QoS-Aware Dynamic CU Selection in O-RAN with Graph-Based Reinforcement Learning

Sebastian Racedo and Brigitte Jaumard
Computer Science and Software Engineering
Concordia University
Montreal (Qc) Canada
brigitte.jaumard@concordia.ca

Oscar Delgado
Systems engineering
Ecole de Technologie Supérieure (ETS)
Montreal (Qc) Canada

Meysam Masoudi
Ericsson
Kista, Sweden

***Abstract*—Open Radio Access Network (O-RAN) disaggregates conventional RAN into interoperable components, enabling flexible resource allocation, energy savings, and agile architectural design. In legacy deployments, the binding between logical functions and physical locations is static, which leads to inefficiencies under time-varying traffic and resource conditions. We address this limitation by relaxing the fixed mapping and performing dynamic service function chain (SFC) provisioning with on-the-fly O-CU selection. We formulate the problem as a Markov decision process and solve it using GRL-DyP, i.e., a graph neural network (GNN)–assisted deep reinforcement learning (DRL). The proposed agent jointly selects routes and the O-CU location (from candidate sites) for each incoming service flow to minimize network energy consumption while satisfying quality-of-service (QoS) constraints. The GNN encodes the instantaneous network topology and resource utilization (e.g., CPU and bandwidth), and the DRL policy learns to balance grade of service, latency, and energy. We perform the evaluation of GRL-DyP on a data set with 24-hour traffic traces from the city of Montreal, showing that dynamic O-CU selection and routing significantly reduce energy consumption compared to a static mapping baseline, without violating QoS. The results highlight DRL-based SFC provisioning as a practical control primitive for energy-aware, resource-adaptive O-RAN deployments.**

***Index Terms*—O-RAN, Deep Reinforcement Learning, Graph Neural Networks, SFC Provisioning, Energy Efficiency.**

I. Introduction

The transition to 5G and the trajectory toward 6G are accelerating adoption of the O-RAN architecture, shifting networks from proprietary, monolithic stacks to disaggregated, virtualized, and intelligent network [1]. By decoupling the Radio Unit (O-RU), Distributed Unit (O-DU), and Centralized Unit (O-CU), O-RAN enables flexible placement and scaling of functions while fostering a multi-vendor ecosystem. These capabilities are underpinned by Software-Defined Networking (SDN) and Network Function Virtualization (NFV), which also support resource partitioning via network slicing [2]. As the O-RAN Alliance advances specifications and deployment profiles, the resulting design space offers greater agility but also introduces substantial orchestration complexity across heterogeneous hardware, fronthaul constraints, and time-varying traffic [1].

However, the same flexibility complicates resource allocation and control. Service function chains (SFCs) must be placed, scaled, and steered across heterogeneous compute and transport resources while meeting slice-specific QoS targets, ranging from high-throughput enhanced Mobile Broadband (eMBB) to Ultra-Reliable Low Latency Communication (URLLC) [3]. In practice, many deployments still rely on static deployment and enforce rigid 1:1 bindings among O-RAN components. Such configurations are often derived from offline capacity planning for peak demand, leading to underutilized hardware and significant energy waste during non-peak hours.

The principles of NFV enable resource and routing decisions to be managed by a centralized controller that maintains a global view of the network's state. This opens the door for more intelligent orchestration methods [3]. Although traditional optimization techniques, such as integer linear programs (ILPs), can find optimal solutions, they often struggle to cope with the scale and dynamism of real-world networks [2]. This complex, dynamic trade-off space is an ideal application for Deep Reinforcement Learning (DRL). A DRL agent, in contrast, can learn complex, non-obvious strategies directly from data patterns, managing the multi-objective problem of maximizing service success while minimizing both latency and energy consumption. Besides using DRL, given the natural structure of network-related problems, the use of graph data is usually beneficial. To work directly with this type of data, we utilize Graph Neural Networks (GNNs) [4].

