Application of Reinforcement Learning and Large Language Models for Energy Optimization in Wireless Networks.

Duong Bui Dang[1], Tra-Giang Le Thi[1], Viet-Hoang Le Mai[1], Hoc Hoang Van[1],
Anh-Tuan Nguyen Duc[1], Thinh Pham Van[2] & Thai-Mai Dinh Thi[1]*.
[1]Faculty of Electronics and Telecommunications, University of Engineering and Technology,
Vietnam National University, Hanoi, Vietnam
[2]Viettel Network, Hanoi, Vietnam
*Corresponding Author

***Abstract*—This paper investigates the combination of Proximal Policy Optimization (PPO) with DeepSeek-R1-Distill-Qwen-7B to improve network performance. Simulations show that the proposed PPO-LLM framework achieves the lowest energy consumption (2038 Wh) across all evaluated methods, including the Enhanced Heuristic Switching Algorithm (EHSA), Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC), Twin Delayed DDPG (TD3), and PPO. This corresponds to a 47% reduction relative to the unoptimized baseline while maintaining a competitive downlink throughput of 28,885 Mbps. Compared to standard Deep Reinforcement Learning (DRL) baselines, PPO-LLM outperforms SAC in energy efficiency by 23% and surpasses TD3 and DDPG in throughput, demonstrating a superior throughput–energy trade-off. These results suggest that LLM-guided reward engineering is a promising approach to automating reward design, enabling efficient and adaptive energy management in 5G heterogeneous networks.**

***Index Terms*—5G mobile networks, small-cell on/off switching, energy efficiency, reward function design, large language models, deep reinforcement learning.**

I. Introduction

The rapid growth of mobile data traffic, driven by the proliferation of smartphones, Internet-of-Things (IoT) devices, and bandwidth-intensive applications, has placed enormous pressure on wireless network infrastructure to deliver higher capacity while maintaining reliable quality of service (QoS). To address this demand, fifth-generation (5G) networks have adopted heterogeneous network (HetNet) architectures, in which dense deployments of small cells (SCs)—including picocells and femtocells—are overlaid on traditional macro base station (BS) grids [1], [2]. This densification strategy significantly improves spectral efficiency and spatial reuse, but it fundamentally alters the energy consumption profile of the network.

Energy consumption has emerged as one of the most critical challenges in modern wireless network operation. Base stations account for approximately 60–80% of the total energy consumption of mobile networks, and a substantial portion of this consumption is static power that is dissipated regardless of traffic load [3]. In dense SC deployments, a large fraction of small cells are lightly loaded or entirely idle during off-peak hours, yet continue to consume near-peak power. This inefficiency not only drives up operational expenditure for network operators, but also contributes significantly to the carbon footprint of the telecommunications sector. Consequently, energy-efficient network management—particularly through intelligent control of BS operational states—has become a priority research area aligned with the green networking goals of beyond-5G systems.

The small cell on/off strategy, which puts underutilized stations into sleep mode and transfers their users to neighboring active stations, is widely recognized as one of the most direct approaches to reduce idle power consumption [1], [2], [3]. However, existing solutions face several important limitations. Traditional heuristic methods [1], [2] reduce computational complexity but make switching decisions without fully accounting for the load impact on neighboring cells, which can easily lead to premature termination and suboptimal energy savings. While the optimal solution can be found through exhaustive search, the decision space grows exponentially as 2^N with the number of small cells N, rendering this approach computationally infeasible in practice.

