Generative AI-Augmented Reinforcement Learning for Enhanced Edge Resource Optimization

Shreya K. Chari*, John S. Vardakas*†, Kostas Ramantas*, Christos Verikoukis‡§
* Iquadrat Informatica S.L., Barcelona, Spain
† Department of Informatics, University of Western Macedonia, Greece
‡ University of Patras, Patras, Greece
§ Industrial Systems Institute (ISI) | Athena Research Center, Patras, Greece
*{s.chari, kramantas}@iquadrat.com, †ivardakas@uowm.gr, ‡cveri@ceid.upatras.gr

***Abstract*—Reinforcement learning (RL) has proven to make proficient resource allocation decisions at the network edge with limited resources and diversified services using real-time interactions with the network environment. However, a heavy dependence on the custom-made reward functions decelerates and often deviates an RL agent's policy convergence. With the proliferation of Generative Artificial Intelligence (GenAI) applications, the RL agent can be enhanced for more optimized performance. In this paper, the pre-trained Large Language Model (LLM) MathΣtral is utilized for a novel rewardless navigation policy within a Soft-Actor Critic (SAC)-based RL environment optimized for edge resource management. The algorithm is designed to address resource allocation challenges in a cloud-edge network that supports heterogeneous service requirements. Edge resources such as CPU, memory and bandwidth are distributed using the RL policy. The proposed GenAI-guided strategy is compared with several algorithms for benchmarking. As a result, the proposed system experiences lower network latencies and higher data rate for every unit of resources spent. Further, resource distribution between the considered algorithms is shown while elaborating the underlying causes of the performance.**

***Index Terms*—Reinforcement learning (RL), Generative-AI (GenAI), Large Language Models (LLMs), Resource management, Cloud-edge networks**

I. Introduction

Modern communication systems are burdened with computationally demanding applications now more than ever. In recent times, RL has emerged as a viable solution at the network edge to resolve complex tasks such as resource allocation, offloading, caching and scheduling [1]. The evolution within the RL domain in terms of optimization of the algorithms to replicate real-world performances and to reach human-like learning capabilities has been astonishing. However, there are few unresolved issues such as slow policy convergence, lack of trainable data, reward shaping and computational overheads that creates hindrance for a RL based system to reach its maximum efficiency.

Simultaneously, GenAI has revolutionized contextual understanding, explainability and adaptive reasoning for tasks involving decision making. Non-trivial tasks such as classifying network attacks, resource block allocation, traffic load prediction and user mobility are some of the areas that can benefit from GenAI [2]. The co-existence of RL and LLM based systems is inevitable and hence integrating these two technologies for optimized performance is the way ahead. Using LLMs, the frequently occurring issues in RL models like reward generalization, error compounding during training, multimodal learning and many other issues are alleviated [3]. RL systems excel in the exploratory phase by constantly learning from interactions with the dynamic environment. Meanwhile, LLMs based on GenAI can summarize the decision-making process through human-like feedback. A fine-tuned LLM prompt can perform real-time analysis relying on the supporting evidence provided. This technique is called Chain-of-Thought (CoT) prompting where LLMs can act as advisors to a RL policy shaping the reward as per the user's intent [4]. This approach would eliminate the need for custom reward shaping within RL pipelines enabling smarter policy convergence. In this paper, a traditional RL reward is replaced with the pre-trained *MathΣtral* model. Further, the performance of RL resource management system utilizing the inferencing abilities of LLM is observed. The main contributions are as follows:

- An edge network based resource management strategy optimizing network parameters such as *latency* and *data rate* is developed. The model training is performed on a realistic cloud-cluster data providing *CPU*, *memory* and *bandwidth* values.
- An RL algorithm using SAC policy for resource distribution is implemented. The algorithm is augmented by GenAI driven rewardless navigation eliminating the need for custom reward shaping.
- The RL-SAC policy using CoT prompting-guided reward function is compared against different policies using custom reward function.
- Lower *latency* and higher network *data rate* for every unit of resource spent is achieved using the formulated algorithm compared to the baseline algorithms. Further, balanced resource allocations can additionally be observed.

