International Journal of Technology

http://ijtech.eng.ui.ac.id

Research Article

Hybrid Reinforcement Learning-Enabled Scheduler for 5G Burst Traffic

Mohamed Mohsen Farouk[1], Chung Gwo Chin[2*], Mardeni Roslee[2], Lee It Ee[2], Pang Wai Leong[3]

[1]Engineering Science, Faculty of Artificial Intelligence and Engineering (FAIE), Multimedia University, 63000 Cyberjaya, Selangor, Malaysia

[2]Faculty of Artificial Intelligence and Engineering (FAIE), Multimedia University, 63000 Cyberjaya, Selangor, Malaysia

[3]School of Engineering Faculty of Innovation & Technology, Taylor's University, 47500 Subang Jaya, Selangor, Malaysia

[*]Corresponding author: gcchung@mmu.edu.my; Tel.: +603-8312 5309; Fax.: +603-8318 3029

Abstract: Resource allocation in wireless networks is inherently complex, a problem intensified in 5G by heterogeneous traffic classes and stringent quality of service (QoS) requirements. This challenge poses significant difficulties for traditional scheduling methods. In this study, we address these limitations using novel hybrid reinforcement learning (RL) architectures evaluated in a dynamic and realistic network environment. We designed and implemented three hybrid RL algorithms: Asynchronous Advantage Actor-Critic integrated with Proximal Policy Optimization (A3C-PPO), A3C with Proximal Policy Optimization and Session Persistence (A3C-PPO-Persistent), and A3C with Twin Delayed Deep Deterministic Policy Gradients (A3C-TD3). These were compared against baseline A3C and Advantage Actor-Critic (A2C) approaches, as well as traditional proportional fair (PF), maximum rate (MR), and Round Robin (RR) schedulers. Simulations were performed in a challenging multicell environment with mobile user equipment and bursty traffic flows across four network traffic types: ultra-low latency (ULL), voice over IP (VoIP), vehicle-to-everything (V2X), and video streaming. Our hybrid RL schedulers showed promising performance in this highly dynamic setting, with A3C-PPO achieving the most balanced overall results, exhibiting 25%–40% lower average jitter and over four times higher packet delivery ratio (PDR) than traditional schedulers under heavy loads. Our results indicate that hybrid RL methods, particularly A3C+PPO, can provide resilient adaptive scheduling that can outperform both conventional techniques and standard RL algorithm models in realistic 5G networks.

Keywords: 5G; Advantage Actor-Critic; Burst traffic; Reinforcement learning

1. Introduction

The evolution of wireless network technology toward 5G and beyond has enabled a variety of new use cases that leverage the improvements afforded by higher throughput, lower latency, and improved reliability (Bikkasani and Yerabolu, 2024; Tan et al., 2022) but have also brought unprecedented complexity in resource management (Benmadani et al., 2025; Tsoulos et al., 2024; Bozis et al., 2024; Carneiro et al., 2023; Gedikli et al., 2022) The fundamental differences in the quality of service (QoS) requirements across the heterogeneous 5G traffic classes, such as enhanced mobile broadband (eMBB), necessitate a high level of throughput (Ibrahim and Ali, 2021; Navarro-Ortiz et al., 2020); ultrareliable low-latency communications (URLLC), which demands near-perfect reliability with sub-1 ms latency (Ibrahimi et al., 2024; Ibrahim and Ali, 2021); or massive machine-type communications (mMTC), which is meant to be traffic for high-density low-power user equipment (UEs) (Ibrahim and Ali, 2021; Navarro-Ortiz et al., 2020).

https://doi.org/10.14716/ijtech.v17i3.8208
Received November 2025; Revised December 2025; Accepted February 2026

Artificial intelligence (AI) and machine learning (ML) are viable approaches for creating effective and flexible network schedulers, with reinforcement learning (RL) recognized as the preeminent type of ML for wireless resource management (Alanazi et al., 2025; Akyildiz et al., 2024). Many popular deep reinforcement learning (DRL) algorithms, including asynchronous advantage actor-critic (A3C) and advantage actor-critic (A2C), may experience training instability during high traffic events and can suffer from value estimation inaccuracies in continuous action spaces (Raza et al., 2025; Del Rio et al., 2024). These challenges must be addressed in the complex, time-varying traffic conditions of real-world 5G networks (Konstantoulas et al., 2025). This suggests that there is significant potential for developing more sophisticated RL approaches, particularly hybrid RL architectures that could better address the unique demands of 5G resource allocation, as shown in the literature (Yan et al., 2025; Sánchez et al., 2022).

In this study, we designed, implemented, and rigorously tested a modular RL-based scheduler for 5G wireless networks, leveraging the A3C framework and multiple based hybrid algorithms (A3C-TD3, A3C-PPO, and A3C-PPO-Persistent) to maintain QoS. Our approach demonstrated transformative performance in high-congestion scenarios while revealing critical trade-offs in scalability and traffic-type optimization while being deployed in an NS3 simulated multi-cell environment with dynamic traffic patterns and diverse traffic types (ULL, VoIP, video, and V2X).

1.1 Related work

Despite the progress made in wireless network research, the task of providing connectivity to heterogeneous 5G networks while maintaining their stringent and varying QoS requirements remains critical and requires further exploration (Guo and Xie, 2025; Bozis et al., 2024; Tsoulos et al., 2024). Mollahasani et al., 2022 proposed two actor-critic-based network schedulers that utilized an Advantage Actor-Critic (A2C) RL algorithm equipped with a multipart reward function, which included channel, delay, and URLLC priority, to perform resource allocation. Their approach outperformed traditional network schedulers in several key metrics, including head-of-the-line (HOL) delays and packet delivery ratio (PDR). Similarly, Wai et al., 2021 designed a model that leverages RL models to perform resource allocation in a 5G network by utilizing predictive analytics to anticipate varying conditions across the network. This work again signifies RL's ability to incorporate real-time feedback from its environment, enabling flexible and dynamic RL-equipped schedulers.

Similar to Mollahasani et al., 2022, the authors in Alsenwi et al., 2021 designed an effective scheduling model for eMBB and URLLC traffic in 5G networks using an optimization-based network slicing approach. The authors employed a Gradient-Based Actor-Critic Learning (PGACL) algorithm that demonstrated a high level of reliability after a relatively short learning period. However, their approach was QoS-unaware and incorporated a high level of complexity.

In (Zhou et al., 2021) and (Elsayed and Erol-Kantarci, 2019), the authors used multiagent RL to address the coexistence of heterogeneous traffic classes in 5G wireless networks. In (Zhou et al., 2021), an RL algorithm was employed for inter-slice resource allocation in URLLC and eMBB traffic with subordinate RL agents responsible for resource allocation within their respective slices. Elsayed and Erol-Kantarci, 2019 employed a decentralized approach to learning, with each next-generation Node B (gNB) adopting a Q-learning algorithm. The gNBs act as agent networks, enabling each gNB to take actions and allocate joint power and resource blocks to its attached UEs. Similar to Elsayed and Erol-Kantarci, 2019 and Zhou et al., 2021, Seid et al., 2021 employed a multi-agent deep RL (MADRL)-based approach to address 5G network resource allocation and scheduling. However, Seid et al., 2021 aimed to deploy unmanned aerial vehicles Unmanned Aerial Vehicles (UAVs) as mobile edge base stations to offload the Internet of Things (IoT) and massive machine-type communications (mMTC) resource allocation tasks.

2. Methodology

Our simulated multicell 5G network was built using Network Simulator 3 (NS-3); the NS3-GYM (Gawłowicz and Zubow, 2019) module was also utilized to implement our RL-enabled scheduler. Furthermore, the 5G Lena module was used to model and implement 5G network standards and mechanisms (Koutlia et al., 2022; Patriciello et al., 2019). Our simulation environment was built to emulate a bursty traffic model with dynamic packet intervals to simulate more realistic and time-varying network demands while accounting for path loss and shadowing.