DRL has recently attracted significant attention as a data-driven approach to network optimization. Algorithms such as Deep Deterministic Policy Gradient (DDPG) [4], Soft Actor-Critic (SAC) [5], Twin Delayed DDPG (TD3) [6], and Proximal Policy Optimization (PPO) [7] have demonstrated competitive performance in dynamic resource management tasks. Nevertheless, a fundamental bottleneck remains: the reward function—which encodes the optimization objective for the RL agent—must be carefully hand-crafted by domain experts. Poorly designed reward functions lead to unstable training, slow convergence, or policies that fail to generalize across varying network conditions. This reward engineering step is inherently difficult, experience-dependent, and does not scale well to heterogeneous multi-objective optimization scenarios. Although Large Language Models (LLMs) have demonstrated remarkable capabilities in code generation and reasoning tasks [8], [9], their potential for automating reward function design in wireless network optimization remains largely unexplored.

Motivated by the above limitations, this paper proposes a comprehensive optimization framework for energy saving in 5G HetNets that integrates classical heuristic methods, DRL algorithms, and LLM-guided reward engineering. The main contributions of this work are as follows:

1) Comprehensive Benchmarking Framework: We develop a high-fidelity 5G NR simulation environment in strict conformance with 3GPP standards [10], [11], [12], [13],

[14], [15], [16]. Within this framework, we conduct a rigorous comparative analysis evaluating established heuristic baselines—including load-aware strategies like EHSA [1]—alongside four prominent DRL algorithms (DDPG, SAC, TD3, and PPO), providing a systematic evaluation of the throughput–energy trade-offs in modern dense networks.

2) LLM-Guided Reward Engineering (PPO-LLM): We propose a novel closed-loop architecture in which DeepSeek-R1 automatically generates and iteratively refines the reward function for the PPO agent based on network state context and performance feedback, eliminating the need for manual reward design.

The remainder of this paper is organized as follows. Section II describes the 5G NR simulation environment and system model. Section III details the proposed DeepSeek-R1–PPO integration architecture, including the prompt engineering pipeline and PPO agent design. Section IV presents experimental results and comparative analysis. Finally, Section V concludes the paper and outlines promising directions for future research.

II. System Model

To ensure that all algorithms are evaluated under identical, physically realistic conditions, we construct a high-fidelity 5G NR simulation environment in strict conformance with 3GPP standards [10], [11], [12], [13], [14], [15], [16].

A. Simulation Architecture and Network Object Model

The system was designed following the Agent-Environment architecture operating in a synchronous mechanism. At each time step, the Agent observes the global state of the network, makes control decisions, and the Environment executes those decisions while simultaneously returning the reward function and the new state. The network environment was built using an object-oriented approach with a strictly hierarchical logical structure: Network & Topology [15], Base Station and Antenna [14], Cell [14], User Equipment (UE)[14].

B. Propagation Model

To ensure the training environment accurately reflects real-world signal interference and attenuation, the system strictly applies the 3GPP propagation standards [10]. The applied scenarios are UMI, UMA, RMA [10]. The channel model is computed based on the following parameters: LOS/NLOS probability and Path Loss [10], Shadow Fading [10], O2I Penetration Loss [10].

C. Link-Level Model

User experience quality is evaluated through a complex link-level computation chain, which helps reinforcement learning algorithms become aware of the trade-off between turning off stations and network quality degradation [1], [17]: SINR computation, Adaptive Modulation and Coding (AMC) [11], HARQ and MIMO Adaptation [11], Uplink Power Control [12].

D. Mobility Management and Resource Allocation

When the system executes algorithms to turn off Small Cells for energy saving, the workload of mobility management increases sharply [2], [3]. The simulation handles this issue through two mechanisms: The handover procedure is triggered according to the 3GPP A3 event [13], combined with the Time-To-Trigger mechanism and 3 dB hysteresis to mitigate the ping-pong effect, only executing when the RSRP of the target cell maintains a stronger level than the current cell for a certain period of time [13]. Simultaneously, PRB resources at each cell are allocated to UEs through scheduling algorithms such as Fair, Round Robin, and Proportional Fair, ensuring a balance between fairness and system performance [11], [16].