The remaining paper is organized as follows. Section II presents the recent advancements in the state-of-the-art combining the power of RL and LLM for edge network management. The system model specifications and problem formulation is described in Section III. Section IV explains the LLM guided rewardless navigation of the RL policy. In

Section V, the experimental setup used for evaluation of the proposed idea is presented. Finally, in Section VI and VII the results and the conclusions of the work are established.

II. Related Work

This paper utilizes GenAI to augment the performance of RL-SAC algorithm at the network edge. Efforts have been made within the state-of-the-art literature to replace complex tasks for the cloud-edge continuum using AI generated content, diffusion models and LLMs. The authors of [5] propose an edge inference framework using LLM ensuring high throughput for edge devices. In [6], a multi-exit U-ViT model for diffusion models that generate images and its impact on energy in a resource constrained deployment is studied. An LLM inference model to calculate the delay of content generation to optimize offloading decisions to the edge is considered in [7]. To improve the decisions for offloading and resource allocation for LLM inference tasks to servers, the authors in [8] suggest a rewardless algorithm with active inferencing. A unified mobile-edge AI-Generated Everything (AIGX) is executed in [9] for optimizing edge device services using prompt engineering. Finally, in the research work [10], Attention-based Diffusion Soft Actor-Critic (ADSAC) is used to AI Generated Content (AIGC) model selection addressing various user requests. Despite these strong contributions, LLMs are not being used to their full potential within the edge infrastructures. GenAI or the LLM's usage so far in the literature can be categorized in the following way: 1) As an interface between the user and the network backend system, 2) AI based content generation which in turn serves as the network load to be offloaded to the servers and 3) for selecting the best reward signals instead of eliminating them completely. Keeping in mind the aforementioned discussion, in this paper we completely eliminate the need for a custom reward shaping and use the powerful inferencing capabilities of LLM to enhance the system performance going beyond the trivial tasks of load generation and being a user interface.

III. System model and Formulation

In this paper, an edge infrastructure with Multi-access Edge Computing (MEC) node $\mathcal{M}$ providing *CPU* c, *memory* m and *bandwidth* b resources to different network services is considered. The services offloaded to the edge node are assumed to be generated by the users while using different applications as shown in Fig.1. The demand requests put forth by the offloaded services are expected to be served by $\mathcal{M}$ as learnt by the RL-SAC policy agent. It is assumed that each service is given minimum amount of resources and they are not left unserved.

For observing the performance enhancement in this work, network *latency* and *data rate* is calculated using the well-known Shannon formula in (4) [11]. Here the *data rate* DR is dependent on the communication channel bandwidth $B_{eff} = bB_{fixed}$ which is a product of the fixed bandwidth and the actual bandwidth utilization. The DR further uses, the SNR which is arrived at using the noise power and received power

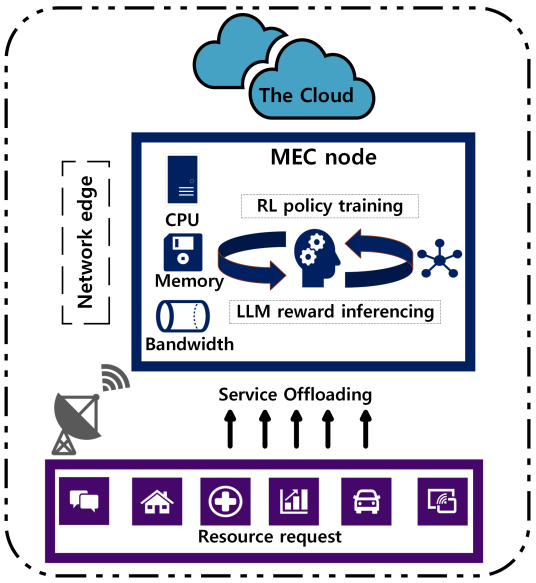

Fig. 1: Network edge infrastructure

R_X through (1)-(3). The network *latency* L is measured by dividing the packet size p by the DR. The values used for calculating DR and L are obtained from available standards and are summarized in Table I [12].

