Constraint-Aware Deep Reinforcement Learning for vRAN Dynamic Placement

Gabriel M. Almeida[1], **Mohammad J. Abdel-Rahman**[2,3], **Kleber V. Cardoso**[1]

[1] Universidade Federal de Goiás - Brazil,
[2]Data Science Dept., Princess Sumaya University for Technology - Jordan,
[3]Electrical & Computer Engineering Department, Virginia Tech, USA.

{gabrielmatheus, kleber}@inf.ufg.br; mo7ammad@vt.edu

Abstract. *The disaggregated and virtualized Radio Access Network (vRAN) is already present in 5G but tends to have increased adoption in 6G, mainly in the context of the Open RAN (O-RAN). Despite the potential benefits, the effective success of disaggregated vRAN depends on the efficient placement of Virtualized Network Functions (VNFs), which is influenced by the demand in the Radio Units (RUs). Several factors create dynamicity in this demand, but the number of users served by each RU is one of the most impacting. This problem has already been tackled in the recent literature, however, the works oversimplify important aspects that make their proposals inappropriate for real-world adoption. In this paper, we introduce a Deep Reinforcement Learning (DRL) agent that is constraint-aware, ensuring the solutions' feasibility. We compare our DRL solution with existing optimization models and evaluate it under different scenarios, including the presence of Mobile Edge Computing (MEC) applications that compete for computing resources. Our contributions include a novel formulation, the implementation of a publicly available DRL agent, and insights into practical application scenarios for disaggregated vRAN optimization.*

1. Introduction

The emergence of B5G and 6G has driven the adoption of Network Function Virtualization (NFV) in the Radio Access Network (RAN), creating the vRAN. This transformation involves converting Base Station (BS) functions into distributed VNFs, offering greater flexibility, efficiency, and cost savings. To enhance RAN efficiency, Mobile Network Operators (MNOs) are exploring various strategies, such as standardizing open interfaces, network function disaggregation, and virtualization. The adoption of NFV in vRAN allows MNOs to deploy radio functions on general-purpose hardware, eliminating the need for specialized equipment. This introduces the concept of vRAN nodes, capable of running vRAN functions in three layers: Central Unit (vCU), Distributed Unit (vDU), and Radio Unit (RU). While vRAN nodes reduce costs and enhance RAN control, they also introduce the challenge of optimizing the placement of vRAN functions across the network to meet the RUs demand [3rd Generation Partnership Project (3GPP) 2018].

Among the proposed solutions is the Cloud RAN (C-RAN) that centralizes baseband processing functions in cloud nodes, enabling independent deployment from RUs and integrated through high-speed transport networks. However, the practical feasibility of this architecture is hampered by its dependence on a high-quality transport network, which results in higher costs. In this context, recent advances presented by O-RAN discuss the use of two functional splits, namely Option 7-2x and CU/DU split F1 [Open

RAN Alliance 2022]. Figure 1 depicts the O-RAN functional splits (S0, S1, S2), illustrating the placement of each vRAN function within different vRAN nodes and their transport requirements considering an RU with 100 MHz bandwidth of spectrum, 32 antenna ports, 8 MIMO layers, and 256 QAM modulation.

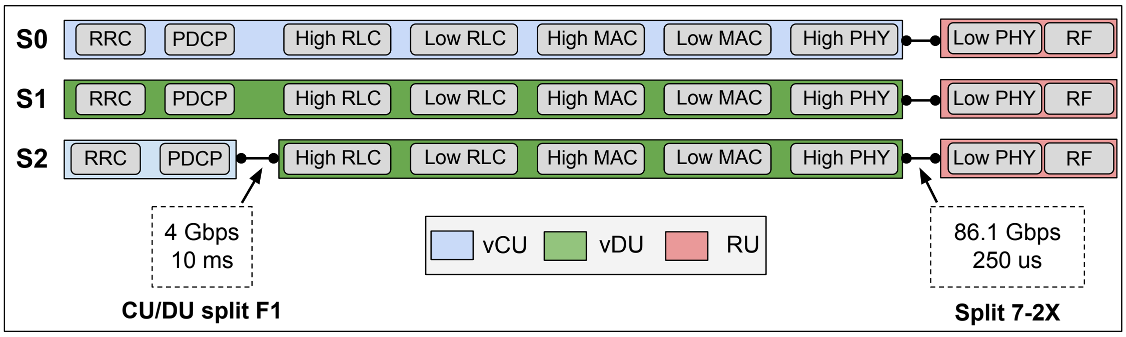


Figure 1. O-RAN functional splits.

1.1. Related Work

The vRAN architecture introduces two key challenges: dynamic network resource allocation and the positioning of virtualized functions. Both problems are addressed in literature as optimization problems, and are solved by classical and stochastic optimization models [Laghrissi and Taleb 2018, Abdel-Rahman et al. 2017], and by non-exact optimization approaches. We refer to this decision problem as vRAN placement problem.

In [Garcia-Saavedra et al. 2018a, Fonseca et al. 2019], the authors provide exact formulations for the vRAN placement problem, solving it optimally through an exact optimization solver. They consider a single CU for placing the vRAN functions, aiming to maximize the vRAN centralization [Larsen et al. 2019]. In [Garcia-Saavedra et al. 2018b], the authors focus on maximizing centralization, considering multiple CUs in the RAN. Although, the authors present improvements in the centralization, their formulation assumes fixed CUs and considers that RUs connected to the same CU must run the same functional split. In this way, the formulation does not explore the capability of the crosshaul to handle dynamic flows, limiting the centralization and cost reduction.

Recent standards [3rd Generation Partnership Project (3GPP) 2018, ETSI 2019] allow up to three vRAN nodes (vCU, vDU, and RU) for dividing the protocol stack, addressed in [Murti et al. 2021, Morais et al. 2023, Almeida et al. 2022b]. In [Murti et al. 2021], the authors optimize the problem using the framework from [Garcia-Saavedra et al. 2018a] with fixed vDU positioning. In [Morais et al. 2023], PlaceRAN is introduced, supporting any number of functional splits, aiming to maximize centralization while minimizing the number of used processing nodes. In [Almeida et al. 2022b] they introduce multipath routing to the transport network based on [Morais et al. 2023], enhancing routing in the transport network.

