# Optimized Multi-Agent Reinforcement Learning for Dynamic Open RAN Slicing

**Femi Elegbeleye**[1]

[1]North West University, South Africa

E-mail: phernet123@gmail.com[1]

**Abstract.** Fifth generation (5G) networks enable radio access network (RAN) slicing, allowing a shared physical infrastructure to be partitioned into multiple logical slices, each tailored to meet heterogeneous service requirements. Slice tenants specify service-level agreements (SLAs) that must be satisfied; however, inefficient resource allocation mechanisms frequently result in SLA violations. Recent studies have adopted constrained multi-agent reinforcement learning (MARL) to dynamically allocate RAN resources, yet these approaches often remain inefficient due to reward misalignment with SLA objectives and unstable exploration in high-dimensional state–action spaces. This paper presents a reward-function–centric analysis of constrained MARL for 5G RAN slicing, examining how different reward designs affect learning efficiency and SLA compliance. Based on this analysis, we propose heuristic-guided reward shaping strategies that explicitly align the learning objective with SLA satisfaction and resource utilization efficiency. In addition, we introduce a state-space optimization technique that aggregates local agent states into a weighted global state representation, thereby reducing state dimensionality while preserving critical SLA-related information. Extensive experimental evaluations demonstrate that the proposed approach achieves a 60% reduction in SLA violations compared to an existing constrained MARL model and a 46% reduction compared to a state-of-the-art model-based reinforcement learning approach. Furthermore, the proposed solution converges faster and requires less training time and computational resources, highlighting its effectiveness and practical applicability for efficient 5G RAN slicing.

**Keywords:** 5G Network, Network Slicing, Open RAN Reward Shaping, SLA-Aware Resource

## 1. Introduction

The physical network is divided into virtualized slices that are tailor-made for a specific type of user equipment (UE) [1]. With increased connectivity demands, service providers have difficulty scaling or entering new markets due to high infrastructure costs. Thus, having a technology such as RAN slicing, which allows service providers to purchase tailor-made slices with specific service level agreements (SLA) to meet key performance indicators and adhere to the required quality of service (quality of service), ultimately, these capabilities enable 5G to be networking as a service, as said by the authors of [2]. The network slices share the base core infrastructure, which requires dynamic resource allocation to ensure that all tenant slices adhere to the SLAs and provide quality of service to all connected UEs. There are multiple different radio access network (RAN) architectures. This study will focus on Open RAN, defined in the survey [3] as a RAN architecture that uses disaggregated components to enable multiple

providers of specific components to be used together. This approach has significant advantages. Slice providers can save costs by avoiding vendor locking by using off-the-shelf components.

Despite the benefits of RAN slicing, this technology faces numerous challenges. Thus, advanced machine learning (AML) techniques are used to solve several problems in resource allocation for RAN slicing, as listed by Yaser Azimi et al. [1], issues such as resource sharing, resource virtualization, isolation among slices, mobility management, energy efficiency, security and privacy, dynamic slice creation and management, and algorithmic aspects of resource allocation. However, this study focuses on resource sharing, where two strategies could be used: static or dynamic resource allocation [1]. Each RAN slicing, each service provider will be given a tailor-made slice that meets specific SLAs; these slices are required to provide quality service (QOS) to the end user. Network traffic and resource utilization are challenging to know beforehand, and understanding the network traffic and resource utilization in advance is impossible due to the dynamic nature of the networking environment. Therefore, when a fixed number of resources is allocated to each slice, resources will be wasted due to the over-allocation of resources, and the quality of service will be reduced when a slice is overburdened due to resource under-allocation. Thus, a dynamic resource allocation mechanism is required. Ioannis A. Bartsiokas et al. [4] show that Unsupervised, supervised, and reinforcement learning are types of advanced machine learning that have been applied to dynamic resource sharing. They each have their quirks and pay particular attention to specific aspects of a problem, such as network traffic prediction, admission control, and real-time resource allocation [5].

Real-time resource allocation requires an AML technique that adapts and learns in a dynamic environment and takes real-time action. This describes the core objective of reinforcement learning (RL). Traditional RL suffers from slow convergence and suboptimal performance when the state-action space is complex, and the policy is difficult to define a priori [6]. Hence, deep learning (DL) is coupled with RL to form a more expressive AML called deep reinforcement learning (DRL). Our contribution will be to extend a DRL model to be more efficient and minimize the total SLA violations. We will also provide reward function analysis in the context of resource allocation for RAN slicing, which we believe has not been focused on. Infrastructure analysis of DRL models leads to efficiency gains and deeper insights into the model's inner workings.

Despite all of the above deep reinforcement learning techniques having been employed to address the resource allocation problem in RAN slicing, these models frequently encounter high service level agreement (SLA) violations due to a lack of constraints during model training and exploration, ineffective cooperation among agents in multi-agent scenarios, and inefficient policy search due to unbounded reward functions and unclear objectives in the reward function. Open RAN architecture has become increasingly prevalent due to its advantages in reducing vendor lock-in and enhancing affordability. We propose an improved constraint-driven multi-agent deep reinforcement learning algorithm based on the model CMAPPO [7], which requires strengthening the reward function objectives and optimizing the global state space; this significantly improves end users' service quality and ensures compliance with SLAs for slice tenants, ultimately enhancing RAN infrastructures' overall performance and reliability. Despite the growing adoption of multi-agent reinforcement learning for resource allocation in Open RAN–enabled 5G network slicing, existing studies predominantly emphasize constraint handling mechanisms and algorithmic sophistication, while paying limited attention to reward function design and state-space efficiency. In particular, the impact of reward misalignment with SLA objectives and the scalability challenges arising from high-dimensional state representations remain insufficiently explored. This gap limits learning efficiency, convergence stability, and practical deployability of MARL-based solutions in Open RAN environments.

