Graph Neural PPO for Joint User Association and Resource Allocation in Open RAN

Kim-Hoan Do*, Tai-Hung Nguyen*✉, Quang-Trung Luu†, Minh-Thanh Nguyen*,
Do-Minh Tran*, and Van-Dinh Nguyen‡
*School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam
†Université Paris-Saclay, CNRS, CentraleSupélec, L2S, 91190, Gif-sur-Yvette, France
‡School of Computer Science and Statistics, Trinity College Dublin, Dublin 2 D02PN40, Ireland
✉Corresponding author. E-mail: hung.nguyentai@hust.edu.vn

***Abstract*—The rapid evolution toward beyond-5G and 6G networks demands intelligent orchestration of heterogeneous radio and computing resources to support ultra-reliable, high-throughput, and large-scale services. Open radio access network (Open RAN) and network slicing technologies enable flexible, software-driven operation through virtualization of RAN components. However, jointly optimizing user association and resource allocation across distributed radio, distributed, and central units (RU-DU-CU) under dynamic traffic and channel conditions remains a challenging mixed-integer problem. To address this, we propose a framework that integrates proximal policy optimization (PPO) with graph sample and aggregation (GraphSAGE)-based neural encoder. The proposed approach, called PPO-GraphSAGE, formulates the joint admission control and resource allocation task as a reinforcement learning model, where the GNN captures spatial and topological dependencies in the Open RAN, while PPO learns adaptive control policies in a sample-efficient and stable manner. Simulation results on a realistic Open RAN topology demonstrate that the proposed PPO-GraphSAGE agent significantly outperforms baseline greedy and heuristic schemes in terms of slice acceptance rate, total throughput, and reward efficiency, confirming its potential as a scalable and topology-aware solution for intelligent resource management in next-generation Open RAN systems.**

***Index Terms*—Deep reinforcement learning, Open RAN, resource allocation, graph neural networks, proximal policy optimization.**

I. Introduction

Network slicing is a key enabler of 5G and beyond networks, allowing a single physical infrastructure to be partitioned into multiple isolated virtual networks (slices) that can be independently managed and optimized [1], [2]. Each slice provides a tailored network environment to support diverse service categories such as enhanced mobile broadband (eMBB), ultra-reliable low-latency communication (uRLLC), and massive machine-type communication (mMTC). These service types impose distinct requirements in terms of bandwidth, latency, reliability, and computational resources [3], [4].

A central challenge in realizing network slicing lies in the embedding problem, which involves mapping virtual network functions (VNFs) and their interconnecting links onto physical resources. Unlike traditional hardware-based systems, VNFs are software-implemented, providing flexibility and scalability in deploying network functions. However, efficient embedding must satisfy multiple, often conflicting, constraints, such as limited computing and bandwidth resources, network topology, and service-level agreements (SLAs), making it a complex multi-dimensional optimization problem.

Recent advances in Open Radio Access Network (Open RAN) architectures further accentuate this complexity. Open RAN decomposes the conventional RAN into radio units (RUs), distributed units (DUs), and central units (CUs) connected in a hierarchical topology [5], [6]. Each layer provides different functionalities and exposes limited resources shared among coexisting network slices. A typical slice, therefore, spans multiple domains, deploying one VNF at each RU, DU, and CU, and requires coordinated radio and computational resource allocation along with bandwidth provisioning for RU–DU–CU links [7], [8]. Efficiently managing this multi-layer, distributed resource environment is crucial for achieving high utilization and guaranteed QoS.

Research on network slice resource management has evolved across several paradigms: classical optimization methods for VNF placement and virtual network embedding [9]–[11]; deep reinforcement learning (DRL) for adaptive admission control and resource allocation [12], [13]; and, more recently, graph-based learning for topology-aware decision-making [14], [15]. DRL algorithms such as deep Q network (DQN), deep deterministic policy gradient (DDPG), and proximal policy optimization (PPO) have shown strong potential for resource orchestration in dynamic wireless systems [16]–[18], while graph neural networks (GNNs) effectively capture spatial correlations and interference relationships in large-scale networks [19]–[21]. Emerging studies that integrate GNNs with DRL have demonstrated promising results for VNF placement and service-function-chain optimization [22].

Despite this progress, existing approaches typically address partial subproblems, focusing on admission control or radio resource block (RB) allocation in isolation, and often neglect the joint optimization across the full RU–DU–CU hierarchy. Moreover, prior reward designs are coarse and fail to capture the fine-grained trade-offs between admission efficiency, resource utilization, and QoS satisfaction.