In this paper, we propose an RL framework that leverages a GNN [4] to learn a joint routing and O-CU selection policy. Our agent's architecture is based explicitly on Graph Convolutional Networks (GCNs) [5] that use convolutional network, enabling it to learn from the underlying network topology and real-time state effectively. We evaluate our approach, called "GRL-DyP," using a realistic 24-hour traffic simulation for the city of Montreal. To isolate and quantify the benefits of dynamic O-CU selection, we compare its performance to two baselines: *(i)* Fix-DRL routing-only GNN-GCN DRL policy with fixed, precomputed O-CU placement, and *(ii)* a traditional CSP baseline (Constrained Shortest Path) algorithm that applies a resource constrained shortest-path routing with

This work was supported by NSERC (under project ALLRP 566589-21) and InnovÉÉ (INNOV-R program) through the partnership with Ericsson. We are grateful to Adel Larabi at GAIA, Ericsson Montréal for clarifying some concepts of the current 5G technology.

978-3-903176-75-1 ©2025 IFIP

fixed, precomputed O-CU placement. Our contributions are as follows:

- We formulate the joint SFC routing and O-CU selection problem as a Markov Decision Process suitable for a single-agent DRL framework.
- We design a GNN-based agent that learns a single, robust policy to handle a full 24-hour traffic cycle for all service types.
- We demonstrate through simulation that our agent outperforms a static placement baseline, achieving significant energy savings while respecting strict service-level latency requirements.

The remainder of this paper is organized as follows. Section II briefly reviews past work on the provisioning. Section III describes the an augmented version of a realistic synthetic dataset derived from the city of Montreal (see [6] for the details). Section IV presents the proposed methodology. Section V presents the performance evaluation and discussion of the results, along with their implications. Section VI concludes the paper and discusses future work.

II. Literature Review

The increasing complexity of network management, particularly in dynamic environments like O-RAN, has spurred significant research into machine learning (ML) and DRL techniques. While supervised ML offers powerful tools for prediction, DRL has emerged as a key enabler for autonomous decision-making in many areas, one of them being networking [7]. DRL can learn complex control policies directly from experience, optimizing for service-level objectives by maximizing a cumulative reward function [8]. This is particularly advantageous for NP-hard problems, such as SFC provisioning, where traditional optimization methods, like ILP, are not able to scale for real-time operations [9]. To further enhance a DRL agent's ability to reason about network problems, GNNs have been proposed as a natural fit for modeling network topologies [4], allowing an agent to make more informed, topology-aware decisions.

Several works have applied these advanced learning techniques to the O-RAN and NFV domains. Ali and Jammal [10] address the proactive VNF scaling and placement in O-RAN using a multi-stage ML framework. By combining an LSTM traffic forecasting with a DRL agent for placement, their system learns to proactively provision resources, reducing the risk of SLA violations that can occur in reactive systems.

Focusing on the joint SFC embedding and routing problem, Tran *et al.* [11] utilize a Deep Q-Learning (DQL) agent. Their agent learns to make sequential node and path selections to provision SFCs under stringent end-to-end delay and bandwidth constraints. Their work demonstrates that a DRL agent can achieve a request acceptance rate of over 95%, comparable to a traditional heuristic algorithm but with a 10-fold reduction in execution time.

The work by Mei *et al.* [12] introduces a hierarchical DRL framework for RAN slicing. Their approach utilizes a high-level controller operating on a large timescale to configure slice parameters (e.g., minimum and maximum data rates) and a low-level controller on a smaller timescale to perform real-time resource scheduling. This demonstrates the power of hierarchical control for managing problems with different temporal granularities. Similarly, Mollahasani *et al.* in [13] use a nested actor-critic model where a high-level agent decides on the optimal placement of a network function (O-DU or O-CU) to balance observability and latency, while a low-level agent handles the specific resource block allocation.

Addressing higher-level orchestration, Staffolani *et al.* [14] introduce PRORL, a DRL agent that learns to allocate resource units from a central pool to various Points of Presence (PoPs). By framing the problem as a Multi-Objective MDP, their agent learns to balance demand satisfaction, resource utilization, and the operational cost of moving resources, achieving an average improvement of 90% over greedy baselines.