The paper contributions are as follows:

- Reward Function Design through Heuristics: Demonstrated that domain-specific heuristics in reward function design significantly improve the stability and efficiency of MARL models for Open RAN slicing.

- Optimized MARL Architecture (MAPPO2): Developed an enhanced MAPPO2 model with state-space optimization techniques (weighted aggregation of local states into a global state), reducing training complexity while maintaining high performance.
- Empirical Performance Gains: Provided evidence that MAPPO2 reduces SLA violations by 60% compared to baseline MAPPO and 46% compared to KBRL, proving that architectural refinements yield tangible improvements under constrained computational resources.

The rest of the paper is divided into six sections. Section 1 presents the introduction; Section 2 is the literature review; Section 3 is the methodology; Section 4 is the results; Section 5 is given in section 5; and Section 6 is the concluding section.

A branch of machine learning that finds its roots in Markov decision theory and is applied to decision-making problems. Markov decision theory uses a mathematical modelling framework called the Markov Decision Process (MDP) to solve sequential decision-making under uncertainty [9]. Thus, reinforcement learning focuses on agents learning to make decisions by interacting with their environment and receiving a reward as a feedback mechanism to drive the learning process, which enables agents to optimize their actions over time. The agents use a policy strategy (policy) to pick an action based on its state; not all policies yield the best results; hence, the agents should use an optimal policy that accumulates rewards for maximum return.

$$v_\pi(s) = E_\pi[G_t | s_t = s] \quad (1)$$

$$v_*(s) = \max v_\pi(s) \ \forall s \in S \quad (2)$$

$$\pi_*(s) = \text{argmax} v_\pi(s) \qquad \forall s \in \quad (3)$$

A value function determines how good a policy is; the best value is defined by equation (2), and the best policy is defined by equation (3). The Bellman equation (4) is used to compute the value function, which has a nonlinear recursive component that requires dynamic programming to solve. When the transition T and reward R are known beforehand, we have a full MDP, which makes value iteration (an application of dynamic programming) possible. However, this is usually not the case for vast dynamic environments, which means the agent should conduct a scientific trial-and-error process of exploration and exploitation at each step so the agent can learn from past experiences.

$$v_\pi(S) = R(s,\pi(s)) + \lambda^X T(s,\pi,s') v_\pi(s') \quad (4)$$

Reinforcement learning (RL) has emerged as a powerful paradigm for decision-making in dynamic and uncertain environments. It enables agents to learn optimal or near-optimal behaviors by interacting with the environment and refining their policies based on the feedback they receive. A central concept in RL is the value of a state under a given policy, which estimates the expected cumulative reward the agent can achieve starting from that state and following the policy over time. This estimation balances immediate rewards with discounted future rewards, enabling agents to consider short-term gains and long-term objectives. Transition dynamics, which capture the probabilities of moving between states after taking actions, further shape these estimates and allow agents to anticipate future outcomes in a structured way. Together, these principles form the foundation for RL-based optimization strategies.

In multi-agent reinforcement learning (MARL), these principles are extended to systems with multiple interacting agents, each making independent decisions that collectively influence the environment. This is particularly relevant in complex domains such as 5G Open RAN slicing, where network resources must be allocated across multiple logical slices serving diverse applications and user demands. Here, agents

represent decision-making entities that balance service-level agreements (SLAs) while sharing common physical infrastructure. Efficient agent coordination is essential, as poor decisions can quickly lead to SLA violations and degraded service quality.

Within this context, reward function design plays a pivotal role. Rewards provide the learning signal that drives agent behavior, and their formulation directly affects how well agents align with system-level goals. Poorly designed reward functions can lead to unintended behaviors, such as over-prioritizing one slice at the expense of others. Contrastingly, carefully constructed reward functions can encourage fairness, efficiency, and SLA compliance. Furthermore, state-space representation has a significant influence on learning performance. High-dimensional or redundant state information can slow convergence and increase computational cost, while compact and well-structured state representations enhance efficiency. Techniques such as aggregating local states into a weighted global state improve scalability and enable agents to capture system-wide dynamics more effectively.

Reward function optimization and state-space representation are complementary strategies for enhancing MARL performance in Open RAN slicing. By integrating these improvements, RL-based models can achieve more efficient resource allocation, reduce SLA violations, and deliver robust solutions that scale effectively with network complexity.

Dynamic resource allocation for radio access network (RAN) slicing in 5G and beyond networks has gained significant attention, with deep reinforcement learning (DRL) and multi-agent reinforcement learning (MARL) emerging as key enablers. Several studies have attempted to address the challenge of allocating resources across multiple slices while minimizing service-level agreement (SLA) violations. However, existing approaches often face scalability, flexibility, and constraint handling limitations, making them less effective in dynamic Open RAN environments.

Nguyen Van Huynh et al. [6] proposed a policy-based DRL technique, deep dual learning, designed for dynamic environments with unknown slice demands. Their approach applies to a combinatorial optimization strategy, dynamically allocating radio, computing, and storage resources. The model is trained using random policy exploration before gradually converging to deterministic policies. While effective in handling uncertainty, the unconstrained nature of policy exploration initially results in SLA violations. Moreover, their model only incorporates constraints on total resource availability, neglecting per-slice SLA requirements. For real-time RAN slicing, however, constraining the Markov decision process (MDP) is necessary to prevent SLA violations during training, ensuring stable policies when deployed in live networks.

In another study, the authors of [14] developed an adaptive interior-point policy optimization method by introducing constrained parameters into the MDP tuple. Each action must satisfy feasibility constraints to avoid SLA violations. These constraints are divided into explicit and implicit categories: resource limits, fairness, spectrum availability, antenna configurations, and transmission time. While the approach successfully reduces infeasible actions, it lacks support for a flexible number of slices, an inherent requirement in Open RAN. It does not address per-tenant variations in SLA demands [7]. As a result, this solution risks higher SLA violations in highly dynamic environments where slice numbers and requirements fluctuate.