Considering that the vRAN placement problem is NP-hard [Morais et al. 2023], [Murti et al. 2022, Almeida et al. 2022a, Almeida et al. 2023] presents non-exact approaches. [Murti et al. 2022] introduces a DRL agent finding near-optimal solutions but with a simplistic problem formulation. In [Almeida et al. 2022a], a DRL agent uses the formulation from [Morais et al. 2023], while [Almeida et al. 2023] presents a metaheuristic using a Genetic Algorithm. Both non-exact models efficiently solve larger-scale problems compared to the optimal model.

None of the works presented solve the vRAN placement problem considering the RUs demand variation over time. In this way, in [Murti et al. 2023, Alba and Kellerer 2022, Joda et al. 2022, Pamuklu et al. 2021, Gupta et al. 2022], the authors discuss the importance of considering RUs demand variation. In [Murti et al. 2023], the authors design a DRL agent to solve the vRAN dynamic placement problem with flexible vRAN nodes placement. However, their approach does not guarantee the attendance of the constraints of the problem, making it possible to try the deployment of infeasible solutions. This means that the solutions found by DRL agent may not be feasible, which is a significant concern since the effective operation of the network depends on adhering to these constraints. In [Alba and Kellerer 2022], the authors introduce a framework for calculating positioning costs, addressing the problem by considering variable demand in RUs through a highly complex exact optimization model.

In [Pamuklu et al. 2021], the authors introduce a DRL agent to the vRAN dynamic placement problem. The agent is implemented using Q-Learning and Sarsa methods, aiming to minimize energy consumption, a common objective in O-RAN resource allocation problems [Abdel-Rahman et al. 2023]. However, the authors also do not consider the need to satisfy the constraints. On the other hand, in [Gupta et al. 2022], the authors solve the placement problem considering the problem constraints and demand variation. However, they use a MILP formulation, which presents high complexity and scalability issues, prompting the exploration of non-exact techniques.

Finally, in [Joda et al. 2022], the authors formulate a DRL agent to the problem, considering in the reward function a positive signal if the agent finds a feasible solution and a negative signal (penalty) for infeasible solutions. This approach mitigates but does not solve the issue found in [Murti et al. 2023], only reducing the number of infeasible solutions. The authors also consider a joint optimization problem, addressing both the vRAN dynamic placement problem and user association. Table 1 presents the contributions of each paper described, and compares them to the contributions our work, where we propose an DRL environment to solve the problem respecting the placement constraints.

Reference	Formulation	CUs	Functional splits	Demand	Constrained
[Garcia-Saavedra et al. 2018a]	MILP	Single	Fixed	Static	Yes
[Fonseca et al. 2019]	MILP	Single	Fixed	Static	Yes
[Garcia-Saavedra et al. 2018b]	MILP	Fixed	Fixed	Static	Yes
[Murti et al. 2021]	MILP	Fixed	Fixed	Static	Yes
[Morais et al. 2023]	BLP	Dynamic	Generic	Static	Yes
[Almeida et al. 2022b]	MILP	Dynamic	Generic	Static	Yes
[Murti et al. 2022]	DRL	Single	Fixed	Static	Yes
[Almeida et al. 2022a]	DRL	Dynamic	Generic	Static	Yes
[Almeida et al. 2023]	GA	Dynamic	Generic	Static	Yes
[Gupta et al. 2022]	MILP	Single	Fixed	Dynamic	Yes
[Alba and Kellerer 2022]	MILP	Single	Generic	Dynamic	No
[Murti et al. 2023]	DRL	Dynamic	Fixed	Dynamic	No
[Joda et al. 2022]	DRL	Single	Fixed	Dynamic	No
[Pamuklu et al. 2021]	DRL	Single	Fixed	Dynamic	No
This work	**DRL**	**Dynamic**	**Generic**	**Dynamic**	**Yes**

Table 1. Related work comparison.

1.2. Our Contributions and Paper Organization

In this work, we present a constraint-aware DRL agent formulation for the vRAN dynamic placement problem, considering the ability to deal with any functional split configuration and demand variation over time. In summary, the main contributions are:

- The vRAN dynamic placement problem considering RUs demand variation and heterogeneous computing nodes able to act as both vCUs and vDUs within the vRAN network.
- A constraint-aware DRL environment to address the vRAN dynamic placement problem, employing principles of safe reinforcement learning.
- Comparing our results with two optimization models from the literature, evaluating our agent in practical scenarios.

This paper is organized as follows. Section 2 presents the system model and problem statement. Section 3 presents the vRAN dynamic placement problem formulation. Section 4 introduces the DRL environment. Section 5 presents the evaluation of our DRL agent. Finally, Section 6 concludes and presents future work.

2. System Model and Problem Statement

In this section, we introduce the system model for our work, outlining the sets, parameters, and notation used in our formulation. Subsequently, we present the problem statement by means of an optimization model that tackles the vRAN dynamic placement problem.

2.1. System Model

Within our vRAN system, we consider a set of RUs, denoted by $\mathcal{B} = \{b_1, \ldots, b_{|\mathcal{B}|}\}$, a set of Computing Nodes (CNs) denoted by $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$, responsible for processing vRAN functions acting as vCU, vDU or both, characterized by their processing (c_m^{Proc}) and memory (c_m^{Mem}) capacities. We assume that CNs are spread across the vRAN topology and connected through a set of transport nodes, represented by $\mathcal{R} = \{r_1, \ldots, r_{\mathcal{R}}\}$. Additionally, we assume that CNs are divided into two layers: edge sites, located closer to the RUs but with limited processing capacity and higher processing costs, and cloud sites, farther from the RUs but equipped with higher computing resources at a reduced cost. We also consider the network core, c_0, positioned closer to the cloud site CNs.

