AD-VRAN: DRL-based Adaptive Deployment of Virtualized RAN in an Open Telco Edge Cloud

Yuan-Yao Lou, Cheng Chen, Ying-Hui Huang, Mung Chiang, *Fellow, IEEE*, Kwang Taik Kim, *Senior Member, IEEE*

***Abstract*—As wireless networks evolve to support diverse applications such as augmented reality (AR), virtual reality (VR), distributed AI, and autonomous driving, traditional RAN architectures like Distributed RAN (D-RAN) and Centralized RAN (C-RAN) struggle to meet increasingly stringent and heterogeneous Quality of Service (QoS) demands. D-RAN offers localized processing for latency-sensitive tasks, while C-RAN centralizes resources for efficiency, but both lack the flexibility to dynamically adapt to varying traffic and application requirements. Emerging 6G networks are shifting toward open, virtualized RAN (vRAN) architectures, enabling real-time resource allocation and network reconfiguration to accommodate fluctuating demands. However, this virtualization introduces complex trade-offs across computing, networking, and application performance metrics, requiring intelligent orchestration. In this work, we propose a deep reinforcement learning (DRL)-based orchestration policy implemented as a RAN intelligent controller (RIC) xApp, designed to optimize vRAN deployment and configuration in multi-cell scenarios. We formulate the joint optimization problem as a Markov Decision Process (MDP) and apply a multi-agent RL (MARL) framework to find the optimal policy. Simulation results demonstrate that our approach achieves the highest composite QoS score compared to traditional architectures, heuristic, and other RL-based methods by adaptively updating the vRAN deployment configurations, reducing resource usage while maintaining equivalent application performance.**

***Index Terms*—Network Function Virtualization, Radio Access Network, Telco Edge Cloud, Radio Intelligent Controller, Multi-agent Reinforcement Learning, Proximal Policy Optimization**

I. Introduction

The cellular networking landscape is undergoing a profound transformation with the advent of Beyond 5G (B5G) and 6G eras. This transformation is driven by a myriad of applications with diverse Quality of Service (QoS) needs, ranging from real-time interactions to high-throughput data processing such as extended reality (XR), autonomous driving, and large language models (LLMs) applications. The traditional RAN architecture faces challenges in accommodating the dynamic nature of these emerging applications. Specifically, D-RAN architecture provides localized processing for low-latency applications but suffers from high deployment costs and limited computing resources. On the other hand, while C-RAN benefits from a centralized computing pool to support computing-intensive tasks, it results in a high throughput requirement over the cross-haul between baseband unit (BBU) and remote radio head (RRH). Traditional cellular network optimization techniques, which often rely on static rules and a fixed architecture, lack adaptability to maintain optimal performance as cellular networks grow more complex.

To address these difficulties, one notable development in the B5G/6G technologies is open and virtualized RAN (vRAN) architectures utilizing shared physical infrastructure in a Telco Edge Cloud (TEC) [1], [2]. 5G core networks (5GC) have already been transitioned to software-based architectures (SBA). Concurrently, RAN functions are also moving away from dedicated hardware and toward virtualized network functions (VNFs) running on Commercial Off-The-Shelf (COTS) servers, enabling RAN functional split (FS) to decompose RAN into virtualized central units (CUs), distributed units (DUs), and radio units (RUs) [1], [3], [4]. Combining RAN FS options with physical functional placement, vRAN architectures can be customized to feature different pros and cons and deployed for various emerging applications.

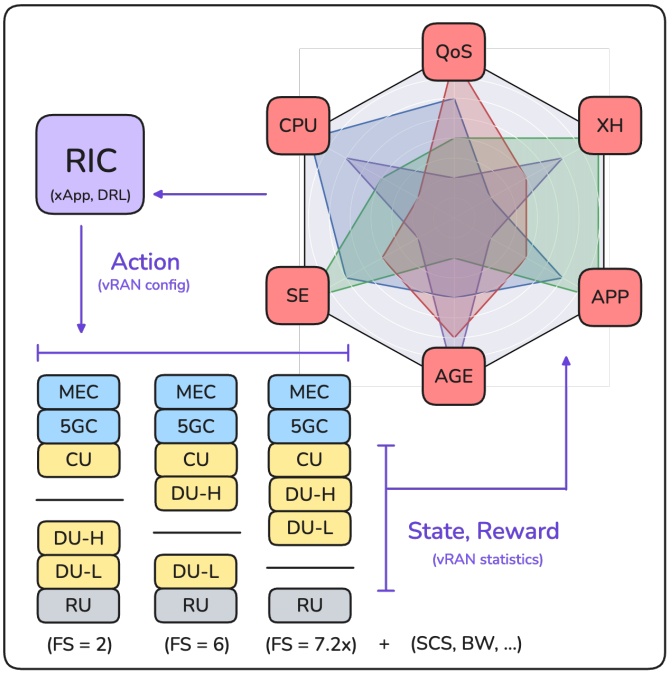

Fig. 1. Overview of vRAN orchestration by RIC in a TEC.

Although the new architecture of cellular networks can be re-programmed to satisfy diverse QoS requirements of service-level agreements (SLAs), optimizing performance necessitates advanced methodologies. Specifically, managing multiple vRAN entities (e.g., CU, DU, and RU) involves navigating multi-dimensional trade-offs between network and application performance. To tackle this challenge, the RIC's micro-service-based applications (e.g., xApps) provide an opportunity by monitoring the network and collecting data metrics from VNFs (e.g., RLC, MAC, PHY) for AI/ML model training and online inference, optimizing the overall performance. Our

research focuses on near real-time RIC and xApps with three key areas of vRAN orchestration:

1) **Resource Utilization of vRAN:** Given that vRAN is open and can operate on COTS servers, characterizing and assessing the computation and network resource consumption by a vRAN is essential for load-balancing among edge cloud servers. The performance of a vRAN is thus constrained by the allocated virtualized resources.
2) **Application Performance and QoS Requirements:** To support applications with diverse QoS requirements, the trade-offs between different vRAN architectures must consider the impact on limited vRAN resources and the application performance. Specifically, although the distributed vRAN architecture eases the traffic load of XH, it consumes more computing resources and thus fewer UEs can be scheduled for data transmission, leading to lower application performance.
3) **AI/ML-based vRAN Orchestration:** Depending on the RAN FS options, the vRAN architecture can be dynamically deployed and adjusted based on RIC policies. Additionally, AI/ML approaches can enhance RIC's ability to orchestrate vRAN, which can further optimize network performance and adapt to dynamic and unpredictable conditions such as varying wireless channel quality, user mobility patterns, and server loads.

In this work, we formulate the joint optimization problem as a Markov Decision Process (MDP) that integrates the multi-dimensional trade-offs of vRAN and application performance. Specifically, we model the performance trade-offs among RAN FS options and configurations in terms of virtualized resource utilization, including CPU and XH link usage, cell throughput, application goodput, and penalty score based on 5QI (5G QoS Identifier) packet delay budget (PDB). Our objective is to develop an optimal learning-based orchestration policy for computational sequential decision-making, enabling adaptive adjustment of the vRAN architecture and configuration parameters to maximize application performance while minimizing the consumption of virtualized resources. Navigating among the complicated state space of the vRAN system and finding optimal policies is challenging, especially in cellular systems where performance metrics mentioned above are interleaved. While prior studies propose heuristic and deep Q-network (DQN)-based RL agents, analyzing the multi-dimensional trade-offs in the vRAN context needs more sophisticated performance modeling and solutions. Our contributions can be summarized as follows:

1) We propose a Deep Reinforcement Learning (DRL)-based xApp to adaptively control vRAN configurations and deployments, specifically addressing numerology and FS options. The multi-agent RL (MARL) training framework for multi-cell scenarios (i.e., one RL agent for each) shows a stable learning progress compared to other RL-based methods, such as DQN and TRPO. (Sec. IV and Sec. V)
2) To quantitatively evaluate the DRL agent's performance relative to a fixed RAN architecture, heuristic, and other RL-based control policies, we develop a 3GPP-compliant system-level simulator (SLS) integrated with a vRAN resource model and an RL framework. (Sec. VI-A)
3) The extensive numerical evaluation indicates that our DRL-based agent strikes a great balance among the five KPI metrics, outperforming traditional fixed architectures, heuristic policies, and other RL-based methods in terms of a composite QoS score. (Sec. VI)

II. System Framework

In this section, we introduce the system framework of a TEC. We begin by explaining the architecture and functionalities of 5GC and vRAN, including the protocol stacks of 3GPP 5G new radio (NR). Then, we detail the RIC framework with the proposed DRL agent as an xApp for RAN orchestration. The high-level overview of TEC is depicted in Fig. 2.

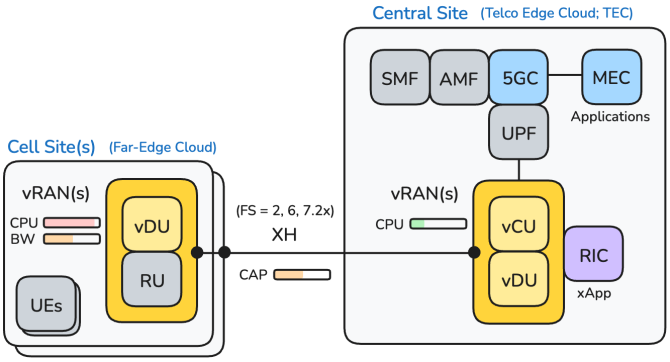

Fig. 2. Overview of TEC system framework.

Deploying an open and virtualized RAN in an edge cloud typically involves two physical locations: a central site and cell site(s). The central site operates a 5GC and may connected with a MEC platform for hosting applications. Furthermore, the open RAN components (RIC, CU, and DU) reside in a containerized environment managed by the edge cloud. Regarding the cell sites, depending on the RAN FS option and the deployment architecture, CU, DU, and RU can be deployed at the cell sites, forming a far-edge cloud. Note that the central site and the cell sites are connected through an integrated backhaul (BH) and fronthaul (FH) interface called crosshaul (XH). In this work, we simulate the end-to-end (E2E) data flow originating from the MEC applications and being routed via 5GC UPF into vRAN entities (CU, DU) and finally, through XH, arriving at the DU and RU of the cell site. In Table I, we present a summary of the variables and notations used within the TEC system framework.

A. *Central Site / Edge Cloud (5GC, RIC, CU, DU, XH)*

We consider that a RIC framework manages containerized vRAN instances for multiple cells N_t in a TEC at time t. The serving cell set for mobile devices consists of only one primary cell in the SLS we developed for this work. Thus, the carrier aggregation is not enabled. For each cell $n \in N_t$, there is a dedicated vRAN with 5G NR protocol stacks to handle control-plane and user-plane traffic. RIC hosts a xApp controller $c \in C_{n,t}$ for each cell $n, |C_{n,t}| = 1, |N_t| = \Sigma_n |C_{n,t}|$. The user equipment (UE) is defined as $k \in K_t$. We detail

the roles and functions of each component in the following paragraphs.

