E-Matro: An Adaptive and Energy-Efficient Framework for Slicing within the O-RAN Architecture

André Cédric Bessala, Guilherme Iecker Ricardo and Gentian Jakllari
IRIT/Toulouse INP-ENSEEIHT, University of Toulouse, France
Email: {andrecedric.bessala, guilherme.ieckerricardo, gentian.jakllari}@toulouse-inp.fr

***Abstract*—The complexity of Open RAN (O-RAN) networks, driven by dynamic traffic and stringent Quality of Service (QoS) requirements, challenges traditional centralized control, leading to suboptimal resource allocation and high energy consumption. We address the problem of energy-efficient, slice-compliant Virtual Network Function (VNF) placement by formulating it as a flow-based optimization model. To solve it, we propose E-Matro, a multi-agent reinforcement learning (MARL) framework that integrates decentralized dApps with a centralized xApp for adaptive and scalable control. Experiments on simulated O-Cloud networks show that E-Matro efficiently learns optimal policies, minimizing energy consumption while ensuring QoS compliance. Among tested RL approaches, Deep Q-Learning demonstrates superior adaptability in complex scenarios, highlighting the potential of hybrid MARL-based control for O-RAN.**

I. Introduction

The telecommunications landscape is evolving rapidly due to the increasing demand for mobile data and the anticipated need for ubiquitous connectivity by 2030 [1]. Fifth-generation (5G) and beyond 5G (B5G) networks aim to address key challenges in wireless communication, including diverse service requirements, large-scale device connectivity, and stringent performance constraints [1], [2].

Network slicing has emerged as a fundamental technique in this evolution, enabling a shift from traditional monolithic network architectures to more flexible and programmable service delivery [3]. This approach allows for the dynamic segmentation of network resources into virtualized and customized network instances, configured to meet the specific requirements of different applications and service providers [3]. By isolating and allocating network capabilities, operators can simultaneously support various communication technologies, including ultra-reliable low-latency communication (URLLC), enhanced mobile broadband (eMBB), and massive machine-type communication (mMTC), while improving resource utilization and operational efficiency [3], [4].

This modular service delivery model aligns with the disaggregation of Radio Access Networks (RAN). A key example is the Open RAN (O-RAN) architecture, developed by the O-RAN Alliance to promote openness and interoperability. As illustrated in Fig.1, O-RAN segments the RAN into three main functions [5]: the Central Unit (CU), the Distributed

ISBN 978-3-903176-72-0 © 2025 IFIP

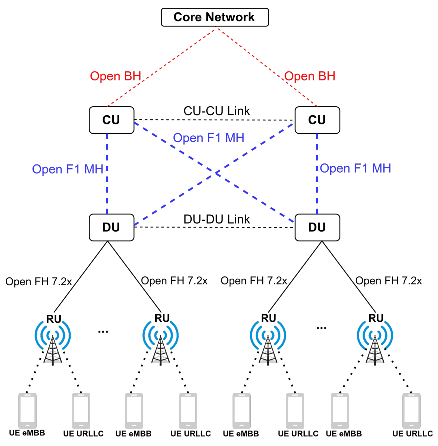


Fig. 1: Example of O-RAN-based Architecture.

Unit (DU), and the Radio Unit (RU), all of which can be deployed on open hardware and edge-cloud infrastructure as Virtual Network Functions (VNFs) [6]. This architecture leverages edge computing for distributed VNF placement, reducing network latency and improving overall performance. Deploying DUs and CUs within the transport network (O-Cloud) enhances RAN slice performance; however, it also introduces a complex resource allocation challenge across RUs, DUs, and CUs, requiring adaptive allocation strategies to balance slice demands and energy efficiency [7], [8].

Recent research has primarily focused on centralized DU/CU control strategies, often implemented via RAN Intelligent Controllers (RICs). For instance, [7] applies deep reinforcement learning (DRL) to optimize joint user association and CU-DU placement through a centralized agent. Similarly, [9] addresses vRAN reconfiguration by orchestrating base station (BS) functional splits with DRL, while [10] explores a DRL-based approach for admission control and VNF placement. However, centralized O-RAN solutions face limitations in dynamic environments where traffic conditions fluctuate. Their lack of adaptability can lead to suboptimal network control and resource allocation.

To mitigate these limitations, distributed applications (dApps) and centralized applications (xApps) provide complementary network control approaches [11]. dApps operate at the O-Cloud edge, enabling localized, low-latency decision-making in response to traffic variations, while xApps run on centralized controllers, leveraging global network insights to assist dApps. A hybrid approach integrating dApps and xApps can enhance system responsiveness by balancing local adaptability with global coordination, allowing real-time traffic-aware adjustments while maintaining network-wide optimization [11].

In this paper, we formally define the problem of energy-efficient, slice-compliant VNF placement and propose a theoretical framework for the design of E-MATRO. E-MATRO is the first O-RAN-based framework that incorporates NF-to-NF traffic steering and function migration to adapt to traffic fluctuations while minimizing energy consumption. It consists of two key components: the *Control xApp* and the *Agent dApp*, which operate in synergy to ensure both slice QoS compliance and energy-efficient VNF placement.

To summarize, our key contributions are as follows:

1) We formalize the problem of minimizing energy consumption while satisfying QoS constraints and reformulate it as an equivalent flow-based optimization problem (§ III).
2) We propose E-MATRO, a multi-agent RL framework that addresses the problem using a hybrid approach combining decentralized dApps and a centralized xApp (§ IV).
3) We demonstrate how E-MATRO can be adapted to different RL techniques, including classical and Deep Q-Learning. Given the distributed nature of the framework, we also introduce a heuristic for resolving action conflicts (§ IV-B).
4) We conduct experiments on O-Cloud networks of varying sizes to evaluate the framework's learning and scalability capabilities under different RL techniques. Results show that E-MATRO successfull learns the optimal policy and converges to an energy-efficient setup, with Deep Q-Learning exhibiting superior adaptability in complex scenarios (§ V).

II. SYSTEM MODEL

Consider the O-RAN architecture (e.g., Figure 1) as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where vertices $\mathcal{V}$ represent *deployment nodes* and arcs $\mathcal{L}$ represent physical links between them. The vertex set is partitioned into two disjoint sets: RUs $\mathcal{V}^{\text{RU}}$ and O-Cloud nodes $\mathcal{V}^{\text{Cloud}}$. Each deployment node has computing capabilities and a core activity. RUs provide wireless access to UEs, serving as traffic entry points[1]. O-Cloud nodes operate either as a (i) *Virtual Infrastructure Manager* (VIM), managing VNFs, or as a (ii) *relay*, forwarding 5G packets.