III. Proposed Method

This section presents the proposed method for the network optimization problem, in which the system is built upon the combination of DeepSeek-R1 and PPO. The overall architecture is designed in a modular fashion, allowing the LLM to assume the role of strategic guidance through reward function generation, while the RL agent carries out the learning and decision-making process within the environment. The interaction mechanism between the components is organized into a closed-loop cycle, enabling the system to continuously adapt and improve performance according to changing network conditions.

A. System Architecture

The system forms a closed-loop optimization cycle in which DeepSeek-R1 generates reward strategies, PPO Agent interacts with the environment and learns network control policies, and the resulting performance metrics are fed back to the LLM for reward refinement. This iterative process enables adaptive network operation under dynamic conditions [8]. Figure 1 illustrates the data flow and interactions between components in the LLM- and RL-based network optimization system .

The system adopts a modular architecture with a clear separation of responsibilities among the LLM module, RL Agent, and simulation environment. To ensure flexibility and scalability, an abstract Interface for LLM (ILLM) is employed to seamlessly integrate various language models, including DeepSeek-R1.

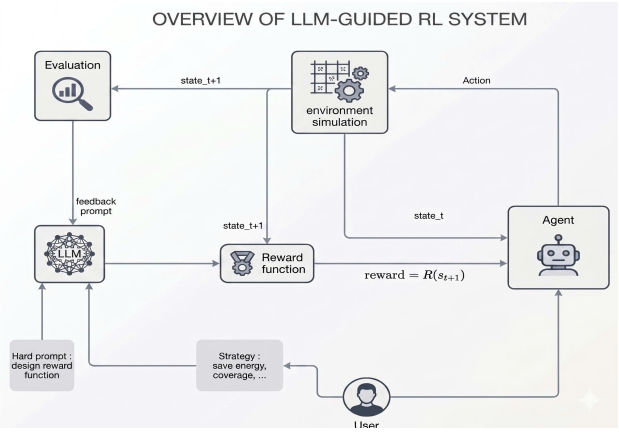

Fig. 1: Network optimization system

B. LLM-Guided Reward Generation

A distinguishing feature of the proposed architecture is the Soft-Defined Reward strategy, in which the reward function is not manually designed but dynamically generated by the DeepSeek-R1 at runtime.

1) LLM-Driven Dynamic Reward Optimization: The Soft-Defined Reward strategy operates through the following pipeline:

- Initialization Check: The system first checks whether a previously generated reward function exists. If available, the function is loaded directly without invoking the LLM.
- Prompt Construction: If no valid reward function is found, the system assembles a reward-generation prompt. The prompt includes the environment state schema, optimization objectives, and execution constraints.
- DeepSeek-R1-Code Generation: The system submits the constructed prompt to the LLM, which generates the reward function in the form of executable Python code.
- Code Extraction and Persistence: The system extracts the generated code from the DeepSeek-R1 response, combines it with the required import statements, and stores it for future reuse.
- Dynamic Loading: The system dynamically loads the saved module and integrates the reward function into the PPO training process.

The reward function objectives and the constants used in the fallback analytical reward are summarized in Table 1.

Parameter	Description	Value
β	Energy decay coefficient: $r_{\text{energy}} = \sum_i e^{-\beta \cdot P_i}$	0.05
α_{th}	Throughput scaling coefficient	0.01
ψ	Handover penalty coefficient	0.1
w_{oos}	Out-of-service penalty weight	2.0
R_{qos}	Base QoS violation penalty	-10.0

Table 1: Reward function parameters and optimization objectives

2) Reward Function Generation through Prompt Engineering: To generate executable reward functions, the reward generation pipeline employs a structured prompt engineering process. Rather than providing only generic requirements, the system supplies the DeepSeek-R1 with a context-rich prompt. Specifically, the prompt includes:

- Function Signature: Defines the exact Python function structure required.
- State Structure: Provides a comprehensive description of the input data, including cell-level information, UE-level information, and aggregated QoS metrics.
- Execution Constraints: Specifies syntax and logic constraints to ensure that the generated code can be executed directly within the simulation environment.
- Historical Context: Provides feedback from previous trials, enabling iterative refinement of the reward function based on prior outcomes.