We represent the vRAN topology as a graph $G = \{V, E\}$, where $V = \{c_0\} \cup \mathcal{R} \cup \mathcal{B} \cup \mathcal{C}$ represent the nodes, and $E = \{e_{ij} : v_i, v_j \in \mathcal{V}\}$ represent the edges. Each edge is characterized by its capacity and latency, denoted by e_{ij}^{Cap} and e_{ij}^{Lat}, respectively. We consider $\mathcal{P}_l$ as the set of k-shortest paths from the core to each RU $b_l \in \mathcal{B}$, where each path is composed of three sub-paths, representing the backhaul, midhaul and fronthaul links, denoted by p_{BH}, p_{MH} and p_{FH}, respectively.

In this work, we focus on the vRAN dynamic placement problem, considering RUs demand variation. To characterize this time-varying behavior, we employ a time discretization represented by a set $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$. We use $\lambda_{b_i}^t \in [0, 1]$ to represent the normalized demand at RU $b_i \in \mathcal{B}$ at time $t \in \mathcal{T}$. We define a set of functional splits $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$ where each functional split has its processing demand at vCU and vDU, represented as $\gamma_{d_r}^{vCU}$ and $\gamma_{d_r}^{vDU}$, a maximum delay requirement in backhaul, midhaul and fronthaul, represented by $\beta_{d_r}^{BH}$, $\beta_{d_r}^{MH}$ and $\beta_{d_r}^{FH}$, and the worst-case throughput demand in backhaul, midhaul, and fronthaul, represented by $\alpha_{d_r}^{BH}$, $\alpha_{d_r}^{MH}$ and $\alpha_{d_r}^{FH}$, respectively.

2.2. Problem Statement

Given $b \in \mathcal{B}$, $c \in \mathcal{C}$, $d \in \mathcal{D}$, c^{Proc}, c^{Mem}, γ_d, β_d, α_d and λ_b, we answer the following questions jointly:

- What is the positioning of vRAN functions for all RUs considering the current demand?
- Which paths are used to route traffic between functions?
- Which functional split is operating for each RU?
- What are the vCUs and vDUs positioning for all RUs?
- What resources are required in each CN to process the current demand?
- When is the most opportune moment to change the placement solution?
- What is the most cost-effective placement solution?

while ensuring the functional splits thoughput, latency and processing requirements, and the CNs and links capacities.

3. Problem formulation

To formulate the vRAN dynamic placement problem, we introduce a set of decision variables $x_{l,i}^{p,r} \in \{0,1\}$ to represent the choice of path $p \in \mathcal{P}_l$ and functional split $d_r \in \mathcal{D}$ to serve RU $b_l \in \mathcal{B}$ at time $t_i \in \mathcal{T}$. The objective of our formulation is to minimize the total cost, which comprises two key components: the processing cost and the migration cost.

The processing cost is contingent on the volume of vRAN functions executed in each CN and their respective costs based on their layer, whether they are deployed in a cloud or edge site. The processing cost can be computed as:

$$\Phi_{t_i}^{Proc} = \sum_{c_m \in \mathcal{C}} \sum_{b_l \in \mathcal{B}} \sum_{d_r \in \mathcal{D}} \sum_{p \in \mathcal{P}_l} \Big(x_{l,i}^{p,r} y_{c_m,b_l}^{vCU} f_{d_r}^{vCU} \omega_{vCU} + y_{c_m,b_l}^{vDU} f_{d_r}^{vDU} \omega_{vDU} \Big), \tag{1}$$

where $y_{c_m,b_l}^{\bullet}$ denotes whether CN $c_m \in \mathcal{C}$ runs the vCU/vDU node for RU $b_l \in \mathcal{B}$, $f_{d_r}^{\bullet}$ represents the amount of functions running at vCU/vDU for configuration $x_{l,i}^{p,r}$, and ω_{vCU} and ω_{vDU} represents the cost of running a given vRAN function at vCU and vDU.

We can compute the migration cost as the number of functions that are changing their positioning. Thus, the migration cost depends on the prior positioning solution and the current positioning solution. We can formulate the migration cost as:

$$\Phi_{t_i}^{Migr} = \sum_{b_l \in \mathcal{B}} \sum_{d_r \in \mathcal{D}} \sum_{p \in \mathcal{P}_l} \left[x_{l,i}^{p,r} N_{t_i}^{t_{(i-1)}}(x) \rho \right], \tag{2}$$

where $N_{t_i}^{t_{(i-1)}}(x)$ represents the number of functions that change position in the current solution, $x_{l,i}^{p,r}$, compared to the previous one, $x_{l,(i-1)}^{p,r}$. ρ denotes the migration cost.

Finally, we can define the objective function as the minimization of the total cost:

$$\underset{x_{l,i}^{p,r}}{\text{minimize}} \qquad \Phi^{Total} = \sum_{t_i \in \mathcal{T}} \Big(\Phi_{t_i}^{Proc} + \Phi_{t_i}^{Migr} \Big). \tag{3}$$

The following constraints need to be satisfied in the problem. Initially, for each RU at each time, exactly one combination of path and functional split must be selected:

$$\sum_{p \in \mathcal{P}_l} \sum_{d_r \in \mathcal{D}} x_{l,i}^{p,r} = 1, \ \forall b_l \in \mathcal{B} \ t_i \in \mathcal{T}. \tag{4}$$

The aggregated demand in the sub-paths, backhaul, midhaul, and fronthaul, should not exceed links capacity, where $z_{e_{ij}}^{p_\bullet}$ represents if link $e_{ij} \in E$ is in path p_{BH}, p_{MH} or p_{FH}:

$$\sum_{d_r \in \mathcal{D}} \sum_{b_l \in \mathcal{B}} \sum_{p \in P_l} \left[x_{l,i}^{p,r} \lambda_{b_i}^{t_i} \left(z_{e_{ij}}^{p_{BH}} \alpha_{d_r}^{BH} + z_{e_{ij}}^{p_{MH}} \alpha_{d_r}^{MH} + z_{e_{ij}}^{p_{FH}} \alpha_{d_r}^{FH} \right) \right] \leq e_{ij}^{Cap}, \ \forall e_{ij} \in \mathcal{E} \ t_i \in \mathcal{T}. \tag{5}$$