There is a set of slices $\mathcal{S}$, where each slice $s \in \mathcal{S}$ specifies QoS requirements in terms of (i) average achievable throughput and (ii) uplink (one-way) delay. Each slice s has an end-to-end average throughput Λ_s (in bps). The total throughput for UEs of slice s associated with RU r is

$$\lambda_{r,s} = \rho_{r,s} \cdot \Lambda_s, \tag{1}$$

[1]For simplicity, we consider only uplink traffic (UE to core network). The proposed approach can be adapted for downlink communication.

where $\rho_{r,s}$ is the number of UEs connected to RU r and subscribing to slice s. Additionally, each slice s imposes a delay tolerance δ_s in the transport network.

Virtualization: Each O-Cloud node supports the virtualization of DU and CU functions, so that it can run one or more instances of that function, each instance associated with a single RU. In this case, the node will process at the DU and/or CU levels the packets produced at their associated RU. We define binary variables

$$x_{r,i}^{\text{VNF}} \in \{0,1\}, \forall r \in \mathcal{V}^{\text{RU}}, \forall i \in \mathcal{V}^{\text{Cloud}}, \text{VNF} \in \{\text{DU, CU}\}, \tag{2}$$

to indicate whether node i hosts RU r's VNF instance (i.e., $x_{r,i}^{\text{VNF}} = 1$) or not (i.e., $x_{r,i}^{\text{VNF}} = 0$).

Each instance of a given VNF processes ω^{VNF} operations per bit of received packets, on average. The total computing workload to virtualize functions in node i is a function of the hosted instances

$$W_i^{\text{V}}(\mathbf{x}) = \sum_{r \in \mathcal{V}^{\text{RU}}} \sum_{s \in \mathcal{S}} \lambda_{r,s} \cdot \left(\omega^{\text{CU}} \cdot x_{r,i}^{\text{CU}} + \omega^{\text{DU}} \cdot x_{r,i}^{\text{DU}}\right), \tag{3}$$

where $\mathbf{x} = (\mathbf{x}^{\text{DU}}, \mathbf{x}^{\text{CU}})$ is the matrix of DU and CU instances. The number of VNF instances in a given O-Cloud node i is bounded by its computing capacity μ_i^{Comp}, in operations per second.

Networking: Each RU must be connected to its corresponding DU and, in turn, the RU's DU must be connected to the RU's CU via a virtual channel between each pair of functions. The virtual channel between RU and DU is called the *fronthaul*, which is implicitly represented by variables $\mathbf{x}^{\text{DU}}$, such that $x_{r,i}^{\text{DU}} = 1$ means that there is a virtual channel between RU r and node i. The virtual channel between DU and CU is called the *midhaul*. It can be represented by the product $x_{r,i}^{\text{DU}} \cdot x_{r,j}^{\text{CU}} = 1$ to indicate that there is a virtual channel between nodes i and j with respect to RU r.

In reality, fronthaul and midhaul channels have each an associated path of physical links in the underlying O-RAN infrastructure. We denote by

$$y_{r,s,\ell} \in \{0,1\}, \forall r \in \mathcal{V}^{\text{RU}}, \forall \ell \in \mathcal{L} \tag{4}$$

the variables indicating whether physical link $\ell \in \mathcal{L}$ is used to build RU r's virtual channels for slice s' traffic (i.e., $y_{r,s,\ell} = 1$) or not (i.e., $y_{r,s,\ell} = 0$). This means that, if there is a virtual channel connecting nodes i and j, then there should be a path between i and j in $\mathcal{G}$. The utilized capacity of link ℓ by UEs from slice s at RU r is represented by

$$L_\ell(\mathbf{y}) = \sum_{r \in \mathcal{V}} \sum_{s \in \mathcal{S}} \lambda_{r,s} \cdot y_{r,s,\ell}, \tag{5}$$

where $\mathbf{y}$ is the matrix indicating the association of links to RUs/slices. The amount of traffic going through link ℓ is bounded by its link capacity μ_ℓ^{Link}, in bits per second. The total computing workload to forward packets in node i is a function of the total volume of incoming traffic at i, i.e.,

$$W_i^{\text{F}}(\mathbf{y}) = \omega^{\text{GF}} \sum_{j \in \mathcal{V} \setminus \{i\}} L_{(j,i)}(\mathbf{y}) \tag{6}$$

where ω^{GF} is the average number of operations per bit of received packet to perform generalized forwarding and $L_{(j,i)}(\cdot)$ is defined in (5). Notice that the forwarding workload considers cases where traffic is sent to a local VNF or to the next node.

Uplink Delay: We define the uplink delay as the time taken for a single bit to travel from the RU to the core. The uplink delay is going to be primarily impacted by the queuing delay at each networking node. Assuming that each node in the virtual transport channels operates simply as an $M/M/1$ queue, the uplink delay experienced by packets generated at RU r is the total average wait time across all of its associated queues, i.e.,

$$D_{r,s}(\mathbf{y}) = \sum_{\ell \in \mathcal{L}} \frac{y_{r,s,\ell}}{\mu_\ell^{\text{Link}} - L_\ell(\mathbf{y})}, \tag{7}$$

where μ_ℓ^{Link} is link ℓ's capacity and $L_\ell(\cdot)$ is the link ℓ's utilized capacity defined in (5).

Energy Consumption: In our system, we consider the total computing workload as the major energy demand point. Then, we can define the nodal energy consumption as

$$E_i(\mathbf{x}, \mathbf{y}) = \epsilon_i \cdot \left(W_i^{\text{V}}(\mathbf{x}) + W_i^{\text{F}}(\mathbf{y}) \right), \tag{8}$$

where ϵ_i is a parameter representing the amount of energy spent per operation and functions $W_i^{\text{V}}(\cdot)$ and $W_i^{\text{F}}(\cdot)$ are the virtualization and forwarding computing workload functions, defined, respectively, in (3) and (6).

III. Problem Definition

We consider a system where traffic flows must be dynamically steered across O-Cloud nodes hosting VNFs. Our goal is to minimize energy consumption while ensuring that all slices meet their performance requirements. As introduced in Section II, this involves two key decisions: (i) function virtualization and (ii) traffic steering. Below, we formalize these aspects as an optimization problem.

Firstly, each RU requires instances of DU and CU functions deployed in cloud nodes, represented by the constraints:

$$\sum_{i \in \mathcal{V}^{\text{Cloud}}} x_{r,i}^v = 1, \quad \forall r \in \mathcal{V}^{\text{RU}}, v \in \text{VNF}, \tag{9}$$

where $\text{VNF} = \{\text{DU}, \text{CU}\}$.