We populated our network with the following four distinct user types representing some main 5G network traffic categories:

1. Ultra-Low Latency (ULL): All packets of 512 bytes with low latency.
2. Voice over IP (VoIP): Small packets of 256 bytes with stringent delay requirements.
3. Video Streaming: Larger 4 KB packets with moderate latency tolerance.
4. Vehicle-to-Everything (V2X): Mobile UEs represented 10% of the total UE count in every simulation and are represented in the results as part of the ULL traffic due to the nature of their QoS requirements.

These traffic types were randomly assigned to UEs based on a weighted percentage algorithm and allowed to connect to their closest gNB. The mobile V2X UEs were allowed to disconnect and reconnect from the gNBs as they moved.

2.1 Dynamic packet intervals (DPIs)

As depicted in Figure 1, our simulated environment used a bursty traffic model to emulate a more realistic time-varying network. This was achieved using exponentially distributed inter-arrival times. Furthermore, to maintain fairness and comparability between runs, we used a fixed average load (λ) function for each traffic class (ULL, VoIP, Video, and V2X). the following expression explains this approach:

$$t_{\text{inter}} = \frac{1}{\lambda}(U), \tag{1}$$

where t_{inter} is the inter-arrival time, $U \sim \text{Uniform}(0, 1)$, and λ is the fixed average load. As shown in Figure 1, this approach allows the following two states for every UE in the network:

1. Burst traffic when $t_{\text{inter}} \ll \frac{1}{\lambda}$ leading to consecutive short intervals.
2. Idle period when $t_{\text{inter}} \gg \frac{1}{\lambda}$ leading to sparse transmission.

The following probability density function governs this variability:

$$f(t) = \lambda e^{-\lambda t}, \tag{2}$$

The proposed design allowed us to maintain a fixed average traffic load through runtime, with periods of high burst density $\sum t_{\text{inter}} < \mathbb{E}[t]$, which are counterbalanced by idle phases $\sum t_{\text{inter}} > \mathbb{E}[t]$, and governed by the following expression:

$$\frac{1}{T}\int_0^T \rho_{\text{transmit}}(t)\, dt = \lambda, \tag{3}$$

Where $\text{I}_{transmit}$ denotes a packet transmission event. This scenario ensures realistic environmental conditions while maintaining consistency through a controlled variance, $\sigma^2 = \frac{1}{\lambda^2}$. These spontaneous burst traffic transmission stress-test schedulers preserve comparability through identical λ values between the iterations.

2.2 Reinforcement learning (RL) framework

In the early stages of this work, we tested two RL algorithms using the same reward function and the designed general network scheduler framework. The first RL algorithm that was tested was Asynchronous Advantage Actor-Critic (A3C). A3C uses a parallel implementation of the advantage actor-critic algorithm with multiple instances to increase efficiency and training stability (Mehta, 2020; Sewak, 2019). These instances computed policy and value updates asynchronously through their own local network copies and then pushed gradient updates to a shared global network (Mnih et al., 2016).

The second RL algorithm we tested was A2C, which used synchronous updates. Unlike A3C's decentralized approach, Advantage Actor-Critic (A2C) employs a single centralized coordinator to aggregate gradients from all agents after identical environmental steps, updates its global parameters, and then synchronizes all copies simultaneously (Sewak, 2019). Although both algorithms are similar and effective in continuous environments (Mollahasani et al., 2022), the asynchronous approach of A3C leads to better computational efficiency. Table 1 presents the RL parameters of the two algorithms in our implementation.

Table 1 The shared RL parameters between A3C and A2C

Parameters	A3C	A2C
Number of neurons	256 × 2 Fully connected layers	256 × 2 Fully connected layers
Discount Factor	0.9	0.9
Exploration Strategy	ϵ-greedy	Pure policy sampling (no ϵ-greedy parameters)
A3C initial exploration (ϵ) rate	1.0	N/A
A3C minimum exploration rate (ϵ-min)	0.01	N/A
A3C multiplicative decay per update (ϵ-decay)	0.995	N/A
Learning rate	0.01	0.01
Advantage Estimate δ	reward + γ*$V(s')$*(1 − done) − $V(s)$	reward + γ*$V(s')$ − $V(s)$ ($V(s') = 0$ if done)
Device	CUDA Accelerated	CUDA Accelerated (No parallelism)
Steps per iteration	2500	2500
Iterations per test	50	50

The core RL parameters listed in Table 1 (discount factor γ=0.9, learning rate α=0.01) were selected based on established ranges commonly used for policy gradient methods and continuous control environments (Ashraf et al., 2021). Focusing on the appropriate medium-term consequences for our 5G autonomous control environment (Ashraf et al., 2021).

3. RL-MAC-Scheduler Operation

As previously discussed, the proposed scheduler uses an RL algorithm to enable dynamic QoS-aware network scheduling. Figure 1 presents the various scheduler components, which are further explained in the following section.

3.1 Reward function

Our model employs a multi-part reward function to guide exploration and learning. This reward function can be divided into the following parts:

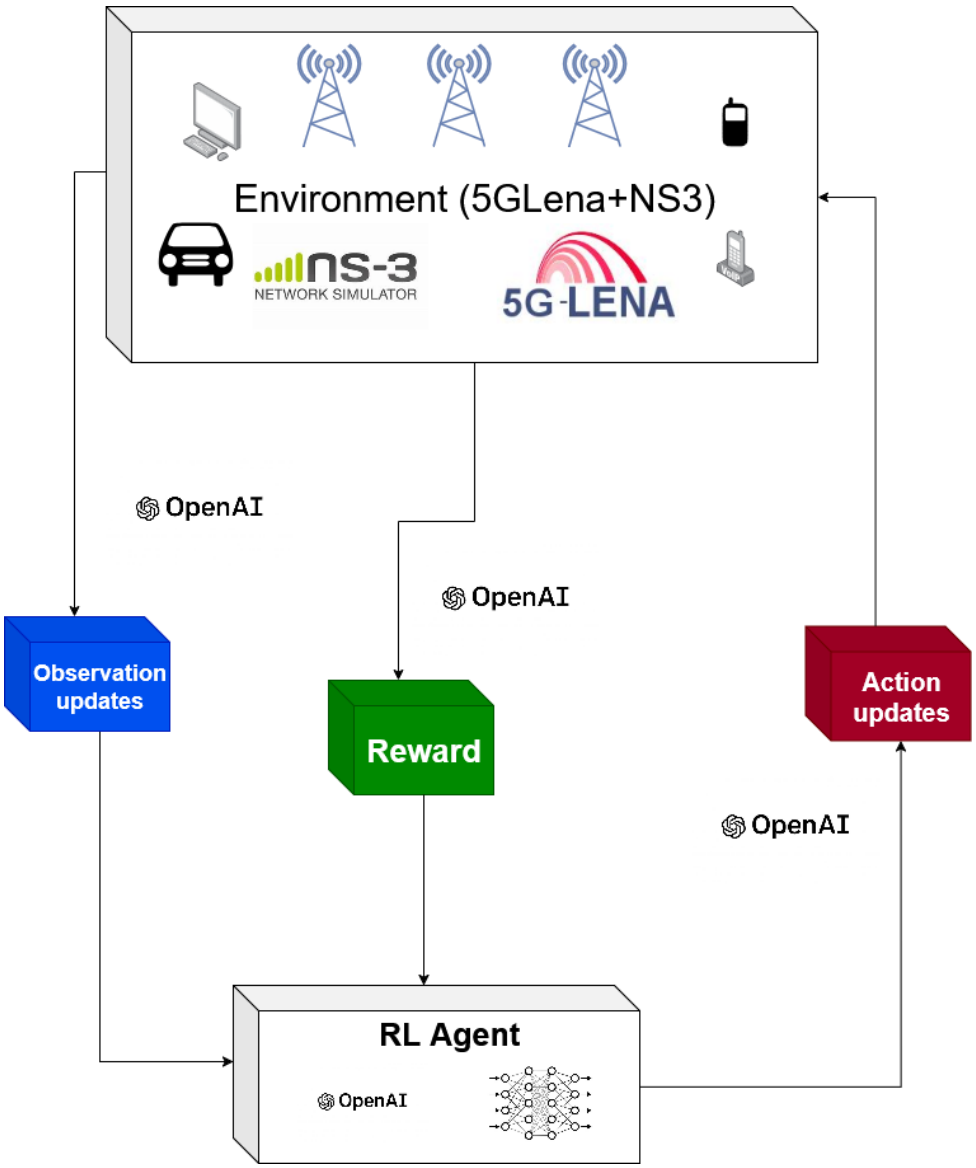

Figure 1 Scheduler function flowchart

3.1.1 Throughput reward (TR)

The first part of the proposed reward function incentivizes efficiency maximization while ensuring numerical stability. The core efficiency matrix (x) in Equation 5 specifies the ratio of achieved downlink throughput to the theoretical maximum, where a small constant ($\epsilon = 10^(-6)$) prevents division by zero during low-traffic periods. This value is transformed through a scaled hyperbolic tangent function ("tanh" (3.0×x)) as described in Equation 4, which normalizes learning by binding the reward value to [-1, 1]. Furthermore, the 3.0 multiplier enhances the sensitivity around important thresholds, enabling the reward function to prioritize substantial efficiency over marginal gains.

$$TR = \tanh(3.0 \times x) \tag{4}$$

$$x = \frac{\text{CurrentDLThroughput}}{\text{PotentialDLThroughput} + \epsilon'} \tag{5}$$

3.1.2 Stability reward

A stability reward (SR) was designed to minimize jitter by ensuring consistent resource allocation. This was achieved using two complementary metrics. First, the resource block group (RBG)-symbol stability component (σ_{rbg}) in Equation 6 is used to quantify resource allocation consistency by computing the difference between the RBG and OFDMA symbol utilization. If $\sigma_{rbg} = 1$, a perfect balance is achieved.

$$\sigma_{rbg} = RBG - SymbolStability \tag{6}$$

$$\sigma_{rbg} = 1 - \left| \frac{RBGUtilisation}{100} - \frac{SymbolUtilisation}{14} \right| \tag{7}$$

The second metric, MCSStability, is formulated in Equation 8 and evaluates the modulation consistency from a normalized deviation midpoint.

$$MCSStability = 1 - \frac{|MCS - 15|}{28} \tag{8}$$

Finally, we combined these two metrics through weighted summation, where the 0.6:0.4 weights reflected the greater emphasis placed on RBG symbol misalignment than MCS variations in our scheduler. We found RBG symbol misalignment to be the most significant metric in our environment. This is expressed in Equation 9:

$$SR = 0.6 \times \sigma_{rbg} + 0.4 \times MCSStability \tag{9}$$

3.1.3 Delay factor (DF)

To enforce an exponential penalty for approaching the packet delay budget (PDB), we implemented a delay factor (DF), which relies on the head-of-line delay ratio (HOLRatio) expressed as.

$$HOLRatio = \frac{HOLDelay}{PDB + \epsilon} \tag{10}$$

The head-of-line delay ratio normalizes the waiting time of the oldest unscheduled packet against the specific PDB threshold of each traffic.

Equations 11 and 12 describe the design of the delay factor to introduce rapidly escalating penalties. For instance, at 50% PDB (HOLRatio = 0.5), the reward could be reduced by approximately 78%, whereas violations (HOLRatio > 1) could incur a near-total penalty, which ensured proactive delay management even before any actual PDB breach occurred.

$$DF = \exp\left(-\text{PenaltySharpness} \cdot \text{HOLRatio}\right) \tag{11}$$

$$\text{PenaltySharpness} = 3.0 \tag{12}$$

3.1.4 Delay penalty (DP)

As shown in the following expression, we applied a delay penalty (DP) to supplement our DF by specifically penalizing violations using a quadratic term as seen in the expression below.

$$DP = (\text{HOLRatio})^2 \tag{13}$$

For example, Equation 13 indicates that the penalty for 2.0 HOLRatio is four times that of a 1.0 violation. For important traffic classes, this design provides a ”big red emergency button” mechanism.

3.1.5 Complete reward function

The complete reward function incorporates all the previous parts into the following equation:

$$\text{TotalReward} = 0.4\,TW \cdot TR + 0.2\,SW \cdot SR + 0.6\,DW \cdot DF - 0.3\,DP \tag{14}$$

Here, unlike fixed-weight approaches, which often fail to balance competing objectives in dynamic environments (Mao et al., 2025), our implementation prioritizes different QoS requirements by employing traffic type sensitive weighting, a method already proven to be effective for multi-objective optimization in complex environments such as 5G networks (Yan et al., 2026).Click or tap here to enter text. The TW scales according to traffic criticality: URLLC (0.6), voice/VoIP (0.8), and other traffic (1.0). Similarly, the stability weight (SW) emphasizes jitter-sensitive applications: URLLC (1.5), voice/VoIP (1.2), and other traffic (0.5). The DW follows the same pattern as that of URLLC (1.2), voice/VoIP (0.8), and other traffic (0.4). These dynamic weights and penalty sharpness parameters were tuned in accordance with established HPO strategies common in RL research (Ashraf et al., 2021). We prioritized configurations that maximized the expected cumulative reward while ensuring that the balance between throughput, stability, and delay was not overfit to a single traffic scenario. The final selected values reflect the relative priority of different aspects of QoS for different service classes, such as URLLC, which requires a higher weight on delay and stability over raw throughput.

3.2 Observation space

The observation space of the proposed RL-enabled scheduler comprises nine normalized metrics representing the current state of the network, as shown in Table 2. These metrics are updated every 100 ms, allowing the scheduler to obtain an accurate and up-to-date view of the current network conditions affecting UE traffic in the downlink direction.

Table 2 Observation space parameters

Metric	Normalization	Description
Current Throughput	DL ÷ 1000	Current downlink throughput (Mbps).
Average Throughput	DL ÷ 1000	Smoothed historical throughput (Mbps).
Potential Throughput	DL ÷ 1000	Theoretical maximum throughput based on channel conditions.
HOL/PDB Ratio	HOL / PDB	Head-of-line delay relative to packet delay budget.
DL CQI	(CQI −5) ÷ 10	Channel quality indicator (0–15), normalized to [0,1].
SINR	SINR ÷ 30	Signal-to-interference-plus-noise ratio (0–30 dB).
RBG Utilization	÷ 100	Percentage of allocated resource block groups.
Symbol Utilization	÷ 14	OFDMA symbols allocated per slot (0–14).
MCS	÷ 28	Modulation and coding scheme index (0–28).

3.3 Action space

The proposed RL-enabled scheduler controls scheduling decisions through four continuously updated parameters (Actions 1–4), which dynamically adjust scheduling decisions. The parameters are as follows:

3.3.1 Actions 1 & 2

The first two actions assigned by the scheduler control the priority weights of UEs during downlink (DL) scheduling, such as assigning a higher weight to a URLLC user during a congestion period.

3.3.2 Action 3

The 3rd action was designed to scale the allocated RBGs and (OFDMA) symbols for the UEs by assigning a value greater than 1 to increase resource intensity for some UEs or a value lower than 1 to conserve resources.