$$\textit{Noise Power} = -174 + 10\log_{10}(B_{fixed}) + NF \quad (1)$$

$$R_{\mathrm{X}} = P_{TX} - PL + G_{TX} \quad (2)$$

$$\mathrm{SNR} = R_{\mathrm{X}} \; - \; \textit{Noise Power} \quad (3)$$

$$DR = B_{\mathrm{eff}} \; \log_2(1 + \mathrm{SNR}) \quad (4)$$

A. Problem formulation

The resource demand from the offloaded services is received in the form of a demand set $\mathcal{D}(t) = \{c_d, m_d, b_d\}$ where c_d is *CPU*, m_d is *memory* and b_d is *bandwidth* requirements. The edge server MEC node $\mathcal{M}$ is responsible for distributing the available resources while maintaining optimal *latency* and *data rate* values. Therefore, the system is posed as an optimization problem maximizing resource allocation set at time t as $\mathcal{A}(t) = \{c_a, m_a, b_a\}$ subjected to constraints as follows:

$$\max_{t \in \mathcal{T}} \mathcal{A}(t) \quad (5)$$

$$r \leq \mathcal{R}, \forall r \; \epsilon\{c_a, m_a, b_a\}, \forall \mathcal{R} \; \epsilon\{C, M, B\} \quad (6)$$

TABLE I: Communication model parameters

Parameter	Symbol	Value
Fixed bandwidth	B_{fixed}	20 MHz
Noise figure	NF	7dB
Transmit power	P_{TX}	46 dBm
Path loss	PL	125 dB
Antenna gain	G_{TX}	5.5 dBi
packet size	p	1 MB

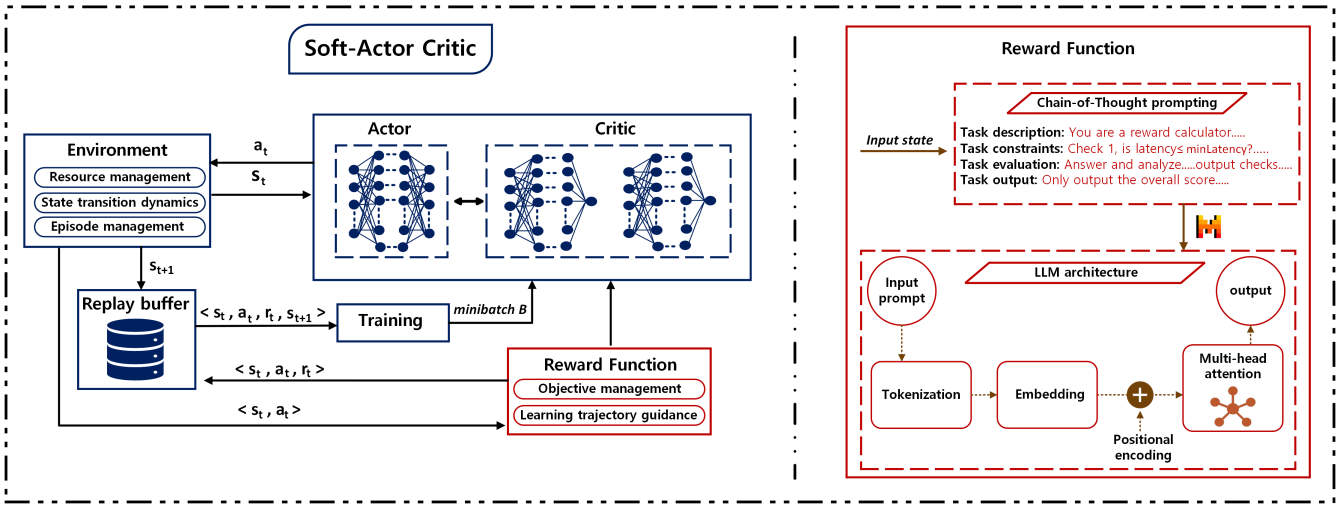

Fig. 2: RL-SAC performance enhancement using LLM based reward guidance

$$\frac{d}{a} \geq 1, \forall d \epsilon \{c_d, m_d, b_d\}, \forall a \epsilon \{c_a, m_a, b_d\} \tag{7}$$

$$L_t \leq L_{max}, \forall t \tag{8}$$

$$DR_t \geq DR_{min}, \forall t \tag{9}$$

Here (6) ensures that the allocations do no exceed the maximum capacity limits. Under or over commitment of resources is controlled by (7). The *latency* L and *data rate* DR are expected to reach minimum thresholds set by (8) and (9).