1) **5GC and MEC:** Defined by 3GPP, the network functions (NFs) of 5GC, such as the Session Management Function (SMF), Access and Mobility Management Function (AMF), and User Plane Function (UPF), are running on COTS hardware with flexible APIs for reconfigurations. We consider that the traffic of MEC applications in the TEC is routed to vRAN through a pre-configured UPF. The UPF encapsulates and decapsulates IP packets using GPRS Tunnelling Protocol (GTP) and carries user data between the vRAN and the 5GC.
2) **RIC and vRAN:** In addition to 5GC and MEC, RIC is a key enabler in applying AI/ML for B5G/6G networks. The E2 interface connects RIC with CU and DU to carry telemetry, control, and policy information. Specifically, xApps can analyze real-time RAN statistics and control vRAN components for optimization by following an ML-based policy. In this work, we focus on the control policy regarding the RAN FS option σ and RAN configurations such as numerology μ or subcarrier spacing (SCS). We consider three types of vRAN architecture based on the RAN FS option 2 (PDCP-RLC split), 6 (MAC-PHY split), and 7.2x (intra-PHY split) [5]. Correspondingly, CU and RU always reside in the central site and the cell site, respectively, while DU can be decomposed into DU-high (RLC, MAC) and DU-low (high PHY) further and located at both sites. We refer to the prior study [6] to assume vDU replicas in each location to enable a faster and lighter live VM migration, and thus our RIC xApp is co-located with vDU. As we assume all components are virtualized and consume computing resources (R_{CPU}), we refer to the prior works for modeling the CPU utilization of the PHY layer and other RAN VNFs located at the cell site [7], [8]. We define the formula of vRAN CPU usage and discuss the related impact factors in Sec. V.
3) **XH (Crosshaul):** The connection between CU, DU, and RU has evolved into an integrated backhaul (BH) and fronthaul (FH) solution called XH. Depending on the RAN FS option σ and numerology μ, the requirement of XH link capacity (R_{XH}) varies. In this work, we measure the bit rate of the XH link based on the existing analysis of three considered RAN FS options (i.e., 2, 6, 7.2x) [9], [10]. The mathematical formulas for XH link usage are defined and explained later in Sec. V.

TABLE I
NOTATIONS FOR TEC SYSTEM FRAMEWORK

Variable	Notation
Time (ms)	t
Cells / vRANs	N_t
vRAN Controllers (xApps)	$C_{n,t}$
UE	k
Total UEs	K_t
Cell UEs (connected)	$K_{n,t}$
Cell UEs (scheduled)	$K^{\text{sched}}_{n,t} \in K_{n,t}$
# of scheduled UEs	$\lvert K^{\text{sched}}_{n,t}\rvert$
Numerology (SCS)	μ
Functional Split (FS) Option	σ
TDD Pattern	TDD
Scheduling Algorithm	ALGO = Proportional Fair (PF)
Resource Block Group	RBG
Modulation Order	M^{mod}
Selected MCS Index	M^{mcs}
Cell Bandwidth	N^{bw}
# of Radio Slots (per ms)	$R^{\text{slot}} = 2^{\mu}$
# of Resource Blocks	$R^{\text{dl\_rb}}$
Transport Block	TB
Transport Block Size (TBS)	S^{tbs}
# of Transmitted TBs	$R^{\text{dl\_tb}}$
Estimated Block Error Rate	$\hat{P}^{\text{bler}}$
# of Allocated Resource Blocks	R^{rb}
Packet Size	S^{pkt}
Downlink Reception (Rx) Bits	S^{dl}
Allocated CPU Resources	R_{CPU}
Allocated XH Link Capacity	R_{XH}

Function	Notation
DU Manager: UE Measurement	$f^{\text{UE}}(K_{n,t})$
DU Manager: Resource Allocation	$f^{\text{RES}}(\mu, N^{\text{bw}}, R^{\text{slot}}, \text{TDD}, t)$
MAC: Scheduling, HARQ	$f^{\text{MAC}}(\text{ALGO}, R^{\text{dl\_tb}}, K_{n,t}, \lvert K^{\text{sched}}_{n,t}\rvert)$
PHY: Transmission (Tx)/Rx	$f^{\text{PHY}}(\text{TB})$

B. *Cell Site / Far-Edge Cloud (DU, RU, UE)*

1) **DU and RU:** We refer to O-RAN reference architecture to decompose DU into DU-high and DU-low. Moreover, for MAC scheduling and data transmission, we design a DU manager in charge of RAN resource allocation (f^{RES}) and UE measurement (f^{UE}) by referencing to the srsRAN software architecture [11]:

$$\begin{cases} f^{\text{RES}}(\mu, N^{\text{bw}}, R^{\text{slot}}, \text{TDD}, t) = R^{\text{dl\_rb}}_{n,t} \\ f^{\text{UE}}(K_{n,t}) = \{(M^{\text{mod}}, M^{\text{mcs}})_{n,t,k}\} \end{cases},$$

where $R^{\text{dl\_rb}}_{n,t}$ is the total resource blocks (RBs) of vRAN n at time t based on the numerology μ, bandwidth N^{bw}, and the TDD pattern. Regarding the output of f^{UE}, the modulation order M^{mod} and the index of modulation coding scheme M^{mcs} (MCS) are derived from the channel model (e.g., RSP, SINR, etc.) of Nokia's 5G SLS named FikoRE [12]. Then we follow 3GPP specifications 38.214 (e.g., the clause 5.1.2.2.1, 5.1.3.1, and 5.1.3.2) to calculate the resource block group (RBG), transport block size S^{tbs} and the block error rate $\hat{P}^{\text{bler}}$. Since the TBS can be larger or smaller than an application packet, we define a transport block (TB) with index i scheduled for UE k in vRAN n at time t as a tuple:

$$\text{TB}_{n,t,k,i} = (\{S^{\text{pkt}}\}_{k,i,j}, S^{\text{tbs}}, \hat{P}^{\text{bler}}, \text{RBG})_{n,t,k,i},$$

where $0 \leq i \leq R^{\text{dl\_tb}} - 1$. The function of the MAC layer f^{MAC} takes a scheduling algorithm (ALGO), the number of RBs $R^{\text{dl\_rb}}_{n,t}$, connected users $K_{n,t}$, and the maximum number of scheduled users $\lvert K^{\text{sched}}_{n,t}\rvert$ as input. The formula for deriving $\lvert K^{\text{sched}}_{n,t}\rvert$ of a vRAN is defined later in Sec. V. The output of f^{MAC} is a series of TBs[1] that will be fed into the PHY abstraction function (f^{PHY}) for data transmission and HARQ retransmission by following Bernoulli distribution with a probability of success given

[1]Depending on the resource allocation and scheduling scheme, there can be one or several TBs transmitted in each TTI for single or multiple UEs.

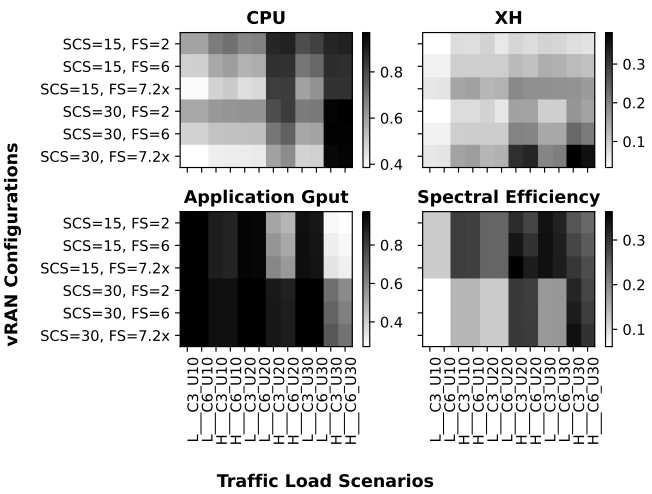

Fig. 3. Heatmaps of RAN KPI metrics: CPU utilization rate, XH link usage, application performance, and spectral efficiency. The values in heatmaps are the average of all cells over 30,000 simulation time steps (i.e., 30 seconds) for 100 random seeds. For the naming of twelve traffic scenarios, H is heavy traffic load, and L is lightweight traffic load. C3 indicates that the number of cells is three, and U10 means there are 10 UEs distributed in each cell.

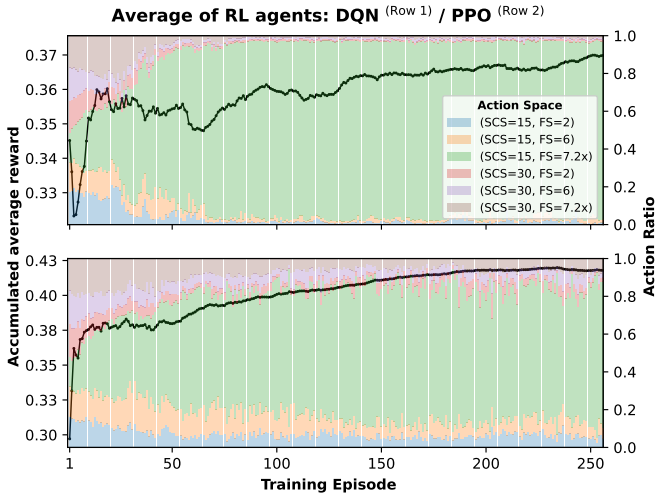

Fig. 4. Training behavior of DQN- and PPO-based RL agents.

by $\hat{P}^{\text{bler}}$. The functions of MAC (f^{MAC}) and PHY (f^{PHY}) layers can then be defined as:

$$\begin{cases} f^{\text{MAC}}(\text{ALGO}, R^{\text{dl\_rb}}_{n,t}, K_{n,t}, |K^{\text{sched}}_{n,t}|) = \{\text{TB}_{n,t,k,i}\}, \\ f^{\text{PHY}}(\text{TB}_{n,t,k,i}) = \begin{cases} \sum_j S^{\text{pkt}}_{t,k,j}, \text{if } p \geq \hat{P}^{\text{bler}}, p \sim U(0,1), \\ 0, \text{otherwise.} \end{cases} \end{cases}$$

2) **UE (User Equipment):** We follow the ITU report (IMT-2020) proposed by 3GPP and other standard organizations to model the UE behavior, including the mobility pattern and channel dynamics [13], [14]. In this work, at time t, a UE can only connect to one cell, $K_{n,t} \cap K_{n',t} = \emptyset, \forall n \neq n'$. Also, each UE is associated with one MEC application whose traffic pattern follows the standardized 3GPP and NGMN models [15].

III. Motivation

As outlined in Sec. II, RAN FS options directly influence vRAN resource utilization and system dynamics, creating intricate performance trade-offs. To analyze these relationships, we conduct preliminary experiments by monitoring the vRAN operation (e.g., f^{MAC}, f^{PHY}, and f^{UE}) across traditional RAN architectures. The evaluation quantifies resource consumption (e.g., CPU utilization, XH link usage) and performance metrics (e.g., spectral efficiency, application goodput) in Fig. 3. We first observe the general pros and cons between traditional RAN architectures that D-RAN (i.e., FS=2) eases the XH traffic load (5–20% link capacity usage) but requires substantial computing resources (60–95% CPU utilization). In contrast, C-RAN (i.e., FS=7.2x) centralizes computing resources, reducing CPU utilization to 40-60%, but requires a high XH throughput of 25–35% link capacity due to intra-PHY communication. We further identify the interleaved trade-offs between vRAN configurations and different traffic scenarios, proving that dynamic vRAN deployment is critical to balance performance and resource efficiency under varying traffic demands. Specifically, by fixing the FS option in heavy traffic load scenarios (e.g., H___C3_U30, H___C6_U30), larger SCS with higher bandwidth improves application goodput and spectral efficiency, increasing both by around 10%. However, it increases CPU and XH usage by 10-20%, reducing the efficiency of resource consumption. The multi-dimensional performance and independent KPI metrics necessitate an intelligent and adaptive vRAN deployment policy. Heuristic and greedy methods, which focus on simplistic optimization goals without awareness of the impact of actions in both the near and long-term, can lead to suboptimal policies by failing to account for these complex interdependencies.

