S-AC-VNE, DRL-AC, DRL-VNE and M-AC-VNE approaches, which achieve 19.64%, 23.56%, 23.6%, and 29.96% gains in revenue over the greedy approach, respectively.

To further shed light on the revenue achieved by different approaches, the average number of in-service SRs during an episode of evaluation, broken down by their priority, is shown in Fig. 7. Due to a limited amount of resources available in the substrate network, there can only be a limited number of in-service SRs at any given time. The number of in-service SRs represents how much resources are being utilized by different approaches. The greedy approach has the lowest number of in-service SRs due to the computing bottleneck at lower-tier sites described in Section VI-C1. The node-ranking approach circumvents this bottleneck, which results in an increase in the average number of in-service SRs over the duration of an episode, though with this approach, the link bandwidth becomes a bottleneck. We can see that the DRL-VNE approach achieves the highest average number of in-service SRs by optimally balancing the compute and bandwidth resource utilization. Fig. 7 also shows that approaches with an AC mechanism have a higher proportion of HP in-service SRs because they can preemptively reject LP SRs to keep the resources available for HP SRs. However, for the other approaches, the proportion of HP and LP SRs is equal to the probability of a request being HP or LP, respectively.

Based on Fig. 6 and Fig. 7, we can conclude that the total InP revenue achieved depends on both the total in-service SRs and the percentage of HP SRs. Optimizing one, but not the other, leads to a loss in potential revenue for the InP. Even though DRL-VNE and S-AC-VNE led to the highest average number of in-service SRs and the highest proportion of HP SRs, respectively, they do not achieve the highest revenue. For S-AC-VNE, the average number of in-service SRs and the achieved revenue is lower than all DRL-based approaches, even the ones using DRL for either AC or slicing. This indicates that although the shaped-reward designed for S-AC-VNE leads to convergence within a reasonable time, it is biased toward optimal AC and non-optimal slicing. Finally, M-

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3222589

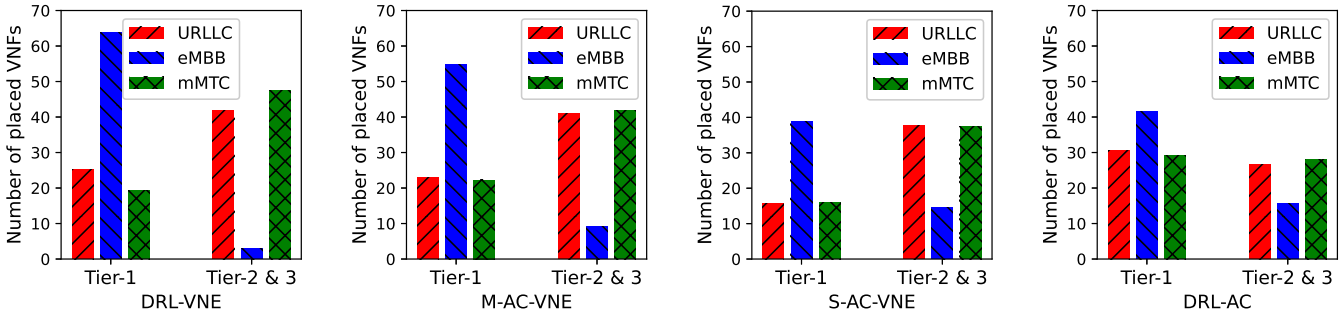


Fig. 10: VNF placement at different sites.

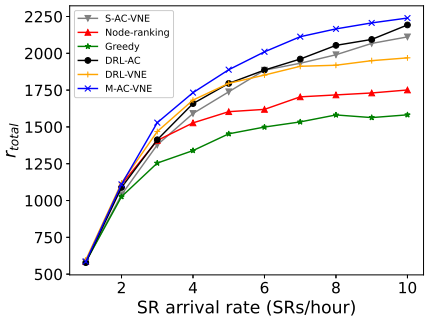


Fig. 11: InP revenue vs. SR arrival rate, training arrival rate: 5 SRs/hour.

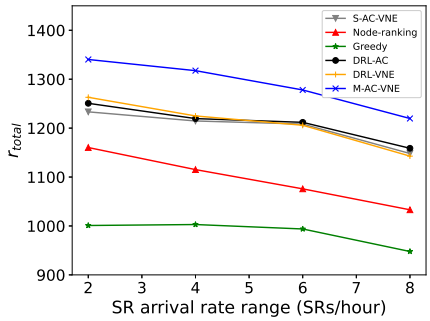


Fig. 12: InP revenue vs. arrival rate range centered around 5 SRs/hour.