3.3.3 Action 4

The 4th action was designed to discourage unfair resource starvation by penalizing the system for every resource-starved UE by assigning a virtual penalty to prevent repeated neglect of the same UE in subsequent transmission time intervals (TTIs). It aims to ensure fairness by dynamically adapting resource allocation based on real-time network load and UE traffic types. Based on the observation space values, the space values of these actions are updated every 100 ms to maximize the reward value obtained by the RL algorithm at the end of every TTI while maintaining the A3C worker's asynchronous updates.

4. Expansion to hybrid A3C algorithms

Based on the demonstrated stability and competitive performance of our foundational A3C framework, we expanded our investigation beyond basic A3C and the well-explored A2C algorithms. Although A2C 5G applications have been extensively documented in the existing literature (Paz-Perez et al., 2024; Mollahasani et al., 2022), we identified a significant opportunity to advance the state-of-the-art by developing and evaluating a suite of hybrid A3C algorithms. These hybrids were specifically designed to enhance the core A3C architecture by integrating advanced RL techniques: Proximal Policy Optimization (PPO), Persistent Learning, and Twin Delayed Deep Deterministic Policy Gradients (TD3) to address the performance limitations in our A3C-based scheduler. This strategic focus on hybridizing A3C allowed us to leverage its inherent parallelization advantages while also targeting performance gaps observed in our simulations.

Additionally, both the base RL models and the proposed hybrid A3C models do add computational complexity primarily stemming from the neural network inference, which is not present in traditional network schedulers; however, while this is a substantial cost, it is a one-time per-decision cost and always performed within the 100 ms decision window.

4.1 A3C+PPO Hybrid: Stabilized Policy Optimization

The A3C+PPO hybrid integrated PPO mechanisms into our A3C framework to mitigate training instability observed during high-congestion events, as it was designed for training deep neural network policies in continuous action spaces (Samidi et al., 2024; Nahhas et al., 2022). While still retaining A3C's asynchronous worker architecture, it introduced the clipped surrogate objective, which replaced A3C's policy update with PPO's clipped objective, as shown in the following equation:

$$L_{\text{clip}}(\theta) = \mathbb{E}\left[\min\left(r(\theta)A,\ \text{clip}\left(r(\theta), 1-\epsilon, 1+\epsilon\right)A\right)\right] \tag{15}$$

where r(θ) is the probability ratio between new and old policies, A is the estimated advantage, and ϵ=0.2 is a clipping parameter. This limited policy updates, preventing large learning steps.

4.2 A3C + PPO + persistent learning hybrid: Cross-session training continuity

The A3C+PPO+Persistent variant was inspired by principles present in meta-learning algorithms, wherein an agent's ability to acquire new skills is improved by leveraging knowledge gained from prior experience (Liu and Xu, 2025). Our design extends our basic A3C framework with persistent state tracking functionality to enable continuous training across multiple

sessions. Its design included the following:

Continuous performance tracking: Lifetime performance indicators were logged to persistent files, allowing the algorithm to track cumulative reward values and training progress over several simulation sessions.

Session recovery: An error-handling system designed to handle common environmental failures in our prolonged NS-3 simulations. This implementation included retry mechanisms and model checkpoints to preserve training integrity across interruptions and system restarts.

4.3 A3C+TD3 Hybrid: Continuous Control Enhancement

To mitigate the value potential overestimation bias in our initial A3C model, which may prevent agents from making the correct decision between exploration and exploitation (Li et al., 2023), we designed the hybrid A3C-TD3 by integrating our initial model with TD3, an effective algorithm for continuous action spaces that typically struggle with efficient environment exploration and may suffer from slow convergence (Wu et al., 2022). However, when incorporated into A3C, it has demonstrated the potential to alleviate this limitation (Wu et al., 2022). Some of its key innovations include the following:

Twin Critic Networks: Two independent critic networks were used to estimate the action values. The minimum of the two Q-values was used for policy updates, theoretically reducing overestimation that could lead to suboptimal scheduling decisions.

Delayed Policy Updates: The actor network was updated less frequently than the critics, and the target networks were updated using Polyak averaging ($\tau = 0.005$). This delayed synchronization further stabilized the training process.

Target Policy Smoothing: Gaussian noise was added to the target actions during critic updates, acting as a regulator against overfitting to sharp Q-function peaks.

4.4 Algorithmic Differentiation Summary

Table 3 summarizes the core methodological distinctions between the four implemented A3C-based algorithms.

Table 3 Comparative summary of A3C and hybrid algorithm features

Feature	A3C	A3C-PPO	A3C-PPO-Persistent	A3C-TD3
Policy Update	Advantage gradients	Clipped surrogate ($\epsilon = 0.2$)	Clipped surrogate ($\epsilon = 0.2$)	Twin delayed DDPG
Action Sampling	ϵ-greedy + Gaussian	Gaussian + Clamping	Gaussian + Clamping	Pure Gaussian noise
Stability Mechanism	None	Gradient clipping	Gradient clipping + Checkpointing	Target smoothing
Exploration Schedule	Multiplicative ϵ-decay	Adaptive σ decay	ϵ-decay	Covariance annealing
Critical Innovation	Parallel workers	Constrained policy updates	Session persistence & recovery	Reduced value bias

5. Simulation results

In this section, we present the results of our simulation experiment, which evaluated the performance of the proposed RL-based network-scheduling approach. First, we present the reward optimization performance of our A2C and A3C scheduler implementations across two network scenarios, which formed the basis of our decision to implement hybrid A3C RL-based algorithms to further improve our scheduler's performance. Next, we compared the final QoS

simulation results of our A2C, A3C, and hybrid A3C models against three traditional schedulers: proportional fair (PF), Round Robin (RR), and maximum rate (MR), under four different network scenarios (30 UEs across 3 gNBs, 40 UEs across 4 gNBs, 50 UEs across 5 gNBs, and 60 UEs across 6 gNBs).

5.1 Jitter results

As shown in Figure 2, our comprehensive jitter analysis shows a clear trend: RL schedulers outperform traditional schedulers under light loads, but this advantage is lost as the network becomes congested. At low loads of 30 and 40 UEs, the RL schedulers demonstrated lower jitter. For 30 UEs, the A3C-PPO scheduler achieved an average jitter of 0.82 ms, which is a 41% improvement over the PF baseline (1.40 ms). This trend continued at 40 UEs, where A3C-PPO (0.86 ms) improved by 55% over PF (1.90 ms). However, at 50 UEs, the traditional schedulers proved to be far more stable. The MR scheduler delivered the best performance with an average jitter of 0.22 ms, which is 59% better than that of the PF (0.54 ms). In contrast, the RL schedulers showed significantly higher jitter; A3C-PPO averaged 0.92 ms (70% worse than PF), and the base A3C model averaged 1.40 ms (159% worse than PF).