To achieve adaptive vRAN deployment, we explore learning for computational sequential decision-making, such as RL algorithms, in a MARL framework for multi-cell scenarios. We train RL agents in a six-cell scenario (i.e., six RL agents) using DQN and PPO algorithms with the same reward function (6) and monitor the training of RL agents (i.e., convergence and vRAN deployment policy). As shown in Fig. 4, the comparison of reward curves between DQN and PPO algorithms reveals significant differences in learning stability and performance convergence. DQN exhibits premature policy stagnation, as evidenced by the action ratio bars becoming predominantly fixed in green (i.e., SCS=15KHz, BW=50MHz, FS=7.2x) after episode 50, indicating that the agent's action selection becomes overly rigid and fails to maintain adequate exploration. This stagnation coincides with unstable convergence patterns where DQN experiences notable reward decreases during the initial training phase around episodes 20-50 and additional fluctuations between episodes 50-150, ultimately leading to a suboptimal accumulated average reward of approximately 0.36-0.37. Conversely, PPO demonstrates superior training results with a smooth improvement trajectory that steadily progresses from 0.35 to 0.42-0.43, representing a higher performance. The action ratio visualization for PPO shows more balanced and dynamic action selection throughout the entire training process, reflecting effective exploration-exploitation dynamics

that prevent premature convergence to suboptimal policies. To validate the findings from above training analysis, we evaluate the performance of DQN and PPO agents using the developed 5G NR SLS detailed in Sec. VI. We also train agents with Trust Region Policy Optimization (TRPO) algorithm, which serves as the foundational algorithm for PPO and employs a KL-divergence constraint to enforce conservative policy updates. Nevertheless, its computational complexity and sensitivity to hyperparameters limited practicality compared to PPO's simplified clipped surrogate objective. We elaborate on the performance evaluation among RL-based methods in Sec. VI-C.

IV. Problem Formulation

Based on the resource consumption and system dynamics of vRAN described in Sec. II, in this section, we formulate a joint optimization problem considering a multi-cell deployment scenario. We introduce two utility functions, $U(t,n)$ and $P(t,k)$, where $U(t,n)$ quantifies the resource utilization of a vRAN at time t and $P(t,k)$ represents the application performance at time t. Here, n represents the vRAN and k denotes the UE. We focus on virtualized resource utilization in terms of radio (bandwidth; BW), computing (CPU), and networking (XH). On the other hand, we measure the average goodput (i.e., the bitrate at which useful data reaches the UE) and the number of expired packets according to 3GPP 5QI for application performance. The definition of utility functions for the reward function design is detailed later in Sec. V-B.

For a given elapsed time T, the joint optimization problem aims to minimize resource utilization over the active vRAN N_t and maximize the average application performance over the connected users $K_{n,t}$, following a vRAN orchestration policy while respecting the resource constraints on BW, CPU, and XH. This problem can be defined as:

$$\max_{\pi} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|N_t|}\sum_{n=1}^{|N_t|}\left[-U(t,n)+\frac{1}{|K_{n,t}|}\sum_{k=1}^{|K_{n,t}|}P(t,k)\right] \quad (1)$$

$$\text{subject to}\begin{cases} U^{\text{BW}}(t,n)\le R_{\text{BW}} \\ U^{\text{CPU}}(t,n)\le R_{\text{CPU}} \quad \text{for all } n\in N_t, \\ U^{\text{XH}}(t,n)\le R_{\text{XH}} \end{cases}$$

where n and k stand for vRAN and UE, and π denotes the vRAN orchestration policy as an xApp in RIC that controls the FS option σ and the numerology μ at each decision point.

Considering a multi-cell distributed deployment scenario with a single dedicated vRAN per cell, we reformulate the problem as a multi-agent model. Each agent in this model is responsible for optimizing its respective vRAN and the associated UEs individually. This approach is adopted to mitigate the complexity arising from the extensive state and action spaces associated with numerous RAN statistics and UEs per cell. Accordingly, the equation (1), subject to the same constraints, is redefined as:

$$\frac{1}{T}\sum_{t=1}^{T}\frac{1}{|N_t|}\sum_{n=1}^{|N_t|}\max_{\pi_n}\left[-U(t,n)+\frac{1}{|K_{n,t}|}\sum_{k=1}^{|K_{n,t}|}P(t,k)\right]. \quad (2)$$

The optimization problem (2) can be directly expressed in the MDP form represented by a tuple $M := < S, A, R, p, \gamma >$ consisting of state, action, reward, transition probability, and discount factor, respectively:

$$\begin{cases} s(t)\in\mathcal{S}: \{U^{\text{CPU}}, U^{\text{XH}}, P^{\text{SE}}, P^{\text{gput}}, S^{\text{penalty}}, |K^{\text{sched}}|\} \\ a(t)\in\mathcal{A}: (\mu,\sigma)\sim\pi_n(\cdot|s_t) \\ r(t)\in\mathcal{R}: -U(t,n)+\frac{1}{|K_{n,t}|}\sum_{k=1}^{|K_{n,t}|}P(t,k) \end{cases},$$

where P^{SE} is the cell spectral efficiency (SE), P^{gput} is the average application goodput, S^{penalty} is the penalty score on the expired packets of applications, and β is a weight parameter. We will define the state space mathematically in the next section by deriving from 3GPP technical specifications and the reference studies. The Bellman equations for the optimal value functions by following the vRAN orchestration policy π_n are defined as follows:

$$V^*(s) = \max_{a\in\mathcal{A}} \mathop{\mathsf{E}}_{s'\sim p(\cdot|s,a)}[R(s,a)+\gamma V^*(s')], \quad (3)$$

$$Q^*(s,a) = \mathop{\mathsf{E}}_{s'\sim p(\cdot|s,a)}[R(s,a)+\gamma\max_{a'\in\mathcal{A}}Q^*(s',a')], \quad (4)$$

where $R(s,a)$ is a function depending on a state-action pair in the reward space $\mathcal{R}$, $V^*(s)$ is the optimal value function of state s regarding CPU and XH usage, SE, and application performance KPIs, and $Q^*(s,a)$ is the optimal action-value function of state s and action a which represents the vRAN update from xApp control policy π_n. The objective of the equation (2) aims to find an optimal control policy π_n of each vRAN n by satisfying the Bellman optimal equations (3) and (4). However, solving this is impractical because it is challenging to model and accurately predict the state transition probability p. More specifically, it is impossible to predict the traffic pattern of applications and estimate the CPU utilization of vRAN U^{CPU} further based on the prediction. Moreover, the crosshaul (XH) link usage U^{XH} also varies according to the wireless channel dynamics of connected user equipment (UE). Thus instead, we rely on a model-free and policy-based RL framework for solving the equation (2).

We define a trajectory $\tau = (s_0, a_0, s_1, a_2, ..., s_{T-1}, a_{T-1})$ representing a sequence of states and actions over time T following a parameterized policy π_θ. For simplicity, the policy π_θ stands for each vRAN's xApp policy π_n.Then the expected return of a policy is defined as:

$$J(\pi_\theta) = \int_\tau P(\tau|\pi_\theta)R(\tau) = \mathop{\mathsf{E}}_{\tau\sim\pi_\theta}[R(\tau)],$$

where $P(\tau|\pi_\theta)$ is the probability of a trajectory given the xApp policy π_θ, and $R(\tau)$ is the reward of a trajectory. One possible form of $R(\tau)$ is a discounted return $\sum_t \gamma^t R(s_t, a_t)$. As the reward of a trajectory can be different forms of reward or value functions, we define a general reward term $\Phi_t = R(\tau)$ and explain the choice of Φ_t for performing optimization in the next section. To find the optimal policy, the gradient of policy performance can be calculated as:

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathop{\mathsf{E}}_{\tau\sim\pi_\theta}[R(\tau)] = \mathop{\mathsf{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_t|s_t)R(\tau)\right].$$

By incorporating policy gradient $\nabla_\theta J(\pi_\theta)$ into the parameter update process, the optimal policy for solving the equation (2) can be estimated as follows:

$$\theta^* = \arg\max_\theta J(\pi_\theta) = \arg\max_\theta \mathop{\mathsf{E}}_{\tau\sim\pi}[R(\tau)],$$
$$\theta_{k+1} = \theta_k + \alpha\nabla_\theta J(\pi_{\theta_k}),$$

where α is the configurable learning rate. We elaborate on the optimization algorithm based on the PPO (Proximal Policy Optimization) algorithm and the design of the proposed MARL model named AD-VRAN as an RIC xApp in the next section.

V. Theoretical Analysis

This section elaborates on the design of the proposed MARL model, AD-VRAN, developed as an xApp within the RIC. We begin by defining the state and action spaces and their functional roles. Following this, we describe the reward-shaping process implemented for modeling training. Subsequently, we present the parameterized neural network architecture for the vRAN control policy, denoted as π_θ. Finally, we introduce the optimization algorithm for AD-VRAN, which leverages the PPO algorithm [16].

A. Design of RL Model

As defined in Sec. IV, the utility functions $U(t,n)$ and $P(t,k)$ represent the resource utilization rate and the application performance at time t. In a TEC, we consider the utilization rates of three virtualized resources necessary for operating a vRAN: radio (BW), computing (CPU), and networking (XH) resources. Concerning application performance, given that vRANs are agnostic to the traffic model, we measure the goodput and design a penalty score for the packets exceeding the 5QI PDB. Note that all elements in the state space are normalized to fall within the range of 0 to 1. This normalization is part of the reward-shaping process which will be elaborated upon subsequently. We present the design of the state space according to the four KPI categories.

1) **Radio Resource (BW):** Since our goal is not to manage bandwidth directly in this work, we analyze scenarios with maximum cell bandwidths corresponding to an SCS of either 15 KHz or 30 KHz equating to 50 MHz or 100 MHz, respectively. As radio resources are scaled up, cell performance is anticipated to enhance. Nevertheless, to adequately train RL agents on the significance of this state, it is crucial to have a more granular range of bandwidth values beyond just two discrete levels. To achieve minimal bandwidth usage, we employ spectral efficiency (SE), P^{SE}, which is mathematically defined as:

$$N^{\text{res}}_{n,t,k} = 12\cdot R^{\text{rb}}_{n,t,k}\cdot 14\cdot 2^\mu\cdot 10^3$$
$$R^{\text{tput}}(t,n) = \frac{\Sigma_k[M^{\text{mod}}_{n,t,k}\cdot R^{\max}_{n,t,k}\cdot N^{\text{res}}_{n,t,k}\cdot(1-O^{\text{dl}})]}{10^6}$$
$$U^{\text{BW}}(t,n) = 50\cdot(\mu+1), \mu\in\{0,1\}; U^{\text{max\_BW}} = 100$$
$$\Rightarrow P^{\text{SE}}(t,n) = \frac{R^{\text{tput}}(t,n)}{U^{\text{BW}}(t,n)}\cdot\left(\frac{R^{\text{max\_tput}}(t,n)}{U^{\text{max\_BW}}}\right)^{-1},$$

where $R^{\text{rb}}_{n,t,k}$ is the number of resource blocks allocated to UE k in vRAN n. Refer to 3GPP specification (38.214), $R^{\text{tput}}(t,n)$ is the cell throughput, and $R^{\text{max\_tput}}(t,n)$ is the theoretical maximum cell throughput of vRAN n at time t based on the maximum coding rate $R^{\max}$, modulation order M^{mod}, and the number of resource elements N^{res}.