3) Self-Correction Loop: To ensure uninterrupted training, the system incorporates an automated self-correction mechanism. When a generated reward function triggers a runtime exception (e.g., division by zero, undefined attribute access, or type mismatch), the system captures the error traceback and appends it to the prompt history. The LLM then uses this feedback to generate a corrected reward function. This process repeats until a valid executable function is obtained. The self-correction workflow is illustrated in Figure 2.

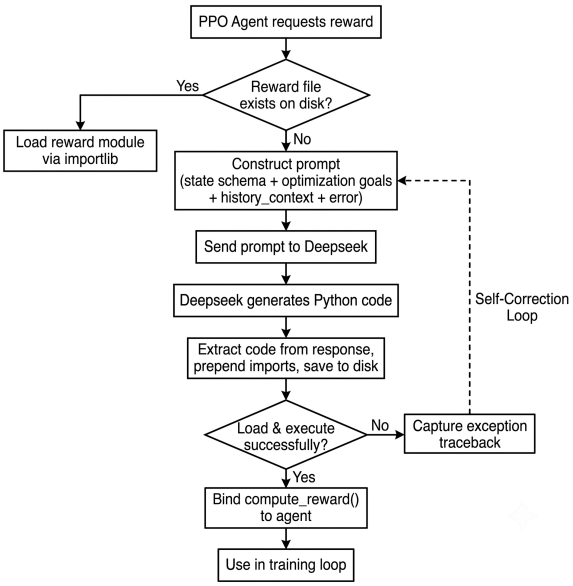

Fig. 2: Self-correction loop for LLM-generated reward function

C. MDP Formulation

To formally characterize the network optimization problem, the system is modeled as a Markov Decision Process (MDP) defined by the tuple (S, A, R, γ), where S and A denote the state and action spaces detailed in the following subsections, R is the reward signal dynamically generated by the DeepSeek-R1 , and γ is the discount factor applied during policy optimization.

1) State Representation: To bridge the gap between the structured network environment and the neural network input, a State Extractor module flattens the nested environment state into a fixed-length continuous vector of dimension D_s. The state dimension is computed dynamically from the topology as $D_s = N_{\text{cell}} \times 2 + N_{\text{UE}} + 4$, where N_{cell} is the number of cells and N_{UE} is the number of UEs. The composition of the state vector is detailed in Table 2.

Feature Group	Dims	Total	Normalization
Cell tx_power	1	N_{cell}	$\div 50.0$
Cell is_active	1	N_{cell}	Binary $\{0, 1\}$
UE is_connected	1	N_{UE}	Binary $\{0, 1\}$
Step throughput	1	1	$\div (N_{\text{UE}} \times 100)$
Step energy	1	1	$\div (N_{\text{cell}} \times 50.0)$
Step handovers	1	1	$\div N_{\text{UE}}$
Step OOS ratio	1	1	$\div N_{\text{UE}}$
Total State Dimension		$D_s = 2N_{\text{cell}} + N_{\text{UE}} + 4$	

Table 2: Composition of the state feature vector

If the total number of extracted features is less than the configured D_s, the vector is zero-padded; if it exceeds D_s, it is truncated. This ensures dimensional consistency regardless of topology size.

2) Action Space and Mapping Mechanism: The action space is a multi-binary vector $\mathbf{a} \in \{0,1\}^{D_a}$, where $D_a = N_{\text{cell}}$ and each element a_i corresponds to the on/off decision for cell i. During training, the Actor network outputs a probability vector $\mathbf{p} \in [0,1]^{D_a}$, from which actions are sampled via a Bernoulli distribution: $a_i \sim \text{Bernoulli}(p_i)$.