IV. Generative-AI guided resource management

In this work, the pre-trained *Math*Σ*tral* model following principles of GenAI is incorporated into the RL-SAC policy iteration. The purpose is to exploit the potential of both RL and GenAI models for advanced resource management with the underlying assumption that both models are bound to co-exist.

A. LLMs for numerical reasoning

LLMs are widely used for Natural Language Processing (NLP) applications owing to their ability in establishing relation between different words in a long sequence of texts. A LLM predicts the probability distribution of a sequence of words $w_1, w_2, ..., w_T$ by estimating the joint probability as:

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, \ldots, w_{t-1}) \tag{10}$$

The recent advancements in prompt engineering has made textual user instructions to code implementation a reality. Combining the prompts with the LLMs prediction skills gives rise to numerical reasoning applications such as ranking, scoring or quantitative calculations. This feat can be attributed to the *self-attention mechanism* that is used by the transformer architecture within the LLMs to establish long-range dependencies and contextual understanding [13]. Each input sequence is converted to token embeddings $x = [x_1, x_2, ..., x_n]$ and for each embedding x a query vector $Q = xW^Q$, key vector $K = Xw^K$ and a value vector $V = xW^V$ is calculated. The attention output is then calculated as in (11) for deciding the relevance of the tokens where d_k is the dimensionality of the key vectors.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{11}$$

By diligently engineering the prompts to the LLM, accurate numerical outputs can be obtained. Therefore, in this paper we take advantage of LLMs capacity to predict rewards for a policy given the task is explained well via prompts.

B. Enhancing RL-SAC efficiency without reward shaping

The resource optimization problem at the network edge is formulated as (S, A, P, R), a Markovian Decision Process (MDP) to be further trained using a SAC policy based RL algorithm. The transition function $P : S \times A \times S \rightarrow [0, \infty]$ maps the state transition from s_t to s_{t+1} to state space S when an action a_t belonging to action space A is taken. The reward function R is the main focus of this work and is acquired from the LLM model. The MDP components are listed as follows:

State: The state $s_t = [c_a, c_d, m_a, m_d, b_a, b_d, L, DR]$ at time t includes the instantaneous resource allocations, demands, *latency* and *data rate*.

Actions: The goal of this work is to allocate resources at the edge node and hence at time t the action space is denoted as $a_t = [c_a, m_a, b_a]$ that is the corresponding allocations to be made for each service request.

LLM driven rewards: For eliminating the need for customized reward shaping, the pre-trained *Math*Σ*tral* model is used [14]. The model is trained on 7 billion parameters specifically for mathematical reasoning. The objective function and the constraints are carefully crafted using CoT style prompts. These prompts dynamically evolve by consuming values from the state space. Fig.2 depicts the mechanism through which the traditional reward function is replaced by