2) **Computing (CPU):** In accordance with the CPU model characterization studies for 5G vRAN [7], [8], our focus is on the RLC, MAC, and PHY layers due to their significant CPU resource demands. Additionally, since the physical functional placement is integrated with RAN FS options, the required computing resources at the cell site are variable. We compute the CPU usage of PHY layer [7] in conjunction with the additional network functions (MAC and RLC) [8] deployed at the cell site as detailed below:

$$U^{\text{CPU}}(t,n) = U^{\text{VNF}}_{n,t}(\sigma) + \xi_1 + \xi_2\cdot|K^{\text{sched}}_{n,t}| + \xi_3\cdot\frac{\Sigma_k M^{\text{mcs}}_{n,t,k}}{|K^{\text{sched}}_{n,t}|},$$

where $\xi_1 = 35.65$, $\xi_2 = 3.9$, $\xi_3 = 0.369$, $U^{\text{VNF}}_{n,t}(\sigma) = 10\cdot(2-\sigma)$, and $|K^{\text{sched}}_{n,t}|$ is the maximum number of users for MAC scheduling of vRAN n at time t.

3) **Networking (XH):** As outlined previously, different RAN FS options impose varying XH link capacity requirements. In this analysis, we consider the dynamic selection mechanism for three specific RAN FS options (i.e., 2, 6, and 7.2x), with zero-based indexing used to denote these options in the SLS. The downlink throughput of the XH is specified in prior studies [5], [9] and is described as:

$$\delta_{\sigma 0} = \begin{cases}\Sigma_k[\max(0,(B^{\text{rlc}}_{n,t,k} - B^{\text{rlc}}_{n,t-1,k}))\cdot 8], \sigma_2 = 0\\ 0, \sigma\neq 0\end{cases}$$
$$\delta_{\sigma 1} = \begin{cases}\Sigma_k(\Sigma_i S^{\text{tbs}}_{n,t,k,i}\cdot 8) + 1.5\cdot|K_{n,t}|, \sigma_6 = 1\\ 0, \sigma\neq 1\end{cases}$$
$$\delta_{\sigma 2} = \begin{cases}\Sigma_k(\Sigma_i 12\cdot 14\cdot R^{\text{rb}}_{n,k,t,i}\cdot M^{\text{mod}}_{n,t,k}), \sigma_{7.2\text{x}} = 2\\ 0, \sigma\neq 2\end{cases}$$
$$\Rightarrow U^{\text{XH}}(t,n) = \frac{10^3\cdot(\delta_{\sigma 0}+\delta_{\sigma 1}+\delta_{\sigma 2})}{R^{\text{XH}}_{n,t}}, \forall\sigma\in\{0,1,2\},$$

where $\delta_{\sigma 0}$ is the amount of data from the MEC application to RLC buffer B^{rlc} (PDCP-RLC split), $\delta_{\sigma 1}$ is the size of data (TBs) from MAC to PHY layer (MAC-PHY split), and $\delta_{\sigma 2}$ is the fronthaul (FH) throughput. Note that, apart from the RAN FS options, the cell bandwidth and SCS also impact the XH requirements [10].

4) **Application Performance:** We measure the application's goodput and design a QoS penalty score based on 5QI PDB. Since our traffic generator involves real-time video streaming, HTTP-based, and FTP traffic models [15], the PDB is set to 10, 100, and 300 ms according to the 3GPP specification (23.501). The goodput, P^{gput}, and penalty scores, S^{penalty}, are normalized using the total number of transmitted bits and the packets, respectively.

$$P^{\text{gput}}_{t,k} = 10^3\cdot\frac{\Sigma_t(\Sigma_j S^{\text{pkt}}_{t,k,j})}{t}, R^{\text{app}}_{t,k} = 10^3\cdot\frac{S^{\text{app}}_{t,k}}{t}$$
$$\Rightarrow P^{\text{gput}}(t,n) = \frac{1}{|K^{\text{sched}}_{n,t}|}\sum_k\frac{P^{\text{gput}}_{t,k}}{R^{\text{app}}_{t,k}},$$

$$S^{\text{penalty}}(t,n) = \frac{1}{|K^{\text{sched}}_{n,t}|} \sum_{k} \frac{N^{\text{exp}}_{t,k}}{N^{\text{app}}_{t,k}},$$

where $S^{\text{app}}_{t,k}$ is the total size of packets in bits generated by application k at time t, $N^{\text{exp}}_{t,k}$ is the number of expired packets of application k at time t, and $N^{\text{exp}}_{t,k}$ is the total number packets generated by application k at time t.

B. State, Action, and Reward Shaping

The optimization objective of (2) and the reward function of the associated MDP M is to derive an optimal policy π_θ for each vRAN n. This policy is designed to minimize the utility function $U(t,n)$ and maximize the utility function $P(t,k)$. Based on the state design described in earlier sections, the state space s_t and the action space a_t are defined as follows:

$$\begin{cases} s_t = \{U^{\text{CPU}}, U^{\text{XH}}, P^{\text{SE}}, P^{\text{gput}}, S^{\text{penalty}}, |K^{\text{sched}}_{n,t}|\} \\ a_t = (\mu,\sigma) \sim \pi(\cdot|s_t), \text{ where } \begin{cases} \mu = \begin{cases} 0 \text{ (SCS = 15 kHz)} \\ 1 \text{ (SCS = 30 kHz)} \end{cases} \\ \sigma = \begin{cases} 0 \text{ (RAN FS = 2)} \\ 1 \text{ (RAN FS = 6)} \\ 2 \text{ (RAN FS = 7.2x)} \end{cases} \end{cases} \end{cases}$$

Then two utility functions $U(t,n)$ and $P(t,k)$ are defined as:

$$\begin{cases} U(t,n) = U^{\text{CPU}} + U^{\text{XH}} + S^{\text{penalty}}, \\ \frac{1}{|K_{n,t}|}\sum_{k=1}^{|K_{n,t}|} P(t,k) = P(t,n) = P^{\text{SE}} + P^{\text{gput}}. \end{cases} \tag{5}$$

Our reward function design process begins with preliminary experiments on the traditional RAN architectures (i.e., fixed vRAN deployment policy) that measure the state values s_t in utility functions $U(t,n)$ and $P(t,n)$. The state values are normalized to the range [0,1] to standardize scoring. As plotted in Fig. 3, the state space exhibits dynamic distribution patterns that vary significantly across simulation scenarios and action spaces, creating a multidimensional optimization challenge. Given that the optimization objective is to minimize resource consumption while maximizing application performance, we first assign a negative weight to $U(t,n)$ and a positive weight to $P(t,n)$. Analysis of state space boundaries revealed that both utility functions operate within a similar absolute value range (i.e., between 0.4 and 1.3). This parity creates convergence challenges during unweighted training, as conflicting reward signals produce oscillating average rewards that destabilize learning. To resolve this, we incorporate several carefully calibrated weight hyperparameters into the reward function. Specifically, we introduce a weight term $(1+|K^{\text{sched}}_{n,t}| \ / \ \beta_1)\cdot\beta_2$ on the negative component $U(t,n)$, which reflects the current traffic load and amplifies the influence of the negative term. This adjustment encourages RL agents to discover policies that yield positive rewards, even under challenging conditions. The degree of this impact can be controlled by tuning the weight parameter β_2 or counterbalanced by modifying the weight β_3 applied to the positive term $P(t,n)$, allowing for flexible adaptation to various training environments and goals. Accordingly, the reward function is formulated as:

$$r_t = -U(t,n)\cdot(1+\frac{|K^{\text{sched}}_{n,t}|}{\beta_1})\cdot\beta_2 + P(t,n)\cdot\beta_3. \tag{6}$$

C. Optimization Algorithm of AD-VRAN

We design our RL agent based on the advantage actor-critic (A2C) architecture. The actor-network generates the control policy π and samples actions a_t based on the current state s_t, $a_t \sim \pi(\cdot|s_t)$. The actor network is composed of multiple fully connected neural network (NN) layers with two distinct output branches. These branches share initial layers for state feature extraction, leading to the outputs for vRAN control, $a_t = (\sigma,\mu)$. The critic network also consists of fully connected NN layers and is tasked with predicting the state value $V^\pi(s_t)$, representing the expected cumulative reward of state s_t. We denote the parameters of the actor and critic networks as θ and ϕ, respectively.

The critic network aims to explore and fit an unknown state value function. The true state value (i.e., the cumulative reward at state s_t with a trajectory of length T) is defined as $V^\pi(s_t) = \sum_{t'=t}^{T}\gamma^{t'-t}r_{t'}$. The objective of the critic is to minimize the mean squared error (MSE) between the state values of the current policy π and the new parameterized policy π_ϕ:

$$\mathcal{L}_c(\phi) = \|V^\pi_\phi(s_t) - V^\pi(s_t)\|^2_2.$$

The actor networks are responsible for populating policies to maximize the fitted state value. We follow the PPO algorithm to define actor's objective as maximizing

$$\mathcal{L}^{\text{CLIP}}(\theta) = \min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right),$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$ measures how policy π_θ performs relative to the old policy π_{θ_k}, ϵ is a hyper-parameter. The generalized advantage estimation $\hat{A}_t$ (GAE) [17] represents the additional reward that could be obtained by the agent by taking a particular action, improving stability and efficiency in training. To ensure sufficient exploration, we further augment the objective as:

$$\mathcal{L}_a(\theta) = \mathcal{L}^{\text{CLIP}}(\theta) + \xi\mathcal{S}(\pi_\theta(\cdot|s_t)),$$

where $\mathcal{S}(\pi_\theta(\cdot|s_t))$ is the entropy bonus [18], [19] and ξ is a balancing hyper-parameter. For a set of trajectory $\mathcal{D}^{\text{trajectories}} = \{\tau_i\}_{i=1,\ldots,N}$, the parameter update process of ϕ and θ for the optimization is described as:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}|T}\sum_{\tau\in\mathcal{D}}\sum_{t=0}^{T}\mathcal{L}_c(\phi),$$

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}|T}\sum_{\tau\in\mathcal{D}}\sum_{t=0}^{T}\mathcal{L}_a(\theta).$$

VI. Numerical Evaluation

To quantitatively evaluate the proposed DRL-based policy, we develop a discrete-event system-level simulator (DE-SLS) by following the 3GPP technical specifications for 5G NR, reference architecture of O-RAN and srsRAN, and the implementation of 5G SLS (e.g., ns-3 5G-LENA, Nokia FikoRE). We extract various data metrics from SLS as state space for each simulation step and convert them into state variables for the RL model training. The overall architecture of our SLS for emulating near-RT RIC in an open vRAN and

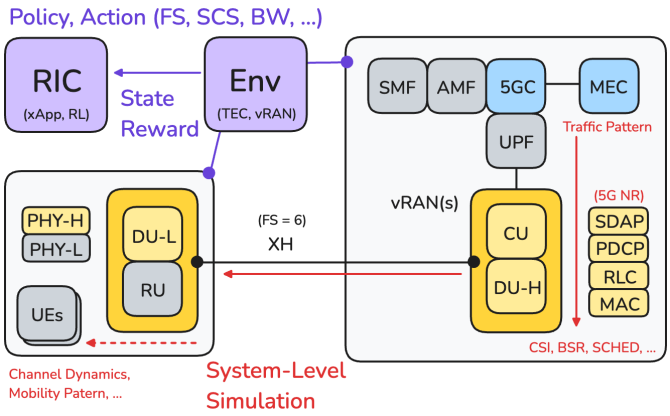

Fig. 5. Overview of SLS for open RAN and RIC in 5G NR.