Flexibility and scalability are fundamental for slice providers, which has motivated further exploration into MARL. Yu Abiko et al. [15] proposed using Ape-X, a distributed MARL algorithm, where agents interact with their environments. In RAN slicing, this translates to assigning an agent per slice. While the approach decentralizes control, it does not constrain agent actions, resulting in potential SLA violations, particularly when introducing new slices.

To address this limitation, Mohammad Zangooei et al. [7] introduced a cooperative learning framework that enhances MARL performance by enabling agents to share aggregated metrics from other slices. By incorporating cooperative behavior, each agent considers global network conditions before allocating resources, thereby improving fairness and reducing SLA violations. Their aggregation method also mitigates the curse of dimensionality, a limitation commonly associated with concatenation techniques. Nonetheless, their framework relies on centralized MARL, restrictive in the distributed Open

RAN architecture, where scalability and resilience require decentralized coordination. Building on cooperative strategies, subsequent research has sought to enrich agent interactions.

The authors of [17] employed an attention-based mechanism during training, prioritizing critical information in cooperative learning. This approach improves training efficiency; however, cooperation is emphasized only during training and not during action execution. Consequently, when agents act independently in real time, the model risks pseudo-cooperative behaviors, leading to SLA violations. In practice, Open RAN requires cooperation that is scoped to avoid greedy agent decisions while preserving slice isolation, ensuring that resource allocation in one slice does not adversely impact another.

A more recent study [18] introduced Multi-Agent Joint Proximal Policy Optimization (MAJPPO), an extension of MAPPO, incorporating a sliding kernel of the global state as a moving average. This method is particularly effective in handling large state spaces with many variables. Similarly, Zangooei et al. [7] used a column-wise weighted sum to aggregate global state construction. However, such approaches are best suited for scenarios with high-dimensional state variables or offline models where autoencoders can be applied to compress the state space into latent representations. In contrast, our study deals with state spaces ranging from 6 to 13 variables, depending on slice type, where such latent approaches are less suitable. We therefore propose using a column-wise average of state variables, which provides a more balanced and interpretable global representation for MARL agents in Open RAN slicing.

In summary, while prior research demonstrates significant progress in applying DRL and MARL to RAN slicing, critical gaps remain. Many models either neglect per-slice SLA constraints, lack flexibility to accommodate varying slice numbers, or rely on centralized frameworks incompatible with Open RAN's distributed architecture. Furthermore, existing approaches to reward function design and state-space representation are limited, often leading to inefficiencies or scalability challenges. Our work addresses these shortcomings by combining optimized reward function heuristics with an aggregated global state representation, enabling scalable, efficient, and SLA-compliant multi-agent learning in dynamic Open RAN slicing environments.

## 2. Related Works

This study models dynamic Open RAN slicing as a constrained multi-agent reinforcement learning (MARL) problem, where each network slice is represented by an autonomous agent responsible for allocating resources in real time. The objective is to minimize service-level agreement (SLA) violations while ensuring scalability and efficiency across heterogeneous and fluctuating slice demands. The system operates within an Open RAN architecture, where a shared pool of spectrum, computing, and storage resources must be distributed among multiple logical slices such as eMBB, URLLC, and mMTC. Each slice is subject to SLA guarantees, and any misallocation can lead to performance degradation, particularly under highly dynamic conditions.

The resource allocation problem is formulated as a constrained Markov Decision Process (CMDP). The state space captures local slice conditions (e.g., traffic load, latency, throughput, buffer occupancy, and channel quality) and aggregates global information, providing system-wide context. The action space represents each agent's resource allocation decisions, expressed in proportional allocations of radio and computing resources. The transition dynamics define how the system evolves after each allocation decision, while the discount factor balances immediate SLA compliance with long-term efficiency. Constraints are embedded within the CMDP to ensure that actions are feasible and do not exceed available resources.

The novelty of our methodology lies in two complementary design choices: constructing an optimized reward function and introducing a state-space aggregation mechanism. Reward shaping plays a decisive role in guiding MARL agents toward desired behaviors. Existing works either apply unconstrained exploration, which exposes the system to excessive SLA violations during training, or adopt rigid constraint enforcement that slows down learning and reduces adaptability. The reward function integrates multiple components in our framework: a penalty for SLA violations proportional to their severity, a positive term for efficient utilization of resources across all slices, and additional penalties for violating

system-level constraints such as total bandwidth or fairness requirements. The reward is therefore sensitive to per-slice SLA satisfaction and aligned with global network efficiency. By embedding these heuristics, agents learn to minimize violations without hindering their ability to explore diverse strategies during training.

The second design choice concerns state-space optimization. In high-dimensional environments such as Open RAN slicing, naïvely concatenating the local states of all slices produces a vast global state that increases training complexity and slows convergence. Previous research has used weighted sums to compress global states, but these risks bias learning toward heavily weighted features. Instead, we propose a column-wise average aggregation. While weighted aggregation provides flexibility, it introduces additional parameters that require careful tuning and may bias the global state toward specific slices or features under dynamic conditions. In contrast, column-wise averaging offers a simple, parameter-free approach that treats all slices uniformly, avoiding unintended dominance of any single metric. Given the relatively low-dimensional state space (6–13 variables per slice), the objective is stable summarization rather than aggressive compression. Moreover, prioritization of critical SLA metrics (e.g., URLLC latency) is already handled in the reward function, making additional weighting in the state representation unnecessary and potentially redundant. corresponding variables across slices are averaged to form a compact and interpretable global state vector. For example, average latency or throughput across all slices provides a useful system-level indicator without overemphasizing any single slice. This approach reduces computational overhead while still preserving critical system dynamics. Each agent can then make decisions based on a combination of its local slice state and the aggregated global state, thereby promoting cooperative learning without undermining slice isolation.