A. Flow Conservation Constraints

Each RU must maintain connectivity between its function instances, requiring valid fronthaul (FH) and midhaul (MH) channels. These dependencies lead to the flow conservation constraints:

$$\sum_{j \in \mathcal{N}_i} y_{r,s,(j,i)} - y_{r,s,(i,j)} = \mathbf{1}_{r=i} - x_{r,i}^{\text{DU}}, \begin{smallmatrix}\forall r \in \mathcal{V}^{\text{RU}}, \\ \forall i \in \mathcal{V},\end{smallmatrix} \tag{10}$$

$$\sum_{j \in \mathcal{N}_i} y_{r,s,(j,i)} - y_{r,s,(i,j)} = x_{r,i}^{\text{DU}} - x_{r,i}^{\text{CU}}, \begin{smallmatrix}\forall r \in \mathcal{V}^{\text{RU}}, \\ \forall i \in \mathcal{V}.\end{smallmatrix} \tag{11}$$

where $\mathcal{N}_i$ is the set of neighbors of node i and $\mathbf{1}_e$ is the indicator function.

B. Resource Capacity Constraints

To ensure resource availability, we impose the following constraints:

Computational Capacity: The total workload on each O-Cloud node must not exceed its available processing power

$$W_i^{\text{V}}(\mathbf{x}) + W_i^{\text{F}}(\mathbf{y}) \leq \mu_i^{\text{Comp}}, \quad \forall i \in \mathcal{V}^{\text{Cloud}}, \tag{12}$$

where μ_i^{Comp} is the computational capacity of node i, and $W_i^{\text{V}}(\cdot)$ and $W_i^{\text{F}}(\cdot)$ are workload functions defined in (3) and (6).

Link Capacity: The total traffic on each link must not exceed its capacity

$$L_\ell(\mathbf{y}) \leq \mu_\ell^{\text{Link}}, \quad \forall \ell \in \mathcal{L}, \tag{13}$$

where μ_ℓ^{Link} is the capacity of link ℓ, and $L_\ell(\cdot)$ is defined in (5).

Delay Constraints: The total uplink delay for slice s at RU r must not exceed its predefined tolerance

$$D_{r,s}(\mathbf{y}) \leq \delta_r, \quad \forall r \in \mathcal{V}^{\text{RU}}, \tag{14}$$

where δ_r is the delay tolerance, and $D_{r,s}(\cdot)$ is defined in (7).

C. Objective Function

Our objective is to minimize the total energy consumption across all nodes given by

$$E(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathcal{V}} E_i(\mathbf{x}, \mathbf{y}), \tag{15}$$

where $E_i(\cdot)$ represents the energy consumption of node i, as defined in (8).

D. Optimization Problem

We now define the Eco-5G Optimization Problem as a non-linear integer program:

Problem 1 (Eco-5G Problem)**.**

$$\begin{array}{ll} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} & (15) \\ \text{subject to} & (2), (4), (9) - (14) \end{array}$$

Proposition 1. Problem 1 is NP-Hard.

We omit the proof of Proposition 1, as it follows from a standard reduction from integer problems with capacity constraints.

E. Flow-Based Approximation

Given the complexity of Problem 1, we introduce a flow-based approximation to reduce the solution space. We define a set $\mathcal{P}_{i,j}^{(k)}$ containing the top k paths between nodes i and j, with the full set denoted by $\mathcal{P} = \{\mathcal{P}_{i,j}^{(k)}\}_{\forall i,j \in \mathcal{V}}$.

Instead of computing paths dynamically, we predefine them and introduce selection variables

$$z_{r,s,p}^c \in \{0, 1\}, \quad \forall r \in \mathcal{V}^{\text{RU}}, s \in \mathcal{S}, p \in \mathcal{P}, c \in \mathcal{C}, \tag{16}$$

where $z_{r,s,p}^c = 1$ if path p is selected for channel $c \in \{\text{FH, MH}\}$ in slice s at RU r.

We enforce flow consistency with:

$$\sum_{p \in \mathcal{P}} z^c_{r,s,p} = 1, \quad \forall r \in \mathcal{V}^{\text{RU}}, s \in \mathcal{S}, c \in \mathcal{C}, \tag{17}$$

ensuring that each flow follows a single path, and

$$z^{\text{FH}}_{r,s,p} \cdot z^{\text{MH}}_{r,s,p'} \leq \mathbf{1}_{p_d = p'_o}, \quad \forall r, s, p, p', \tag{18}$$

ensuring that fronthaul and midhaul paths are contiguous.

Problem 2 (Flow-Based Eco-5G Problem)**.**

$$\begin{aligned} &\underset{\mathbf{z}}{\text{minimize}} && (15) \\ &\text{subject to} && (16), (17), (18), (12) - (14) \end{aligned}$$

Note that each assignment of variable $\mathbf{z}$ uniquely determines the values of variables $\mathbf{x}$ and $\mathbf{y}$. Thus, the functions and constraints from Problem 1 can be applied interchangeably.

Proposition 2. Problem 2 is NP-Hard.

We omit the proof, as it follows from a known reduction from knapsack-constrained optimization problems.

Remark 1. For sufficiently large k, Problems 1 and 2 yield the same optimal solution. While this formulation does not reduce complexity, it allows for good approximations by tuning k, as discussed in Section IV.

IV. E-MATRO

In this section, we discuss how to solve Problem 2 using a Multi-Agent Reinforcement Learning approach. We refer to it as Eco Migration And Traffic Re-routing in O-RAN, or simply as E-MATRO. In our approach, in addition to their role in hosting VNFs, active nodes operate as agents capable of making decisions. The first step is to model each agent's behavior as an independent Markov Decision Process (MDP).

A. Markov Decision Process

The MDP $= (\mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ is a tuple defining the set of observable states $\mathcal{O}$, the set of actions $\mathcal{A}$, the transition probabilities $\mathcal{T}$, and the reward function $\mathcal{R}$. Although agents base their decisions on the same MDP, their states may evolve differently.

States: The state $o_i(t) \in \mathcal{O}$ represents the observed status of agent $i \in \mathcal{V}$ at time t. Each state is a tuple denoted by $o_i(t) = (\theta^{\text{RU}}_i, \{\theta^{\text{DU}}_{r,i}\}_{r \in \mathcal{V}^{\text{RU}}}, \{\theta^{\text{CU}}_{r,i}\}_{r \in \mathcal{V}^{\text{RU}}}, \{\theta^{\text{Path}}_{r,i}\}_{r \in \mathcal{V}^{\text{RU}}})$ whose components are

- RU Indicator θ^{RU}_i: A binary component indicating whether the agent is an RU $\theta^{\text{RU}}_i = 1$ or an O-Cloud server $\theta^{\text{RU}}_i = 0$.
- DU Indicator $\theta^{DU}_{r,i}$: A binary component indicating whether or not the agent hosts the DU of RU r, which is derived from the decision variable $x^{\text{DU}}_{r,i}$.
- CU Indicator $\theta^{CU}_{r,i}$: A binary component indicating whether or not the agent hosts the CU of RU r, which is derived from the decision variable $x^{\text{CU}}_{r,i}$.
- Routing Path $\theta^{\text{Path}}_{r,i} \in \mathcal{P} \cup \{\text{null}\}$: A list of segments to reach the next VNF based on the current agent's role:

1) If the agent is an RU, i.e., $\theta^{\text{RU}}_i = 1$, then $\theta^{\text{Path}}_{r,i}$ is the path to its DU.
2) If the agent is a DU, i.e., $\theta^{\text{RU}}_i = 0$ and $\theta^{\text{CU}}_r = 1$, then $\theta^{\text{Path}}_{r,i}$ is the path to RU r's CU.
3) If the agent is a CU, i.e., $\theta^{\text{RU}}_i = 0$ and $\theta^{\text{CU}}_{r,i} = 1$, or a simple relay, i.e., $\theta^{\text{RU}}_i = \theta^{\text{CU}}_{r,i} = \theta^{\text{CU}}_{r,i} = 0$, then there is no routing decision and $\theta^{\text{Path}}_{r,i} = \{\text{null}\}$.