TABLE II
CONFIGURATIONS OF SLS INTEGRATED WITH AN RL FRAMEWORK AS RIC IN A 5G NR VRAN.

Environment Setting	**Value**
# Cells	21
# Cells (active)	3, 6
# Users (total)	30, 60, 90, 120, 180
# Users (per cell)	10, 20, 30
Environment	Urban Macro [13], [14]
Inter-Site Distance (ISD)	500 (m)
Mobility	Manhattan
Traffic Model	FTP, HTTP, Video-streaming [15]
User Speed	3 (km/h)
Radio Resource Alloc.	RAT#0
Resource Block Group	4 (RB)
TDD Pattern	DDDDUDDDDU
Scheduling Algo.	Proportional Fair (PF)
HARQ	3 retx; 20 processes (per user)
Reliability	99.999%
Target BLER (TBLER)	10%
Channel Model	ITU-R M.2412-0 [14]

Simulation Setting	**Value**
Time Step	1 (ms)
vRAN Update	100 (ms)
Evaluation Time	30,000 (ms)
CPU Resource	1 (per vRAN)
XH Capacity	1 (Gbps; per vRAN)
Cell Bandwidth (BW)	50, 100 (MHz)
Numerology (μ)	0 (SCS=15 kHz), 1 (SCS=30 kHz)
FS Option (σ)	0 (FS 2), 1 (FS 6), 2 (FS 7.2x)

TEC environment is illustrated in Fig. 5. In the following subsections, we first elaborate on our SLS's features and the E2E data flow in the downlink direction. Then, we describe the RL training process and explain the performance evaluation figures in Sec. VI-C.

A. System-Level Simulator

We develop an SLS to simulate the operation of 5G NR network functions – including the RLC, MAC, and PHY layers as described in Sec. II – within an open and vRAN setup, accurately replicating realistic network conditions. The environment and simulation configurations are listed in Table II. Following the technical documents and models from 3GPP (i.e., TS 23.501 [20], TS 38.214 [21], and TR 38.901 [22]), ITU reports (i.e., M.2135-1 and M.2412-0), ns-3, and NGMN

Algorithm 1 AD-VRAN

Initialize hyperparameters γ, ϵ, and ξ
Following Table II, initialize VNFs for central site and cell sites, distribute UEs (numbers, locations, application models, and mobility patterns) among multiple cells
Initialize state $s_t \leftarrow s_0$
for $S = 1$ to $S_{\max}$ **do**
  while M is not full **do**
    Sample action $a_t \sim \pi_\theta(\cdot|s_t)$
    Execute vRAN with configuration a_t for 100ms, observe reward r_t and next state s_{t+1}
    Store tuple (s_t, a_t, r_t) in M
    if Trajectory ends **then**
      Reinitialize state $s_t \leftarrow s_0$
    else
      Update state $s_t \leftarrow s_{t+1}$
    end if
  end while
  for $e = 1$ to $\lfloor R \times (\|M\|/B) \rfloor$ **do**
    Sample B tuples from M
    Compute $\mathcal{L}_c(\phi)$ and $\mathcal{L}_a(\theta)$ using the samples
    Update critic parameters: $\phi \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}_c(\phi)$
    Update actor parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_a(\theta)$
  end for
  Clear memory buffer M
end for

[15], we evaluate multi-cell scenarios in an Urban Macro environment with 500m inter-site distance, where the distributed UEs are equipped with Manhattan mobility patterns [13], [14]. Traffic pattern incorporates NGMN and 3GPP models in ns-3 for FTP (truncated log-normal file size distribution with 2MB mean), HTTP (with main and embedded objects following log-normal and Pareto distributions), and real-time video streaming (deterministic frame structure with Pareto-distributed packet sizes) [15]. IP packets of these MEC applications are segmented based on the maximum transmission unit (MTU) of 1500 bytes and buffered at the MEC platform. Since the RLC buffer size is set to 1 MB, application packets will fill up the RLC buffer and be scheduled for transmission by the network functions f_{MAC} and f_{PHY}.

Additionally, to emulate the RIC platform, RAN statistics defined in Sec. V-A are transmitted from the SLS to an integrated RL framework at each time step t. The optimization goal of the proposed MARL model AD-VRAN is described in Sec. IV, with the optimization algorithm detailed in Sec. V-C. The following section will cover the model training procedure for solving the joint optimization problem (2).

B. DRL Model Training

In the training stage, we collect several trajectories containing (s_t, a_t, r_t) tuples and store these in a memory buffer M. Each trajectory is defined with a duration of 10,000 ms, and the agent executes sample actions at every policy update interval (i.e., 100 ms simulation step), which counts as a single training step within a trajectory. Given that our SLS operates on a 1

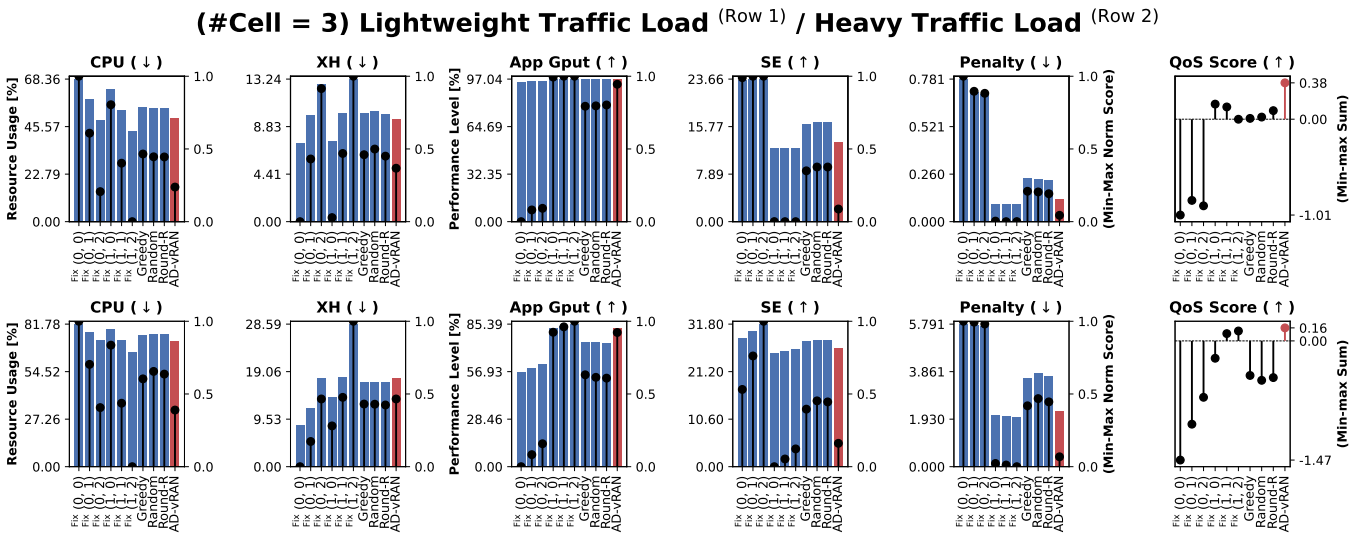

Fig. 6. Multi-dimensional trade-offs analysis: CPU utilization, XH usage, application Goodput (Gput), spectral efficiency (SE), penalty. The naming of fix policies follows (SCS μ, FS option σ). For example, (0, 2) means SCS equals 15 KHz and FS option is 7.2x. The bars are the resource usage or performance level. The black vertical line is the corresponding min-max normalization score. The QoS score is the sum of the min-max value of KPI metrics.

ms interval, all metrics are averaged over the previous 100 ms interval. To achieve generalization, instead of training multiple agents on a specific model or scenario, we train our agents in a variety of configurations involving different numbers of distributed UEs, traffic models of UEs, and the related parameters of traffic models. Periodically, we use batched samples from M to update the actor and critic networks. The precise algorithm is detailed in Algorithm 1. The actor and critic for each vRAN are trained for $S_{\max}$ =512$\|M\|$ training steps with the same learning rate of 0.0001. The memory buffer size $\|M\|$ is 256, the batch size B is 64, and the sample reuse time R is 20. The hyperparameters γ, ϵ, and ξ are set to the common PPO implementation values of 0.95, 0.2, and 0.001, respectively. The hyperparameters β_1, β_2, and β_3 of the reward function are set to 30, 1.1, and 0.9.

C. Performance Evaluation

To evaluate the DRL-based policy, we compare our agent (xApp `AD-VRAN`) with six traditional RAN architectures (i.e., fixed SCS and FS option), two heuristic approaches, namely random selection (random combination) and round-robin (Round-R). The evaluation results are shown in Fig. 6. We also compare our method with other RL-based approaches (i.e., epsilon-greedy action selection, DQN, and TRPO). Due to the page limitations, the implementation details are elaborated in the Appendix section of the full version of this manuscript [23], and the evaluation results are presented in Fig. 7. We collect the simulation results from 100 random seeds and average each metric over 30,000 time steps. Metrics measured include CPU usage, XH link usage, application goodput, SE, and penalty score under two traffic scenarios to reflect the resource utilization rate and application performance of vRAN. Due to the various distributions (i.e. mean value) of each metric, to quantify the overall performance, we first apply min-max normalization[2] on each metric, aligning the metrics within the same scoring unit (represented by black inner bars). Following the optimization goal, we compute the performance using an equal-weighted sum of metrics: $-\text{CPU} - \text{XH} + \text{Gput} + \text{SE} - \text{Penalty}$. The trade-offs between each metric and the performance of the proposed approach are discussed below:

1) **Lightweight Traffic Load**: In this scenario, 40% of UEs have FTP traffic, 30% have HTTP traffic, and the other 30% have video streaming traffic. The bar charts indicate that our agent strikes a balance between each metric and thus achieves the highest QoS score. For example, our approach provides high application performance while consuming lower CPU and XH resources. As we will explain the behavior of the proposed DRL method in Fig. 8, our agent is able to switch between different policies to avoid those leading to high resource consumption with poor application performance. Specifically, although the fixed policy with FS option 2 (e.g., $\sigma = 0$), a distributed RAN architecture, reduces the XH load demands, it requires higher computing resources at the cell site, resulting in a lower application performance level.
2) **Heavy Traffic Load**: Given that FTP traffic [15] represents the heaviest traffic pattern, all distributed UEs are engaged in FTP traffic in this scenario. The bar chart shows an increase in CPU utilization rate and XH link throughput with higher server loads. Comparing both scenarios, baseline policies show inconsistent performance. Despite this, our agent maintains the highest QoS score with a good balance between resource consumption and performance KPIs. Although the fixed policy (i.e., $\mu = 1$, $\sigma = 2$) has a high performance level and achieves a similar QoS score to our approach, it consistently consumes more XH