To overcome these limitations, this work introduces a unified DRL framework that combines PPO with a GraphSAGE-based GNN encoder for joint user association and resource allocation in Open RAN. The proposed approach enables

adaptive policy learning that explicitly accounts for resource dynamics, network topology, and service heterogeneity. Our contributions can be summarized as follows:

- We design a comprehensive Open RAN model integrating radio, computational, and topological constraints across RU–DU–CU entities;
- We develop a DRL agent, called PPO–GraphSAGE, that jointly performs admission control, function placement, and discrete resource allocation using topology-aware graph representations;
- Through extensive simulations, we demonstrate that the proposed method achieves superior slice acceptance rate, throughput, and resource efficiency compared to baseline schemes, confirming its effectiveness for intelligent Open RAN management.

II. System Model

We consider an Open RAN consisting of three hierarchical functional entity classes RUs, DUs and CUs. Each RU attaches to one or more DUs, and each DU attaches to one or more CUs, forming a two-tier aggregation topology. Nodes expose finite radio and/or compute resources that are shared across coexisting network slices; available resources together with topology feasibility determine admissible VNF embeddings.

A. Physical Infrastructure

Let the physical infrastructure be represented by the triplet $\mathcal{N} = \big(\mathcal{R}, \mathcal{D}, \mathcal{C}\big)$, where $\mathcal{R}$, $\mathcal{D}$ and $\mathcal{C}$ are finite sets of RUs, DUs, and CUs, respectively. Indices $r \in \mathcal{R}$, $d \in \mathcal{D}$ and $c \in \mathcal{C}$ denote individual nodes.

Each RU r is provisioned with a maximum transmit power $P_r^{\max}$ and a maximum number of resource blocks (RBs) $M_r^{\mathrm{RB,max}}$. Each DU d has a computing capacity $C_d^{\max}$ (e.g., CPU cycles per unit time) and each CU c has computing capacity $U_c^{\max}$. Physical connectivity is captured by binary adjacency indicators $L_{r,d}^{\mathrm{RD}} \in \{0,1\}$ and $L_{d,c}^{\mathrm{DC}} \in \{0,1\}$, where $L_{r,d}^{\mathrm{RD}} = 1$ (resp. $L_{d,c}^{\mathrm{DC}} = 1$) if and only if a feasible RU–DU (resp. DU–CU) physical link exists. The large-scale/channel gain between RU r and the user(s) associated with slice request k is denoted by $g_{r,k}$. The discrete set of admissible total transmit-power levels is denoted by $\mathcal{P}$. The bandwidth of a single RB is B_{RB} (Hz) and the noise power per RB is denoted by σ^2 (W).

B. Slice Requests

Slice requests arrive dynamically and are queued for admission decisions. Here, arrivals are modeled as a Poisson process with rate λ, though the framework is agnostic to the specific arrival law. Let $\mathcal{K}(t)$ denote the set of pending requests at decision epoch t and $K(t) = |\mathcal{K}(t)|$. An arrived slice request $k \in \mathcal{K}(t)$ is described by the tuple $q_k = \big(R_k^{\mathrm{req}}, D_k^{\mathrm{DU}}, D_k^{\mathrm{CU}}\big)$, where R_k^{req} is the requested user-plane throughput (bps), D_k^{DU} and D_k^{CU} are DU and CU compute demands (CPU cycles per unit time). A request k is admissible only if there exists at least one triplet $(r,d,c) \in \mathcal{R} \times \mathcal{D} \times \mathcal{C}$ such that: (i) $L_{r,d}^{\mathrm{RD}} = 1$ and $L_{d,c}^{\mathrm{DC}} = 1$ (topology feasibility), (ii) an allocation of RBs and transmit power yields an achieved throughput $\geq R_k^{\mathrm{req}}$, and (iii) DU and CU compute demands can be satisfied by residual capacities.