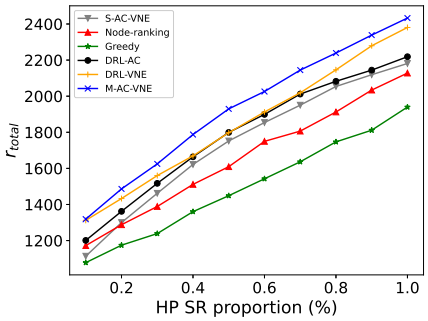


Fig. 13: InP revenue vs. HP SR proportion, training HR SR proportion: 0.5.

AC-VNE achieves the highest average revenue by optimally balancing between these two factors.

2) AC and slicing statistics

Another metric of interest for performance analysis of a slicing solution in the literature is the acceptance ratio, which is defined as the proportion of the total SRs that are embedded in the substrate network. In slicing-only approaches (*i.e.*, when there is no AC policy), an SR might be rejected only if there is not enough resources available (*i.e.*, SR not feasible), but in the joint AC and slicing problem, an SR is rejected either by the AC action, when there are no resources available, or when the slicing action violates constraints (*i.e.*, invalid slicing). Fig. 8 shows, for each approach, the acceptance ratio and the reason for rejecting the remaining SRs. Comparing the acceptance ratio with the average number of in-service SRs (*cf.*, Fig. 7), we can observe that the relative values differ slightly because DRL-based approaches take SRs' operation times into account which can result in a higher or lower acceptance ratio. However, the insights from Fig. 7 can generally be translated to acceptance ratio, *e.g.*, DRL-VNE has the highest acceptance ratio followed closely by M-AC-VNE. Approaches equipped with an AC policy have a higher proportion of requests rejected due to AC action rather than infeasible SR. In addition, the proportion of SRs rejected due to an invalid slicing (*i.e.*, constraint violations) is also negligible for all these approaches. Together these two observations prove that the shaped reward design of these approaches works as intended, *i.e.*, it makes the RL agent reject low-paying SRs using the AC action rather than constraint violations.

3) Resource utilization

Figures 9a and 9b show the average compute resource utilization at different tiers and the average Tier-1 to Tier-2 link bandwidth utilization for different approaches during evaluation, respectively. Node-ranking approach has the highest bandwidth utilization since all VNFs are placed at the farthest node, consuming a large amount of bandwidth. Similar bandwidth utilization can be observed for the DRL-AC approach as it uses node-ranking for slicing. Both M-AC-VNE and DRL-VNE approaches have lower average bandwidth utilization, demonstrating that DRL-based slicing conserves more bandwidth by placing fewer bandwidth-hungry SRs on the Tier-1 to Tier-2 link. This is further corroborated by Fig. 9a, which indicates that these approaches achieve the highest resource utilization in Tier-3, while still maintaining high resource utilization at Tier-1 and Tier-2.

4) VNF placement

Fig. 10 shows the average number of VNFs placed at different sites, by the DRL-based approaches, broken down by the type of service. We can observe that DRL-VNE achieves high resource utilization by placing the highest number of eMBB VNFs on the Tier-1 site, and the highest number of mMTC and URLLC VNFs on higher tiered sites. In general, as the number of eMBB VNFs on higher tiered sites increases, the corresponding cumulative number of VNFs on these sites decreases due to the bottleneck created by eMBB VNFs on the Tier-1 to Tier-2 link. Finally, the VNF placement by M-AC-

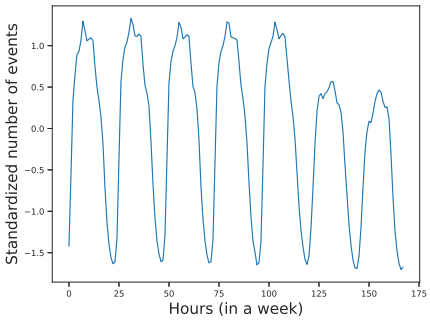


Fig. 14: Weekly traffic pattern of Internet events in Milan [49].