[2] $x_{\text{Min-Max}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$, e.g., $\text{CPU}^{(0,1)}_{\text{Min-Max}} = \frac{0.58614 - 0.43493}{0.68368 - 0.43493} = 0.60791$

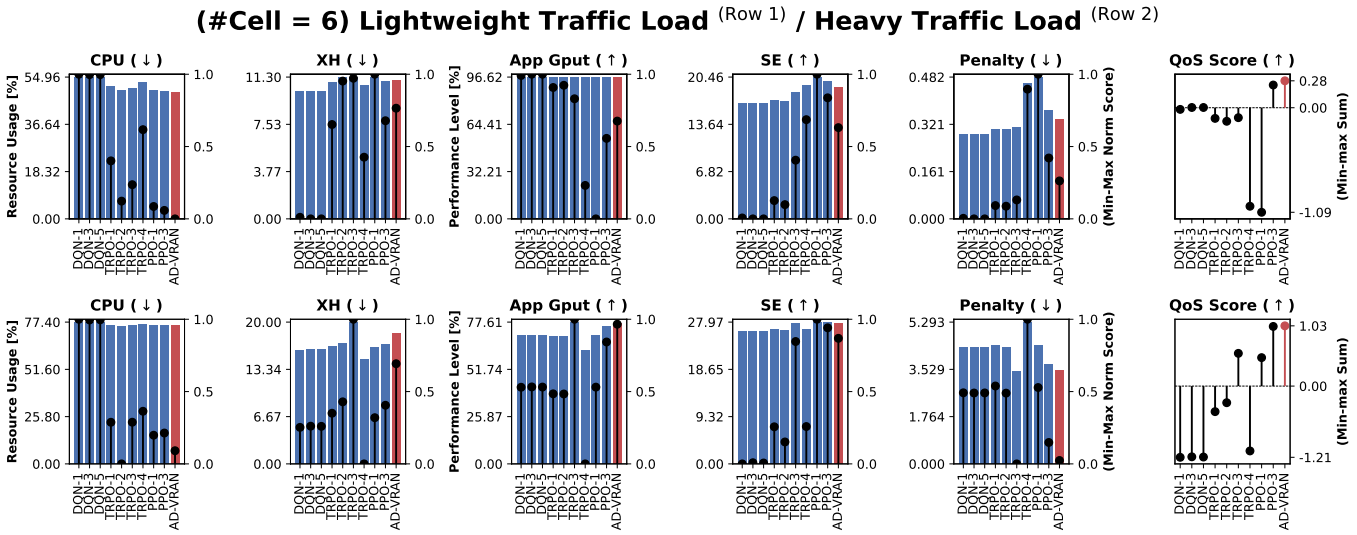

Fig. 7. Performance comparison between RL agents trained by DQN, TRPO, and PPO algorithms with varying hyperparameters.

resources due to the intra-phy FS option and has a lower SE score. Generally, higher goodput corresponds to increased SE. However, as SE in this context is calculated over fixed bandwidths of 50 or 100 MHz, as defined in Sec. V, simply doubling the bandwidth does not produce a directly proportional increase in throughput. This is due to the impact of channel variability and the constrained computational capacity available for UE scheduling, ultimately leading to a lower SE score.

3) **RL-based Methods**: In addition to comparing the epsilon-greedy policy with the PPO-based agent (AD-VRAN), we further evaluate RL agents in a six-cell environment—i.e., deploying six RL agents—using DQN, TRPO, and PPO algorithms. These agents are trained under various hyperparameter configurations, including learning rate, replay buffer size, and batch size, to facilitate a comprehensive performance comparison. Detailed descriptions of the algorithmic implementations and parameter settings are provided in the Appendix of the full manuscript [23]. As illustrated in Fig. 7, the composite QoS scores obtained by DQN agents exhibit little variation, attributable to their simplistic but ineffective exploration strategy and suboptimal replay buffer management, which leads to inferior policy performance. In the case of TRPO agents, the results are inconsistent across different traffic load conditions, primarily due to the sensitivity of hyperparameters such as the KL divergence constraint—which governs the size of the trust region for model updates—and the number of conjugate gradient iterations. Consequently, TRPO agents may require scenario-specific hyperparameter tuning to achieve optimal performance. In contrast, PPO agents, built upon the A2C architecture and employing periodic replay buffer refresh, demonstrate greater adaptability. While all RL agents eventually converge during training, PPO agents achieve superior and more consistent performance by substantially reducing resource consumption while maintaining equal or higher application performance. Therefore, PPO agents yield more robust learning outcomes. Notably, the PPO-2 agent is selected as the AD-VRAN agent because, unlike PPO-3—which employs a larger batch size resulting in significantly longer training times—PPO-2 offers a favorable trade-off between efficiency and performance.

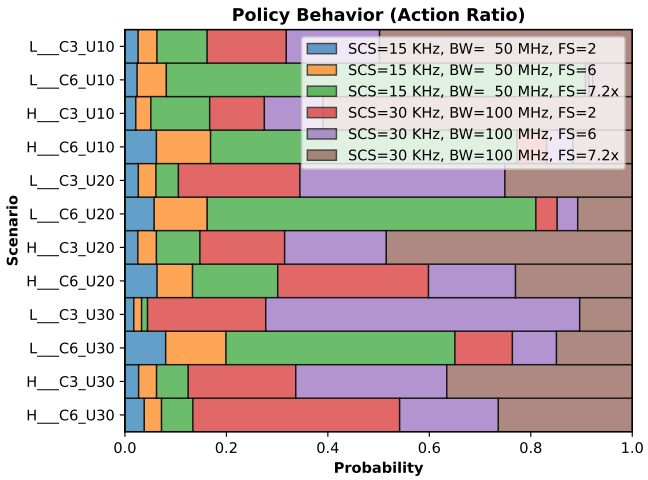

Fig. 8. Policy decisions of vRAN deployment and configurations.

4) **Policy Decisions**: In addition to the overall performance, we also investigate the proposed DRL policy's behavior in relation to vRAN deployment (i.e., FS and SCS). Fig. 8 illustrates the decision ratios of our agent across various simulated scenarios. The variance between each stacked bar confirms the agent's robust generalizability to accommodate not only varying UE counts but also various traffic types. For instance, compared to the scenario of lightweight traffic, while our agent prefers $\mu = 1$ (i.e., SCS 30 KHz) in heavy FTP traffic scenarios for a higher application performance, it also adapts to the number of distributed UEs by switching between different RAN FS

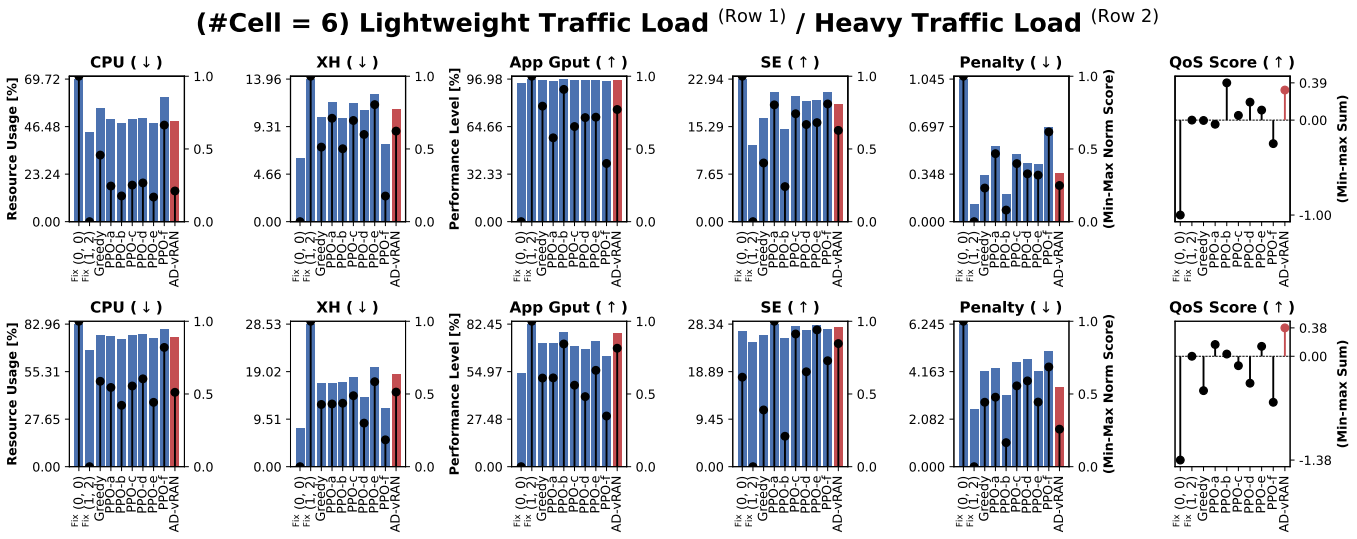

Fig. 9. Performance evaluation of the ablation study by PPO-based agents trained with the removal of algorithm components in the designated reward function. From PPO-a to PPO-f, the removed component is P^{gput}, P^{SE}, $|K_{n,t}^{\text{sched}}|/\beta_1$, S^{penalty}, U^{XH}, and U^{CPU}, respectively.

options. Specifically, with the number of distributed UEs increasing, our agent selects $\sigma = 2$ (i.e., FS 7.2) with lower probabilities to avoid high XH link throughput.

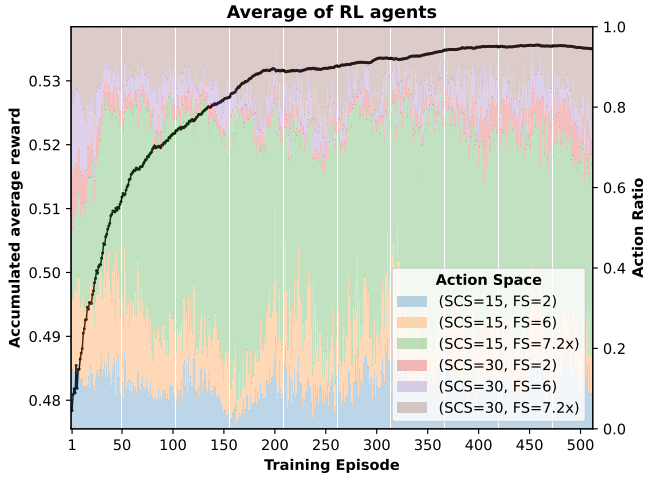

Fig. 10. Convergence analysis of MARL model training.