C. Link-level Throughput Model

Consider an allocation $(n_{k,r}, p_{k,r})$ serving request k at RU r. The per-RB transmit power is defined as

$$\rho_{k,r} \;=\; \begin{cases} \dfrac{p_{k,r}}{n_{k,r}}, & n_{k,r} > 0, \\ 0, & n_{k,r} = 0 \end{cases} \tag{1}$$

where $p_{k,r}$ denotes the total transmit power assigned to request k at RU r, and $n_{k,r}$ is the number of allocated RBs. The resulting per-RB signal-to-noise ratio (SNR) is expressed as

$$\mathrm{SNR}_{r,k} \;=\; \frac{\rho_{k,r}\, g_{r,k}}{\sigma^2} \tag{2}$$

with $g_{r,k}$ representing the channel gain and σ^2 denoting the noise power. The achievable throughput is then given by

$$R_{k,r}^{\mathrm{ach}} \;=\; n_{k,r}\, B_{\mathrm{RB}} \log_2\big(1 + \mathrm{SNR}_{r,k}\big) \tag{3}$$

where B_{RB} is the bandwidth per resource block.

III. Problem Formulation

We introduce binary assignment variables

$$x_{k,r,d,c} = \begin{cases} 1, & \text{request } k \text{ is mapped to } (r,d,c), \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

with $x_{k,r,d,c} \in \{0,1\}$. For a request k admitted at RU r, we denote by $n_{k,r} \in \mathbb{Z}_{\geq 0}$ the number of RBs allocated and by $p_{k,r} \in \mathcal{P}$ the total transmit power allocated to that request at RU r. The per-RB transmit power for (k,r) is defined shortly. Unless ambiguity arises, we omit the explicit dependence of $n_{k,r}$ and $p_{k,r}$ on the chosen DU/CU.

The orchestrator aims to admit and embed requests so as to maximize slice admission while using radio and compute resources efficiently. We formulate a weighted objective that trades off admitted requests and achieved throughput:

$$\max \sum_{k \in \mathcal{K}(t)} \sum_{r \in \mathcal{R}} \sum_{d \in \mathcal{D}} \sum_{c \in \mathcal{C}} \Big(\alpha\, x_{k,r,d,c} \;+\; \beta\, \frac{R_{k,r}^{\mathrm{ach}}}{R_k^{\mathrm{req}}} x_{k,r,d,c} \Big) \tag{5}$$

where $\alpha > 0$ and $\beta \geq 0$ are tunable weights (setting $\beta = 0$ reduces the objective to pure admission maximization). Below we list the constraints that must hold for all relevant indices.

Total transmit power at RU r cannot exceed its installed power budget:

$$\sum_{k \in \mathcal{K}(t)} \sum_{d \in \mathcal{D}} \sum_{c \in \mathcal{C}} p_{k,r}\, x_{k,r,d,c} \;\leq\; P_r^{\max}, \qquad \forall r \in \mathcal{R}. \tag{6}$$

The number of RBs assigned at RU r is bounded by its RB capacity:

$$\sum_{k \in \mathcal{K}(t)} \sum_{d \in \mathcal{D}} \sum_{c \in \mathcal{C}} n_{k,r}\, x_{k,r,d,c} \;\leq\; M_r^{\mathrm{RB,max}}, \quad \forall r \in \mathcal{R}. \tag{7}$$

The cumulative DU processing demand of slices mapped to DU d must not exceed $C_d^{\max}$:

$$\sum_{k\in\mathcal{K}(t)}\sum_{r\in\mathcal{R}}\sum_{c\in\mathcal{C}} D_k^{\mathrm{DU}}\, x_{k,r,d,c} \;\le\; C_d^{\max}, \qquad \forall d\in\mathcal{D} \tag{8}$$

and the CU processing demand obeys

$$\sum_{k\in\mathcal{K}(t)}\sum_{r\in\mathcal{R}}\sum_{d\in\mathcal{D}} D_k^{\mathrm{CU}}\, x_{k,r,d,c} \;\le\; U_c^{\max}, \qquad \forall c\in\mathcal{C}. \tag{9}$$

A mapping is valid only if exist underlying physical links:

$$x_{k,r,d,c} \le L_{r,d}^{\mathrm{RD}}; \quad x_{k,r,d,c} \le L_{d,c}^{\mathrm{DC}}, \qquad \forall k,r,d,c. \tag{10}$$

Each request is mapped to at most one RU–DU–CU triplet, and assignment variables are binary:

$$\sum_{r\in\mathcal{R}}\sum_{d\in\mathcal{D}}\sum_{c\in\mathcal{C}} x_{k,r,d,c} \le 1, \qquad \forall k\in\mathcal{K}(t). \tag{11}$$

For any admitted mapping the achieved throughput must satisfy the request's minimum:

$$x_{k,r,d,c}\, R_{k,r}^{\mathrm{ach}} \;\ge\; x_{k,r,d,c}\, R_k^{\mathrm{req}}, \qquad \forall k,r,d,c. \tag{12}$$

The optimization problem defined by (5)–(12) is a mixed-integer, combinatorial program. Its feasible set and objective scale combinatorially with the number of pending requests and physical nodes, rendering exact solution methods impractical for large instances and motivating scalable heuristics and learning-based policies (see Sec. IV).