The Action Mapper translates the binary action vector into concrete environment control commands. Specifically, for each cell i:

- If $a_i \geq 0.5$ and the cell is currently in `SLEEP` state $\Rightarrow$ send command `CONTROL_CELL` with state = `ACTIVE`.
- If $a_i < 0.5$ and the cell is currently in `ACTIVE` state $\Rightarrow$ send command `CONTROL_CELL` with state = `SLEEP`.
- If the desired state matches the current state $\Rightarrow$ no action is sent (reducing communication overhead).

The action mapping parameters are summarized in Table 3.

Parameter	Value
Dimension D_a	N_{cell}
Action type	Multi-binary (Bernoulli)
Decision threshold	0.5
Control command	`CONTROL_CELL`
Target states	`ACTIVE` / `SLEEP`
Optimization	Skip if target = current

Table 3: Action mapping configuration

D. PPO Agent Design

1) Agent Architecture: The PPO agent employs an Actor-Critic architecture with two separate Multi-Layer Perceptron (MLP) networks operating in parallel:

Actor Network: This network maps the state vector s to a probability vector for the cell on/off decision. It consists of two hidden layers of 64 neurons with Tanh activation and a Sigmoid output layer:

$$h_1^{(a)} = \tanh(W_1^{(a)} s + b_1^{(a)}), \quad h_2^{(a)} = \tanh(W_2^{(a)} h_1^{(a)} + b_2^{(a)}) \tag{1}$$

$$\mathbf{p} = \sigma(W_3^{(a)} h_2^{(a)} + b_3^{(a)}), \quad a_i \sim \text{Bernoulli}(p_i) \tag{2}$$

Critic Network: This network estimates the state value function $V(s)$ for computing the Advantage Function. It shares the same hidden structure as the Actor but outputs a single scalar without activation:

$$V(s) = W_3^{(c)} h_2^{(c)} + b_3^{(c)} \tag{3}$$

Both networks are jointly optimized using the Adam optimizer. The composite loss function is:

$$\begin{aligned} \mathcal{L} = & \underbrace{-\min\left(r_t \hat{A}_t,\ \text{clip}(r_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\right)}_{\text{Clipped Policy Loss}} \\ & + \underbrace{c_v \cdot \text{MSE}(V(s_t), G_t)}_{\text{Value Loss}} \\ & - \underbrace{c_e \cdot H[\pi(\cdot|s_t)]}_{\text{Entropy Bonus}} \end{aligned} \tag{4}$$

where $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the importance sampling ratio.

The PPO algorithm is selected for its key advantages: it does not require a large Replay Buffer (unlike Off-Policy methods such as SAC or TD3), it supports both discrete and continuous action spaces, and the Clipping mechanism ensures stable training by preventing excessive policy updates.

The detailed layer-by-layer configuration of the Actor and Critic networks is summarized in Table 4.

Network	Layer	Input dim	Output dim	Activation
Actor	Hidden 1	D_s	64	Tanh
	Hidden 2	64	64	Tanh
	Output	64	D_a	Sigmoid
Critic	Hidden 1	D_s	64	Tanh
	Hidden 2	64	64	Tanh
	Output	64	1	None (Linear)

Table 4: Layer-by-layer configuration of the Actor–Critic neural networks

The PPO hyperparameters used in the system are listed in Table 5.

Parameter	Description	Default Value
α_{lr}	Learning rate (Adam optimizer)	3×10^{-4}
γ	Discount factor	0.99
K (K_epochs)	Number of PPO update epochs per batch	10
ϵ (eps_clip)	Clipping parameter for surrogate loss	0.2
c_v	Value loss coefficient	0.5
c_e	Entropy bonus coefficient	0.01
Optimizer	Gradient descent method	Adam

Table 5: PPO agent hyperparameters

2) Memory Buffer: The system uses an on-policy Memory Buffer to accumulate transitions $(s_t, a_t, \log \pi(a_t|s_t), r_t, d_t)$ during environment interaction. The buffer stores five parallel lists: states, actions, log-probabilities, rewards, and terminal flags.