D. Convergence of MARL Training

In addition to the performance comparison between different RL-based methods, PPO also shows strength in training by employing entropy to encourage exploration and multiple sample reuse time for model update. At the same time, PPO leverages the advantage function and the A2C framework within its clipped objective, optimizing the policy to maximize advantage while maintaining proximity to the previous policy. We observe and depict the behavior of our RL agents during training in Fig. 10. The reward curve progresses through three distinct stages. The initial rapid learning phase from episodes 1 to 50, where average reward surges as agents quickly grasp basic environment interactions. Followed by an intermediate phase from episodes 50 to 200, showing slower but steady improvement through refining policy decisions. The convergence can be observed starting around episode 200, although there are some small oscillations until episode 350. On the other hand, the utilization of the action space plotted in bars reveals critical exploration-exploitation dynamics. During the first 200 episodes, the varying action selection ratios reflect active exploration, matching with the steepest reward gains as agents test diverse strategies. After episode 200, stabilized action distributions indicate a shift toward exploitation from learned policies, aligned with the early convergence signal around episode 200. Lastly, the persistent minor fluctuations in both reward and action ratios between episodes 300 and 500 suggest exploratory samplings even during exploitation, preventing premature policy stagnation.

E. Ablation Study

To train RL agents to grasp the environmental dynamics, the variables in the designated state space are formed by various RAN metrics, including the number of allocated RBs, cell throughput, average MCS, etc., capturing the fluctuating RAN statistics. Based on these state variables, we design the reward function (6) for RL agents to learn the complex trade-offs and the ultimate optimization goal, achieving effective model updates in training. To examine the importance of these components in handling the abrupt environmental changes, such as traffic surges caused by diverse application models and varying numbers of UEs due to dynamic mobility patterns, we conduct an ablation study by training six PPO-based agents through removing state variables in the reward function (6) that contribute to the performance and robustness of AD-VRAN training algorithm.

As shown in Fig. 9, the PPO agents trained with an incomplete reward function (i.e., PPO-a, PPO-b, etc.) suffer from worse and unstable performance across different evaluation

scenarios. Specifically, although the PPO-b agent, which is trained without the state variable P^{SE}, outperforms the AD-VRAN agent slightly in the lightweight traffic, its performance is close to fixed RAN architecture (i.e., SCS=30kHz, FS=7.2x) under the heavy traffic loads. As for our proposed solution, the AD-VRAN agent has the second highest composite QoS score in the lightweight traffic, but achieves the best performance in a more stringent scenario with heavy traffic. This indicates that the designated state space is effective and necessary in handling the abrupt environmental changes for training RL agents to perform consistently, proving to be the best orchestration strategy in a dynamic network environment.

VII. Related Work

A. 5G NR System-level Simulator

In this paper, we dynamically select the RAN FS options and numerology (i.e., subcarrier spacing, SCS) in our SLS. We develop our 5G NR SLS based on the code architecture and the channel model in [12] and add novel features regarding the modeling of vRAN resource consumption and dynamic vRAN reconfiguration. Although there are many well-known SLSs for 5G NR modeling and simulations, such as ns-3 5G-LENA [24], OMNeT++ Simu5G [25], and NVIDIA Sionna [26], none of them implements the vRAN resource consumption modeling, which affects the performance of RAN and applications. Our SLS uniquely focuses on modeling CPU consumption and cross-Haul (XH) link usage, specifically capturing how these factors impact system performance within a vRAN context. For example, CPU limitations directly affect the maximum number of UEs that can be scheduled for data transmission. These vRAN-specific features—such as detailed resource consumption tracking and their effect on end-to-end performance—are not present in any open-source SLSs available today.

B. Adaptive Functional Split

Considering diverse applications and different system loads of base stations, intelligent selection of RAN FS options can utilize virtualized resources (e.g., BW, CPU) more efficiently and provide a better quality of service (QoS) to users. For example, the work [6] builds two function migration binaries and demonstrates that switching RAN FS options at runtime with a certain packet loss ratio is possible. JOBFR [27] aims to jointly optimize the base station sleeping mode, RAN FS options, and routing in vRAN heuristically. SO-RAN [28] creates isolated RAN slices by optimally placing RAN and MEC functions per slice using a distributed method. The two most relevant studies of this work are as follows. LOFV [29] intelligently selects RAN FS options and allocates virtualized resources to minimize management costs of vRAN while respecting users' demands. Another study [30] proposes to select the best possible RAN FS option, maximizing the centralization gain given the constraints on the channel bandwidth (BW) and XH link capacity. However, their state spaces are either traffic loads or the quantity of resource allocation without performance metrics, such as application and cell throughput. We also implement epsilon-greedy and deep Q-learning methods in our evaluation by referring to [29] and [30], respectively, and the results prove that our proposed PPO method with the distributed MARL model is more capable of addressing the joint optimization problem defined in Sec. IV. Lastly, none of these approaches address the trade-offs introduced by numerology (i.e., SCS) and FS options affecting CPU consumption, XH capacity requirements, cell throughput, and application performance.

C. Data-driven RAN Analytics

Data-driven optimization requires tremendous data collection of RAN statistics (e.g., channel quality, radio resource allocation, etc.). Additionally, online reconfiguration demands control messages frequently enforced according to a policy. With the O-RAN-defined E2 interface, we can interactively communicate between control application xApps within the RIC and the Open RAN architecture. A rich literature has utilized this paradigm to enable radio information analytics for diverse applications. For instance, the authors of [31] build a prototype of the radio network information service (RNIS) [32] through REST APIs, allowing MEC applications to access contextual radio information (e.g., CQI, RSRP, and RSRQ) on the UE end. The work [33] implements closed-loop control of scheduling policy for each network slice as an xApp in O-RAN by analyzing Key Performance Measurements (KPMs) from RAN nodes such as CQI, MCS, and the number of TBs. The recent advancement of the well-known 5G SLS, ns3-oran, integrates and extends the OpenRAN Gym to mimic the behavior and components of the O-RAN architecture for RIC application development [34]. This work integrates our SLS with an RL framework to emulate RIC operations for vRAN deployment and reconfiguration by analyzing RAN statistics defined as state/observation space in Sec. V-B.

VIII. Concluding Remarks

The openness of the modern RAN architecture is promising in serving emerging applications with diverse QoS requirements. Leveraging RIC and learning for computational sequential decision-making, the deployment architectures and vRAN configurations can be adaptively controlled based on the varying number of UEs and cell traffic loads, ensuring the best orchestration policy. This paper proposes a DRL-based RIC xApp as a control policy to optimize overall performance through periodic updates to vRAN configurations (i.e., RAN FS options, numerology). We first formulate the joint optimization problem as an MDP in the multi-cell scenario and employ a MARL framework for finding the optimal policy. We design model-free, policy-based DRL agents with quantitative KPI metrics related to application performance and vRAN resource consumption, including radio (BW), computing (CPU), and networking (XH) resources. We compare our approach with six traditional and fixed RAN architectures, two heuristic strategies, and RL-based methods (i.e., epsilon-greedy, DQN, and TRPO). An extensive numerical analysis using a developed 3GPP-compliant SLS demonstrates that our approach strikes a great balance between five metrics and

outperforms other policies in the composite QoS score. Our DRL-based policy yields up to a 14% reduction in resource usage while preserving equivalent application performance compared to other approaches. The future work aims to deploy a real testbed of 5G NR networks and test the algorithm in actual systems with a telemetry system emulating the behavior of RIC for the monitoring of virtualized resource consumption and performance metrics.

REFERENCES

[1] C. G. Brinton, M. Chiang, K. T. Kim, D. J. Love, M. Beesley, M. Repeta, J. Roese, P. Beming, E. Ekudden, C. Li, G. Wu, N. Batra, A. Ghosh, V. Ziegler, T. Ji, R. Prakash, and J. Smee, "Key focus areas and enabling technologies for 6g," *IEEE Communications Magazine*, vol. 63, no. 3, pp. 84–91, 2025.

[2] NTT DOCOMO, "5G open RAN ecosystem whitepaper." https://www.docomo.ne.jp/english/corporate/technology/whitepaper_5g_open_ran.

[3] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, dec 2020.

[4] P. Abdisarabshali, K. T. Kim, M. Langberg, W. Su, and S. Hosseinalipour, "Dynamic D2D-assisted federated learning over O-RAN: Performance analysis, MAC scheduler, and asymmetric user selection," 2024. [Online]. Available: https://arxiv.org/abs/2404.06324

[5] 3GPP, "NG-RAN; architecture description," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.401, 01 2018, version 15.0.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3219

[6] A. M. Alba, J. H. G. Velásquez, and W. Kellerer, "An adaptive functional split in 5G networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 410–416.

[7] S. Pramanik, A. Ksentini, and C. Fabiana Chiasserini, "Characterizing the computational and memory requirements of virtual RANs," in *2022 17th Wireless On-Demand Network Systems and Services Conference (WONS)*, 2022, pp. 1–8.

[8] C. Wei, A. Kak, N. Choi, and T. Wood, "5GPerf: Profiling open source 5G RAN components under different architectural deployments," in *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*, ser. 5G-MeMU '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 43–49.

[9] 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.

[10] S. Lagén, L. Giupponi, A. Hansson, and X. Gelabert, "Modulation compression in next generation RAN: Air interface and fronthaul tradeoffs," *IEEE Communications Magazine*, vol. 59, no. 1, pp. 89–95, 2021.

[11] S. R. S. (SRS), "Software Architecture – srsRAN Project," https://docs.srsran.com/projects/project/en/latest/dev_guide/source/\software_arch/source/index.html.

[12] D. G. Morín, D. Medda, A. Iossifides, P. Chatzimisios, A. G. Armada, A. Villegas, and P. Peréz, "An extended reality offloading IP traffic dataset and models," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 6820–6834, 2024.

[13] M. Series, "Guidelines for evaluation of radio interface technologies for IMT-advanced," *Report ITU*, vol. 638, no. 31, 2009.

[14] ——, "Guidelines for evaluation of radio interface technologies for IMT-2020," *Report ITU*, vol. 2512, no. 0, 2017.

[15] B. Bojovic and S. Lagen, "Enabling NGMN mixed traffic models for ns-3," in *Proceedings of the 2022 Workshop on Ns-3*, ser. WNS3 '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 127–134. [Online]. Available: https://doi.org/10.1145/3532577.3532602

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

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

[18] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[20] 3GPP, "System Architecture for the 5G System (5GS)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 06 2018, version 15.2.0, Release 15. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[21] ——, "NR; physical layer procedures for data," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.214, 10 2023, version 17.7.0, Release 17. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3216

[22] ——, "Study on channel model for frequencies from 0.5 to 100 GHz," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.901, 11 2020, version 16.1.0, Release 16. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3173

[23] Y.-Y. Lou, C. Chen, Y.-H. Huang, M. Chiang, and K. T. Kim, "AD-VRAN: DRL-based adaptive deployment of virtualized RAN in an open telco edge cloud," https://kimkt.com/publications/AD-VRAN-tech.pdf, 2025, [Online; accessed 19-June-2025].

[24] K. Koutlia, B. Bojovic, Z. Ali, and S. Lagén, "Calibration of the 5G-LENA system level simulator in 3GPP reference scenarios," *Simulation Modelling Practice and Theory*, vol. 119, p. 102580, 2022.

[25] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5G–an OMNeT++ library for end-to-end performance evaluation of 5G networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, 2020.

[26] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," 2023. [Online]. Available: https://arxiv.org/abs/2203.11854

[27] Y. Xu, H. Li, Z. Zhu, Y. Chen, L. Wang, Z. Lu, and X. Wen, "Joint optimization of base station sleeping, functional split, and routing selection in virtualized radio access networks," in *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, 2023, pp. 1–6.