IV. Reinforcement Learning Approach

To obtain scalable and adaptive policies for joint admission control and resource embedding under stochastic slice arrivals and time-varying wireless channels, we first derive a reinforcement learning model. A model-free DRL is then employed as a scalable algorithm capable of learning effective policies without explicit knowledge of system dynamics or traffic statistics.

A. Reinforcement Learning Model

In our RL framework, decisions are made at discrete epochs indexed by $t = 0, 1, 2, \ldots$ At each epoch, the environment is represented by a state s_t summarizing the residual resources, active slice requests, and channel conditions. Based on this state, the agent selects an action a_t that specifies whether to admit a given request and how to allocate radio and computational resources across RU, DU, and CU nodes. The environment then returns a scalar reward r_t, reflecting both the quality of service delivered to the admitted slices and the efficiency of resource utilization, and transitions to a new state s_{t+1} according to the underlying system dynamics.

The learning task is to determine a stochastic policy $\pi_\theta(a|s)$, with parameter θ, maximizes the expected discounted return

$$J(\pi_\theta) \;=\; \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{\infty}\gamma^t r_t\Big], \tag{13}$$

where $\gamma\in(0,1]$ is the discount factor controlling the trade-off between immediate and future rewards. A larger γ emphasizes long-term performance, while a smaller value encourages short-sighted decisions prioritizing short-term gains.

States. At each decision epoch t, the agent observes a structured state that jointly captures residual resources, traffic context, and network conditions:

$$\begin{aligned} s_t = \big(&\mathbf{P}^{\mathrm{res}}(t),\ \mathbf{C}^{\mathrm{DU,res}}(t),\ \mathbf{U}^{\mathrm{CU,res}}(t),\ \mathbf{M}^{\mathrm{RB,res}}(t),\\ &\mathcal{Q}^{'}(t),\ L_{r,d}^{\mathrm{RD}},\ L_{d,c}^{\mathrm{DC}},\ \mathbf{G}(t),\ \mathcal{P},\ M_{\max}^{\mathrm{RB}}\big). \end{aligned} \tag{14}$$

The first group encodes available resources across RUs, DUs, and CUs, together with the remaining RB budgets. The second group describes the set of pending service requests,

$$\mathcal{Q}^{'}(t) = \{q_k^{'}\}_{k=1}^{K(t)}, \qquad q_k^{'} = (R_k^{\mathrm{req}},\ D_k^{\mathrm{DU}},\ D_k^{\mathrm{CU}},\ a_k), \tag{15}$$

where a_k indicates request activity. The third group reflects structural and physical information, including topology (RU–DU and DU–CU adjacencies), instantaneous channel gains, admissible power levels, and per-request RB limits.

This compact yet expressive representation ensures that both feasibility constraints and performance trade-offs are explicitly available to the policy, enabling informed admission and embedding decisions without redundant features.

Action. The agent issues an action that jointly specifies admission and resource embedding for a single decision (one request per epoch, or a batched set of decisions in extended formulations). We consider the per-epoch action

$$a_t = \big(k_t,\ \delta_t,\ r^\star,\ d^\star,\ c^\star,\ n_{k_t,r^\star},\ p_{k_t,r^\star}\big) \tag{16}$$

where $k_t \in \mathcal{K}(t)$ selects a pending request, $\delta_t \in \{0,1\}$ is the admission indicator (0 = reject, 1 = accept), $(r^\star, d^\star, c^\star)$ denotes the chosen RU/DU/CU triplet, $n_{k_t,r^\star}$ is the number of allocated RBs (integer, $0 \le n_{k_t,r^\star} \le M_{\max}^{\mathrm{RB}}$), and $p_{k_t,r^\star} \in \mathcal{P}$ is the total transmit power. During action sampling infeasible choices are masked using pre-computed feasibility checks derived from constraints (6)–(12).

Reward. For a feasible allocation of request k at RU r, the normalized throughput is defined as

$$\theta_{k,r} = R_{k,r}^{\mathrm{ach}} / R_k^{\mathrm{req}}, \tag{17}$$

which captures the degree to which the demand is satisfied relative to the target requirement. To reduce wasteful resource usage, we introduce the following penalty

$$\phi(\theta) \;=\; \eta\,(\theta-1)^+, \qquad (x)^+ = \max\{0,x\} \tag{18}$$

with $\eta > 0$ controlling the severity of the penalty.