The set of states has a size complexity of $O((k+1)|\mathcal{V}^{\text{RU}}|^3)$, where k is the number of redundant paths between each pair of nodes introduced in Section III-E.

Actions: Each agent has the same set of actions $\mathcal{A}$. The agents are capable of learning two distinct types of actions, as outlined below:

1) Function Migration: If a host i serves as the DU/CU for a RU r, meaning $x^{DU}_{r,i} = 1$ or $x^{CU}_{r,i} = 1$, then it can choose to migrate that function to another active node j at any moment. We denote each of these actions by $a^{\overrightarrow{DU}}_{r,i,j}$ and $a^{\overrightarrow{CU}}_{r,i,j}$, respectively. We refer to set $\mathcal{A}^{\overrightarrow{DU}}_i = \left\{a^{\overrightarrow{DU}}_{r,i,j} : \forall r \in \mathcal{V}^{\text{RU}}, \forall j \in \mathcal{V}\right\}$ and set $\mathcal{A}^{\overrightarrow{CU}}_i = \left\{a^{\overrightarrow{CU}}_{r,i,j} : \forall r \in \mathcal{V}^{\text{RU}}, \forall j \in \mathcal{V}\right\}$ as groups of migration actions available for agent i. Note that, for either $a^{\overrightarrow{DU}}_{r,i,j}$ or $a^{\overrightarrow{CU}}_{r,i,j}$, if $i = j$, we say that the action is to not migrate the corresponding function. Note that, if agent i is an RU, i.e., $\theta^{\text{RU}}_i = 1$, or simply a relay node, i.e., $\theta^{\text{RU}}_i = \theta^{\text{CU}}_{r,i} = \theta^{\text{CU}}_{r,i} = 0$, then migration action is not available.
2) Traffic Steering: Any RU or DU functions can decide to change the path of the traffic stream generated at RU r to the subsequent function, i.e., DU or CU, respectively. We denote by $a_{r,s,i,p}$ the action of agent i to adopt path $p \in \mathcal{P}$ to carry r's traffic to the next function. We refer to the set $\mathcal{A}_i = \left\{a_{r,s,i,p} : \forall r \in \mathcal{V}^{\text{RU}}, \forall s \in \mathcal{S}, \forall q \in \mathcal{P}_i\right\}$ as a group of steering actions available for agent i. Note that, for steering action $a_{r,s,i,p}$, if the new chosen path is the same as the current path, i.e., $p = \theta^{\text{Path}}_{r,i}$, then we say that the action is to not steer the traffic.

The actions set has a size complexity of $O(|\mathcal{V}^{\text{RU}}|(2|\mathcal{V}| + k|\mathcal{S}|)$.

Transition Probabilities: In our MDP formulation, an agent's action request does not necessarily guarantee that the action will be executed as intended. Due to the multi-agent nature of the problem, conflicts may arise when multiple agents request actions that cannot be simultaneously accommodated, leading to deviations from the expected state transitions. As a result, the next state is not solely determined by an individual agent's decision but rather by the collective interaction of all agents, making the transition dynamics uncertain. This unpredictability prevents us from accurately estimating the transition probabilities, which are typically required in model-based approaches. Consequently, we adopt a model-free reinforcement learning method, which does not rely on explicit transition probability estimation but instead learns optimal policies directly from the experienced rewards.

Reward: After taking an action, the agent will experience a reward. In our setup, after resolving eventual action conflicts, the system will move to a new state. This state is intrinsically mapped to Problem 2's variables. Then, as our goal is to minimize energy consumption, our reward is built based on function 15. Additionally, we wish to embed the notion of feasibility by penalizing our reward if unfeasible solutions are found. To capture these two notions, we use Problem 2's Lagrangian function as follows

$$\begin{aligned}G(\mathbf{z}) = E(\mathbf{z}) &+ \sum_{r\in\mathcal{V}^{\text{RU}}}\sum_{s\in\mathcal{S}}\sum_{c\in\mathcal{C}}\alpha^{\text{C1}}_{r,s,c}\left[\sum_{p\in\mathcal{P}} z^{c}_{r,s,p} - 1\right]\\ &+ \sum_{r\in\mathcal{V}^{\text{RU}}}\sum_{s\in\mathcal{S}}\sum_{p,p'\in\mathcal{P}}\alpha^{\text{C2}}_{r,s,p,p'}\left[z^{\text{FH}}_{r,s,p}\cdot z^{\text{MH}}_{r,s,p} - \mathbf{1}_{p_d=p'_o}\right]\\ &+ \sum_{i\in\mathcal{V}}\alpha^{\text{C3}}_{i}\left[W^{\text{V}}_i(\mathbf{z}) + W^{\text{F}}_i(\mathbf{z}) - \mu^{\text{Comp}}_i\right]\\ &+ \sum_{\ell\in\mathcal{L}}\alpha^{\text{C4}}_{\ell}\left[L_\ell(\mathbf{z}) - \mu^{\text{Link}}_\ell\right]\\ &+ \sum_{r\in\mathcal{V}^{\text{RU}}}\sum_{s\in\mathcal{S}}\alpha^{\text{C5}}_{r,s}\left[D_{r,s}(\mathbf{z}) - \delta_{r,s}\right],\end{aligned} \tag{19}$$

where parameters $\alpha^{\text{C1}}_{r,s,c}$, $\alpha^{\text{C2}}_{r,s,p,p'}$, α^{C3}_{i}, $\alpha^{\text{C4}}_{\ell}$, and $\alpha^{\text{C5}}_{r,s}$ are Lagrangian multipliers for constraints (17), (18), (12), (13), and (14), respectively.