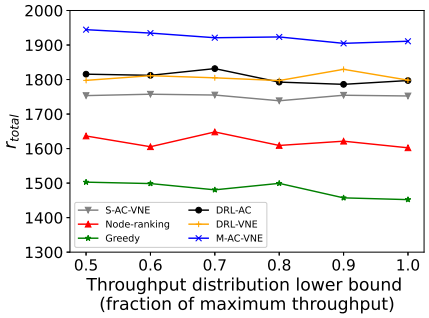


Fig. 15: InP revenue vs. throughput range.

VNE is quite close to that of DRL-VNE proving that the AC and slicing agents are able to cooperate, and the addition of a DRL-agent for AC does not adversely affect the performance of the slicing agent.

5) Robustness

To test the robustness of the trained model under diversified conditions, we vary the arrival rate, priority, and throughput of SRs. First, we introduce previously unseen load conditions to the agents, by deviating the SR arrival rate from the one used during training (*i.e.*, 5 SRs/hour). Fig. 11 displays the robustness of different approaches against varying 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 heuristic-based approaches also increases. At the highest point, the total revenue difference between M-AC-VNE, and the node-ranking and greedy approaches reaches 27.90% and 45.13%, 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 are fewer chances of creating bottlenecks.

We also evaluate the generalization of the DRL-based approaches under real-world traffic pattern when the agents are only trained on the mean SR arrival rate. For this purpose, we utilize the Telecom Italia dataset [49] to generate dynamic SR arrival rates. Fig. 14 shows the standardized weekly traffic pattern of Internet events in Milan. We use min-max scaling to scale the traffic to different ranges centered around the mean arrival rate of 5 SRs/hour used for training. Fig. 12 shows the performance of different approaches as the range around the mean varies. The DRL-based approaches are still superior to heuristic-based approaches in terms of performance, and M-AC-VNE continues to generate the highest revenues. However, the revenue decreases as the range increases. This is due to the fact that revenue decrease is steeper for lower than mean arrival rates as compared to the revenue increase for higher than mean arrival rates (*cf.*, Fig. 11).

Additionally, we evaluate the robustness under varying percentages of HP SRs. Fig. 13 shows the revenue achieved by different approaches in this scenario. It is evident that the M-AC-VNE is able to generalize to this scenario as well and maintain its lead in the achieved long-term InP revenue. When the proportion of HP SRs increases or decreases from 0.5, AC becomes less effective and the performance difference between M-AC-VNE and DRL-VNE decreases. This shows that although DRL-AC has better training performance, M-AC-VNE is needed to retain good performance under various conditions. In cases with no HP SRs, or with only HP SRs, the M-AC-VNE performs similar to DRL-VNE, but is still able to achieve higher revenue when compared to other baseline approaches as it is able to avoid resource bottlenecks. Additionally, DRL-based AC 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 InP revenue. However, as the HP SRs become oversaturated, the rate of increase decreases.

Finally, we test the different approaches in the scenario where the throughput requested by the SRs of the same slice type can vary. For this purpose, we consider the slice types' throughput values used during training to be the maximum throughput for SRs of those slice types. Whereas, during evaluation, the throughput demand by an SR is uniformly selected from a range below that maximum value. The lower bound of this uniform distribution is a certain fraction of the maximum throughput. Fig. 15 shows the InP revenue achieved as the lower bound of the uniformly distributed throughput demand increases from half to maximum throughput (*i.e.*, constant throughput demand). In general, we can see that as the lower bound decreases, the InP revenue increases. This is expected since as throughput decreases, compute and bandwidth resource demand decreases as well and the InP is able to accommodate more SRs. Moreover, we can observe that the proposed approach retains its performance delta across different ranges of throughput values.

VII. Conclusion

In this paper, we addressed the problem of joint slicing and AC in 5G C-RAN. Considering diverse slice constraints, future traffic patterns, and limited resource availability in the substrate network, while maximizing an InP's long-term revenue makes the problem complex. This complexity calls for close collaboration between the slicing and AC modules. We propose a MADRL-based approach consisting of two agents with shaped rewards that can effectively address slicing and AC problems jointly. We show that the designed rewards lead to convergence and cooperation between agents, and DRL-

based approaches that address only one aspect of the problem lead to a loss in potential InP revenue. The proposed approach achieves as much as 29.96% higher revenue when compared to approaches based on heuristics and approaches that address only one problem optimally (*i.e.*, using DRL). Our results also show that multi-agent DRL achieves 8.62% higher long-term InP revenue and leads to faster convergence, when compared to a single-agent DRL approach that jointly addresses the slicing and AC problems. Additionally, our results indicate that the proposed approach is able to generalize to different arrival rates and proportions of HP SRs, as well as dynamic and real traffic patterns that differ from those used during training.