The instantaneous reward r_t associated with the agent's decision on request k_t is then given by

$$r_t = \begin{cases} r_{\mathrm{rej}}, & \delta_t = 0,\\ r_{\mathrm{inf}}, & \delta_t = 1 \wedge \text{infeasible},\\ r_{\mathrm{acc}} + \theta_{k_t,r^\star} - \phi(\theta_{k_t,r^\star}), & \delta_t = 1 \wedge \text{feasible} \end{cases} \tag{19}$$

where $r_{\mathrm{rej}} < 0$ is a mild penalty for rejection, $r_{\mathrm{inf}} \ll 0$ is a strong penalty for admitting an infeasible allocation, and $r_{\mathrm{acc}} > 0$ is a positive credit for feasible admission. To prevent instability due to outliers, reward values may be clipped below a chosen floor.

B. Proximal Policy Optimization (PPO)

PPO is a widely used RL algorithm that helps agents learn how to make better decisions through trial and error [23]. PPO belongs to the family of policy gradient methods, which directly improve the agent's strategy for choosing actions. A key feature of PPO is its clipped objective function, which stabilizes learning by limiting how much the policy can change during each update. This function is defined as:

$$\mathcal{L}_{\text{clip}} = \mathbb{E}\big[\min(r_t(\theta)\cdot A, \text{clip}(r_t(\theta), 1\pm\epsilon)\cdot A)\big] \tag{20}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the estimated advantage, and ϵ is a small constant that defines the clipping range. This formulation helps PPO maintain stable and conservative updates, making it a robust choice for training reinforcement learning agents.

C. Graph Sample and Aggregation

Graph sample and aggregation (GraphSAGE), proposed in [24], is a method for learning node representations in graphs. It supports inductive learning, allowing embeddings for new nodes not seen during training. This is useful for large and changing graphs like social networks. GraphSAGE samples a fixed number of neighbors and aggregates their features to represent each node.

The main idea of GraphSAGE is to combine a node's features with its neighbors' using an aggregation function across layers. The embedding of node v at layer k is:

$$h_v^{(k)} = \sigma\left(W^{(k)}\mathcal{A}^{(k)}(\{h_u^{(k-1)} : u\in\mathcal{N}(v)\}\cup\{h_v^{(k-1)}\})\right) \tag{21}$$

where $h_v^{(k)}$ is the node embedding, $\mathcal{N}(v)$ is the set of neighbors, $W^{(k)}$ is a learnable weight matrix, σ is a non-linear function, and $\mathcal{A}^{(k)}$ is the aggregation function.

D. Learning Architecture

The proposed PPO–GraphSAGE algorithm uses an actor-critic framework trained with PPO integrated with GraphSAGE, as described in Algorithm 1. Its policy network includes a structured encoder that captures the spatial layout and resource status of the Open RAN. A graph neural network encodes the RU/DU/CU hierarchy and node-level capacities to create resource-aware embeddings. At the same time, a lightweight multilayer perceptron encodes slice request features such as throughput demand, DU/CU computation needs, and holding time. These embeddings are combined and used by multiple output heads to produce logits for four decisions: request selection, admission, RU/DU/CU assignment, and resource allocation (RBs and transmit power).

The algorithm follows these steps. In each episode, the agent selects a random action a and performs it in the environment $\mathcal{E}$ (Line 6). After that, the agent saves the outcome of the action in the buffer $\mathcal{D}$ (Line 7). If the buffer becomes full or the environment reaches a terminal state, the agent starts optimizing the neural network.

To perform optimization, the agent first computes the values of A and R (Line 9). Then, for each minibatch in the buffer, the agent calculates the loss for the neural network. To support stable and effective learning, the algorithm uses several methods, such as normalized advantage estimation, entropy regularization with annealing, and gradient clipping. The clipped surrogate loss $\mathcal{L}_{\text{clip}}$ helps stabilize policy updates by preventing large changes in the policy ratio. The value loss $\mathcal{L}_V$ reduces errors in predicting the value function (Line 11). After completing the optimization, the agent updates the neural network and finishes the episode (Line 12).

Algorithm 1 Our Proposed PPO-GraphSAGE Algorithm.