For each functional split, there is a maximum latency demand $\beta_{d_r}^{XH}$ in each sub-path (p_{BH}, p_{MH}, p_{FH}), which depends on the chosen functional split $d_r \in \mathcal{D}$. We represent this constraint as follows:

$$\sum_{e_{ij} \in \mathcal{E}} x_{l,i}^{p,r} z_{e_{ij}}^{p_{XH}} e_{ij}^{Lat} \leq \beta_{d_r}^{XH}, \forall b_l \in \mathcal{B}, p \in \mathcal{P}_l, t_i \in \mathcal{T}. \tag{6}$$

Finally, the processing demand in each CN must not exceed its processing capacity:

$$\sum_{f_s \in \mathcal{F}} \sum_{b_l \in \mathcal{B}} \sum_{D_r \in \mathcal{D}} \sum_{p \in \mathcal{P}_l} \left(x_{l,i}^{p,r} \lambda_{b_i}^{t_i} y_{c_m,b_l}^{vCU} \gamma_{d_r}^{vCU} + y_{c_m,b_l}^{vDU} \gamma_{d_r}^{vDU} \right) \leq c_m^{Proc}, \ \forall c_m \in \mathcal{C}. \tag{7}$$

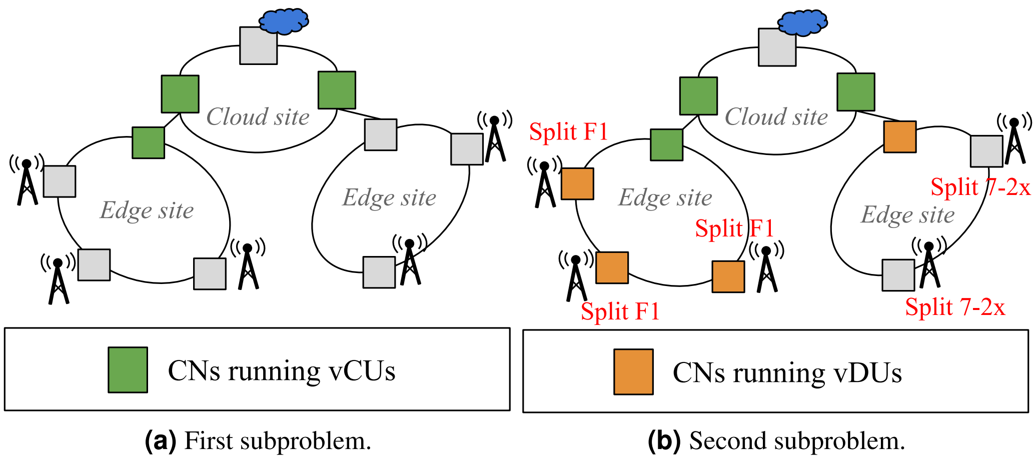
The presented formulation is a Mixed Integer Linear Programming (MILP) model, a class of problems widely acknowledged as NP-hard. Furthermore, it is worth noting that even the non-dynamic formulation of the vRAN placement problem is proved to be NP-hard, with scalability issues [Morais et al. 2023]. Thus, the formulation in this work address the vRAN dynamic placement problem by solving instances for each time window, which remains as challenging as its non-dynamic version. This emphasizes the necessity to explore non-exact approaches, such as AI/ML-based solutions to the problem.

4. Reinforcement Learning Solution

Due to the high complexity of the formulation presented in Section 3, solving the problem optimally, using classical optimization solvers, is computationally expensive. This means that optimization solvers have limited scalability in terms of efficiently solving large instances and consume significant computational resources during execution. On one hand, exact optimization models may not be suitable for practical problem-solving due to their high cost. On the other hand, such models can be widely used as a baseline for comparison and performance evaluation of other methods, such as non-exact formulations.

In this section, we present DRL environment formulation to address the vRAN dynamic placement problem. First, we subdivide the problem formulated in Section 3 into two subproblems, aiming to reduce the number of actions and states in the presented formulation. Then, we discuss the constrained-aware architecture developed in this work.

The vRAN dynamic placement problem is a joint optimization problem, where one of the decisions is the vRAN nodes placement, vCUs and vDUs, in CNs spread across the network. However, the larger the number of processing nodes and RUs, the higher is the complexity to solve the problem, since the placement of vCUs/vDUs is a combinatorial problem with exponential solution space ($\Theta(|\mathcal{C}|^{|\mathcal{B}|})$). In this way, to minimize the complexity of the joint optimization problem, we present a two subproblems division, where the first subproblem address the vCUs placement, and the second subproblem solve the funcional split selection for all RUs in the network. Figure 2 illustrate the subproblems proposed in our formulation.



(a) First subproblem. **(b)** Second subproblem.

Figure 2. Proposed subproblems division.

4.1. First Subproblem: vCU Placement

We represent the actions of the vCU placement subproblem as $z_l^m = \{0, 1\}$, which indicates whether the vCU of RU $b_l \in \mathcal{B}$ is running at CN $c_m \in \mathcal{C}$. In this way, solving the first subproblem defines the vCU placement for all RUs. However, to calculate the solution, we assume that the vRAN current state is known, containing data about the current functional splits selection, the current RUs demand, the current CNs available resources and the transport network resources.

Although the vCU placement presents a combinatorial solution space, this decision problem can be reduced to a classical optimization problem known as the bin packing problem. In the bin packing problem, a set of items, each with a specific size, must be efficiently placed into a limited number of bins, while minimizing wasted space. The objective is to find the most space-efficient way to allocate items, considering the items size, bin capacities, and minimizing the number of bins used. The similarity of both problem allows us to reduce our first subproblem into the bin packing problem, where the vCUs are treated as items, the item size is their processing demand, the CNs serve as bins, and the CNs processing capacity represents the bin capacities. In this way, by solving the bin packing problem, we also solve our first subproblem, i.e., the vCU placement problem.