3) PPO Update Mechanism: When the accumulated timestep count reaches the configured `update_timestep` threshold, the PPO update procedure is triggered. The update process consists of the following steps:

- Discounted Return Computation: Rewards are processed in reverse chronological order. At each terminal state, the cumulative reward is reset to zero. The discounted return at timestep t is $G_t = r_t + \gamma \cdot G_{t+1}$, and the resulting tensor is normalized to zero mean and unit variance for training stability.
- K-Epoch Optimization: For K epochs, the current policy re-evaluates all stored transitions. The importance sampling ratio : $r_t(\theta) = \exp\left(\log \pi_\theta(a_t|s_t) - \log \pi_{\theta_{\text{old}}}(a_t|s_t)\right)$, and the advantage is $\hat{A}_t = G_t - V_\theta(s_t)$.
- Composite Loss: The total loss function combines three terms: Clipped Policy Loss, Value Loss, and Entropy Bonus as defined in Equation (4).
- Policy Synchronization: After the K-epoch update, the old policy network $\pi_{\theta_{\text{old}}}$ is synchronized by copying the weights of the updated network: $\theta_{\text{old}} \leftarrow \theta$.

The training loop configuration is summarized in Table 6.

Parameter	Description	Default Value
max_episodes	Maximum number of training episodes	1000
max_timesteps	Maximum timesteps per episode	200
update_timestep	Buffer size before PPO update is triggered	2000
γ	Discount factor for return computation	0.99
Reward normalization	$(\mathbf{G} - \mu_G)/(\sigma_G + 10^{-7})$	Enabled

Table 6: Training loop configuration

IV. Results

A. Simulation Setup

The simulation system was set up and run for a fixed duration, during which agents performed actions such as adjusting states (cell mode, base station mode) and parameters (tx_power) with the objective of minimizing the total network power consumption and maximizing throughput for users.

1) Network Topology Configuration: The system was configured with a simulation period from 00:00:00 to 00:50:00, corresponding to a total duration of 50 minutes, where each time step was set to 1 second to ensure sufficient resolution for monitoring and evaluating system fluctuations over time.

The simulation deploys 6 gNBs, each configured with 3 sector antennas, yielding 18 sectors in total in accordance with the 3GPP standard [14]. Each sector employs a directional antenna with 3D radiation pattern conforming to 3GPP TR 38.901 Table 7.3-1 [10], and operates across 3 frequency layers (700/1800/3500 MHz), resulting in 54 cells (ID 0–53) across the entire network. A total of 200 UEs are randomly distributed within a 6×6 km^2 coverage area.

Parameter	Value
Area size	6000 m × 6000 m
Terrain type	UMi
Simulation seed	42
Grid resolution	160
Minimum RSRP threshold	−120 dBm
Handover Hysteresis delay	3 dB
Cell selection load index	load_actual

Table 7: Terrain parameters and general configuration

UEs were simulated with mobility models including stationary, random movement, directional movement, and random waypoint, with speeds ranging from 0 to 15 m/s. Each UE has a DL demand of 1-20 Mbps, concentrated during peak hours (8-10h, 18-21h). System performance was evaluated through metrics such as DL/UL throughput, energy consumption, number of handovers, number of disconnected UEs, UL QoS ratio, and corresponding cumulative values. The coverage map for the network used in the algorithms is described in Figure 3.