1: **Input:** Environment $\mathcal{E}$, policy π_θ, value V_ϕ, discount γ, GAE λ, clip ϵ, steps T, epochs K;
2: **Initialize:** Parameters (θ,ϕ), optimizer, entropy coefficient β, and buffer $\mathcal{D}\leftarrow\varnothing$;
3: **for** each episode **do**
4: Reset environment: $s\leftarrow\text{Reset}(\mathcal{E})$;
5: **while** not done **do**
6: Sample $a\sim\pi_\theta(\cdot|s)$ and execute in $\mathcal{E}\Rightarrow(s',r,\text{done})$;
7: Store $(s,a,r,\log\pi_\theta(a|s),V_\phi(s))$ in $\mathcal{D}$; set $s\leftarrow s'$;
8: **if** $|\mathcal{D}|\geq T$ or done **then**
9: Compute: $A=\text{GAE}(r,V_\phi,\gamma,\lambda)$, $R=A+V_\phi(s)$;
10: **for** K epochs over minibatches **do**
11: Recompute $\log\pi_\theta$ and V_ϕ, then evaluate losses: $\mathcal{L}_{\text{clip}}=\mathbb{E}\big[\min(r_t(\theta)\cdot A,\text{clip}(r_t(\theta),1\pm\epsilon)\cdot A)\big]$, $\mathcal{L}_V=\mathbb{E}\big[(R-V_\phi)^2\big]$;
12: Update (θ,ϕ) by maximizing: $\mathcal{L}_{\text{clip}}+\beta\,\mathcal{H}[\pi_\theta]-c_1\,\mathcal{L}_V$;
13: **end for**
14: Clear $\mathcal{D}$; anneal β if required;
15: **end if**
16: **end while**
17: Store metrics: reward, acceptance, throughput;
18: **end for**
19: **return** π_θ^*.

V. Performance Evaluation

A. Setup

The proposed reinforcement learning framework is implemented in a custom-built simulator that emulates the Open RAN system model described in Sec. II. Slice requests arrive according to a Poisson process with rate λ, and each request specifies a throughput demand together with DU and CU compute requirements.

Wireless propagation is modeled by combining large-scale path loss and small-scale fading. The path loss follows the 3GPP urban macrocell model, with reference attenuation of 128.1 dB at 1 km and a distance-dependent exponent of 37.6. Each RU is equipped with an array of eight antennas, and the small-scale fading between RU–UE pairs is represented by independent Rayleigh fading coefficients. The effective channel gain is obtained by superimposing fading and path loss, and the received signal-to-noise ratio (SNR) is calculated per resource block after incorporating noise power.

A total of 30 UEs are randomly deployed within a circular service area of 2 km diameter, while RUs are evenly placed within the same region. Each UE is associated with its nearest

RU. The deployment of UEs and RUs, together with nearest-RU association.

The agent is trained using PPO-GraphSAGE over 5000 decision episodes with discount factor $\gamma = 0.99$, clipping parameter $\epsilon = 0.2$, and learning rate 1×10^{-4}. Performance is benchmarked against three baseline strategies: a random allocation, a greedy admission policy, and a throughput-to-resource heuristic.

Table I summarizes the main simulation parameters.

TABLE I: Simulation parameters.

Parameter	Value
Nb. of RUs/DUs/CUs $(\lvert\mathcal{R}\rvert, \lvert\mathcal{D}\rvert, \lvert\mathcal{C}\rvert)$	$(5, 3, 3)$
Nb. of slices/RBs $(\lvert\mathcal{K}\rvert, \lvert\mathcal{B}\rvert)$	$(30, 137)$
RB's bandwidth (B_{RB})	360 kHz
RU's power budget $(P_r^{\max})$	43 dBm
CU's/DU's capacity $(C_d^{\max}, U_c^{\max})$	$(100, 100)$ units
Nb. of training episodes	5000

The training and evaluation are conducted on a PC equipped with an Intel® Core™ i5-8565U CPU (1.80 GHz, 4 cores/8 threads), 16 GB RAM, and running Ubuntu 20.04 LTS.

B. Results

In this section, we evaluate the performance of the proposed PPO–GraphSAGE user-to-RU mapping scheme and compare it with several baseline approaches, including Greedy, Round Robin, and Random policies. The evaluation metrics include acceptance rate, throughput, and average reward value.