The reduction presented offers the advantage of transforming our first subproblem into an optimization problem in the literature. However, it is worth noting that the bin packing problem is known to be NP-hard, depending on the number of items and bins. Fortunately, the literature provides effective algorithms for solving the bin packing problem. We use the First-Fit Decreasing heuristic [Dósa 2007] that finds solutions with an approximation ratio of 1.7.

4.2. Second Subproblem: Functional Split Selection

Once the first subproblem addresses the vCU placement, we must now deal with the remaining decisions of the vRAN dynamic placement problem. This entails positioning the vRAN functions at CNs, respecting the positioning constraints. Although considering a two subproblems formulation can enhance RL environment efficiency, both subproblems in our formulation are NP-hard, since even with the vCUs/vDUs positioning previously defined, solve the vRAN placement problem remains NP-hard [Murti et al. 2021]. In this way, to tackle the vRAN dynamic placement problem, we formulate our second subproblem as the positioning of vRAN functions, achieved through the selection of the functional split for each RU in the network. Since both subproblems are solved sequentially, in the second subproblem, we assume the vCU placement defined by the First-Fit Decreasing heuristic in the first subproblem.

The combination of these subproblems allows us to model the second subproblem as an RL environment. The RAN environment exhibits complex and intricate natural behaviors, making it a suitable candidate for representation as an RL environment. Consequently, as we aim to address the vRAN dynamic placement problem, our goal for the RL agent is to optimize the placement of vRAN functions with the primary objective of minimizing processing costs. Additionally, we want the agent to learn when it is cost-efficient to adapt the placement solution based on the varying demand patterns of the RUs.

To address the second subproblem, we formulate a Markov Decision Process (MDP) as $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mu \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ represent the set of states and actions, $\mathcal{R}$ represent the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} = [0, 1]$ captures the transition probabilities between states and actions, and μ signifies the initial state distribution. Within this framework, we introduce a parameterized resource orchestration policy, denoted as $\pi_\theta : \mathcal{S} \rightarrow Pr(\mathcal{A})$, which maps states to probability distributions over actions, where θ denotes the neural network parameters.

We define the state $\mathbf{s}_t$ to include all vRAN current information, representing the vRAN functions positioning, the vCUs positioning, the CNs available resources and the RUs demand. Thus, the policy π_θ generates an action $\mathbf{a}_t$, indicating the selection of functional splits for all RUs in the vRAN, where $a_l^r = \{0, 1\}$ denotes if split $d_r \in \mathcal{D}$ is selected for RU $b_l \in \mathcal{B}$. The reward function is defined as the solution total cost (Equation (3)), considering processing and migration costs.

As the second subproblem represents an NP-hard decision problem, the MDP involves an exponentially large number of actions in each state. We can define the size of the action space as $\Theta\left(|\mathcal{T}| \times |\mathcal{C}|^{|\mathcal{B}|}\right)$. To address this challenge, we developed an Actor-Critic framework using DRL principles to train the agent.

Figure 3 illustrates the proposed DRL framework to the vRAN dynamic placement problem. We implement an Actor-Critic environment to strikes a balance between exploration and exploitation, comprising two key components, the actor (policy network) and critic (value function approximator), this architecture facilitates the vRAN dynamic placement problem learning, where the actor selects actions, while the critic assesses their value, updating the policy optimization. This collaborative approach is suitable for our formulation since we have a large action space and state space, enhancing the learning process due to its interactive behavior.

A constraint-aware solution to the vRAN dynamic placement is essential, since in practice the RAN environment present a series of network requirements, e.g., latency and

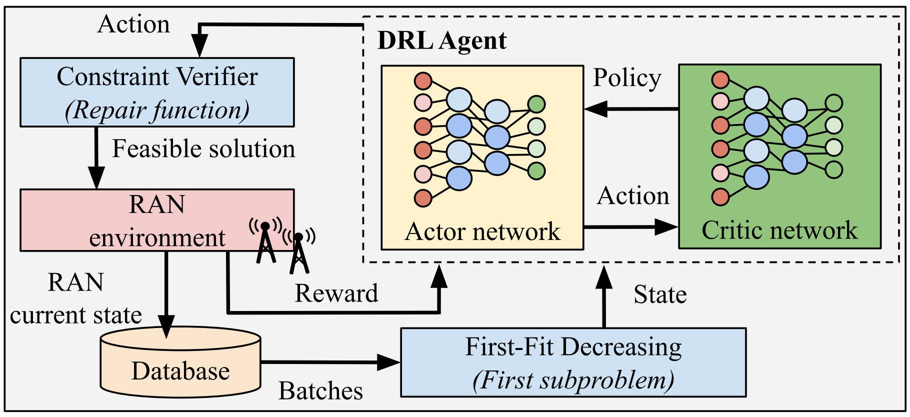


Figure 3. DRL environment of our vRAN-DPRL formulation.

throughput. In this way, while the MDP framework does not explicitly account for these constraints, we have integrated a Constraint Verifier module based in the principles of Safe Reinforcement Learning [Liu et al. 2021]. This module acts as a critical gatekeeper, evaluating the feasibility of the agent selected actions before they are applied in the RAN environment. This proactive approach prevents the introduction of infeasible or disruptive changes into the network, maintaining the integrity and stability of the vRAN placement solution. The Constraint Verifier does not impose a significant overhead on training, as its time complexity is polynomial, defined as $\mathcal{O}(|\mathcal{B}|)$.

The repair function implemented is based on Genetic Algorithms models, where random modifications are applied to the chromosomes of infeasible individuals until they become feasible [Almeida et al. 2023]. This approach is well-established in the literature and yields efficient results as it avoids many infeasible actions mapping to the same feasible solution. Thus, our approach does not impact the agent exploration and exploitation throughout the learning process. Since, the reward returned by the RAN environment is calculated based on the feasible solution applied by the constraint verifier.