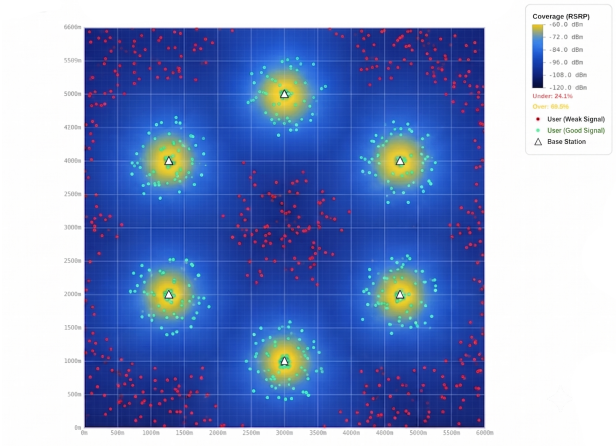

Fig. 3: Coverage map for the network used in the algorithms

B. Results and Evaluation

This subsection presents the experimental results to evaluate and compare the effectiveness of the proposed methods for the energy saving problem, based on the integration of heuristic algorithms such as EHSA with theory-based optimization techniques, combined with DRL, and the application of DeepSeek-R1 in reward function design for the PPO agent. The results below will clarify the degree of improvement, stability, and differences between the methods.

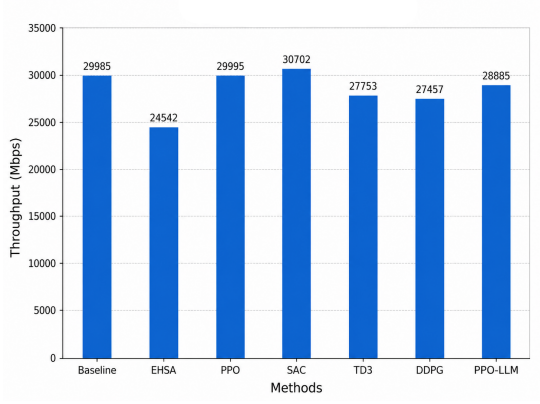

Fig. 4: Throughput comparison among evaluated algorithms

1) Throughput Performance Analysis: Figure 4 illustrates the throughput performance of the considered optimization methods. Among all algorithms, SAC achieves the highest throughput with 30702 Mbps, followed by PPO with 29995 Mbps and the Baseline scheme with 29985 Mbps. The proposed PPO-LLM framework obtains a throughput of 28885 Mbps, which is slightly lower than SAC and PPO but still remains competitive compared with other DRL-based methods.

Compared with TD3 and DDPG, PPO-LLM improves the throughput from 27753 Mbps and 27457 Mbps to 28885 Mbps, respectively. This indicates that the proposed framework can maintain a relatively high data transmission capability while operating under an energy-aware optimization objective. Although PPO-LLM does not achieve the maximum throughput, the obtained result shows that the integration of LLM-assisted reward guidance can support the PPO agent in making

more balanced resource allocation decisions without causing a significant degradation in network throughput.

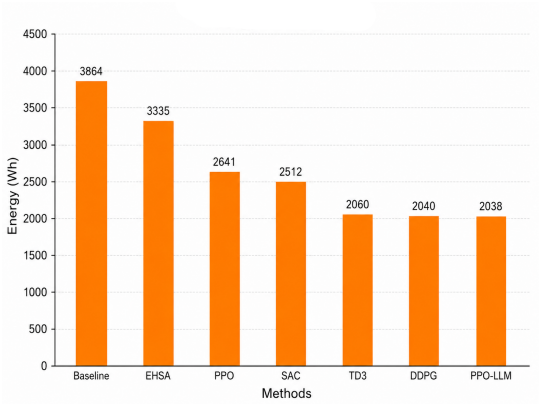

Fig. 5: Energy consumption comparison among evaluated algorithms

2) Energy Consumption Analysis: As shown in Figure 5, all optimization-based methods reduce energy consumption compared with the Baseline scheme. The Baseline consumes the highest energy, reaching 3,864 Wh, while the proposed PPO-LLM framework achieves the lowest energy consumption of 2,038 Wh. This corresponds to an energy reduction of approximately 47.3% compared with the Baseline.

Although TD3 and DDPG also obtain low energy consumption values of 2,060 Wh and 2,040 Wh, respectively, PPO-LLM still provides the best result. Moreover, compared with SAC, PPO-LLM reduces energy consumption by about 18.9%, while maintaining competitive throughput performance.