Algorithm 1: RL-SAC policy augmented by LLM

Initialize replay buffer $\mathbb{D}$
policy parameters θ
Set target networks ϕ_i
for *episode=1 to N* **do**
  Obtain initial state S_t
  for *steps=1 to M* **do**
    Sample action a_t from policy $\pi_\theta(.|s_t)$
    Receive next state s_{t+1}, LLM reward r_t, info d
    Store $(s_t, r_t, a_t, s_{t+1}, d)$ in $\mathbb{D}$
    if *training* **then**
      Randomly sample a minibatch $\mathbb{B}$
      Calculate the Q-function targets y_t
      $r_t + \gamma(min\mathcal{Q}_T(s_{t+1}, a_{t+1})) - \alpha log\pi_\theta(a_{t+1}|s_{t+1})$
      Update Q-function $\mathcal{L}_Q$
      $\frac{1}{|\mathbb{B}|} \sum_{s'\epsilon B} (Q(s_t, a_t) - y_t)^2$
      Update policy $\mathcal{L}_\pi$
      $\frac{1}{|B|} \sum_{s\epsilon B} (min\mathcal{Q}(s_t, a_\theta(s_t))) - \alpha log\pi_\theta(a_\theta(s_t)|s_t)$
      Update target network parameters
      $\phi_i^{targ} \leftarrow \tau\phi_i^{targ} + (1-\tau)\phi_i$
      Update temperature coefficient α
    end
    Let $s_t \leftarrow s_{t+1}$
    Store model weights
  end
end

prompts that are engineered to output a reward score using a LLM architecture.

RL policy: SAC is a powerful model-free policy employing entropy regularization and two critic networks for achieving model stability [15]. A policy $\pi_\theta(a_t|s_t)$ maps the states to the actions. This policy then maximizes $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t(r(s_t, a_t)) + \alpha H(\pi_\theta(.|s_t))]$ where $\gamma^t(r(s_t, a_t)$ is the discounted rewards controlled by discount factor γ and $H(\pi_\theta(.|s_t))$ is the entropy term regulated by temperature coefficient α. Further, γ decides the relative worth of the immediate rewards obtained in comparison to the future rewards while α maintains a trade-off between exploration and exploitation. In Algorithm 1, the RL-SAC implementation used for this work along with LLM-based reward mechanism is described.

V. Experimental framework

The framework designed for conducting the experiments is described in this section. The particulars of the dataset used for RL policy training and the system setup for the experiments is mentioned followed by the baseline algorithms used for benchmarking.

A. Training dataset and system settings

Dataset: Microsoft Azure's traces for packing offers allocation workloads [16]. These workloads are associated with Virtual Machine (VM) requests which in turn are related to the tenant, type and priority. The dataset stored in a sqlite database format has two tables VM requests and VM types which are pre-processed in conjunction. The *CPU, memory* and *bandwidth* utilization values are extracted for the experiments serving as the resource demand set $\mathcal{D}$.

Experiment settings: The experiments were run using NVIDIA GH200 resources [17]. The algorithms were simulated on Python utilizing Tensorflow and PyTorch modules [18]–[20]. The RL policy is trained twice within an episode after every 512 steps. The actor and critic network's learning rate is set at 0.0001 and 0.001 for faster value-function fitting. The minibatch $\mathbb{B}$ size is set to 64 while using the Adam optimizer. Additionally, the discount factor γ is kept at 0.90 and a temperature coefficient α of value 0.2 is used. The state space is initialized again at the end of each episode.

B. Performance comparison

The performance of the GenAI guided RL-SAC policy is benchmarked against a RL-SAC policy, a Bayesian optimizer and a random policy. The custom reward function used for these algorithms is mentioned in (12). The rewards follow the same constraints and objectives as presented in the problem formulation; every time the policy makes an acceptable allocation an award is given to the reward score and every time it violates a constraint, the reward score is penalized. The award R_{award} is a function of the difference between the resources $\Delta_t = e_{target} - e_{actual}$ where $e_{target}\epsilon\mathcal{D}$ and $e_{target}\epsilon\mathcal{A}$. Whether the constraints have been met or not is decided by δ_t.

$$reward_t = \begin{cases} +R_{\text{award}}(\Delta_t), & \text{if } \delta_t = 1, \\ -R_{\text{penalty}}, & \text{if } \delta_t = 0. \end{cases} \tag{12}$$

The baselines used are elaborated below:

- **SAC:** A SAC policy using the above mentioned reward mechanism is also implemented to showcase the difference in learning on changing the reward function.
- **Bayesian optimization:** It is a black-box approach suitable for resource optimization in continuous action spaces. A probability distribution based on a Gaussian process is used for sampling. It is an efficient tool for searching parameter space and is often a popular choice in resource optimization problems [21].
- **Random:** It is a uniformly distributed random policy that sample actions a_t without any exploration bias. Each action is equally likely to be sampled independent of the prior selection hence appropriate as a baseline as well.