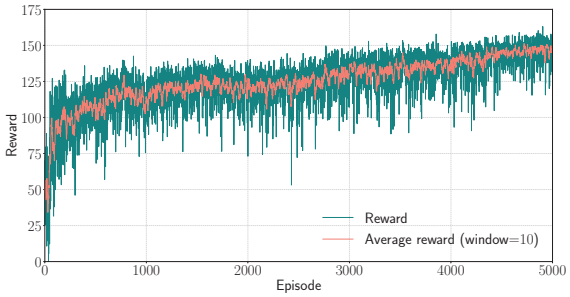
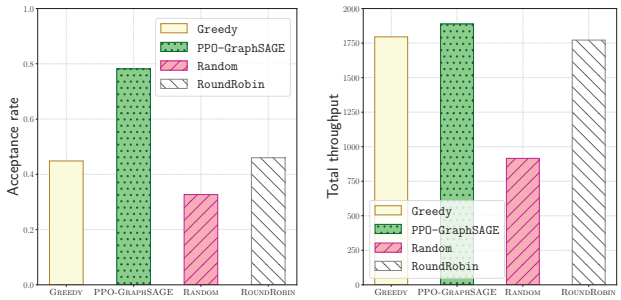


Fig. 1: Reward evolution during PPO-GraphSAGE training.

Figure 1 illustrates the training process of the PPO–GraphSAGE agent. The reward curve exhibits a consistent upward trend as the number of episodes increases, and the moving average becomes smoother over time, indicating that the agent is gradually learning an effective policy scenarios.

Although the reward has not fully converged within the reported training horizon, the trajectory suggests that further improvement is achievable if the training is extended with a larger number of episodes. This behavior is consistent with the on-policy nature of PPO–GraphSAGE, where stable convergence typically requires longer training. From a practical perspective, this implies that the proposed framework has the potential to achieve even higher efficiency and robustness when applied to larger-scale deployments or more demanding traffic.



(a) Acceptance rate. (b) Total throughput.

Fig. 2: Compared performance of PPO–GraphSAGE with benchmark schemes (Greedy, Random, and RoundRobin), in terms of (a) acceptance rate and (b) total throughput.

Fig. 3: Average reward value of the compared algorithms.

Figure 2 provides a performance comparison in terms of acceptance rate and total throughput. As shown in Fig. 2a, PPO–GraphSAGE achieves the highest acceptance rate among all schemes, significantly outperforming the baseline methods. Similarly, Fig. 2b demonstrates that PPO–GraphSAGE attains the best throughput performance. These results confirm that PPO–GraphSAGE enables more efficient utilization of radio resources.

Finally, Fig. 3 illustrates the average reward values of different schemes. The PPO–GraphSAGE approach achieves the highest reward, whereas the baseline algorithms yield lower values. This outcome further validates that PPO–GraphSAGE, when integrated with the GNN-based model, is able to capture the complex user-to-RU mapping relations and provide superior decision-making compared with traditional heuristic approaches.

Overall, the simulation results demonstrate that the proposed PPO–GraphSAGE method achieves consistent and significant performance gains in terms of acceptance rate, throughput, and reward, confirming its effectiveness for intelligent resource allocation in Open RAN scenarios.

VI. Conclusion

This paper investigates the problem of slice admission control and embedding in Open RAN. We first formulate this problem as a mixed-integer combinatorial optimization model. To obtain a practical solution, we design a reinforcement learning framework operating on a layered RU/DU/CU architecture, which explicitly accounts for radio and compute budgets, RU/DU/CU connectivity, available resource blocks, and per-UE QoS requirements. Subsequently, we propose PPO–GraphSAGE, a DRL agent trained using PPO and equipped with a GraphSAGE-based encoder to capture network topology and residual capacities, complemented by a lightweight request encoder.

Extensive simulations have demonstrated that the proposed PPO–GraphSAGE algorithm consistently outperforms Random, Greedy, and Round Robin baselines across loads in acceptance rate, aggregate throughput, and average reward the gains of jointly learning admission and placement under dynamic traffic.

For future work, we will extend the framework to multi-slice scenarios and richer wireless conditions (shadowing, mobility), integrate an explicit end-to-end latency model, and benchmark against an ILP upper bound to quantify optimality gaps. We will also explore distributed/multi-agent training aligned with practical Open RAN deployments and transfer across network topologies and traffic regimes.

VII. Acknowledgment

This research is funded by the Australia-Vietnam Strategic Technologies Centre under the seed grant agreement for the project entitled "Building a Platform to Enable Future Research on Optimization Methods for Resource Allocation in Network Slicing for 5G/6G Networks."