Considering that a set of states $\mathbf{o}_t = \{o_{i,t}\}_{i\in\mathcal{V}}$ of all agents in a given instant t can be represented by an assignment of variables $\hat{\mathbf{z}}$, we can write $G(\mathbf{o}_t) = G(\hat{\mathbf{z}})$. We use function (19) defined over the set of states to quantify the benefit of the system to be in such a global state. Then, we define the reward function as the variability of this quantity, i.e.,

$$R(\mathbf{o}_t, \mathbf{a}_t) = G(\mathbf{a}_t(\mathbf{o}_t)) - G(\mathbf{o}_t), \tag{20}$$

where, with some abuse of notation, $\mathbf{a}_t$ represents both the set of actions taken by each agent at instant t and a function for state transition $\mathbf{a}_t : \mathcal{O}^{|\mathcal{V}|} \to \mathcal{O}^{|\mathcal{V}|}$ based on that same set of actions.

Policy: Our goal is to find an optimal policy $\pi^*_i(o_i(t))$, which is typically defined as selecting the action that maximizes the MDP's Q-value. The Q-value $Q(o_i(t), a_i(t))$ is a function that represents the expected cumulative reward for the action $a_i(t)$ taken by agent i in state $o_i(t)$ at time step t and following the policy thereafter. We define an optimal policy using ϵ-Greedy algorithm as follows

$$\pi_i(o_i(t)) = \begin{cases}\arg\max\limits_{a\in\mathcal{A}_i} Q(o_i(t), a), & \text{with prob. } 1-\epsilon\\ \text{random } a \in \mathcal{A}_i, & \text{with prob. } \epsilon,\end{cases} \tag{21}$$

where $\epsilon \in [0,1]$ is used to balance exploration-exploitation. However, computing maximum $Q(o_i(t), a)$ in (21) exactly (e.g., using Bellman optimality equation) requires knowledge of the MDP's transition probabilities, which are unknown in our case. Then, as previously mentioned, we resource to a model-free approach where the Q-value can be inferred using different Q-Learning algorithms.

B. E-MATRO*'s Design*

E-MATRO is a hybrid framework where a portion of the functionalities is performed directly by O-Cloud servers, implemented as dApps, and another part is performed by the Near-RT RIC, implemented as an xApp. The framework consists of two phases: (i) a setup phase, where agents will perform the initial preparations for the algorithm's correct operation, and (ii) the main phase, defined by the action-reward loop where the learning is effectively taking place. In the following, we discuss each phase in detail.

1) Setup Phase: The setup phase is managed by an xApp deployed on the Non-RT RIC, responsible for three tasks:

(i) **Path sets definition**, i.e., compute each agent's set of paths in the O-Cloud network. Parameter $k > 1$, i.e., the number of alternative paths between each pair of agents, and the path-selection strategy defines, for each $i \in \mathcal{V}$, its path set $\mathcal{P}^{(k)}_i = \{p^{(1)}_{i,j}, \ldots, p^{(k)}_{i,j}, \forall j \in \mathcal{V} : j \neq i\}$. Note that $\mathcal{P}^{(k)}_i$ later defines the agents' states and available action sets.

(ii) **Initial state definition**, i.e., determine an initial DU-CU placement for each RU. We compute any feasible solution $\hat{\mathbf{z}}'$ for Problem 2 given set $\mathcal{P}^{(k)}_i$ as input. This solution is then mapped to an initial state $o_i(0)$, for each agent i.

(iii) **Q-Learning Deployment**, in which data structures supporting Q-Learning-based algorithms are implemented. We then explore two distinct solutions: one requiring the implementation of a Q-Table, and the other relying on a Q-Network.

Once the setup is complete, we proceed to the main phase, where the primary learning loop takes place.

2) Main Phase: The main phase is implemented as the "Agent" dApp directly on every O-Cloud server. At every time step t, agent i, who is in state $o_i(t)$ chooses an action $a_i(t) \in \mathcal{A}_i$, leading to a new state $o_i(t+1)$. Such action is chosen using policy (21) based on the latest Q-values provided by the "Control" xApp. The next state depends on resolving eventual unfeasible or conflicting actions from different agents (we discuss more about it in Section IV-B4). Each agent i announces its transition tuple $(o_i(t), a_i(t), o_i(t+1))$ to the "Control" xApp, which stores the received transitions of all agents at time step t. After M time steps, the "Control" xApp generates a reward $R(o(t), a(t))$, given by (20), concerning the observed batch of actions. This reward is then used to update the Q-Table or the parameters of the Q-Network, depending on the implementation of E-MATRO. This loop goes on until the end of the observation horizon T.

3) Q-Learning Implementations: E-MATRO allows for different implementations in the representation of Q-values.

A straightforward approach is the classic Q-Learning method, where a Q-Table is maintained in the "Control" xApp and updated every M time steps using the Bellman optimality equation:

$$Q^*(\mathbf{o}(t), \mathbf{a}(t)) = \mathbb{E}\left[R^{(M)}(t) + \gamma^M \max_{\mathbf{a}'} Q^*(\mathbf{o}(t+M), \mathbf{a}')\right], \tag{22}$$

where γ is the discount factor, and $R^{(M)}(t)$ is defined in (20).

Alternatively, Q-values can be approximated using a deep neural network (DNN), denoted as $Q(o_i(t), a_i(t); \Phi)$. Here, the input to the network is the state $o_i(t)$ of agent i at time step t, and the output is a set of Q-values corresponding to all possible actions.

The DNN $Q(o_i(t), a_i(t); \Phi)$ is trained using *experience replay*, where past transitions (o, a, r, o') are stored in a FIFO buffer and sampled randomly for training. The training process aims to minimize the difference between the predicted Q-values and the *target values*, which are computed using an n-step bootstrapping method:

$$y^{\text{Target}} = \sum_{k=0}^{n-1} \gamma^k r(t+k) + \gamma^n \max_{\mathbf{a}'} Q(\mathbf{o}(t+n), \mathbf{a}'; \Phi^-), \quad (23)$$

where r is the received reward, γ is the discount factor, and $Q(o', a'; \Phi^-)$ is the Q-value of the next state o', estimated using a separate target network with parameters Φ^-. The neural network parameters Φ are updated by minimizing the mean squared error (MSE) loss over a batch of N samples:

$$L(\mathbf{\Phi}) = \frac{1}{N} \sum_{k=1}^{N} \left(y^{\text{Target}} - Q^*(\mathbf{o}(k), \mathbf{a}(k); \Phi) \right)^2. \quad (24)$$

4) Conflict Resolution: In a multi-agent system, each agent acts in its own interest, which can sometimes lead to conflicting actions. These conflicts arise when resources that were available at the time decisions were made become insufficient to accommodate all selected actions. To address this issue, we introduce a conflict resolution protocol that ensures actions are executed without creating infeasible system states.

The resolution process is structured as follows: first, all agents select their actions according to the policy defined by the Q-Learning algorithm. These actions may include function migration or traffic steering. Each acting agent signals its intended action to a target agent, which may be the recipient of a migrated function or the destination of steered traffic. The signaling mechanism is implemented using end-to-end Explicit Congestion Notification (ECN), allowing acting agents to assess the congestion status of intermediate routers. Additionally, the target agent provides feedback on its current computational utilization.