VI. Result evaluation

In this section, the performance of Gen-AI guided RL-SAC is compared against several other considered baselines. *Latency* L and *data rate* DR as described in Section III is calculated for each state s_t while using different policies. The *CPU, memory* and *bandwidth* distributions across the policies are further presented. The results are averaged over the last three episodes keeping in mind that the RL policy converges towards the end after extensive exploration of the state space.

Rewards: Any desired behavior for an objective using RL is shaped by the rewards it receives within the MDP structure. The RL agent's learning trajectory depends on how well a reward value is able to reflect on the conditions an environment

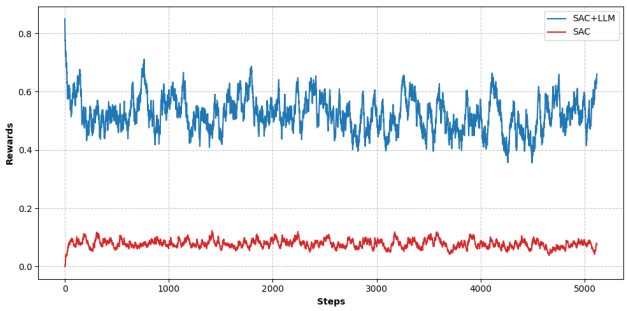

Fig. 3: Normalized received rewards

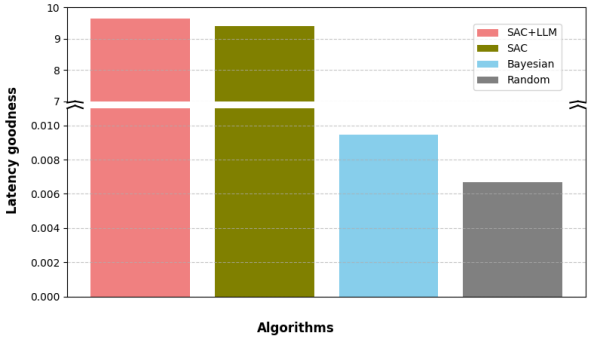

Fig. 4: Performance evaluation using latency

can be found in which is further signaled to the model training process. In Fig.3 the normalized rewards obtained from the SAC policy while using the LLM-guided reward function and the traditional reward function is shown. The rewards are smoothed using exponential smoothing for better understanding. It can be seen that the policy using LLM-guided rewards results in higher rewards signaling better state conditions to the model training process.

Network performance: A *goodness* metric is calculated to measure performance of the algorithms. The *goodness measure* is defined as $goodness = \{Latency, Datarate\}\frac{b_d}{b_a}$ averaged over episodes where the *latency* was observed in seconds and *data rate* in Mbps. The purpose of this metric is to visualize the effective *latency* or *data rate* achieved for the resource units spent. For example, if the *latency* or the *data rate* value is high when there is resource over-allocation that is, $\frac{b_d}{b_a} < 1$, the *goodness* value is low indicating high resource expenditure and vice versa. The *goodness* metric ensures fair resource provisioning by not exaggerating the performance indicators such as *latency* and *data rate* at the cost of over or under-allocation.

From the discussion, it can be summarized that higher *goodness* measure means better and efficient performance. From Fig.4 and Fig.5, it is clear that the proposed RL-SAC using a LLM powered reward function has the highest *goodness* values for both *latency* and *data rate* compared to all the other algorithms. The SAC policy using custom reward function has comparatively lower goodness values which can be attributed to uneven resource distribution because of the limitations posed by custom rewards. The worst performance is seen by the bayesian and random policy as they do not have the ability to train from replay buffers halting them from foreseeing many states. The proposed RL-SAC with LLM guided rewards benefit from LLMs ability to dynamically change its thought process while evaluating each state. Hence, the RL policy is prevented from diverging and reaching extreme values.