[28] B. Ojaghi, F. Adelantado, and C. Verikoukis, "SO-RAN: Dynamic RAN slicing via joint functional splitting and MEC placement," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 2, pp. 1925–1939, 2023.

[29] F. W. Murti, S. Ali, G. Iosifidis, and M. Latva-Aho, "Learning-based orchestration for dynamic functional split and resource allocation in vRANs," in *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, jun 2022.

[30] H. Gupta, A. Antony Franklin, M. Kumar, and B. R. Tamma, "Traffic-aware dynamic functional split for 5G cloud radio access networks," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022, pp. 297–301.

[31] ETSI, "ETSI GS MEC 012 V2.2.1: multiaccess edge computing (MEC); radio network information API," https://www.etsi.org/deliver/etsi_gs/MEC/001_099/012/02.02.01_60/\protect\@normalcr\relaxgs_MEC012v020201p.pdf.

[32] S. Arora, P. A. Frangoudis, and A. Ksentini, "Exposing radio network information in a MEC-in-NFV environment: the RNISaaS concept," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 306–310.

[33] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing machine learning-based xApps for open RAN closed-loop control on programmable experimental platforms," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5787–5800, 2023.

[34] W. Garey, T. Ropitault, R. Rouil, E. Black, and W. Gao, "O-RAN with machine learning in ns-3," in *Proceedings of the 2023 Workshop on Ns-3*, ser. WNS3 '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 60–68. [Online]. Available: https://doi.org/10.1145/3592149.3592157

[35] F. Rezazadeh, L. Zanzi, F. Devoti, H. Chergui, X. Costa-Pérez, and C. Verikoukis, "On the specialization of FDRL agents for scalable and distributed 6G RAN slicing orchestration," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3473–3487, 2023.

[36] X. Lin, D. Yu, and H. Wiemann, "A primer on bandwidth parts in 5G new radio," *5G and beyond: Fundamentals and Standards*, pp. 357–370, 2021.

[37] A. Martínez Alba and W. Kellerer, "Dynamic functional split adaptation in next-generation radio access networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3239–3263, 2022.

APPENDIX A
IMPLEMENTATION OF RL-BASED AGENTS

A. Epsilon-Greedy Policy of Action Selection

In this rule-based decision system, actions are selected using a pure ϵ-greedy policy without any training or learned value function. Given the current environment state s_t and the previously taken action a_{t-1}, a custom scoring (or reward) function is applied to compute a performance score. This score is used to update a lookup table $\texttt{score\_board}[a]$, which stores the highest observed score for each possible action a. At each time step, the agent selects the best-known action with probability $1-\epsilon$, or a random action with probability ϵ:

$$a_t = \begin{cases} \texttt{random action}, & \text{with probability } \epsilon, \\ \arg\max_a \texttt{score\_board}[a], & \text{with probability } 1-\epsilon. \end{cases}$$

This policy enables simple exploration over a discrete action space without using any function approximation, gradient updates, or value iteration. The score board acts as a hand-coded memory of observed performance associated with each action.

B. Deep Q-Networks (DQN)

DQN is a value-based reinforcement learning algorithm that combines Q-learning with deep neural networks to approximate the action-value function $Q(s,a)$. The goal is to learn a policy that selects actions maximizing the expected cumulative reward. At each iteration, the Q-network is trained to minimize the temporal difference (TD) error between the current Q-value estimate and a target value computed using the Bellman equation:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\theta(s,a) - y\right)^2\right],$$
$$\text{where } y = r + \gamma \max_{a'} Q_{\theta^-}(s',a').$$

Here, $\mathcal{D}$ is a replay buffer of past transitions used for experience replay, and Q_{θ^-} is a periodically updated target network that stabilizes training. By using experience replay and a target network, DQN reduces the correlation between samples and mitigates instability due to non-stationary targets.

In practice, the Q-network is updated via stochastic gradient descent, typically using mini-batches sampled from $\mathcal{D}$, and the target network parameters θ^- are synchronized with the online network parameters θ every fixed number of steps. DQN is sensitive to hyperparameters such as the learning rate, target update frequency, and exploration strategy; common choices include ϵ-greedy policies with linear annealing of ϵ and target updates every 10^3–10^4 steps.

C. Trust Region Policy Optimization (TRPO)

TRPO is a precursor to PPO that offers stronger theoretical guarantees on *monotonic policy improvement*, though at a higher computational cost. Unlike vanilla policy gradient methods that keep the updated policy close to the old one in *parameter space*, TRPO improves stability by constraining the *Kullback–Leibler (KL) divergence* between the new and old policies in *distribution space*. It maximizes the surrogate objective

$$\mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)} A^{\pi_{\text{old}}}(s_t,a_t)\right]$$

subject to a trust region constraint on the expected KL divergence:

$$\mathbb{E}_t\left[\text{KL}\left(\pi_{\text{old}}(\cdot|s_t) \,\|\, \pi_\theta(\cdot|s_t)\right)\right] \le \delta.$$

We use Taylor's series to expand the terms up to the second order around the current parameters. The second-order term of the surrogate objective is comparatively small and is therefore ignored, resulting in the following constrained quadratic optimization problem:

$$\max_\theta \; g^\top(\theta - \theta_{\text{old}}) \quad \text{subject to} \quad (\theta-\theta_{\text{old}})^\top F(\theta-\theta_{\text{old}}) \le 2\delta,$$

where g is the policy gradient and F is the Fisher Information Matrix (FIM) estimated from the KL divergence. The analytical solution to this problem is given by

$$\theta_{\text{new}} = \theta_{\text{old}} + \sqrt{\frac{2\delta}{x^\top F x}} \cdot x, \quad \text{where } x = F^{-1}g.$$

TABLE III
TRAINING PARAMETERS OF RL-BASED AGENTS: DQN, TRPO, PPO

Algorithm	Model Name	Learning Rate	Buffer Size	Batch Size	Clip Range	Epsilon (start, end, decay)	CG Iteration
DQN	DQN-1	0.001	128	32	—	(0.05, 1.0, 10000)	—
	DQN-2	0.0001	128	32	—		—
	DQN-3	0.001	256	64	—		—
	DQN-4	0.0001	256	64	—		—
	DQN-5	0.001	512	128	—		—
	DQN-6	0.0001	512	128	—		—
TRPO	TRPO-1	0.001	128	—	—	—	5
	TRPO-2		128	—	—	—	10
	TRPO-3		256	—	—	—	5
	TRPO-4		256	—	—	—	10
	TRPO-5		512	—	—	—	5
	TRPO-6		512	—	—	—	10
	TRPO-7		256	—	—	—	15
PPO	PPO-1	(actor, critic) = (0.0001, 0.0001)	128	32	0.2	—	—
	PPO-2		256	64		—	—
	PPO-3		512	128		—	—

Since forming F explicitly is infeasible for high-dimensional policies, the natural gradient direction x is computed approximately by solving $Fx = g$ using the *conjugate gradient* method, typically with 10–20 iterations. Each Fisher-vector product Fv is computed efficiently using *Hessian-vector products* via automatic differentiation of the KL divergence.

A *backtracking line search* is then performed along the computed direction to select a step size $\alpha \in \{1.0, 0.5, 0.25, \dots\}$ that satisfies two conditions: (i) the surrogate objective improves, and (ii) the KL divergence constraint is satisfied. This final step compensates for the approximation error introduced by the Taylor expansion and ensures stable policy updates without requiring manual learning rate tuning.

APPENDIX B
DISCUSSION AND FUTURE WORK

Through a series of numerical evaluations, our DRL-based approach (i.e., A2C and PPO-based RL agents) is demonstrated to be adaptive and superior to the traditional RAN architectures, heuristic, greedy, and other RL-based methods in terms of the QoS score. By observing the multi-dimensional metrics as state variables, our agent intelligently selects beneficial vRAN architecture and configurations as action spaces to achieve higher rewards, while respecting the virtualized resource constraints. We discuss several open problems and extended research below.

A. Scalability of MARL Training in AD-vRAN

The proposed MARL model trains RL agents based on PPO algorithms for each vRAN instance of the cell. The scalability of the proposed MARL model training is affected by several factors, including but not limited to the number of UEs and the application traffic of UEs. As more UEs are connected to the cell and scheduled for transmission, the RIC xApp needs to process a higher amount of data from the vDU for the state space observation. Also, if there are multiple PDU sessions of each UE established for different applications, the complexity and the loading of the computing process of vDU and xApp will also increase. In our implementation, the required memory size and computing iterations linearly grow with the number of UEs and the total number of applications. Utilizing more parallel computing resources with efficient parallelization strategies (e.g., multi-thread processing for line 7 and GPU processing for line 15 in Algorithm 1) can mitigate this linear growth complexity.

B. Joint Decision Making of MARL Model

It is possible to explore and integrate with other AI/ML approaches for improving the policy decisions. For instance, the authors in [35] propose a federated RL (FDRL) approach to add one more control layer over the multiple DRL agents to manage network slice resources in a federated manner, maximizing the acceptance ratio of the service level agreement (SLA). In our work, multiple DRL agents aim to minimize the resource consumption and maximize the application performance by analyzing the RAN statistics collected individually. Adding the federation layer over our proposed MARL model to allow inter-agent information exchange can achieve higher performance generalization by utilizing the diverse learning experience updated by the individual replay buffer. This might help improve the performance and robustness against more dynamic and complex traffic scenarios.

C. Implications of Subcarrier Spacing and Functional Split in 5G NR System

When transitioning between bandwidth parts (BWPs) configured with different SCS values, the switching latency is governed by the timing requirements associated with the smaller SCS. According to the standard, two types of BWP switching delays are defined: Type 1 and Type 2. Type 1 typically corresponds to a duration of two slots for the majority of user equipments (UEs), whereas Type 2 is subject to less stringent delay constraints [36]. In scenarios where the BWP switch involves different SCS configurations (e.g., from 15 kHz to 30 kHz), the switching delay is computed based on the slot length of the smaller SCS. For instance, with a 15 kHz SCS (where each slot duration is 1 ms), the switch incurs a delay of 2 ms (i.e., two slots). This interval provides the UE with adequate time to reconfigure radio parameters, retune the local oscillator, and achieve synchronization with the new numerology prior to resuming data transmission or reception. The proposed vRAN orchestration mechanism operates on a 100 ms periodicity; thus, the relatively minor 2 ms BWP switching delay is insignificant and does not impact overall system performance.

With respect to the overhead incurred by switching between FS options, a previous study [6] developed a system based on srsRAN [11] to dynamically change the FS option at the base station and analyze the associated trade-offs between packet loss and user-perceived latency. Experimental results demonstrated that zero packet loss can be achieved during a migration period of up to 15 ms in a non-virtualized 5G NR environment. Furthermore, [37] reports that virtualization technologies, such as virtual machines or containers, can facilitate live migration of network functions across physical hosts.

D. Resource Characterization of vRAN

Despite the proposed DRL-based agent's superior performance over traditional, heuristic, and RL-based approaches (i.e., epsilon-greedy, DQN, and TRPO), the vRAN CPU model [7], [8] implemented in our SLS lacks realism, being derived from a laboratory-scale testbed. However, simulated experiments underscore the need for precise vRAN characterization in resource consumption and performance evaluation to facilitate open RAN utilizing AI/ML techniques.