Our proposed MADRL-based approach can be improved in terms of generalization in several ways. We have not thoroughly evaluated the scalability of the proposed approach over large and complex topologies. Furthermore, the leveraged models for the slicing and AC agents require re-training with the slightest topological variations, *e.g.*, addition or removal of a node or link. In the future, we plan to investigate the use of Graph Neural Networks to accommodate complex and dynamic metro 5G RAN topologies. Our model definition also does not directly support SRs for slice types with a specification different from the ones used during training. A viable solution to this problem is to feed the model directly with the delay and throughput requirements. Finally, we will investigate dynamically scaling the resources allocated to SRs based on their predicted demand. This can allow the network to accommodate more SRs by leveraging over-booking.

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.

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, Feb. 2018, version 1.0.

[3] M. Shehata, A. Elbanna *et al.*, "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 *et al.*, "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 *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] R. Boutaba, N. Shahriar *et al.*, *Managing Virtualized Networks and Services with Machine Learning*. Wiley Online Library, 2021.

[7] N. Van Huynh, D. Thai Hoang *et al.*, "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.

[8] S. Matoussi, I. Fajjari *et al.*, "Deep Learning based User Slice Allocation in 5G Radio Access Networks," in *IEEE Conference on Local Computer Networks (LCN)*, 2020, pp. 286–296.

[9] M. Sulaiman, A. Moayyedi *et al.*, "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.

[10] H. Yu, F. Musumeci *et al.*, "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 *et al.*, "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] R. Boutaba, M. A. Salahuddin *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Springer Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.

[14] 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.

[15] D. Bega, M. Gramaglia *et al.*, "Optimising 5G infrastructure markets: The business of network slicing," in *IEEE Conference on Computer Communications (INFOCOM)*, 2017, pp. 1–9.

[16] ——, "A machine learning approach to 5G infrastructure market optimization," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 498–512, 2019.

[17] V. Sciancalepore, L. Zanzi *et al.*, "ONETS: Online network slice broker from theory to practice," *IEEE Transactions on Wireless Communications*, vol. 21, no. 1, pp. 121–134, 2022.

[18] M. R. Raza, C. Natalino *et al.*, "Reinforcement learning for slicing in a 5G Flexible RAN," *Journal of Lightwave Technology*, vol. 37, no. 20, pp. 5161–5169, Oct. 2019.

[19] J. Koo, V. B. Mendiratta *et al.*, "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics," in *IEEE Conference on Network and Service Management (CNSM)*, 2019, pp. 1–5.

[20] Z. Gao, S. Yan *et al.*, "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.

[21] R. Solozabal, J. Ceberio *et al.*, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, 2019.

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

[23] F. W. Murti, J. A. Ayala-Romero *et al.*, "An optimal deployment framework for multi-cloud virtualized radio access networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2251–2265, 2021.

[24] A. Garcia-Saavedra, X. Costa-Perez *et al.*, "FluidRAN: Optimized vRAN/MEC orchestration," in *IEEE Conference on Computer Communications*, 2018, pp. 2366–2374.

[25] 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.

[26] S. Matoussi, I. Fajjari *et al.*, "5G RAN: Functional Split Orchestration Optimization," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1448–1463, 2020.