5. Evaluation

In this section, we evaluate the DRL agent proposed in Section 4. First, we analyze the training phase of our DRL agent, evaluating the convergence to the optimal policy. Then, we evaluate the solutions found by the agent during the inference phase across various scenarios. All the data and code used in this evaluation is publicly available at GitHub[1].

We evaluate two scenarios: first, a fully provisioned RAN where CNs can process all RUs demand; second, an augmented scenario introducing Mobile Edge Computing (MEC) applications competing with RUs for CNs resources. Our evaluation employs a real-world RAN topology with 51 CNs and 47 RUs connected via an annular-based midhaul. We introduce a cloud site layer with 2 and 4 CNs. In each setting, we compare the DRL agent's solution with the optimal implementation outlined in Section 3.

To implement our DRL environment, we used the Stable Baselines3 version 1.8.0, an OpenAI Gym-based framework to train and evaluate RL agents. We implemented the RL environment using Python version 3.8.10. The optimization models were implemented with the IBM CPLEX solver, version 12.8, docplex library version 2.25.236, and Python version 3.8.10. All experiments were conducted on a computer node equipped

[1] https://github.com/LABORA-INF-UFG/paper-GMK-2024

with an Intel Core i7-10700F CPU @ 2.90GHz, 1 TB SSD, and 32 GB of memory. Table 2 describe the parameters used to evaluate our model in all scenarios.

Parameters	Values
$\lvert\mathcal{C}\rvert$ and $\lvert\mathcal{B}\rvert$	51 CNs and 47 RUs
$\lvert\mathcal{D}\rvert$ and $\lvert\mathcal{P}_l\rvert$	3 split options (Fig. 1) and k-shortest path [Morais et al. 2023]

CNs processing capacity {central site, edge site} (cores)	Fully provisioned RAN: {16, 4}
	Scenario 50% MEC demand: {8, 4}
	Scenario 75% MEC demand: {4, 4}

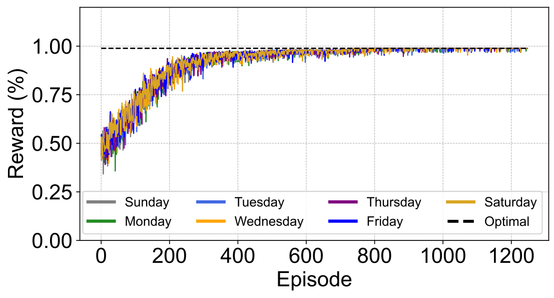
CNs memory capacity {central site, edge site} (GBs)	Fully provisioned RAN: {32, 8}
	Scenario 50% MEC demand: {16, 8}
	Scenario 75% MEC demand: {8, 8}

Dataset	53 weeks of tracing, resulting in 8,904 timesteps

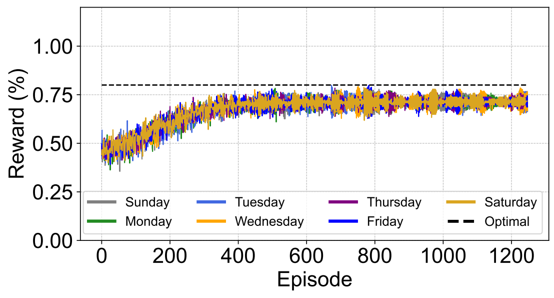
Table 2. Experiment parameters.

5.1. Training Phase

To train our agent, we used a dataset with demand variation data, of 53 weeks and a set of RUs with uniform characteristics, i.e. transmission power and bandwidth. The minimum throughput and maximum latency requirements considered for each functional split are presented in Fig. 1. During training, we pass to the agent the inputs following the demand variation dataset, ensuring that the demand scenarios do not repeat during the training. In this way, we ensure that, during the training phase, the agent will process each day of the dataset only once. This approach showcases the applicability of our solution in an online environment, where metrics are processed in real-time and do not repeat, which makes our solution proper to be deployed at the O-RAN AI/ML framework [Lee et al. 2021].



(a) 4 CNs at cloud site.



(b) 2 CNs at cloud site.

Figure 4. Reward during training phase.

Figure 4 illustrates the reward during the training phase. Note that in the initial 400 episodes, the agent exhibits exploratory behavior since it does not have prior knowledge of the problem. Starting from episode 600, the agent starts to converge to the best-known solution found. However, the agent still demonstrates exploratory behavior, since each episode represents distinct demand scenarios that have not been encountered before. In this way, in each episode, the agent is learning the demand variation pattern of the network, considering the migration cost, which encourages the agent to alter the placement solution only when it is cost-effective rather than changing it solely based on necessity.

Additionally, it is worth noting that the reward curves during the training phase exhibit variations between the scenarios with 4 CNs and 2 CNs at the central site. This variance can be attributed to the decreased availability of processing resources at the cloud site, which affects the solution's cost as there is a need to utilize edge site CNs to meet the demand of the RUs. This variance illustrate that scarce resource scenarios are tricky to train and solve, specially in constrained DRL environments, due to the high complexity landscape in the search space, which impacts the agent's exploration and exploitation.

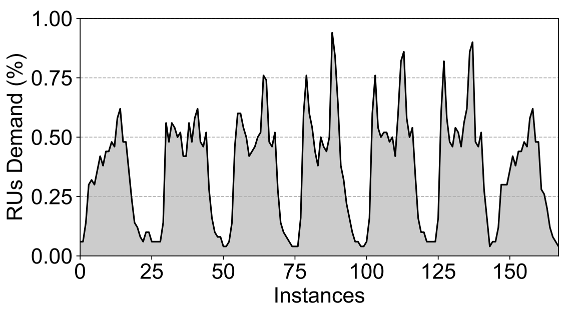
Scenario	Step time (sec)	Episode time (sec)	Training time (sec)
4 CNs at cloud site	0.148002	3.108072	1292.801
2 CNs at cloud site	0.147237	3.092384	1286.119

Table 3. Training time results.