To address the inherent graph structure of computer networks, GNNs have been proposed to enhance learning capabilities. The work by Rusek *et al.* in [7] specifically highlights this potential with their GNN-based model, RouteNet. They demonstrate that a GNN trained on one network topology can successfully generalize to make accurate performance predictions on larger, unseen topologies, a critical capability for real-world networking applications.

While these works establish a strong foundation, several research gaps remain. A common limitation is the choice of topology: solutions are often tested on small networks that seldom exceed a dozen nodes. Secondly, the types of constraints are not always consistent across the board: some solutions consider delay limits for requests without including VNF processing requirements, while others focus on resource allocation without per-flow routing decisions. To address these gaps, our work considers a larger, more realistic network topology and integrates multiple constraints, including latency, bandwidth, CPU capacity, and, critically, the energy consumption of both network nodes and function processing, an aspect often overlooked in prior SFC provisioning studies.

III. Network Topology and Problem Statement

A. Network Topology

The physical network is represented as a directed graph $G = (V, E)$, where V represents the set of physical nodes and E is the set of fiber or microwave links. A subset of these nodes hosts compute resources and can run multiple co-located VNFs (e.g., a O-DU and a O-CU). This subset of nodes is heterogeneous and has bounded CPU, memory and storage resources. A bandwidth capacity and a propagation delay characterize each bidirectional link $\ell \in L$. Our scenario includes RUs distributed across the region. These RUs handle both uplink and downlink traffic flows.

Each flow is defined by its service type, source/destination, required bandwidth, and a maximum end-to-end latency constraint. More details on each service SFC can be found in Table I and in [15].

TABLE I: SFC (values adapted from [15])

Service	Class	SFC	Bandwidth	Latency	# Request
Cloud gaming (CG)	eMBB	UPF_{CG}- CU - DU - RU	4 Mbps	80 ms	[40-55]
Augmented reality (AR)	eMBB	UPF_{AR}- CU - DU - RU	100 Mbps	20 ms	[1-4]
VoIP	eMBB	RU - DU - CU - UPF_{VoIP} UPF_{VoIP}- CU - DU - RU	64 Kbps	100 ms	[100-200]
Video streaming (VS)	eMBB	UPF_{VS}- CU - DU - RU	4 Mbps	100 ms	[50-100]
Massive IoT (MIoT)	mMTC	RU - DU - CU - UPF_{MIoT}	[1 -50] Mbps	10 ms	[10-15]
Industry 4.0 (I4.0)	uRLLC	RU - DU - CU - $\text{UPF}_{\text{I4.0}}$ $\text{UPF}_{\text{I4.0}}$- CU - DU - RU	70 Mbps	15 ms	[1-4]

B. Problem Statement

Given the network topology, the set of possible locations for each VNF type, and the 24-hour traffic, the problem is to provision each incoming SFC request dynamically. This involves two joint decisions:

1) O-CU Selection: Selecting a suitable O-CU from the set of pre-deployed candidate nodes to host the flow's O-CU function.
2) Routing: Determining a physical path from the flow's source to its destination while satisfying all the constraints.

A provisioned SFC is considered valid only if it satisfies all of the following constraints: Let p be the selected end-to-end path for a flow, consisting of a sequence of links $\ell \in p$. Let $V_{\text{SFC}} \subset V$ be the set of nodes where the flow's VNFs are placed.

- **CPU Capacity:** For the selected O-CU node, $v_{\text{CU}} \in V_{\text{SFC}}$, its available CPU resources must be sufficient to handle the demand generated by the flow φ:

$$\text{CPU}_{\text{free}}(v_{\text{CU}}) \geq Demand_{\text{CPU}}(\varphi). \tag{1}$$

- **Bandwidth Capacity:** For every link ℓ in the chosen path p, the available bandwidth must be greater than or equal to the flow's requirement:

$$\text{BW}_{\text{free}}(\ell) \geq \text{BW}_{req}(\varphi) \qquad \ell \in p. \tag{2}$$