Training uses a policy gradient reinforcement learning algorithm adapted for the multi-agent setting. Agents begin with an exploration phase, during which they experiment with diverse allocation strategies while guided by the optimized reward function that discourages extreme SLA violations. This phase allows agents to gather sufficient experience across various system conditions. As training progresses, exploration gradually transitions to exploitation, where agents refine their policies toward deterministic behaviors, prioritizing SLA satisfaction and efficient utilization. Unlike unconstrained exploration methods, our approach ensures that the policies exposed during training remain feasible and realistic for deployment in live systems. The evaluation environment simulates traffic patterns that range from stable to highly dynamic, with slice numbers and demands varying across time. Depending on the slice type, each slice is modeled with between six and thirteen state variables. This setup allows us to test the scalability of the proposed method under heterogeneous conditions. Our framework is benchmarked against three baselines: a constrained MARL model without reward shaping, a state-of-the-art model-based RL approach, and cooperative MARL frameworks using weighted-sum global states. Performance is assessed using multiple metrics, including SLA violation rate, throughput fairness, convergence time, and computational cost.

The experimental results demonstrate that our method achieves significant performance improvements. SLA violations are reduced by 60% compared to baseline constrained MARL and by 46% compared to the model-based RL method, requiring less training time and computational resources. This highlights the effectiveness of combining heuristic reward design with state-space aggregation in improving MARL performance for Open RAN slicing. The methodology integrates domain-specific heuristics into the reinforcement learning process to achieve robust and scalable resource allocation in Open RAN environments. The proposed solution balances local decision-making with global cooperation, ensuring that agents act efficiently without becoming greedy or violating isolation principles. By reducing training complexity and improving reward alignment, the methodology provides a practical and deployable model for dynamic RAN slicing that addresses the shortcomings of existing centralized or unconstrained approaches.

## 3. Results and Discussion

Three phases of experimentation were used, each focusing on different comparisons concerning the algorithms and number of stages; each theme of the experiments brings different insights into the proposed model. In each phase, 2 or 3 experiments were conducted using the specifications summarized in Table 1:

**Table 1.** Experimental Setup

| Category | Parameter | Value |
|---|---|---|
| eMBB GBR traffic model | UE arrivals | Poisson process with arrival rate = 2 users/min |
| | UE connection time | Exponentially distributed with a mean = 30 seconds |
| | Bit rate | 0.5 Mb/s |
| eMBB non-GBR traffic model | UE arrivals | Poisson process with arrival rate = 5 users/min |
| | UE connection time | Exponentially distributed with a mean = 30 seconds |
| | Burst arrivals | Poisson process with arrival rate = 1 burst/second |
| | Burst length | Exponentially distributed with mean = 500 packets |
| | Burst bit rate | 1 Mb/s |
| mMTC traffic model | mMTC devices | 1000 |
| | Transmission periods | *{*10, 50, 100, 150, 200, 250, 500, 1000*}* seconds |
| | Packet repetitions | *{*2, 4, 8, 16, 32, 64, 128*}* |
| | Packet size | 1000 bits |
| Radio channel parameters | Transmitted power | 30 dBm |
| | Base station (BS) antenna gain | 15 dBi |
| | BS antenna pattern | Section 4.2.1, TS 36.942 |
| | Cell range | 2 Km |
| | Noise figure | 9 dB |
| | Interference plus noise per RB | -110 dBm |
| | Carrier frequency | 2 GHz |
| | Propagation model | Macro cell urban (Section 4.5.1, TS 36.942 |
| | Fading models | Pedestrian A, Typical Urban, Vehicular A (Annex B.2, TS 36.104 |

The metric we focus on is each model's total number of SLA violations and the cumulative violations across the stages. A moving average of the SLA violations is plotted with a window of 1% or 10% of the number of stages. This is done to smooth the plots and create a curve that is better to compare, instead of discrete plots that do show trends or convergence. The base models used in the experiments are Advantageous Actor-Critic (A2C), Single-agent Proximal Policy Optimization (PPO), MAPPO based on [7], and Kernel-based reinforcement learning proposed by [8].

*3.1. Phase 1: Long Stage Runs of MAPPO2 Compared to Base Models*

In phase 1, Short Stage Runs MAPPO2 Compared to Base Models. In this phase, we ran experiments for 500 stages comparing A2C, PPO, MAPPO, and MAPPO2 (our proposed improvements to MAPPO). Figures 1, 2, and 3 present the SLA violation results and cumulative violation trends for Experiments 1, 2, and 3, respectively.