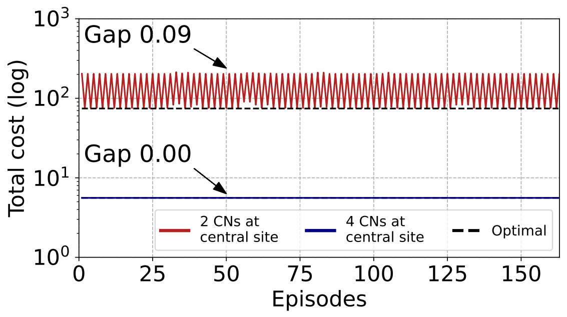
Table 3 presents the training time of our DRL agent. For each scenario, we conducted 30 training runs using the same dataset, and the results in Table 3 represent the average time across these 30 runs. It is noteworthy that our agent consistently completes each step in less than 148 milliseconds. In each step, the agent successfully solves the problem for a given hour of the day. Furthermore, our agent demonstrates the capability to solve each episode in approximately 3 seconds, indicating its efficiency in processing a complete day, i.e., 24 hours of tracing. Notably, to conclude the training phase and process all 53 weeks of tracing, the agent requires less than 22 minutes. In essence, our DRL agent can efficiently process an entire year of data traces in under 22 minutes.

5.2. Scenario of a Fully Provisioned RAN

To evaluate our agent, we consider one week of demand variation data that was not part of the training dataset. To compare the DRL agent solutions, we implemented the optimal model described in Section 3 using CPLEX solver. Given that both the DRL agent and the optimal model solve the same problem, we can assert that the solution obtained by the optimal model represents the best possible solution that the agent can find. In this experiment, we considered two scenarios, with 2 and 4 available CNs at the central site.



(a) RUs demand used in inference phase.



(b) Comparing the DRL agent solutions to the optimal solution.

Figure 5. Fully provisioned RAN results.

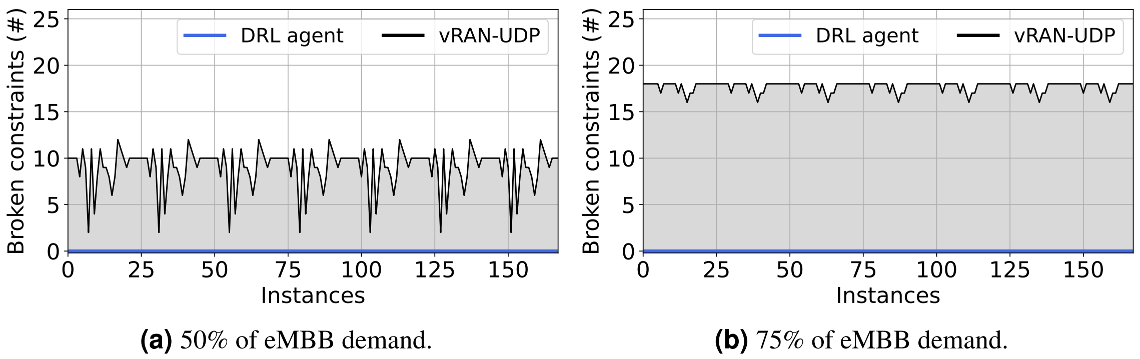
Figure 5 illustrates the results from both simulation scenarios. In Figure 5a, we depict the demand variation employed in both simulations. Additionally, Figure 5b showcases the optimal gap of the solutions for each simulation scenario. The solutions found by the DRL agent are in proximity to the optimal solutions, showing that the optimal policy learned by the agent is representative, leading the agent to find only optimal solutions when solving the problem considering 4 CNs at central site. However, due to the non-exact nature of RL methods, in some instances, the DRL agent may not find the optimal solution. In this case, the agent finds solutions close to optimality, with 9% of gap to the optimum value, when considering 2 CNs at central site. This behavior is expected, since it is more complex to solve the problem in scenarios with scarce resources, and shows the importance of a constrained DRL agent.

On one hand, the solutions found by the agent in the fully provisioned RAN scenario exhibit a maximum gap of 9% from the optimal solution. On the other hand, this behavior may become more critical in scenarios with higher resource competition with different demand variation patterns.

5.3. Scenario with MEC Application

In this scenario, we evaluate our DRL agent in a demand-intensive edge environment, considering the coexistence of MEC applications and vRAN functions. We assume that MEC applications and RUs competes for CNs processing resources addressing two scenarios, with 50% and 75% of MEC processing demand. In these experiments, we also compare our DRL agent solutions with the optimal implementation of a state-of-the-art formulation found in the literature.

In [Murti et al. 2023], the authors formulate the vRAN dynamic placement problem. However, they do not consider problem constraints in a prohibitive manner. Consequently, the solutions found by the proposed model may violate network constraints, albeit incurring penalties. Thus, in this evaluation scenario, we compared the number of constraints broken by our DRL agent with the vRAN Unconstrained Dynamic Placement (vRAN-UDP) model based on [Murti et al. 2023].



(a) 50% of eMBB demand. **(b)** 75% of eMBB demand.

Figure 6. Number of broken constraints for each MEC application scenario.

Figure 6 illustrates the constraints violated by both models, our DRL agent, and the vRAN-UDP. We conducted evaluations under two scenarios, each featuring 50% and 75% utilization of MEC application resources, e.g. eMBB demand. The RUs demand is depicted in Figure 5 (a). In both scenarios, our constraint-aware framework, empowers our agent to identify solutions without violating any problem constraints, ensuring stability and functionality of all RUs. In contrast, the solutions generated by vRAN-UDP fail to exhibit such compliance. This discrepancy arises from the representation of vRAN dynamic placement problem constraints solely as penalties in the objective function, allowing vRAN-UDP to neglect constraints. This constraint violation is more pronounced under conditions of intense competition, as observed in the scenario with 75% of eMBB demand. In this case, vRAN-UDP breaks more than 15 constraints in all instances. Even in the scenario with 50% eMBB demand, although the number of broken constraints decreases, vRAN-UDP still presents RUs with degraded service. This underscores the significance of employing a constraint-aware formulation.