- **QoS Constraint:** The total end-to-end delay, comprising the sum of propagation delays ($\delta_{prop}(\ell)$) on the links and the processing delays at each VNF node ($\delta_{proc}(v, \varphi)$), must not exceed the flow's maximum allowed latency ($L_{\max}(\varphi)$):

$$\sum_{\ell \in p} \delta_{prop}(\ell) + \sum_{v \in V_{\text{SFC}}} \delta_{proc}(v, \varphi) \leq L_{\max}(\varphi). \tag{3}$$

The overall objective is to learn a policy π that, for each incoming flow, finds a valid provisioning solution that is optimal with respect to our defined reward function. The goal is not to find all possible valid paths, but to find the one that best maximizes the cumulative reward, which encapsulates the trade-off between successfully granting the flow and minimizing the energy cost of the chosen path and O-CU selection. This implicitly maximizes the number of successfully provisioned flows over the 24-hour period by using resources efficiently.

C. Power Model

To evaluate the agent's performance, we model the network's power consumption. For the processing nodes (O-DU, O-CU, UPF), we use the classical and widely used linear power model (P_{node}) proposed by Fan *et al.* [16] which relates CPU utilization (u^{CPU}_{node}) to power draw. For transport nodes acting as routers, we use a simpler model based on idle power (P^{IDLE}_{net}) and per-bit forwarding energy (P^{dyn}_{net}).

$$P_{node} = P^{\text{IDLE}}_{node} + (P^{\max}_{node} - P^{\text{IDLE}}_{node}) \times u^{\text{CPU}}_{node} \tag{4}$$

$$P^{\text{IDLE}}_{net} = \sum_{r \in R} P^{\text{IDLE}}_{r} + \sum_{\ell \in L} P^{\text{IDLE}}_{\ell} \tag{5}$$

$$P^{\text{DYN}}_{net} = \sum_{r \in R} \left(\frac{P^{p}_{r}}{8L} + \frac{P^{\text{SF}}_{r}}{8} \right), \tag{6}$$

where R is the set of routers, L is the set of links, P^p is per-packet processing power, and P^{SF} is per-byte store and forward power.

IV. Methodology

To address the dynamic SFC provisioning challenge, we model the problem as a MDP and simulate an RL environment where the agent sequentially provisions one flow per episode, we assume that the resources of each flow remain in the network per hour, meaning that per hour we reset the overall state of the network. The environment exposes the network graph, resource states, and traffic requests sampled from the 24-h trace, enabling the agent to interact with a dynamic simulator. The agent is trained using Maskable PPO [17], an extension of PPO [8] that handles dynamic action spaces.

A. MDP Formulation

The state $s_t \in S$ comprises *(i)* a task vector with flow metadata and *(ii)* node features encoding topology and resources. The environment also provides an action mask at each step to nullify invalid actions. Details on these components are as follows: *(i)* **Task Vector ('obs'):** A low-dimensional vector with flow-specific information: the service ID, the current hour of the day, the current SFC segment index, the agent's current location index, the target location index for the current segment, the graph distance to the target, and the remaining latency budget. *(ii)* **Graph Features ('node_features'):** An $N \times F$ matrix describing the network, where N is the number of nodes and F is the number of features per node. These features are done per node and include: a multi-hot encoding—a binary vector where multiple entries can be active, indicating all logical functions a node can host (e.g., O-DU, O-CU, UPF)—, the normalized number of connections the node has to other nodes, and the real-time available CPU of the node (if it has compute capacity), which provides a direct signal for network congestion.

We define a joint action space A for the agent, allowing for the joint decision of routing and embedding. The action space is a discrete set of size $D_{\max} + 1$, where $D_{\max}$ is the maximum out-degree of any node in the graph.

- **Routing Actions ($a \in [0, D_{\max} - 1]$):** Correspond to a 'MOVE' action, where the agent traverses a link to a neighboring node.
- **Embedding Action ($a = D_{\max}$):** Corresponds to an 'EMBED' action. This action is only available during the O-DU-to-O-CU routing segment and allows the agent to select its current location for the O-CU function.

The action mask ('action_mask') is a binary vector indicating which actions are valid in the current state, s_t. This mask is generated by the environment at each step to prevent the agent from selecting actions that would violate hard constraints.