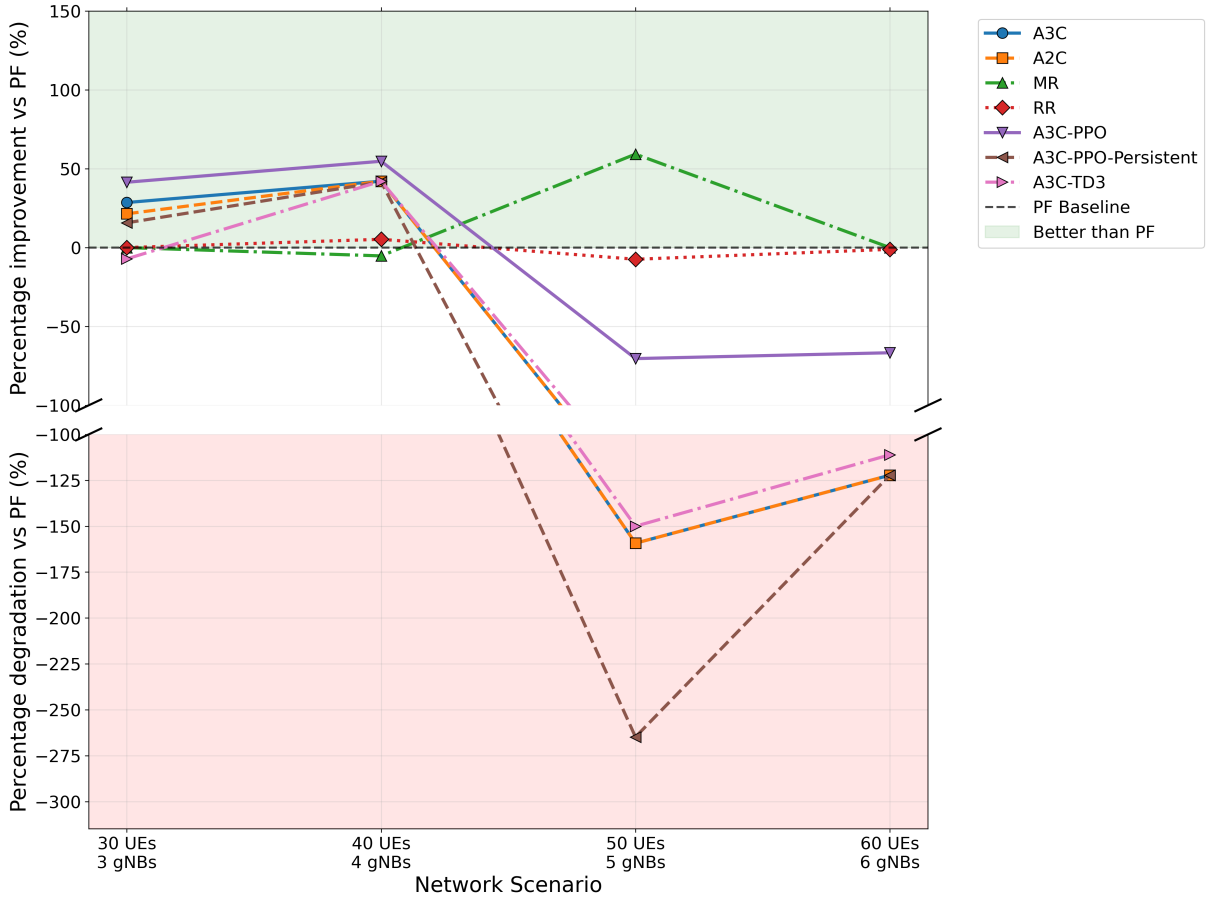

Figure 2 Jitter percentage of improvement vs. PF

This performance gap persisted at 60 UEs. The traditional schedulers MR, RR, and PF all maintained consistently low jitter, averaging between 0.90 and 0.91 ms. However, the RL schedulers degraded further, with A3C-PPO averaging 1.50 ms (67% worse than PF) and the base A3C model reaching 2.00 ms (122% worse than PF).

5.2 PDR results

As shown in Figure 3, the Packet Delivery Ratio (PDR) results highlight a critical strength of our RL schedulers: consistent, high reliability under all network conditions, while traditional

schedulers suffered severe degradation under congestion. All schedulers performed well at 30 and 40 UEs. The RL and traditional schedulers were closely matched at 30 UEs, with A3C-PPO at 99.18% and PF at 99.33%. At 40 UEs, our RL schedulers took the lead, with A3C-PPO maintaining 99.01% reliability, which is a 6% improvement over PF, which had dropped to 93.5%.

The most significant difference emerged at 50 UEs. Here, the traditional schedulers experienced a catastrophic performance drop. The PDR of PF collapsed to 16.8%, and the MR and RR fell to 28.9% and 15.9%, respectively. In stark contrast, our RL schedulers maintained near-perfect reliability, with A3C-PPO achieving 99.38% reliability, which is a 492% improvement over the failing PF baseline.

Finally, at 60 UEs, the traditional schedulers recovered, and performance converged once more. The PF scheduler achieved 99.33%, whereas our best RL model, A3C-PPO, delivered 99.30%.

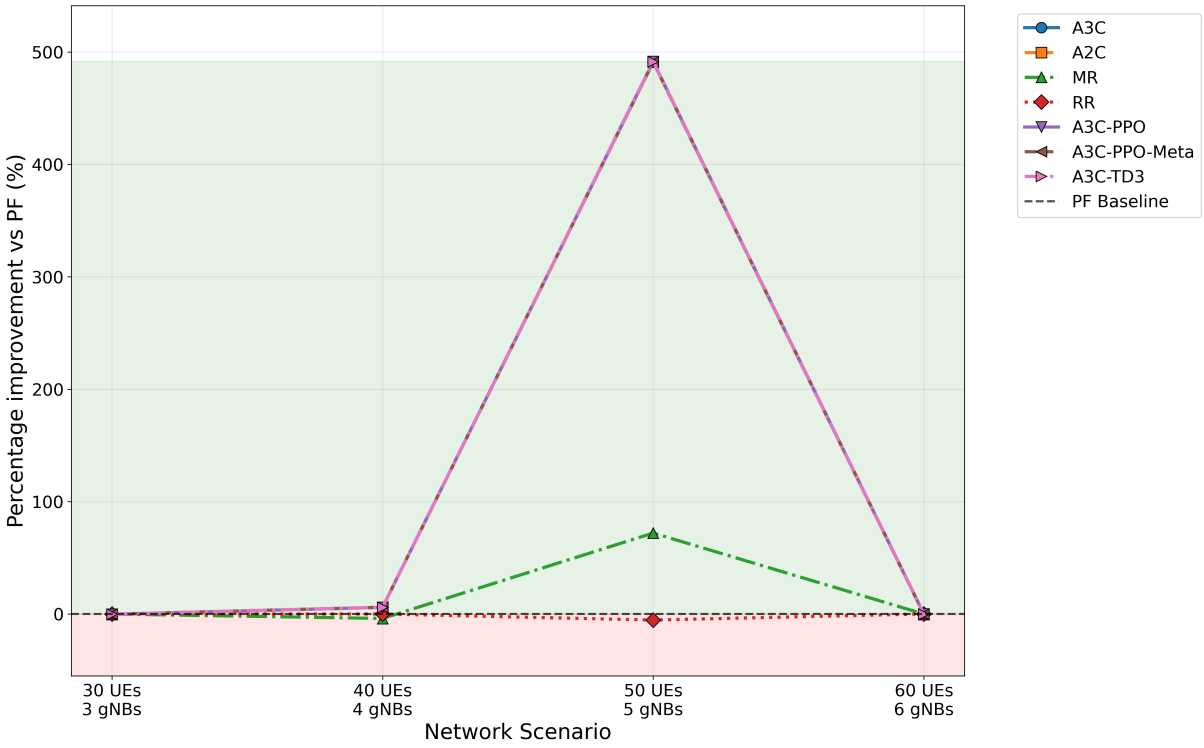

Figure 3 Percentage of improvement in PDR vs. PF

5.3 Delay results

As shown in Figure 4, the delay simulation results show a clear pattern, where traditional schedulers are efficient under low congestion but collapse completely under load, whereas our RL schedulers provide consistent, adaptive performance across all scenarios. Traditional schedulers were significantly more efficient at 30 UEs. The PF scheduler maintained an average delay of 6.8 ms. In comparison, our RL schedulers were slower, with the best-performing scheduler, A3C-PPO, averaging 8.89 ms (31% higher than PF). This changed dramatically at 40 UEs. While our RL schedulers maintained consistent delays between 10.7 and 10.9 ms, the traditional schedulers suffered a catastrophic collapse, with the PF scheduler's average delay skyrocketing to 180.2 ms. The catastrophic performance of the traditional scheduler continued at 50 UEs. The RL schedulers again maintained low latency, with A3C-PPO achieving the best average delay of 7.18 ms. Meanwhile, the PF scheduler became practically unusable, with an average delay of 945.6 ms. This represents a performance degradation of over 13,000% compared with the A3C-PPO scheduler. At 60 UEs, the traditional schedulers recovered but were still outperformed.