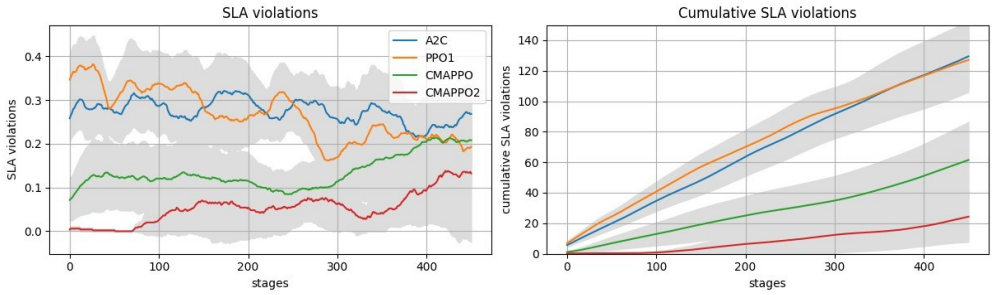


**Figure 1.** Comparative Performance of MAPPO2 and Base Models - Experiment 1

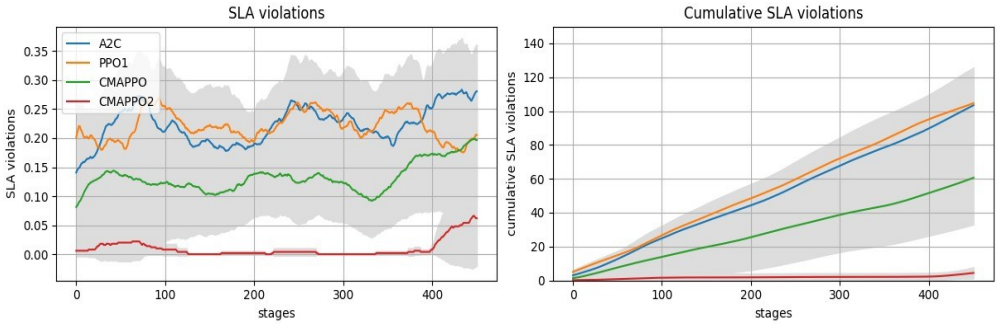


**Figure 2.** Comparative Performance of MAPPO2 and Base Models - Experiment 2

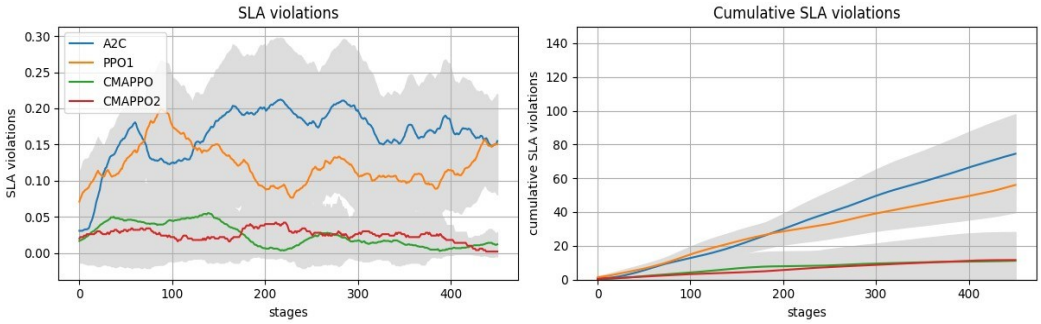


**Figure 3.** Comparative Performance of MAPPO2 and Base Models - Experiment 3

As shown in Figure 3, CMAPPO2 (red line) has the lowest SLA violations between 0.0 and slightly over 0.1, compared to the CMAPPO (green line), which has SLA violations above 0.1. CMAPPO and CMAPPO2 have lower SLA violations than benchmarked single-agent models A2C and PPO. We also see that CMAPPO2 had close to no violations for the first few stages, whilst all other models started with higher SLA violations. The second graph shows the cumulative SLA violation over time, and the trend for CMAPPO is steeper than that of CMAPPO2, therefore having a higher violation rate. The total violation of CMAPPO2 is a little over 20 violations in 400 stages, CMAPPO has about 60 violations, and both A2C and PPO1 have about 130 total violations after 500 stages.

In experiment 4, CMAPPO2 has the lowest SLA violations, which remain flat between stages 100 and 400, while the other models fluctuate and have higher SLA violations. This is depicted by the cumulative

SLA violation graph, where CMAPPO2 has fewer than 20 violations in total, and the sudden spike after stage 400 mainly influences this. What is interesting to see is that between stages 0 and 100, CMAPPO2 had a slight increase in SLA violations but managed to stabilize very quickly, which is a positive effect for a model that will use online learning in a real-world setting. In experiment 5, CMAPPO2 and CMAPPO have similar SLA violation results, which are still lower than A2C and PPO. Keeping violations low from the start, coupled with recovering and stabilizing when an increase in SLA violations happens, plays a significant role in the total violations for CMAPPO2, which is lower on average. The proposed reward function proves to be a helpful improvement from CMAPPO [21], which has its reward function influenced by an agent's action.

### *3.2. Phase 2: Long Stage Runs of MAPPO2 Compared to Base Models*

In this phase, the short-stage 1000 performance of MAPPO2 is compared with KBRL, with results shown in Figures 6, 7, and 8, corresponding to Experiments 6, 7, and 8, respectively. Unlike in phase 1 of experimentation, where CMAPPO2 and CMAPPO were run and evaluated on the same device, this phase uses the performance results presented by the authors in [22], who used a significantly more powerful machine than we did. The results show the potential of the model architecture we propose. In all experiments, KBRL starts with a very high violation and stabilizes, often after 200 stages. This is the number one factor explaining why the model produces a higher cumulative SLA violation. In experiment 6, we see that CMAPPO2 has two significant increases in SLA at around stage 200 and just after stage 700, but with a quick stabilization, showing that the model can learn to reduce its SLA violation if any comes up. Despite these two increases, the model still has a lower cumulative SLA violation than the KBRL model. Experiment 7 shows that the CMAPPO2 model performs significantly better than KBRL; CMAPPO2 has a cumulative SLA violation of less than 20, while KBRL has a cumulative SLA violation of just above 60. This 40% is attributed to CMAPPO2 mostly being stable during evaluation, with only a slight increase during stage 400. In this experiment, the model shows that with better training, it can be stable without any SLA violation spikes. Furthermore, experiment 8 shows our proposed model's extended stability period with very low SLA violations. The policy approximated by the model can learn quickly when slight increases in SLA violations happen. These results make our proposed model comparable to a model-based reinforcement learning model trained and evaluated with a better machine [23]. The proposed aggregation improves interpretability by preserving the physical meaning of each variable (e.g., average latency, throughput), unlike weighted sums, where feature contributions may be obscured. Conceptually, averaging provides a balanced global view of all slices without bias.