The task vector has 7 dimensions; node features form an $N \times F$ matrix (F=5 per node: roles, degree, CPU, etc.). $D_{\max}$ is the maximum node out-degree (8 in our topology), so the joint action space size is $D_{\max} + 1$. Routing is performed per-flow, step by step, until the SFC is provisioned.

To guide the agent towards the dual objectives of QoS satisfaction and energy efficiency, the total reward for an episode is a summation of components awarded at each step. The agent's objective is to maximize the cumulative discounted reward, where the reward at each step $RW(s_t, a_t)$ is composed of several event-driven components. A conceptual formula for the total reward of a complete trajectory τ is:

$$RW_{\text{total}}(\tau) = RW_{terminal} + \sum_{t \in \tau} (rw_{\text{shaping}} + rw_{\text{intermediate}} - p_{\text{energy}}). \quad (7)$$

Each term of the reward function is defined as follows:

- **Dense Shaping Reward (rw_{shaping}):** At every 'MOVE' action, the agent receives a small, immediate reward proportional to the reduction in shortest-path distance to its next target. It also receives a penalty for creating loops by revisiting nodes.
- **Intermediate Success Reward ($rw_{\text{intermediate}}$):** Upon completing a segment (either by arriving at a pre-defined VNF or by performing a valid 'EMBED' action), the agent receives a large positive reward. This reward is penalized by the number of hops taken within that segment to encourage path efficiency.
- **Energy Penalty (p_{energy}):** The intermediate reward for a successful 'EMBED' action is penalized based on the estimated power consumption of the chosen node. Furthermore, the final success reward is penalized by the total energy consumed by all nodes in the chosen path. This directly guides the agent to find energy-efficient solutions.
- **Terminal Reward (RW_{terminal}):** At the end of the episode, a significant positive reward is given for successfully provisioning the entire SFC. A significant negative penalty is given for any action that leads to a terminal state by violating a constraint (e.g., exceeding the latency budget or reaching a dead end).

B. GCN Architecture

Our GCN architecture comprises: **GCN Backbone:** Three GCNConv layers that process the 'node_features' matrix and the graph's 'edge_index' to produce a rich embedding for each node in the network. **Feature Embedding:** Separate embedding layers for the categorical features in the 'obs' vector (service ID, hour, function type, etc). **MLP Heads:** The GNN embedding of the agent's current node is concatenated with the embedded features and fed into a shared MLP. The output of this MLP is then passed to two separate linear layers to produce the policy logits (for the Actor) and the state-value prediction (for the Critic).

V. Results & Discussion

We now discuss the validation of the proposed GRL-DyP models and algorithms.

A. Energy Savings of GRL-DyP

Our scenario consist of 300 O-RUs, 10 O-DUs, 6 O-CUs and 7 UPFs that are distributed across a 5G network. Figure 1 illustrates the initial placement of the logical functions used in our scenario. To do the placement, we first analyzed the daily traffic generated by all service slices, as shown in Figure 2. From this analysis, we identified the 24-hour period with the highest total traffic. This peak-demand data was then used to determine the placement of the logical functions (in our scenario each slice corresponding to a single service type). Traffic traces in Figure 2 are the raw dataset; training samples flows from this day, while evaluation uses disjoint subsets. We evaluate two DRL agents: (1) Fix-DRL, which learns a routing-only policy for a fixed 1:1 O-DU-to-O-CU mapping, and (2) GRL-DyP, which learns the joint routing and dynamic O-CU selection policy (both RL agents were trained for 1M steps using Maskable PPO from Stable-Baselines3 contrib [ref paper or library?], with roll-out buffer 4096, batch 128, lr=$3e^{-4}$, $\gamma = 0.99$, clip=0.2, and GCN hidden dim=64. Training used sampled flows; testing used a separate 24-hour traffic dataset with similar distribution patterns, ensuring evaluation under comparable but unseen traffic conditions. Figures 4–5 report the total energy and latency across the entire subset, not a single request, with flows provisioned dynamically to emulate online arrivals). We compare them against a CSP algorithm that uses the same fixed 1:1 mapping and routes flows via the shortest delay valid path. The simulation uses a realistic 24-hour traffic trace from the city of Montreal, see [6] for the details. Our primary baseline is a heuristic that reflects how a network operator might eagerly route traffic, with all flows served with minimal delay. Figure 3 shows the learning curves for our two DRL agents. Both agents exhibit stable learning, with the average episode reward converging to a high positive value. This demonstrates that the Maskable PPO algorithm, guided by our reward structure, is capable of solving the complex routing and embedding tasks.