The PF scheduler averaged 9.0 ms, while our best RL scheduler, A3C-PPO, delivered a 10% lower average delay of 8.1 ms.

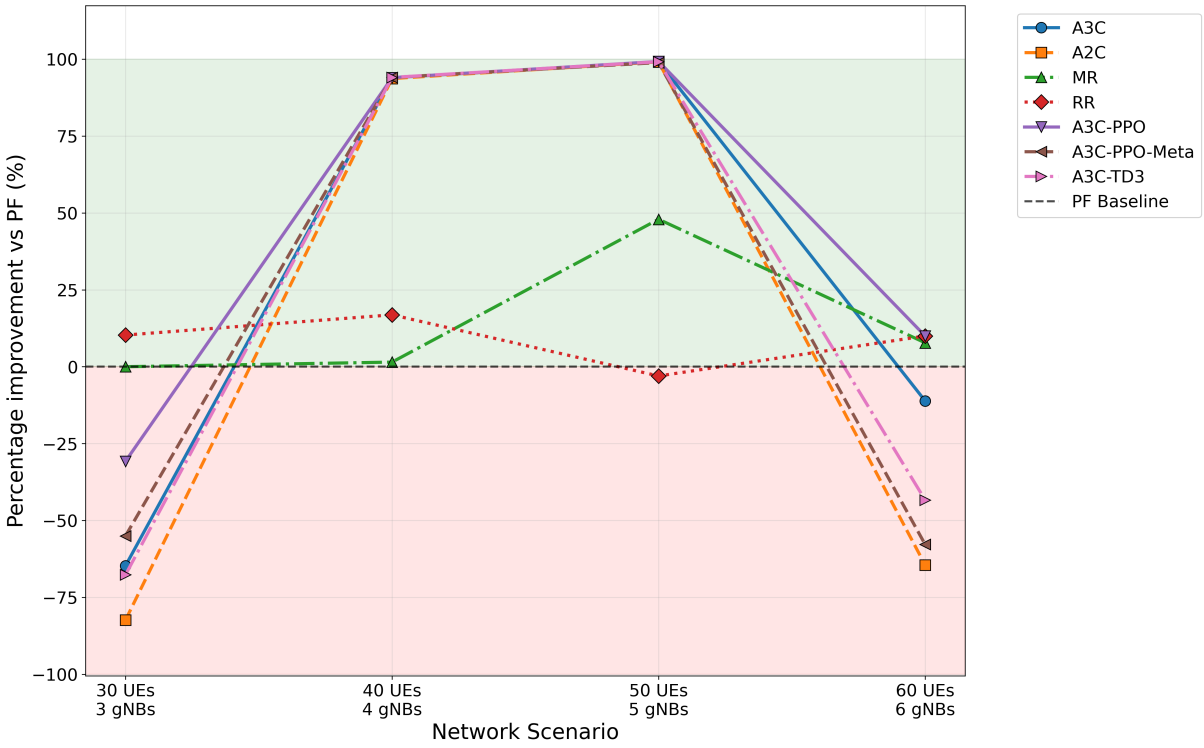

Figure 4 Delay percentage of improvement vs. PF

5.4 Throughput results

As shown in Figure 5, no scheduler managed to lead throughput in all categories. At 30 UEs, our RL schedulers demonstrated a clear advantage. The A3C-TD3 model achieved the highest average throughput of 1.05 Mbps, which is an 114% improvement over the PF baseline, which averaged 0.49 Mbps.

This advantage was reversed at 40 and 50 UEs, where traditional schedulers handled the increased load more efficiently. At 40 UEs, the PF throughput increased to 2.63 Mbps, which was 62%–66% higher than our RL schedulers (A3C-PPO: 1.00 Mbps, A3C: 0.89 Mbps). The gap widened at 50 UEs, where PF achieved 4.15 Mbps, outperforming the best RL scheduler, A3C-PPO-Persistent (1.27 Mbps), and others by 70%–79%.

However, at 60 UEs, our RL schedulers regained the performance lead. The A3C-PPO-Persistent hybrid model achieved the highest average throughput at 1.47 Mbps, an 18% improvement over the PF scheduler, which averaged 1.24 Mbps.

6. Discussion

Our simulation results revealed a complex performance landscape where our RL-enabled scheduler demonstrated significant and transformative advantages over traditional schedulers but still fell short in certain metrics. Furthermore, each of our hybrid A3C-equipped variants addressed specific limitations of the foundational A3C framework with varying degrees of success.

The A3C-PPO hybrid demonstrated the most consistent performance improvements. It was particularly effective in maintaining a reliable PDR, with an overall average of 99.38% at 50 UEs, which is an improvement of 492% over the catastrophic average of 16.8% for PF. This represents a transformative advantage in reliability under load. While still managing to deliver competitive jitter control, with an overall average of 0.82 ms at 30 UEs compared to PF's 1.40 ms, which is a 41% improvement. This general robustness can be attributed to the

PPO's clipped objective function, which was designed to train deep neural network policies in continuous action spaces and is meant to constrain policy updates and prevent performance collapse during high-congestion events (Samidi et al., 2024; Nahhas et al., 2022).