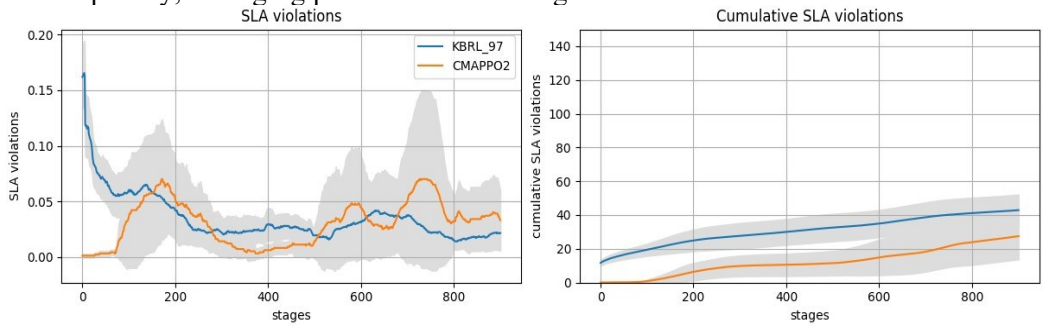


**Figure 6.** MAPPO2 vs. KBRL - Short-Stage Experiment 3

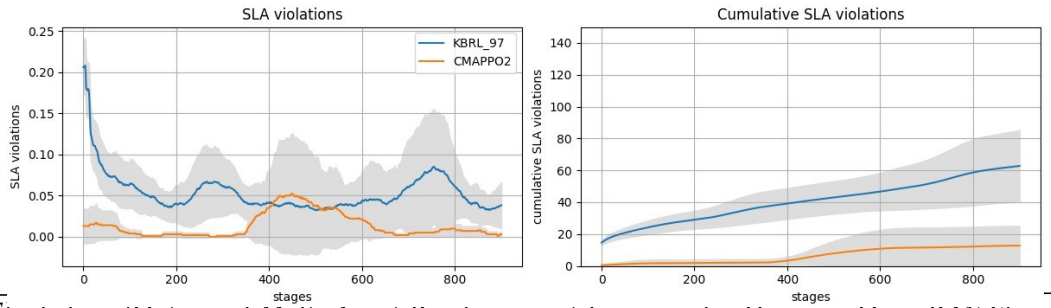

**Figure 7.** MAPPO2 vs. KBRL - Short-Stage Experiment 3

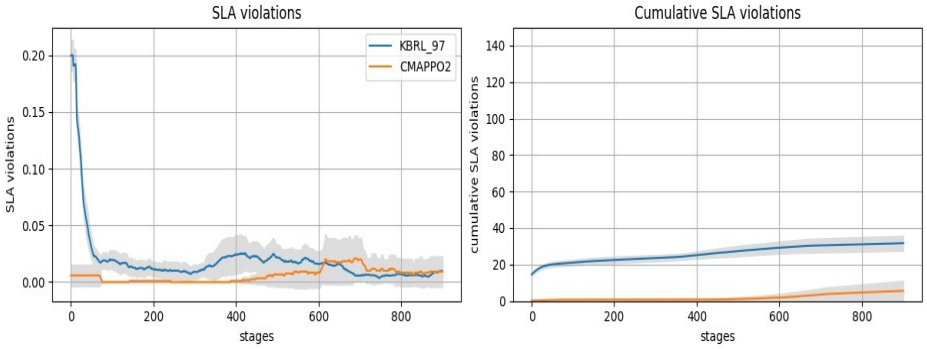


**Figure 8.** MAPPO2 vs. KBRL - Short-Stage Experiment 2

*3.3. Phase 3: Long Stage Runs of MAPPO2 Compared to Base Models*

For this phase, we extended the stages and compared the long-term performance of MAPPO2 with the base models. Figures 9 and 10 present the long-term SLA violation results for Experiment 1 and Experiment 2, respectively.