The primary results of our study are shown in Figures 4 and 5. The GRL-DyP policy identifies O-CU nodes where embedding can be performed, reducing total energy consumption during evaluation by 1.3% (447.8 kWh) vs. CSP and 2.85% (1,039.5 kWh) vs. Fix-DRL over 24 h, while still meeting all demand and latency constraints.

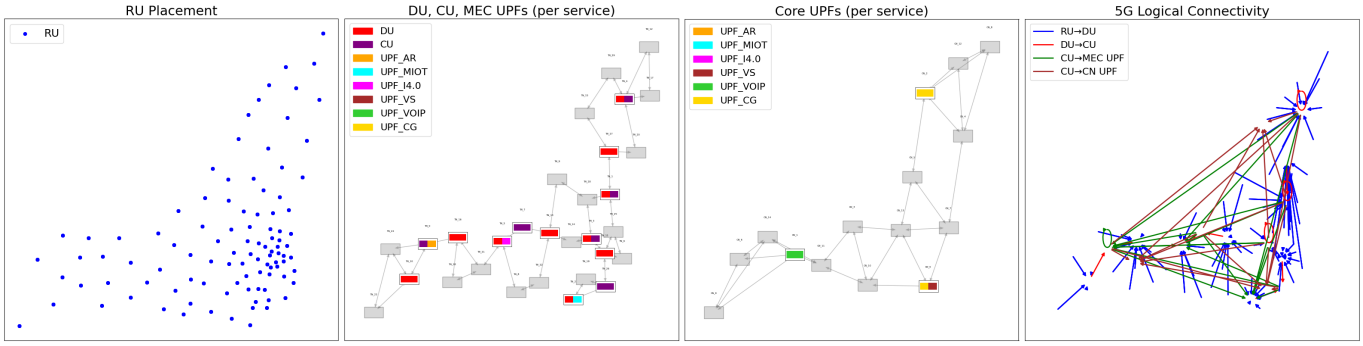

Fig. 1: Fixed placement of logical functions capable of accommodating a dynamic traffic over 24 hours

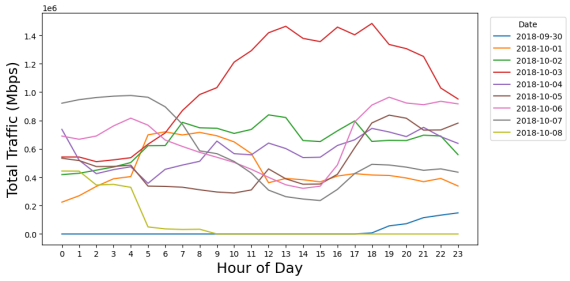

Fig. 2: Overall traffic per hour, by day

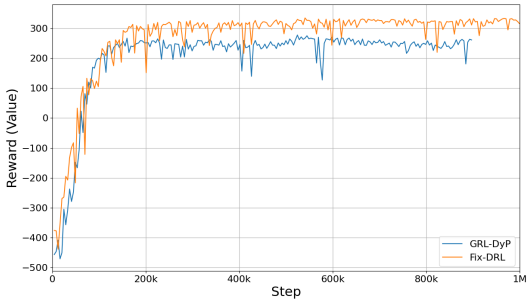

Fig. 3: Learning curves for the GRL-DyP and Fix-DRL

Furthermore, the GRL-DyP is able to learn a routing strategy that is more congestion-aware than the simple shortest-path with constraints baseline. As seen in Figure 5, the agent maintains a consistently similar or lower average latency, particularly during the peak traffic hours.