With knowledge of both network congestion levels and computational resource availability, the acting agent evaluates whether its intended action satisfies Problem 2's constraints. If the action results in a feasible solution, the involved agents proceed with the implementation. Otherwise, the agent selects an alternative action from its action set. For simplicity, we assume that if no feasible action is available, the agent defaults to not performing migration or steering. Conflict resolution follows a hierarchical order, starting with CUs, followed by DUs, and finally RUs.

V. Numerical Results

In this section, we evaluate the performance of the E-MATRO framework (introduced in Section IV) in solving Problem 2. The analysis covers network stress conditions (for scalability) and various parameter settings.

A. Experimental Setup

We implement E-MATRO using Pytorch on a Linux server with a 12th Intel(R) i7-12700 CPU, 64GB RAM, and NVIDIA RTX A4000 GPU. Table I details the parameters used to determine Problem 2's functions, extracted from [12][2].

Parameter	Value
Number of RUs	2 (light-load), 8 (heavy-load)
Number of DUs	2
Number of CUs	2
Link capacity (μ_ℓ^{Link})	5 Gbps
Computational capacity (μ_i^{Comp})	50 GOPS
Processing operations ω^{DU}	25 OPS/bit
Processing operations ω^{CU}	40 OPS/bit
Forwarding operations (ω^{GF})	9.5×10^{-12} OPS/bit
Total throughput ($\lambda_{r,s}$)	40 Mbps, 100 Mbps
Delay tolerance(δ_s)	25 ms (URLLC), 50 ms (eMBB)
Amount energy per operation ϵ_i	3.30 kWh/GOPS
Amount energy per forwarding ϵ_{ij}	$[0.0303, 0.727]$ kWh/GB

TABLE I: Setup network parameters.

Network: In this work we use three different topologies for the O-Cloud network. First, the "Toy" topology (the same as in Fig 1), and two real-world topologies, GEANT [14] and Abilene network [15]. We generate the graph corresponding to each topology using NetworkX [16]. Evaluations are performed under two traffic conditions: light-load, with 2 RUs connected to each DU, and heavy-load, with 8 RUs connected to each DU.

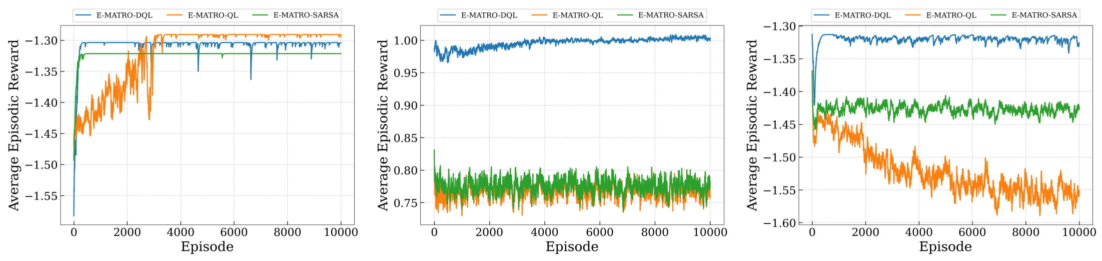
Algorithms: We assess the performance of the proposed E-MATRO algorithm with three different implementations: Deep Q-Learning (DQL), classic Q-learning (QL), and SARSA.

Paths: The number of paths between agent connections RU-DU, DU-DU, DU-CU and CU-CU can singificantly increase with the number of nodes in the topology, hindering the ability to solve instances of larger topologies. As proposed in Section III, we employ an algorithm for $k = 2$ shortest paths to generate the routes for the set $\mathcal{P}$.

Performance Metrics: The performance assessment is based on the following evaluation metrics:

- Average Episodic Reward: It measures the total reward gathered by each method in each episode, averaged over 10-step duration episodes.
- Objective Value: It quantifies the optimization performance of each method based on the defined energy cost function defined in Problem 2.
- Network Link Utilization: It evaluates how efficiently each method allocates and uses available bandwidth for the selection of routing paths.

[2]In [12], the energy cost is set at $\sigma = 0.165\$/kWh$, reflecting the U.S. average [13], but we used the values to compute the average energy consumption for each processed Byte (kWh/GB).



(a) Toy Topology (b) Abilene Topology (c) GEANT Topology

Fig. 2: Average episodic reward over time across diverse network topologies.

B. Performance Evaluation

Learning Capabilities: First, we evaluate the ability of E-MATRO to learn the optimal policy under different learning algorithms. We show the corresponding experimental results on Figure 2 for a system where UEs generate traffic with rate of 40 Mbps in the range of 2 RUs. Each figure consists of three plots, one per considered network topology, displaying the average episodic reward evolution over time, summarized across all the 10000 episodes.

In the Toy topology (Figure 2a), we observe that all algorithms successfully converge to a learned policy. Although E-MATRO-QL exhibits a noisier convergence process, it achieves the best overall performance. This is because Q-Learning typically converges to the optimal policy, and in this simple scenario, it can maintain accurate Q-value estimates. Moreover, it outperforms E-MATRO-DQL, whose performance is likely hindered by approximation errors in the Q-values. E-MATRO-SARSA shows slightly lower performance, likely due to following an ϵ-greedy policy, which incorporates more exploratory actions.

As the system's complexity increases (see Figures 2b and 2c), the learning process becomes noisier, although it maintains a consistent average over episodes. Notably, the performance of E-MATRO-QL and E-MATRO-SARSA is severely impacted, as these algorithms struggle to scale effectively to larger systems. In contrast, E-MATRO-DQL successfully captures the complexity of these scenarios and integrates it into its Q-Network approximations, allowing it to adapt more effectively.

Observation 1: E-MATRO is capable of converging to an optimal policy across networks of varying complexity, provided that the appropriate learning algorithm is selected for each scenario.

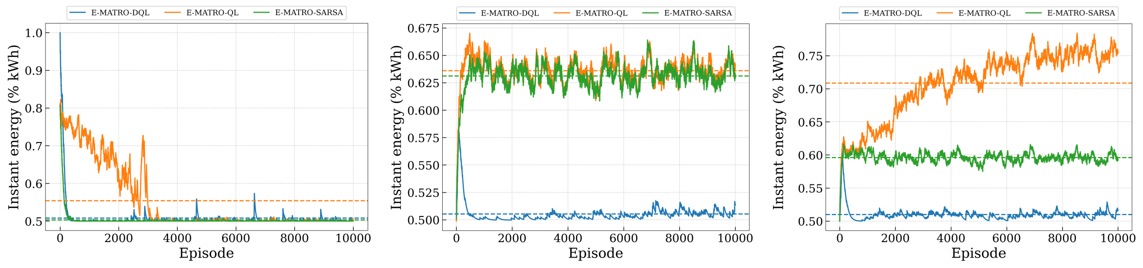
Energy Consumption: In Figure 3, we observe the energy consumption evolution over time. Similarly to the previous discussion, we plot the results for all three learning algorithms and for each one of the three considered topologies. The energy consumption at each time step is computed by applying function (15) at each time step. We normalize the value of the energy consumption over all different topologies so that we can compare the performance in scenarios of different complexity levels. We recall that the function depends on variables $\mathbf{z}$ that can be used to summarize the agents' states. At every time step, after resolving any eventual conflicting actions, the energy consumption can be effectively measured.