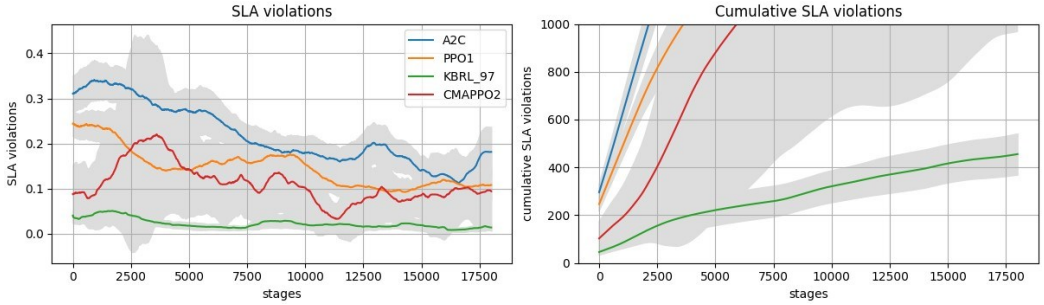


**Figure 9.** MAPPO2 and Base Models - Long Stage Run (Experiment 1)

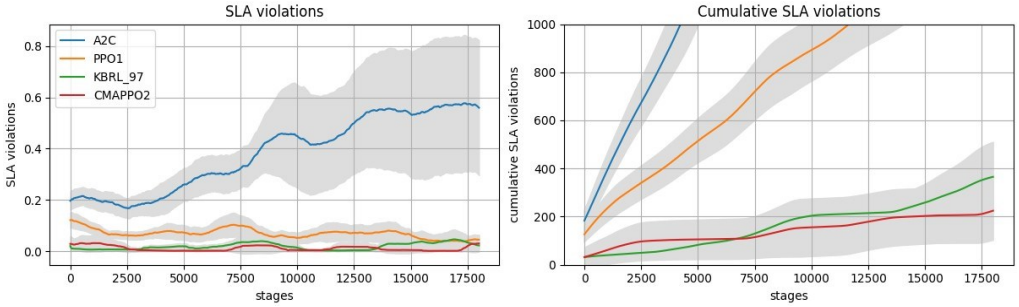


**Figure 10.** MAPPO2 and Base Models - Long Stage Run (Experiment 2)

Therefore, we then ran a second experiment with a different seed and tuned hyperparameters, and we showed significant improvement in the CMAPPO2 model; A2C has an increasing rate of SLA violation, showing no sign of stabilizing. PPO starts with a higher SLA violation than CMAPPO2 and KBRL, but its SLA violations do not decrease fast enough to have a cumulative SLA violation close to the ones attained by CMAPPO2 and KBRL. KBRL maintained a low violation rate and had excellent stabilization from the start. However, CMAPPO2 starts with a slightly higher SLA violation but eventually learns to decrease its SLA violation and stabilize below KBRL, yielding a lower cumulative SLA violation. Phase 3 experiments were challenging due to machine performance and processing power. We often reached

maximum CPU capacity and had to restart the whole experiment. The results are promising, showing that the improvements made on CMAPPO significantly impact the overall model performance. Empirically, Section 3 shows that trends in aggregated variables align closely with SLA violations and reward dynamics, e.g., increases in average latency correspond to higher violation rates, while stable averages align with sustained performance. The model's ability to quickly stabilize after disturbances further indicates that the aggregated state captures essential system dynamics without significant information loss.

*3.4. Phase 4: Long-Term Reward Analysis*

In the final phase, we critically analyse the proposed model's long-term running effects. We have a deeper look at the reward during the evaluation of experiment 8, which will show why the model performed well and where improvements can be made to make the model always perform at its highest level. Figure 11 presents the smoothed long-term reward trend for this evaluation.

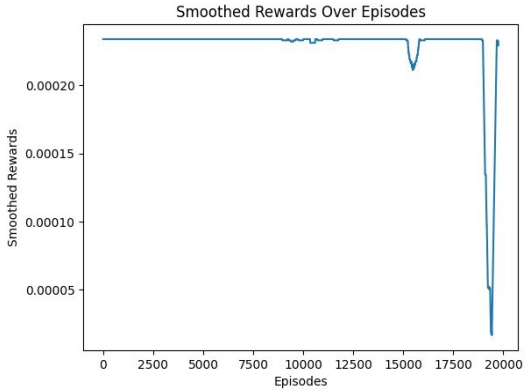


**Figure 11.** Long-Term Reward Analysis - Part 2

Figure 11 shows a smoothed version of the rewards during evaluation. Since rewards in our environment are discrete, we applied a moving average with a window of 250 to make the graph continuous and more precise so that we could see significant points in the longer evaluation run. The graph shows the model having a stable, converged, high reward for an extended period, with slight decreases in reward in some cases, but mostly stable. High rewards represent lower SLA violations; conversely, lower rewards represent high SLA violations.

The model's performance is stable for about 14000 stages and dips around stage 15000; this indicates an increase in violations, which decreases the model's rewards. However, a quick recovery occurs, and the policy manages to guide the model to lower SLA violations again, stabilizing the model. Later on, there's a significant dip in rewards towards the end, which indicates signs of destabilization during edge cases of channel conditions running out of resources, where drastic PRB resources have to be reorganized for all slices; this brings a slight concern that the model needs to have a forward look in admission control to avoid having this drastic dip and having to restabilize again. Figure 12 presents the corresponding discrete graph of rewards:

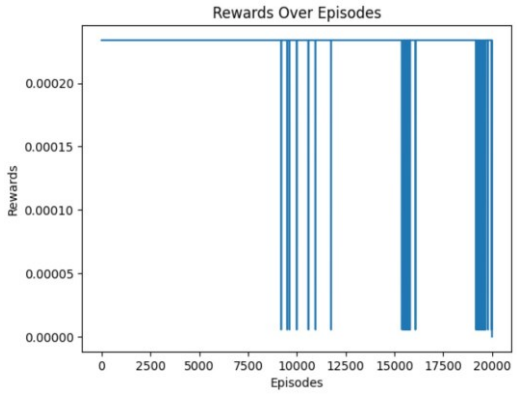


**Figure 12.** Discrete Long-Term Reward Analysis - Part 2

## 4. Conclusion

This work reinforces the argument that multi-agent reinforcement learning (MARL) holds strong potential for addressing the complex challenge of dynamic resource allocation in Open RAN slicing. Unlike prior studies that often focus narrowly on performance metrics without examining the architectural underpinnings of MARL models, we have shown that carefully integrating heuristic-driven reward design and state aggregation into the learning framework yields substantial gains in efficiency, stability, and adaptability. Our enhanced algorithm, MAPPO2, consistently outperformed the baseline MAPPO model by an average of 60%, surpassed KBRL by 46% in shorter training runs, and demonstrated the capacity to closely match the performance of more computationally demanding model-based approaches, even under constrained hardware conditions. Importantly, MAPPO2 exhibited robust stability, sustaining high reward values over extended training horizons and rapidly recovering from occasional increases in SLA violations.

The findings highlight two key insights. First, carefully designing reward functions guided by heuristics that align with domain-specific constraints plays a decisive role in shaping agent behaviors and accelerating convergence. Second, optimizing the state-space representation through aggregation reduces training overhead without sacrificing critical system dynamics. Together, these innovations present a methodology that improves simulation performance and lays the groundwork for more deployable MARL-based solutions in real-world Open RAN environments. While the results are promising, we recognize certain limitations. A broader hyper-parameter search, facilitated by more powerful computational resources, would likely yield further performance and convergence speed improvements. Moreover, as the experiments were conducted within a simulated environment, future studies should extend evaluation to physical or emulated testbeds to better capture the unpredictability and heterogeneity of real-world network conditions.