The proposed policy GRL-DyPachieves significant energy savings without compromising service quality, as confirmed by all six service slices meeting their latency requirements (Table II)

B. Discussion

The results highlight a clear advantage for dynamic, AI-driven orchestration. The primary source of energy efficiency is the agent's learned ability to perform dynamic O-CU selection. By optimizing globally the O-CU selection for each traffic flow, the GRL-DyP policy consolidates traffic (Figure 4, lower off-peak energy) onto fewer active O-CUs, a significant improvement over the static 1:1 O-DU-to-O-CU mapping used by the baselines. While a shortest-path algorithm provides the lowest theoretical delay, our DRL agent learns a more robust, congestion-aware routing policy. This is evident during peak hours, where the agents remain competitive on latency by intelligently routing around network bottlenecks, ensuring QoS is met under dynamic loads (see Figure 5). The choice between these algorithms presents a clear trade-off. A shortest-path baseline is straightforward for non congested networks where energy is not a concern. In contrast, our agent is superior for networks with fluctuating traffic loads and multi-objective operational goals, such as balancing performance and energy cost.

It is worth noting that an actual network has a higher number of nodes/locations/links, this will increase the energy gain that we can see using a AI-driven orchestrator such as the proposed RL agent. Finally, these findings demonstrate that moving from a static, pre-calculated resource allocation model to a dynamic, AI-driven one can yield substantial benefits. Our agent's ability to learn a flexible 1:m mapping proves the value of moving beyond the rigid assumptions of current static O-RAN deployments. This capacity to learn complex, multi-objective behaviors from a reward signal represents a tangible step toward the zero-touch network automation envisioned by the O-RAN alliance, enabling more autonomous, adaptive, and energy-efficient network management.

VI. Conclusion

We have shown that a reinforcement learning agent augmented with a graph neural network can learn a topology-aware, multi-objective policy that jointly performs SFC routing and O-CU selection in an O-RAN setting. By enabling dynamic choice among candidate O-CU locations, the learned policy discovers energy-aware embeddings that outperform static, precomputed placements while preserving QoS guarantees. These results underscore the promise of AI-driven orchestration for realizing the flexibility envisioned by O-RAN supporting autonomous, adaptive, and energy-efficient network management under time-varying demand.

Future work will extend this framework toward a fully multi-agent design for slice-level coordination and incorporate

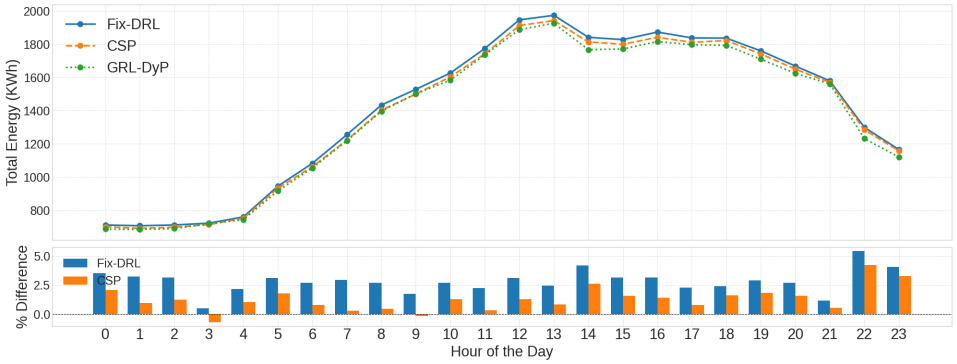

Fig. 4: Total energy consumption per hour comparison

TABLE II: Combined Per-Slice Performance Metrics

Slice	Fix-DRL		CSP		GRL-DyP	
	Avg. Latency (ms)	Avg. Links Used	Avg. Latency (ms)	Avg. Links Used	Avg. Latency (ms)	Avg. Links Used
Augmented reality	6.76	8.27	5.48	8.02	6.82	8.62
Cloud Gaming	4.41	7.08	3.45	6.81	4.46	7.31
Industrie 4.0	3.74	6.36	3.41	5.73	3.75	6.39
Massive IoT	4.86	9.67	4.35	8.36	4.68	8.79
Video streaming	4.51	6.64	4.10	5.99	4.47	6.49
VoIP	2.17	7.26	1.97	6.46	2.05	6.63