Resource management: Fig.6 showcases the average of network resources $\mathcal{A}(t) = \{c_a, m_a, b_a\}$ distributed by the algorithms given the system environment. The resources were distributed by the algorithms based on the resource demands $\mathcal{D}(t) = \{c_d, m_d, b_d\}$ put forth by the system state space. The network resources are assumed to be equally available within the MEC node $\mathcal{M}$ for dispensing. It can be observed that the RL-SAC policy with the LLM rewards distributes the *CPU, memory* and *bandwidth* almost equitably avoiding over or under-allocation. The SAC policy with custom reward function slightly under-allocates the *bandwidth* resources which also explains its performance in the *goodness* metrics. Bayesian optimization seems to be performing the worst in this regard by over-allocating the *CPU* resources followed by the random policy.

Such a resource distribution can be credited to LLMs context-aware scoring when compared to a fixed reward function which is not that flexible. Further, a LLM-guided RL policy is able to maintain a fair trade-off between satisfying the performance indicators and resource allocation without the need of tuning or shaping a reward. In turn, the RL agent learns to generalize the feedback received in a better way adapting itself to any new resource demand which is yet to be encountered.

Scalability: As networks become more and more complex both systems and algorithms need to move towards automation with once in a while human feedback for greater efficiency. Reward shaping as introduced is extremely challenging and becomes cumbersome in non-trivial multi-objective scenarios. Moreover, black-box models such as RL lack interpretability and explainability making it extremely difficult for any intervention. Moving forward, LLM- guided RL policies seem like the ideal mix for faster exploration and policy convergence.

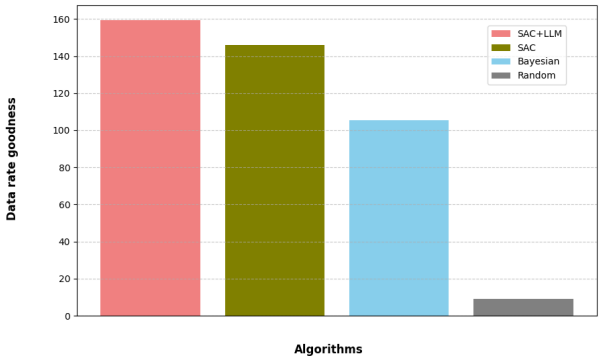

Fig. 5: Performance evaluation using data rate

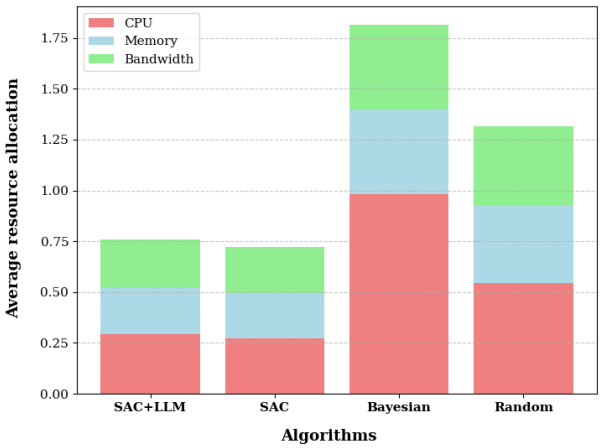

Fig. 6: Resource distribution performance

VII. Conclusion and Future Work

A novel GenAI augmented RL technique for resource management at edge network MEC nodes is proposed within this paper. A SAC policy trained on publically available Azure traces with reward shaping using the pre-trained *Math*Σ*tral* LLM is implemented. The primary objective is to manage edge resources such as *CPU, memory* and *bandwidth* while maintaining service standards. The proposed algorithm is then compared with benchmark algorithms in terms of *goodness* metric and resource allocation. The SAC policy utilizing rewards from the LLM outperforms SAC policy with fixed custom reward, Bayesian optimizer and random policy. Detailed reasoning for the improved performance in terms of balanced resource distribution of *CPU, memory* and *bandwidth* is provided.