Looking ahead, future work will explore automated design of reinforcement learning components, such as leveraging genetic programming to optimize reward functions dynamically, and extending the framework to multi-objective optimization problems. Such advancements would better prepare MARL-based solutions for deployment in live 5G and 6G networks, where competing objectives such as energy efficiency, latency guarantees, and fairness must be balanced concurrently. In conclusion, this study demonstrates that a fundamental re-examination of MARL architecture, particularly reward shaping and state-space design, can lead to significant performance improvements. By addressing both efficiency and stability, MAPPO2 represents a practical step toward solving the pressing problem of dynamic resource allocation in Open RAN slicing, bridging the gap between simulation-based research. Please ensure that affiliation is written as completely as possible, including the country. Affiliations are written with 25mm indentation and align with the abstract. If the authors come from different affiliations, superscript numbering must be used after each last name to refer to the author with the affiliation. Superscript numbering should not be entered with a footnote because it will enter references in the wrong place (at the bottom of the page or at the end of the document). Make sure that each superscript numbering connects

the name of the author and affiliation, starting from 1. Do not add footnotes until all author names are linked to their affiliation.

## 5. Future Works

Although MAPPO2 has demonstrated strong performance in simulated environments, several directions remain for further research. A key step is to evaluate the model on real-world testbeds to validate its robustness under practical conditions such as varying traffic, hardware limitations, and diverse user demands. This would also allow benchmarking against industry-grade solutions in 5G and emerging 6G scenarios. Further improvements can be achieved through broader hyper-parameter optimization, supported by large-scale computational resources or distributed training environments. Automating elements of the reinforcement learning pipeline also presents a promising path. For example, genetic programming or neural architecture research could be used to dynamically generate reward functions or optimize state representations, reducing reliance on manual engineering. Finally, extending the model to handle multi-objective optimization problems, such as balancing latency, energy efficiency, fairness, and throughput, would bring the framework closer to practical deployment. These research directions will strengthen the applicability of MARL-based RAN slicing solutions and accelerate their transition from simulation to large-scale, real-world networks.

## 6. References

[1]. Y. Azimi, S. Yousefi, H. Kalbkhani, and T. Kunz, "Applications of machine learning in resource management for run-time slicing in 5G and beyond networks: A survey," *IEEE Access*, vol. 10, pp. 106581–106612, 2022.

[2]. L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong, "Network slicing: Recent advances, taxonomy, requirements, and open research challenges," *IEEE Access*, vol. 8, pp. 36009–36028, 2020.

[3]. S. K. Singh, R. Singh, and B. Kumbhani, "The evolution of radio access network towards Open RAN: Challenges and opportunities," pp. 1–6, 2020.

[4]. I. A. Bartsiokas, P. K. Gkonis, D. I. Kaklamani, and I. S. Venieris, "ML-based radio resource management in 5G and beyond networks: A survey," *IEEE Access*, vol. 10, pp. 83507–83528, 2022.

[5]. H. P. Phyu, D. Naboulsi, and R. Stanica, "Machine learning in network slicing—a survey," *IEEE Access*, vol. 11, pp. 39123–39153, 2023.

[6]. N. Van Huynh, D. Thai Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and fast real-time resource slicing with deep dueling neural networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[7]. M. Zangooei, M. Golkarifard, M. Rouili, N. Saha, and R. Boutaba, "Flexible RAN slicing in Open RAN with constrained multi-agent reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 280–294, 2024.

[8]. J. J. Alcaraz, F. Losilla, A. Zanella, and M. Zorzi, "Model-based reinforcement learning with kernels for resource allocation in RAN slices," *IEEE Transactions on Wireless Communications*, vol. 22, pp. 486–501, 2023.

[9]. C. C. White and D. J. White, "Markov decision processes," *European Journal of Operational Research*, vol. 39, no. 1, pp. 1–16, 1989.

[10]. Y. Li, "Deep reinforcement learning," 2018.

[11]. R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," vol. 12, 1999.

[12]. F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," vol. 30, 2017.

[13]. L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 38, no. 2, pp. 156–172, 2008.
[14]. Y. Liu, J. Ding, and X. Liu, "A constrained reinforcement learning-based approach for network slicing," in *Proc. IEEE ICNP*, pp. 1–6, 2020.
[15]. Y. Abiko et al., "Flexible resource block allocation to multiple slices for radio access network slicing using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 68183–68198, 2020.
[16]. D. Horgan et al., "Distributed prioritized experience replay," 2018.
[17]. F. Lotfi, F. Afghah, and J. Ashdown, "Attention-based Open RAN slice management using deep reinforcement learning," pp. 6328–6333, 2023.
[18]. W. Zhao et al., "Research on the multi-agent joint proximal policy optimization algorithm controlling cooperative fixed-wing UAV obstacle avoidance," *Sensors*, vol. 20, no. 16, 2020.
[19]. I. Vilà, J. Pérez-Romero, O. Sallent, and A. Umbert, "A multi-agent reinforcement learning approach for capacity sharing in multi-tenant scenarios," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9450–9465, 2021.
[20]. N. Ghafouri et al., "A multilevel deep RL-based network slicing and resource management for O-RAN-based 6G cell-free networks," *IEEE Transactions on Vehicular Technology*, pp. 1–12, 2024.
[21]. M. Sulaiman et al., "Coordinated slicing and admission control using multi-agent deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1110–1124, 2023.
[22]. M. K. S. Sibeko, "Network slicing custom code and models," 2024. [Online]. Available: https://github.com/MkSibeko/RANSLICING/
[23]. F. Elegbeleye, M. Mbodila, A. Mabovana, and O. Esan, "Data privacy on using four models: A review," in *International Conference on Electrical, Computer and Energy Technologies*, pp. 1–6.