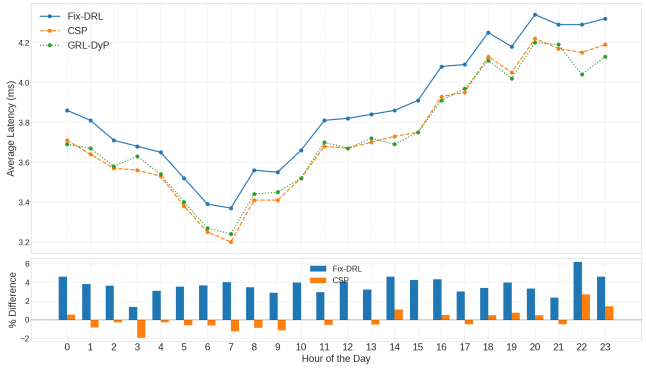

Fig. 5: Average latency per hour comparison

dynamic sleep modes for network nodes to further reduce energy consumption. Additional directions include safety-aware constraint handling, explainability of decisions, and evaluation on hardware-in-the-loop testbeds.

References

[1] O-RAN Alliance, "O-RAN working group 6 (cloudification and orchestration) cloud architecture and deployment scenarios for O-RAN virtualized RAN," 2024.

[2] K. Kaur, V. Mangat, and K. Kumar, "A comprehensive survey of service function chain provisioning approaches in SDN and NFV architecture," *Computer Science Review*, vol. 38, p. 100298, 2020.

[3] 3GPP, "System architecture for the 5G system (5GS)," ThreeGPP, Technical Specification (TS) 23.501, 12 2021, TS 23.501, Version 17.3.0.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 4–24, 2021.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, pp. 1–14, 2016.

[6] B. Jaumard and J. Ziazet, "5G E2E network slicing predictable traffic generator," in *Int'l Conference on Network and Service Management (CNSM)*, 2023, pp. 1–7.

[7] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of graph neural networks for network modeling and optimization in SDN," in *ACM Symposium on SDN Research*, 2019, p. 140–151.

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

[9] B. Jaumard, C. Boudreau, and E. Janulewicz, "Dynamic service function chaining provisioning with reinforcement learning graph neural networks," in *3rd GNNet Workshop on Graph Neural Networking Workshop*. Association for Computing Machinery, 2024, p. 53–58.

[10] K. Ali and M. Jammal, "Proactive VNF scaling and placement in 5G O-RAN using ML," *IEEE Transactions on Network and Service Management*, pp. 174–186, 2024.

[11] T. Tran, B. Jaumard, H. Duong, and K.-K. Nguyen, "Joint service function chain embedding and routing in cloud-based NFV: A deep Q-learning based approach," in *5G World Forum (5GWF)*, Montreal (Qc) Canada, 2021, pp. 171–175.

[12] J. Mei, X. Wang, K. Zheng, G. Boudreau, A. B. Sediq, and H. Abou-Zeid, "Intelligent radio access network slicing for service provisioning in 6G: A hierarchical deep reinforcement learning approach," *IEEE Transactions on Communications*, pp. 6063–6078, 2021.

[13] S. Mollahasani, M. Erol-Kantarci, and R. Wilson, "Dynamic CU-DU selection for resource allocation in O-RAN using actor-critic learning," in *IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[14] A. Staffolani, V.-A. Darvariu, L. Foschini, M. Girolami, P. Bellavista, and M. M. Foschini, "PRORL: Proactive resource orchestrator for open RANs using deep reinforcement learning," *IEEE Transactions on Network and Service Management*, pp. 3933–3944, 2024.

[15] J. Ziazet, B. Jaumard, A. Larabi, and N. Huin, "Placement of logical functionalities in 5G/B5G networks," in *IEEE Future Networks World Forum (FNFW)*, Baltimore (MD) US, 2023, pp. 1 – 7.

[16] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH International Symposium on Computer Architecture (ISCA)*, vol. 35, no. 2, pp. 13–23, 2007.

[17] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *The International Florida AI Research Society Conference (FLAIRS)*, vol. 35, pp. 1 – 6, 2022.