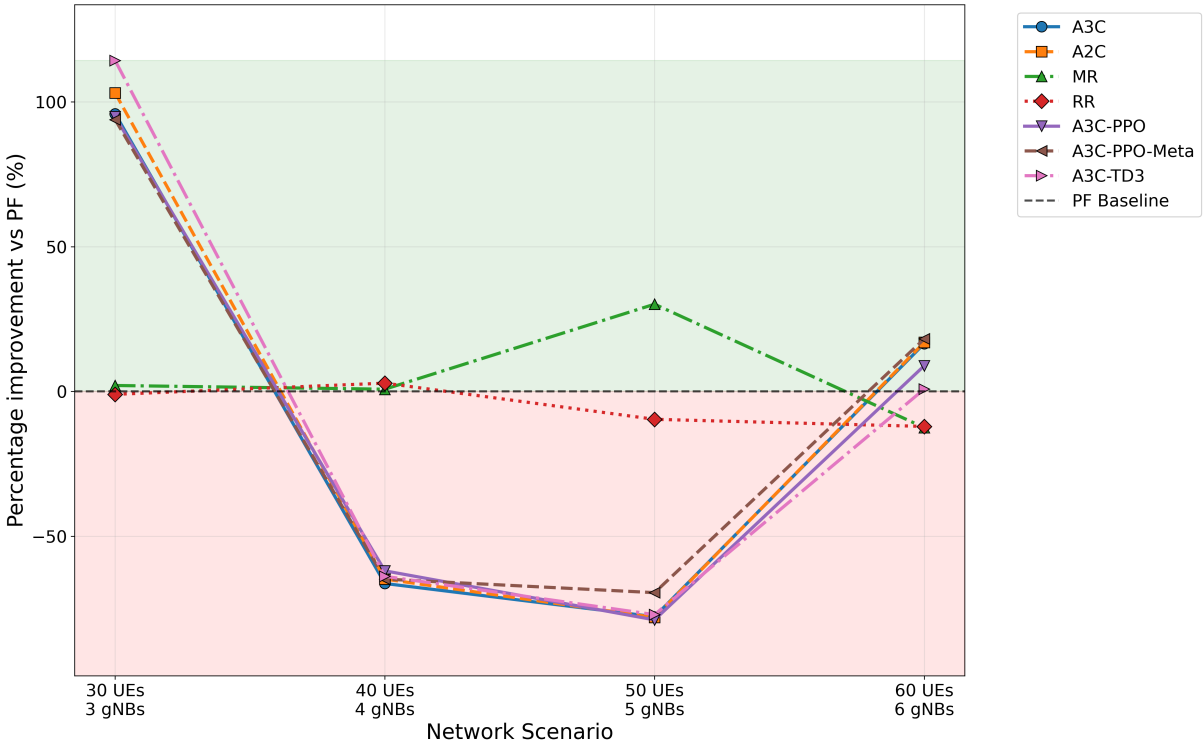

Figure 5 Percentage of improvement vs. PF

The A3C-PPO-Persistent hybrid exhibited a high degree of performance volatility. Even though it occasionally improved performance, such as achieving an 18% higher overall average throughput (1.47 vs. 1.24) at 60 UEs, it suffered from inconsistent jitter performance, with an overall average of 1.97 ms at 50 UEs, which is 70% worse than PF's average of 0.54 ms. This suggests that the persistent training mechanism, while enabling session recovery, introduced additional variance in performance stability, which is a somewhat known challenge in meta-reinforcement learning, where preserving knowledge across sessions may lead to interference and instability when adapting to new instances of the same task (Hattori et al., 2023).

The A3C-TD3 hybrid achieved remarkable performance in particular scenarios, delivering an 114% improvement in overall average throughput at 30 UEs over PF (1.05 vs. 0.49). This shows that the twin-critic architecture addressed the value overestimation bias in our initial A3C implementation. This bias could have impaired our model's decision-making regarding when to explore and when to exploit. However, it lacked overall consistency, particularly in terms of jitter stability under congestion.

Notably, traditional schedulers still managed to surpass our RL-enabled scheduler under certain conditions. At 50 UEs, MR (0.22 ms) and RR (0.58 ms) exhibited significantly superior performance compared with the highest-performing RL model, A3C-PPO (0.92 ms), which was 70% less effective than PF (0.54 ms). This specific shortcoming under high load could stem from an inherent trade-off in our multi-part reward function, where we strongly emphasize avoiding delay budget violations to protect PDR, which may have come at the cost of higher jitter. Furthermore, the exploding action space may have challenged the agent's exploration strategy under extreme congestions. Traditional schedulers also demonstrated superior throughput efficiency under moderate loads, with RR achieving an average of 2.71 Mbps at 40 UEs and 3.75 Mbps at 50 UEs, in contrast to the best-performing RL-enabled schedulers, namely, A3C-PPO-Persistent (1.27 Mbps) and A3C-PPO (1.00 Mbps). Nonetheless, the disastrous decline in reliability under medium loads, such as PF's average PDR falling to 16.8% at 50 UEs compared to 99.38% for

A3C-PPO, underscores their intrinsic vulnerability in adaptive network contexts and highlights the potential improvements offered by hybrid A3C algorithms.

7. Conclusions

In conclusion, among all the evaluated approaches, our A3C-PPO-equipped scheduler emerged as the most balanced and reliable option, achieving an optimal balance between algorithmic sophistication and practical stability. The integration of PPO's clipped surrogate objective addressed the training instability of the base A3C implementation by preventing destructive policy updates during high-congestion events. However, no implementation of our RL-enabled scheduler managed to lead in every performance metric and was sometimes outperformed by traditional schedulers, as seen in jitter at 50/60 UEs and throughput at 40/50 UEs. This proves that while our implementation has strengths worth pursuing, there is still room for improvements in future works. One direction for future work could be exposing the agent to even more complex environments and for longer iteration cycles, potentially allowing for more robust policy refinement. Furthermore, implementing TD3's twin-critic concept in the A3C-PPO model could help combat possible value overestimation issues without sacrificing the PPO's improvements. Lastly, deployment feasibility relies on a few factors such as computational overhead, existing hardware and, decision latency and while much of that falls outside this study's scope, we can theorize that the primary practical challenge for real-world deployment is ensuring the robustness and generalization of the RL policy across unseen, real-world traffic distributions.

Acknowledgements

This research was supported by the Ministry of Higher Education (MOHE) through the Fundamental Research Grant Scheme (FRGS/1/2023/TK07/MMU/03/1).

Conflict of Interest

The authors have no conflicts of interest to declare.

Declaration of AI

During the preparation of this work, DeepSeek and Paperpal Preflight were used in order to improve readability and to check for any grammar or spelling mistakes in the manuscript. After using these tools/services, the content was reviewed and edited as needed.

References

Akyildiz, H. A., Gemici, O. F., Hokelek, I., & Cirpan, H. A. (2024). Hierarchical reinforcement learning based resource allocation for RAN slicing. *IEEE Access*, *12*, 75818–75831. https://doi.org/10.1109/ACCESS.2024.3406949

Alanazi, R., Obayya, M., Alghamdi, A. M., Nemri, N., Alshahrani, S., Alduaiji, N., Hasanin, T., & Sorour, S. (2025). Machine learning-driven routing optimization for energy-efficient 6G-enabled wireless sensor networks. *Alexandria Engineering Journal*, *129*, 877–888. https://doi.org/10.1016/j.aej.2025.07.032

Alsenwi, M., Tran, N. H., Bennis, M., Pandey, S. R., Bairagi, A. K., & Hong, C. S. (2021). Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning approach. *IEEE Transactions on Wireless Communications*. https://doi.org/10.1109/TWC.2021.3060514

Ashraf, N. M., Mostafa, R. R., Sakr, R. H., & Rashad, M. Z. (2021). Optimizing hyperparameters of deep reinforcement learning for autonomous driving based on whale optimization algorithm. *PLOS ONE*, *16*(6), e0252754. https://doi.org/10.1371/journal.pone.0252754

Benmadani, H. E., Azni, M., Alharbi, T. E., Alzaidi, M. S., & Tounsi, M. (2025). Deep reinforcement learning-based dynamic scheduling for real-time applications in LTE and 5G. *IEEE Access*, *13*, 33555–33570. https://doi.org/10.1109/ACCESS.2025.3541531

Bikkasani, D., & Yerabolu, M. (2024). AI-driven 5G network optimization: A review of resource allocation, traffic management, and network slicing. *American Journal of Artificial Intelligence*, *8*(2), 55–62. https://doi.org/10.11648/j.ajai.20240802.14

Bozis, E. Z. G., Sagias, N. C., Batistatos, M. C., Kourtis, M. A., Xilouris, G. K., & Kourtis, A. (2024). Enhancing 5G performance: A standalone system platform with customizable features. *AEU - International Journal of Electronics and Communications*, *187*. https://doi.org/10.1016/j.aeue.2024.155515

Carneiro, D. P. Q., Cardoso, A. A., & Vieira, F. H. T. (2023). Adaptive resource allocation in 5G systems using reinforcement learning. *Neural Computing and Applications*, *35*(13), 9421–9435. https://doi.org/10.1007/s00521-023-08406-2

Del Rio, A., Jimenez, D., & Serrano, J. (2024). Comparative analysis of A3C and PPO algorithms in reinforcement learning: A survey on general environments. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3472473

Elsayed, M., & Erol-Kantarci, M. (2019). Reinforcement learning-based joint power and resource allocation for URLLC in 5G. *IEEE GLOBECOM*, 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9014032

Gawłowicz, P., & Zubow, A. (2019). Ns-3 meets OpenAI Gym: Machine learning for networking research. *MSWiM*, 113–120. https://doi.org/10.1145/3345768.3355908