Method	**Throughput** (Mbps)	**Energy** (Wh)	**Efficiency Evaluation**
Baseline	29985	3864	Low
EHSA	24542	3335	Moderate
PPO	29995	2641	Good
SAC	**30702**	2512	High
TD3	27753	2060	Very high
DDPG	27457	2040	Very high
PPO-LLM	*28885*	**2038**	**Optimal**

Table 8: Summary of network performance metrics

Overall, PPO-LLM provides the most favorable energy-saving capability while preserving a high throughput level. These results indicate that the proposed framework achieves a better trade-off between throughput and energy consumption, leading to improved energy efficiency compared with the other evaluated algorithms.

V. Conclusions and Future Research Directions

A. Conclusions

This paper proposed a PPO-LLM framework integrating Proximal Policy Optimization with LLM-guided reward engineering for energy-efficient small-cell control in 5G HetNets. By replacing manual reward design with a closed-loop LLM-based generation pipeline, the proposed framework reduced energy consumption by approximately 47% relative to the unoptimized baseline while maintaining a downlink throughput of 28,885 Mbps, outperforming all six evaluated baselines in overall energy efficiency. These results validate LLM-guided reward engineering as a scalable, expert-independent approach for intelligent energy management in next-generation wireless networks.

B. Future Research Directions

Future work will focus on three directions. First, validating PPO-LLM on real network traces or operator-grade testbeds to assess robustness beyond simulation assumptions. Second, expanding the LLM's role from reward design to policy advising and explainable RL, particularly to improve performance under high-load conditions where the throughput gap with SAC remains. Third, extending the framework to multi-agent reinforcement learning (MARL) to enable distributed decision-making in ultra-dense deployments beyond the current 6 gNB architecture.

References

[1] C. Luo and J. Liu, "Load based dynamic small cell on/off strategy in ultra-dense networks," in *IEEE WCSP*, Hangzhou, China, 2018, pp. 1–6.
[2] J. Wu, J. Liu, and H. Zhao, "Dynamic small cell on/off control," in *WCSP*, 2016, pp. 1–5.
[3] H. Wu, X. Xu, Y. Sun, and A. Li, "Energy efficient base station on/off," in *IEEE WCNC*, 2017, pp. 1–6.
[4] T. P. Lillicrap et al., *Continuous control with deep reinforcement learning*, arXiv:1509.02971, 2015.
[5] T. Haarnoja et al., "Soft actor-critic," in *ICML*, 2018, pp. 1861–1870.
[6] S. Fujimoto et al., "Addressing function approximation error," in *ICML*, 2018, pp. 1587–1596.
[7] J. Schulman et al., *Proximal policy optimization algorithms*, arXiv:1707.06347, 2017.
[8] H. Zhou et al., "Large language model for telecommunications: A survey," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 3, pp. 1955–2005, 2025.
[9] D. Wang et al., *Enhancing llms for telecommunications*, arXiv:2503.24245, 2025.
[10] 3GPP, *Study on channel model for frequencies from 0.5 to 100 ghz*, TR 38.901, 2020.
[11] 3GPP, *Nr; physical layer procedures for data*, TS 38.214, 2020.
[12] 3GPP, *Nr; physical layer procedures for control*, TS 38.213, 2020.
[13] 3GPP, *Nr; radio resource control (rrc)*, TS 38.331, 2020.
[14] 3GPP, *Nr; overall description*, TS 38.300, 2020.
[15] 3GPP, *System architecture for 5gs*, TS 23.501, 2020.
[16] 3GPP, *Policy and charging control*, TS 23.203, 2020.
[17] J. S. Pujol-Roig et al., "Deep reinforcement learning for cell on/off energy saving," in *IEEE GLOBECOM*, Madrid, Spain, 2021, pp. 1–7.