[27] R. Boutaba, N. Shahriar *et al.*, "AI-driven closed-loop automa-

tion in 5G and beyond mobile networks," in *ACM Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility (FlexNets)*, 2021, p. 1–6.
[28] S. Ayoubi, N. Limam *et al.*, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018.
[29] ORAN Alliance, "O-RAN: Towards an open and smart RAN," 2018. [Online]. Available: https://www.o-ran.org/resources
[30] 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.
[31] G.sup.5GP, "5G wireless fronthaul requirements in a PON context," ITU-T, G.Sup66 Supplement, Jul. 2019.
[32] C. Desset, B. Debaillie *et al.*, "Flexible power modeling of LTE base stations," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2012, pp. 2858–2862.
[33] D. Szczesny, A. Showk *et al.*, "Performance analysis of LTE protocol processing on an ARM based mobile platform," in *International Symposium on System-on-Chip*, 2009, pp. 056–063.
[34] Small Cell Forum, "Small cell virtualization functional splits and use cases Rel. 7.0," 2016.
[35] 3GPP, "Transport requirement for CU&DU functional splits options," 3rd Generation Partnership Project (3GPP), discussion R3-162005, Aug. 2016.
[36] A. Maeder, M. Lalam *et al.*, "Towards a flexible functional split for cloud-RAN networks," in *European Conf. on Networks and Comm. (EuCNC)*, 2014, pp. 1–5.
[37] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html
[38] J. Schulman, F. Wolski *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[39] J. Schulman, S. Levine *et al.*, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.
[40] J. Schulman, P. Moritz *et al.*, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.
[41] M. Andrychowicz, F. Wolski *et al.*, "Hindsight experience replay," in *Advances in neural information processing systems*, vol. 30, 2017.
[42] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *International Conference on Machine Learning (ICML)*, 1999, p. 278–287.
[43] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.
[44] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
[45] G. Brockman, V. Cheung *et al.*, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.
[46] E. Liang, R. Liaw *et al.*, "RLlib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2018, pp. 3053–3062.
[47] R. Liaw, E. Liang *et al.*, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.
[48] A. Paszke, S. Gross *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.
[49] Telecom Italia, "Telecommunications - SMS, Call, Internet - MI," 2015. [Online]. Available: https://doi.org/10.7910/DVN/EGZHFV

Muhammad Sulaiman (Student Member, IEEE) received BS in Electrical Engineering from the National University of Sciences and Technology, Pakistan, in 2019. He joined the University of Waterloo for a Master of Mathematics (MMath) in Computer Science in 2020 and transferred to the Ph.D. program soon after. His research interests include reinforcement learning, optimization of computer networks, and autonomous management and orchestration of mobile networks.

Arash Moayyedi received his BS degree in computer engineering from the Sharif University of Technology, Iran, in 2020. During his bachelor's degree, he held research positions at the University of Vienna, Austria, and KTH Royal Institue of Technology, Sweden. He is currently a master's student in computer science at the University of Waterloo, under the supervision of Professor Raouf Boutaba, conducting research on AI-based orchestration in 5G mobile networks.

Mahdieh Ahmadi received the BS degree in computer engineering from the University of Tehran, Iran, in 2013, and the M.Sc. and Ph.D. degrees in computer engineering from Sharif University of Technology, Iran, in 2015 and 2020, respectively. She is currently a Postdoctoral Researcher at the David R. Cheriton School of Computer Science, University of Waterloo, Canada. Her main research areas include performance evaluation and optimization of computer networks.

Mohammad A. Salahuddin (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from Western Michigan University in 2003 and 2014, respectively. He is currently a Research Assistant Professor of Computer Science with the University of Waterloo. His research interests include the Internet of Things, content delivery networks, network softwarization, network security, and cognitive network management. His co-authored publications have received numerous awards, including ACM SIGMETRICS best student paper, IEEE/IFIP NOMS best paper, and IEEE CNOM best paper. He serves on the TPC for international conferences and is a reviewer for various peer-reviewed journals, magazines and conferences.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3222589

Raouf Boutaba (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in computer science from Sorbonne University in 1990 and 1994, respectively. He is currently a University Chair Professor and the Director of the David R. Cheriton School of Computer science at the University of Waterloo (Canada). He is the founding Editor-in-Chief of the IEEE Transactions on Network and Service Management (2007-2010) and served as the Editor-in-Chief of the IEEE Journal on Selected Areas in Communications (2018-2021). He is a fellow of the IEEE, the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada.

Aladdin Saleh is currently priming research and innovation activities at Rogers communications, among them the joint research partnership with the University of Waterloo on 5G and emerging technologies. He has over 20 years of industry experience in mobile telecom in Canada. He earned a Ph.D. degree in electrical and electronic engineering and an MBA in International Management, both from the University of London in the UK. He taught and conducted research on next-generation wireless networks at several universities as a full-time professor, Adjunct Professor, and a visiting researcher. He is currently an Adjunct Professor with the Cheriton School of Computer Science at the University of Waterloo.