6. Conclusion and Future Work

This work introduced a constraint-aware deep reinforcement learning environment for the vRAN dynamic placement problem. Our results demonstrate that our agent can learn an optimal policy, yielding solutions with a maximum optimality gap of 9%. Furthermore, we conducted practical scenario evaluations, including a fully provisioned RAN and a scenario featuring resource competition due to the presence of MEC applications. A comparison with a state-of-the-art unconstrained model revealed that our agent successfully adheres to constraints across all evaluated problem instances. As future work, we intend to incorporate characteristics of the O-RAN Intelligent Controllers in the formulation to make the proposed DRL agent compliant with the O-RAN specifications. Additionally, exploring the evaluation of network metrics based on a heterogeneous set of users connected to each base station presents an interesting avenue for future research.

Acknowledgments

This work was supported by CAPES, MCTIC/CGI.br/São Paulo Research Foundation (FAPESP) through the Project Smart 5G Core And MUltiRAn Integration (SAMURAI) under Grant 2020/05127-2, by CNPq through the Project Universal under Grant 405111/2021-5, by RNP/MCTIC, Grant No. 01245.010604/2020-14, under the 6G Mobile Communications Systems project, and Program OpenRAN@Brasil.

References

3rd Generation Partnership Project (3GPP) (2018). System Architecture for the 5G (Release 15). Technical report, 3GPP.

Abdel-Rahman, M., MAZIED, E., HASSAN, F., Teague, K., AL-SHAGGAH, A., MACKENZIE, A., Midkiff, S., and Cardoso, K. V. (2023). A stochastic optimization framework for joint RAN intelligent controller placement and RAN nodes assignment in O-RAN networks. *TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.24212046.v1*.

Abdel-Rahman, M. J., Mazied, E. A., Teague, K., MacKenzie, A. B., and Midkiff, S. F. (2017). Robust controller placement and assignment in software-defined cellular networks. In *Proc. IEEE Int. Conf. Comp. Commun Netw (ICCCN)*, pages 1–9.

Alba, A. and Kellerer, W. (2022). Dynamic functional split adaptation in next-generation radio access networks. *IEEE Transactions on Network and Service Management*, 19(3):3239–3263.

Almeida, G. et al. (2022a). Deep reinforcement learning for joint functional split and network function placement in vRAN. In *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, pages 1229–1234.

Almeida, G. M. et al. (2023). A genetic algorithm for efficiently solving the virtualized radio access network placement problem. In *Proc. IEEE Int. Conf. Commun. (ICC)*, pages 1874–1879.

Almeida, G. M., Pinto, L. d. L., Both, C. B., and Cardoso, K. V. (2022b). Optimal Joint Functional Split and Network Function Placement in Virtualized RAN With Splittable Flows. *IEEE Wirel. Commun. Lett.*, 11:1684–88.

Dósa, G. (2007). The tight bound of first fit decreasing bin-packing algorithm. In *Proc. Int. Symp. on Combin. Algor. Prob. and Exper Method.*, pages 1–11. Springer.

ETSI (2019). TS 138 401 – 5G; NG-RAN; Architecture description (3GPP TS 38.401 version 15.5.0 Release 15).

Fonseca, F., Correa, S., and Cardoso, K. (2019). Optimizing Allocation and Positioning in a Disaggregated Radio Access Network Aware of Paths Through the Core Infrastructure. In *Proc. Simp. Bras. Red. Comp. e Sis. Dist. (SBRC)*, pages 791–804.

Garcia-Saavedra, A., Costa-Perez, X., Leith, D. J., and Iosifidis, G. (2018a). FluidRAN: Optimized vRAN/MEC Orchestration. In *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pages 2366–2374.

Garcia-Saavedra, A. et al. (2018b). WizHaul: On the Centralization Degree of Cloud RAN Next Generation Fronthaul. *IEEE Trans Mob Comput*, 17(10):2452–2466.

Gupta, H., Antony Franklin, A., Kumar, M., and Tamma, B. R. (2022). Traffic-Aware Dynamic Functional Split for 5G Cloud Radio Access Networks. In *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, pages 297–301.

Joda, R. et al. (2022). Deep reinforcement learning-based joint user association and CU–DU placement in O-RAN. *IEEE Trans. Netw. Service Manag.*, 19(4):4097–4110.

Laghrissi, A. and Taleb, T. (2018). A survey on the placement of virtual resources and virtual network functions. *IEEE Commun. Surv. Tutor.*, 21(2):1409–1434.

Larsen, L. M. P. et al. (2019). A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks. *IEEE Commun. Surv. Tutor.*, 21(1):146–172.

Lee, H., Jang, Y., Song, J., and Yeon, H. (2021). O-RAN AI/ML Workflow Implementation of Personalized Network Optimization via Reinforcement Learning. In *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, pages 1–6.

Liu, Q., Choi, N., and Han, T. (2021). Constraint-aware deep reinforcement learning for end-to-end resource orchestration in mobile networks. In *Proc. IEEE Int. Conf. Netw. Prot. (ICNP)*, pages 1–11.

Morais, F. Z. et al. (2023). Placeran: Optimal placement of virtualized network functions in beyond 5g radio access networks. *IEEE Trans. Mob. Comput.*, 22(9):5434–5448.

Murti, F., Ayala-Romero, J. A., Garcia-Saavedra, A., Costa-Pérez, X., and Iosifidis, G. (2021). An Optimal Deployment Framework for Multi-Cloud Virtualized Radio Access Networks. *IEEE Trans. Wirel. Commun.*, 20(4):2251–2265.

Murti, F. W. et al. (2022). Constrained deep reinforcement based functional split optimization in virtualized rans. *IEEE Trans. Wirel. Commun.*, 21(11):9850–64.

Murti, F. W. et al. (2023). Deep reinforcement learning for orchestrating cost-aware reconfigurations of vRANs. *IEEE Trans. Netw. Service Manag.*

Open RAN Alliance (2022). Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN. Technical report, O-RAN Alliance. white paper.

Pamuklu, T. et al. (2021). Reinforcement learning based dynamic function splitting in disaggregated green open RANs. *Proc. IEEE Int. Conf. Comm. (ICC)*, pages 1–6.