References

[1] Q.-T. Luu, S. Kerboeuf, and M. Kieffer, "Foresighted resource provisioning for network slicing," in *Proc. IEEE International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1–8.

[2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 2, pp. 1376–1411, 2023.

[3] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource allocation for network slicing in 5G telecommunication networks: A survey of principles and models," *IEEE Network*, vol. 33, no. 6, pp. 172–179, 2019.

[4] Q.-T. Luu, S. Kerboeuf, and M. Kieffer, "Admission control and resource reservation for prioritized slice requests with guaranteed SLA under uncertainties," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 3136–3153, 2022.

[5] F. Kavehmadavani, V.-D. Nguyen, T. X. Vu, and S. Chatzinotas, "Intelligent traffic steering in beyond 5G open RAN based on LSTM traffic prediction," *IEEE Trans. Wire. Commun.*, vol. 22, no. 11, pp. 7727–7742, 2023.

[6] V.-D. Nguyen, T. X. Vu, N. T. Nguyen, D. C. Nguyen, M. Juntti, N. C. Luong, D. T. Hoang, D. N. Nguyen, and S. Chatzinotas, "Network-Aided Intelligent Traffic Steering in 6G O-RAN: A Multi-Layer Optimization Framework," *IEEE J. Select. Areas Commun.*, vol. 42, no. 2, pp. 389–405, 2024.

[7] F. Kavehmadavani, V.-D. Nguyen, T. X. Vu, and S. Chatzinotas, "Empowering Traffic Steering in 6G Open RAN With Deep Reinforcement Learning," *IEEE Trans. Wire. Commun.*, vol. 23, no. 10, pp. 12 782–12 798, 2024.

[8] F. Kavehmadavani, T. X. Vu, V.-D. Nguyen, and S. Chatzinotas, "Intelligent User Association and Scheduling in Open RAN: A Hierarchical Optimization Framework," *IEEE Trans. Commun.*, pp. 1–1, 2025.

[9] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 240–252, 2016.

[10] Q.-T. Luu, M. Kieffer, A. Mouradian, and S. Kerboeuf, "Aggregated resource provisioning for network slices," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[11] E. Author and F. Author, "Network function placement in virtualized radio access network with reinforcement learning based on graph neural network," *Electronics (MDPI)*, 2024, replace DOI/URL.

[12] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[13] K. Suh, S. Kim, Y. Ahn, S. Kim, H. Ju, and B. Shim, "Deep reinforcement learning-based network slicing for beyond 5G," *IEEE Access*, vol. 10, pp. 7384–7395, 2022.

[14] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, 2017.

[15] J. A. H. Sánchez, K. Casilimas, and O. M. C. Rendón, "Deep reinforcement learning for resource management on network slicing: A survey," *Sensors*, vol. 22, no. 8, p. 3031, 2022.

[16] S. Jo, U. Kim, J. Kim, C. Jong, and C. Pak, "Deep reinforcement learning-based joint optimization of computation offloading and resource allocation in F-RAN," *IET Communications*, vol. 17, pp. 549–564, 2023.

[17] T. Mai, H. Yao, N. Zhang, W. He, D. Guo, and M. Guizani, "Transfer reinforcement learning aided distributed network slicing optimization in industrial IoT," *IEEE Trans. Ind. Inform.*, vol. 18, no. 6, pp. 4308–4316, 2021.

[18] M.-T. Nguyen, Q.-T. Luu, T.-H. Nguyen, D.-M. Tran, T.-A. Do, K.-H. Do, and V.-H. Nguyen, "Accelerating network slice embedding with reinforcement learning," in *Proc. IEEE International Conference on Communications and Electronics (ICCE)*, 2024.

[19] M. Eisen and A. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 2977–2991, 2020.

[20] Z. Wang, M. Eisen, and A. Ribeiro, "Learning decentralized wireless resource allocations with graph neural networks," *IEEE Trans. Signal Process.*, vol. 70, pp. 1850–1863, 2022.

[21] P. Tam, S. Ros, I. Song, S. Kang, and S. Kim, "A survey of intelligent end-to-end networking solutions: Integrating graph neural networks and deep reinforcement learning approaches," *Electronics*, vol. 13, p. 994, 2024.

[22] S. Ros, P. Tam, I. Song, S. Kang, and S. Kim, "Handling efficient VNF Placement with Graph-Based reinforcement learning for SFC Fault Tolerance," *Electronics*, vol. 13, no. 13, p. 2552, 2024.

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

[24] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.