For future work, agentic-AI along with human-in-the-loop feedback system can be considered for performance enhancement. Multi-agent based continual learning and explainability is also an interesting avenue. Hierarchical decomposition of larger objectives into smaller tasks using LLM can further accelerate policy convergence. To conclude, GenAI can be positioned within a RL pipeline in many ways to boost efficacy.

VIII. Acknowledgement

This research was funded by the AC3 (Agile and Cognitive Cloud-edge Continuum management) project (grant no. 101093129).

References

[1] N. C. Luong et al., “Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3133-3174, 2019.

[2] H. Zhou et al., “Large Language Model (LLM) for Telecommunications: A Comprehensive Survey on Principles, Key Techniques, and Opportunities," IEEE Communications Surveys & Tutorials, 2024.

[3] Y. Cao et al., “Survey on Large Language Model-Enhanced Reinforcement Learning: Concept, Taxonomy, and Methods," IEEE Transactions on Neural Networks and Learning Systems, pp. 1-21, 2024.

[4] Jason Wei et al., “Chain-of-thought prompting elicits reasoning in large language models," in Proc. NeurIPS, 2022.

[5] X. Zhang et al., “Beyond the Cloud: Edge Inference for Generative Large Language Models in Wireless Networks," IEEE Transactions on Wireless Communications, vol. 24, no. 1, pp. 643-658, January 2025.

[6] M. Tian, Z. Liu, C. Qiu, X. Wang, D. Niyato and V. C. M. Leung, "Enabling Collaborative and Green Generative AI Inference in Edge Networks," in Proc. IEEE Global Communications Conference, 2024.

[7] H. Zhou et al., “Generative AI as a Service in 6G Edge-Cloud: Generation Task Offloading by In-Context Learning," IEEE Wireless Communications Letters, vol. 14, no. 3, pp. 711-715, March 2025.

[8] J. Fang, Y. He, F. R. Yu, J. Li and V. C. Leung, “Large Language Models (LLMs) Inference Offloading and Resource Allocation in Cloud-Edge Networks: An Active Inference Approach," in Proc. IEEE 98th Vehicular Technology Conference (VTC2023-Fall), 2023.

[9] Y. Liu et al., “Optimizing Mobile-Edge AI-Generated Everything (AIGX) Services by Prompt Engineering: Fundamental, Framework, and Case Study," IEEE Network, vol. 38, no. 5, pp. 220-228, September 2024.

[10] Y. Liu, X. Lin, S. Li, G. Li, Q. Mao and J. Li, “Towards Multi-Task Generative-AI Edge Services with an Attention-based Diffusion DRL Approach," in Proc. IEEE International Conference on Smart Cloud (SmartCloud), 2024.

[11] C. E. Shannon, “A mathematical theory of communication,” The Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, 1948.

[12] ETSI, “Study on International Mobile Telecommunications (IMT) parameters for 6.425 - 7.025 GHz, 7.025 - 7.125 GHz and 10.0 - 10.5 GHz," 3GPP TR 38.921 version 17.1.0, 2022.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” Advances in Neural Information Processing Systems, vol. 30, 2017.

[14] Mathstral. [Online]. Accessed: April 28, 2025. Available: https://mistral.ai/news/mathstral

[15] Soft Actor Critic. [Online]. Accessed: April 28, 2025. Available: https://spinningup.openai.com/en/latest

[16] Ori Hadary et al., "Protean: VM Allocation Service at Scale," in Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI 2020), November 2020.

[17] NVIDIA GH200 Grace Hopper Superchip. [Online]. Accessed: April 28, 2025. Available: https://www.nvidia.com/en-us/

[18] Python. [Online]. Accessed: April 28, 2025. Available: https://www.python.org/

[19] Tensorflow. [Online]. Accessed: April 28, 2025. Available: https://www.tensorflow.org/

[20] PyTorch. [Online]. Accessed: April 28, 2025. Available: https://pytorch.org/

[21] Yan, Jia and Lu, Qin and Giannakis, Georgios B., “Bayesian Optimization for Online Management in Dynamic Mobile Edge Computing," IEEE Transactions on Wireless Communications, pp. 3425–3436, vol. 23, April 2024.