Gedikli, A. M., Koseoglu, M., & Sen, S. (2022). Deep reinforcement learning-based flexible preamble allocation for RAN slicing. *Computer Networks*, *215*. https://doi.org/10.1016/j.comnet.2022.109202

Guo, Y., & Xie, Y. (2025). Adaptive network planning for 5G/6G networks under burst traffic. *IEEE Communications Letters*. https://doi.org/10.1109/LCOMM.2025.3537863

Hattori, R., Hedrick, N. G., Jain, A., Chen, S., You, H., Hattori, M., Choi, J.-H., Lim, B. K., Yasuda, R., & Komiyama, T. (2023). Meta-reinforcement learning via orbitofrontal cortex. *Nature Neuroscience*, *26*(12), 2182–2191. https://doi.org/10.1038/s41593-023-01485-3

Ibrahim, A. A., & Ali, W. A. E. (2021). High gain, wideband and low mutual coupling AMC-based millimeter wave MIMO antenna for 5G NR networks. *AEU - International Journal of Electronics and Communications*, *142*. https://doi.org/10.1016/j.aeue.2021.153990

Ibrahimi, K., Jouhari, M., Sow, S., Ayoub, F., Kamili, M. E., & Chougdali, K. (2024). Reinforcement learning for optimized resource allocation in 5G URLLC. *CommNet 2024*. https://doi.org/10.1109/CommNet63022.2024.10793310

Konstantoulas, I., Loi, I., Tsimas, D., Sgarbas, K., Gkamas, A., & Bouras, C. (2025). A framework for user traffic prediction and resource allocation in 5G networks. *Applied Sciences*, *15*(13). https://doi.org/10.3390/app15137603

Koutlia, K., Bojovic, B., Ali, Z., & Lagén, S. (2022). Calibration of the 5G-LENA system level simulator in 3GPP reference scenarios. *Simulation Modelling Practice and Theory*. https://doi.org/10.1016/j.simpat.2022.102580

Li, S., Tang, Q., Pang, Y., Ma, X., & Wang, G. (2023). Realistic actor-critic: A framework for balancing value overestimation and underestimation. *Frontiers in Neurorobotics*, *16*, 1081242. https://doi.org/10.3389/fnbot.2022.1081242

Liu, L., & Xu, Z. (2025). Combining meta reinforcement learning with neural plasticity mechanisms for improved AI performance. *PLOS ONE*, *20*(5). https://doi.org/10.1371/journal.pone.0320777

Mao, L., Ma, Z., & Li, X. (2025). A multi-task dynamic weight optimization framework based on deep reinforcement learning. *Applied Sciences*, *15*(5), 2473. https://doi.org/10.3390/app15052473

Mehta, D. (2020). State-of-the-art reinforcement learning algorithms. *International Journal of Engineering Research & Technology*, *8*(12), 717–722. https://doi.org/10.17577/IJERTV8IS120332

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *arXiv preprint*. https://doi.org/10.48550/arXiv.1602.01783

Mollahasani, S., Erol-Kantarci, M., Hirab, M., Dehghan, H., & Wilson, R. (2022). Actor-critic learning-based QoS-aware scheduler for reconfigurable wireless networks. *IEEE Transactions on Network Science and Engineering*, *9*(1), 45–54. https://doi.org/10.1109/TNSE.2021.3070476

Nahhas, A., Kharitonov, A., & Turowski, K. (2022). Deep reinforcement learning techniques for solving hybrid flow shop scheduling problems: Proximal policy optimization and asynchronous advantage actor-critic. *HICSS 2022*. https://doi.org/10.24251/HICSS.2022.206

Navarro-Ortiz, J., Romero-Diaz, P., Sendra, S., Ameigeiras, P., Ramos-Munoz, J. J., & Lopez-Soler, J. M. (2020). A survey on 5G usage scenarios and traffic models. *IEEE Communications Surveys and Tutorials*, *22*(2), 905–929. https://doi.org/10.1109/COMST.2020.2971781

Patriciello, N., Lagén, S., Bojovic, B., & Giupponi, L. (2019). An end-to-end simulator for 5G NR networks. *Simulation Modelling Practice and Theory*, *96*. https://doi.org/10.1016/j.simpat.2019.101933

Paz-Perez, A., Tato, A., Escudero-Garzas, J. J., & Gomez-Cuba, F. (2024). Flexible reinforcement learning scheduler for 5G networks. *IEEE ICMLCN 2024*, 566–572. https://doi.org/10.1109/ICMLCN59089.2024.10625129

Raza, W., Farooq, M. U. B., Ijaz, A., Manalastas, M., & Imran, A. (2025). AI-powered resilience: A dual-approach for outage management in dense cellular networks. *Computer Communications*, *236*. https://doi.org/10.1016/j.comcom.2025.108129

Samidi, F. S., Radzi, N. A. M., & Aripin, N. M. (2024). Reinforcement learning model selection for resource allocation and subcarrier spacing optimization in 5G sliced spectrum networks. *IEEE ICAEE 2024*. https://doi.org/10.1109/ICAEE62924.2024.10667637

Sánchez, J. A. H., Casilimas, K., & Rendon, O. M. C. (2022). Deep reinforcement learning for resource management on network slicing: A survey. *Sensors*. https://doi.org/10.3390/s22083031

Seid, A. M., Boateng, G. O., Mareri, B., Sun, G., & Jiang, W. (2021). Multi-agent deep reinforcement learning for task offloading and resource allocation in multi-UAV IoT edge networks. *IEEE Transactions on Network and Service Management*, *18*(4), 4531–4547. https://doi.org/10.1109/TNSM.2021.3096673

Sewak, M. (2019). Actor-critic models and the A3C. In *Deep reinforcement learning*. Springer. https://doi.org/10.1007/978-981-13-8285-7_11

Tan, K. H., Lim, H. S., & Diong, K. S. (2022). Modelling and predicting quality-of-experience of online gaming users in 5G networks. *International Journal of Technology*, *13*(5), 1035–1044. https://doi.org/10.14716/ijtech.v13i5.5866

Tsoulos, G., Athanasiadou, G., Zarbouti, D., Nikitopoulos, G., Tsoulos, V., & Christopoulos, N. (2024). Empirical analysis of 5G deployments: A comparative assessment of network performance with 4G. *AEU - International Journal of Electronics and Communications*, *186*. https://doi.org/10.1016/j.aeue.2024.155479

Wai, J. H., Lee, Y. L., & Ke, F. (2021). Combined metric-based resource scheduling for 5G networks. *IEEE MICC 2021*, 19–24. https://doi.org/10.1109/MICC53484.2021.9642139

Wu, J., Wu, Q. M. J., Chen, S., Pourpanah, F., & Huang, D. (2022). A-TD3: An adaptive asynchronous twin delayed deep deterministic algorithm. *IEEE Access*, *10*, 128077–128089. https://doi.org/10.1109/ACCESS.2022.3226446

Yan, P., Lu, J., Zeng, H., & Hou, Y. T. (2025). Near-real-time resource slicing for QoS optimization in 5G O-RAN using deep reinforcement learning. *arXiv preprint*. https://doi.org/10.48550/arXiv.2509.14343

Yan, P., Lu, J., Zeng, H., & Hou, Y. T. (2026). Near-real-time resource slicing for QoS optimization in 5G O-RAN using deep reinforcement learning. *IEEE/ACM Transactions on Networking*, *34*, 1596–1611. https://doi.org/10.1109/TON.2025.3628209

Zhou, H., Elsayed, M., & Erol-Kantarci, M. (2021). RAN resource slicing in 5G using multi-agent correlated Q-learning. *IEEE PIMRC 2021*. https://doi.org/10.1109/PIMRC50174.2021.9569358