In Figure 3a, we see that all implementations of E-MATRO are able to achieve the same the level of energy utilization in the Toy topology. Notice that E-MATRO-QL has a noisier and slightly longer convergence due the more aggressive nature of the QL algorithm. For Abilene, in Figure 3b, E-MATRO-QL and E-MATRO-SARSA present similar behavior, oscillating around the same average value. On the other hand, E-MATRO-DQL shows better performance by staying consistently at the lowest energy consumption level. A similar behavior is observed in Figure 3c for the GEANT topology.

Observation 2: E-MATRO is able to minimize energy consumption for DQL algorithms with a fast and stable convergence.

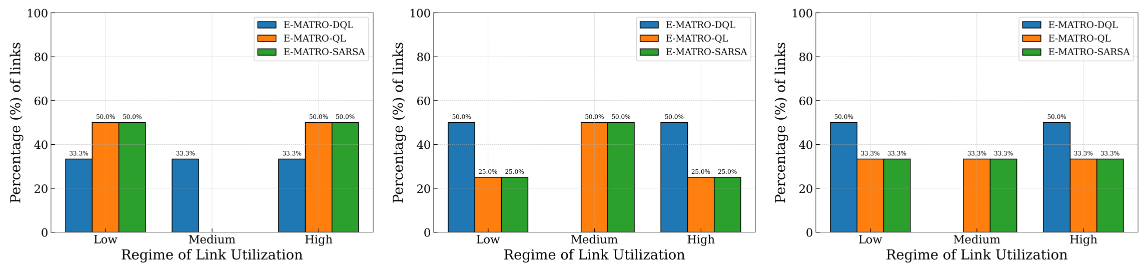
Link Utilization: In an effort to quantify the number of migration and steering actions, we propose to study the link utilization, i.e., the fraction of time each link is being used for transmission. In Figure 4, we present the fraction of links under specific levels of link utilization, which we organize into three categories: Low ($< 20\%$), Medium ($21-60\%$), and High ($> 60\%$) link utilization. This distribution serves as a metric to evaluate each method's network load balancing efficiency, providing insight into how effectively each approach distributes traffic across available resources. In this experiment, we expose the system to heavy-traffic load conditions, where we consider 8 RUs with UEs generating traffic at a rate of 100 Mbps.

E-MATRO-DQL demonstrates superior energy-aware traffic management across all topologies. In the Toy topology, E-MATRO-DQL achieves a perfectly balanced distribution (33.3% Low, 33.3% Medium and 33.3% High), strategically utilizing the full spectrum of link capacities to optimize energy consumption. This contrasts with both E-MATRO-QL and E-MATRO-SARSA, which show less sophisticated resource allocation (50% Low, 50% High), failing to leverage medium-utilization links for optimal traffic distribution. In the more complex Abilene topology, E-MATRO-DQL maintains an efficient split distribution (50% Low, 50% High), concentrating traffic on fewer high-utilization paths while keeping others minimally used—directly supporting our energy minimization objective. Conversely, E-MATRO-QL and E-MATRO-SARSA display a fragmented split pattern (25.0% Low, 50% Medium,



(a) Toy Topology (b) Abilene Topology (c) GEANT Topology

Fig. 3: Evolution of normalized instantaneous energy consumption over time across diverse network topologies.



(a) Toy Topology (b) Abilene Topology (c) GEANT Topology

Fig. 4: Percentage of links under different utilization regimes across diverse network topologies.

and 25.0% High), spreading traffic across more paths and consequently increasing energy consumption despite the topology's greater connectivity. Similarly, in the GEANT topology, E-MATRO-DQL maintains its strategic bimodal distribution (50% Low, 50% High), while E-MATRO-QL and E-MATRO-SARSA resort to a less energy-efficient uniform distribution across all utilization levels (33.3% across Low, Medium, and High). This consistent performance across increasingly complex network topologies demonstrates E-MATRO-DQL ability to make energy-optimal routing decisions regardless of network scale or traffic conditions.

Observation 3: E-MATRO-DQL consistently outperforms E-MATRO-QL and E-MATRO-SARSA in energy-aware traffic management by strategically distributing traffic to optimize energy consumption across all topologies. Its ability to adapt to increasing network complexity while maintaining an energy-efficient bimodal distribution highlights its effectiveness in minimizing energy usage compared to less optimized allocation patterns.

VI. Related Work

Virtual function placement in the context of disaggregated RAN architectures, mainly O-RAN, is an active research topic, where the problem is often formalized as an optimization model and tackled using different heuristics and ML techniques [17], [18]. In this section, we focus on studies that consider energy consumption in their performance model as well as distributed and multi-agent solution approaches.

The work presented in [19] investigates the energy-efficient and delay-constrained joint resource allocation and DU selection problem within O-RAN, with a primary focus on optimizing energy efficiency. In a similar vein, [20] formulates a generalized network function placement problem with functional split options, solving it as a Binary Integer Linear Programming (BILP) problem in three stages. While their objective is to minimize computing resource costs and maximize the aggregation of VNFs, their approaches do not incorporate the energy cost of deployment and overall computing, which is a key consideration in our formulation.

Additionally, the study in [21] addresses the joint optimization of flow-split distribution, congestion control, and scheduling, enabling intelligent traffic steering in O-RAN. The aim is to efficiently and adaptively direct traffic to the appropriate RUs while ensuring compliance with latency constraints. However, this work does not account for the function placement in O-RAN or the operational energy cost — two critical components of our approach.

Moreover, [10] investigates a related context by addressing joint admission control, resource activation, VNF placement, resource allocation, and traffic routing. Their approach leverages model predictive control (MPC) to dynamically adjust to updated traffic forecasts. However, unlike our work, these studies rely on a single-time-scale optimization framework, either assuming that RIC controllers perform optimization in parallel or neglecting the possibility that certain operations could be executed across different time scales.

In O-RAN, RL-based solutions have been widely adopted for dynamic network function placement and route selection. For instance, [7] optimizes the placement of CUs and DUs within regional O-Clouds under user-level slicing, aiming to minimize end-to-end delay and deployment costs, while considering processing and bandwidth constraints. This study employs deep Q-learning as the DRL approach.

The authors in [9] address the vRAN reconfiguration by optimizing the gNB functional split, vCU/vDU placement, resource allocation, and routing to minimize network costs. To mitigate dimensionality challenges, the study proposes a model-free DRL framework utilizing D3QN. Similarly, [22] investigates real-time RIC co-located with the DU, optimizing system throughput and latency for near-real-time applications using a multi-agent DRL approach.

In [23], authors assume that DU/CU functionalities are powered by renewable energy sources and employ a multi-agent Q-learning distributed management system. However, this study does not incorporate extensive service considerations. Despite the advancements in RL-based approaches, existing studies often lack updated forecasts or dynamic decision-making across varying time scales under traffic uncertainty. To the best of our knowledge, our work is the first to jointly optimize VNF resource allocation in O-Clouds and traffic steering between VNFs, with the objective of minimizing end-to-end energy consumption in O-RAN through VNF migration.

VII. Conclusion

In this paper, we addressed the challenge of energy-efficient VNF placement and traffic steering in O-RAN, focusing on minimizing end-to-end energy consumption while ensuring slice compliance. We formalized the problem as a flow-based optimization and developed E-Matro, a novel Multi-Agent RL framework that integrates decentralized dApps and a centralized xApp. This hybrid approach ensures both localized, real-time traffic adaptation and global network coordination, effectively responding to dynamic traffic conditions. Our framework offers a new solution by incorporating VNF-to-VNF traffic steering and function migration, facilitating energy efficiency in the face of fluctuating demands.

Through rigorous experimentation on O-Cloud networks of various sizes, we demonstrated that E-Matro converges to optimal, energy-efficient configurations, with Deep Q-Learning showing superior adaptability in complex, large-scale scenarios. The results highlight the potential of our framework to significantly improve the scalability and efficiency of O-RAN, positioning it as a viable solution for 5G&B networks. Future work will explore further energy models and extensions of E-Matro to handle even more dynamic and heterogeneous network environments.

Acknowledgements

This work was supported by ANR under the France 2030 program, grant NF-NAI: ANR-22-PEFT-0003.

References

[1] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten, “The road towards 6g: A comprehensive survey,” *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334–366, 2021.
[2] C. Han, Y. Wu, Z. Chen *et al.*, “Network 2030 a blueprint of technology, applications and market drivers towards the year 2030 and beyond,” *International Telecommunication Union*, 2018.
[3] NGMN Alliance, “5g white paper 2,” Tech. Rep., 2024.
[4] H. Cheng, S. D’Oro, R. Gangula, S. Velumani, D. Villa, L. Bonati, M. Polese, T. Melodia, G. Arrobo, and C. Maciocco, “Oranslice: An open source 5g network slicing platform for o-ran,” in *MobiCom*, 2024.
[5] A. Lacava, M. Polese, R. Sivaraj, R. Soundrarajan, B. S. Bhati, T. Singh, T. Zugno, F. Cuomo, and T. Melodia, “Programmable and customized intelligence for traffic steering in 5g networks using open ran architectures,” *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 2882–2897, 2023.
[6] M. Polese, L. Bonati, S. D’oro, S. Basagni, and T. Melodia, “Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges,” *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.
[7] R. Joda, T. Pamuklu, P. E. Iturria-Rivera, and M. Erol-Kantarci, “Deep reinforcement learning-based joint user association and cu–du placement in o-ran,” *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4097–4110, 2022.
[8] 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 Journal on Selected Areas in Communications*, 2023.
[9] F. W. Murti, S. Ali, G. Iosifidis, and M. Latva-aho, “Deep reinforcement learning for orchestrating cost-aware reconfigurations of vrans,” *IEEE Transactions on Network and Service Management*, 2023.
[10] N. Fryganiotis, E. Stai, I. Dimolitsas, A. Zafeiropoulos, and S. Papavassiliou, “Dynamic, reconfigurable and green network slice admission control and resource allocation in the o-ran using model predictive control,” in *2024 IFIP Networking Conference (IFIP Networking)*. IEEE, 2024, pp. 1–9.
[11] S. D’Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, “dapps: Distributed applications for real-time inference and control in o-ran,” *IEEE Communications Magazine*, vol. 60, no. 11, pp. 52–58, 2022.
[12] Z. Xu, H. Ren, W. Liang, Q. Xia, W. Zhou, P. Zhou, W. Xu, G. Wu, and M. Li, “Near optimal learning-driven mechanisms for stable nfv markets in multitier cloud networks,” *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2601–2615, 2022.
[13] S. Maxenti, S. D’Oro, L. Bonati, M. Polese, A. Capone, and T. Melodia, “Scalo-ran: Energy-aware network intelligence scaling in open ran,” in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 891–900.
[14] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, “Providing public intradomain traffic matrices to the research community,” *Computer Communication Review (SIGCOMM)*, vol. 36, no. 1, pp. 83–86, 2006.
[15] “Abilene, 2004.” [Online]. Available: http://www.cs.utexas.edu/yzhang/research/AbileneTM/
[16] “Networkx.” [Online]. Available: https://networkx.org/
[17] H. Hojeij, M. Sharara, S. Hoteit, and V. Vèque, “On flexible placement of o-cu and o-du functionalities in open-ran architecture,” *IEEE Transactions on Network and Service Management*, vol. 22, no. 1, pp. 660–674, 2025.
[18] H. Hojeij, G. I. Ricardo, M. Sharara, S. Hoteit, V. Vèque, and S. Secci, “On flexible association and placement in disaggregated ran designs,” *Computer Communications*, vol. 238, p. 108166, 2025.
[19] T. Pamuklu, S. Mollahasani, and M. Erol-Kantarci, “Energy-efficient and delay-guaranteed joint resource allocation and du selection in o-ran,” in *2021 IEEE 4th 5G World Forum (5GWF)*. IEEE, 2021, pp. 99–104.
[20] F. Z. Morais, G. M. F. de Almeida, L. Pinto, K. V. Cardoso, L. M. Contreras, R. da Rosa Righi, and C. B. Both, “Placeran: Optimal placement of virtualized network functions in beyond 5g radio access networks,” *IEEE Transactions on Mobile Computing*, vol. 22, no. 9, pp. 5434–5448, 2022.
[21] 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 Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 389–405, 2023.
[22] W.-H. Ko, U. Ghosh, U. Dinesha, R. Wu, S. Shakkottai, and D. Bharadia, “Edgeric: Empowering real-time intelligent optimization and control in nextg cellular networks,” in *21st NSDI 24*, 2024, pp. 1315–1330.
[23] T. Pamuklu, M. Erol-Kantarci, and C. Ersoy, “Reinforcement learning based dynamic function splitting in disaggregated green open rans,” in *IEEE International Conference on Communications*, 2021, pp. 1–6.