DRL-enabled SLO-aware Task Scheduling for Large Language Models in 6G Networks

Abdelkader Mekrache
EURECOM
France
abdelkader.mekrache@eurecom.fr

Adlen Ksentini
EURECOM
France
adlen.ksentini@eurecom.fr

Christos Verikoukis
University of Patras and ISI/ATH
Greece
chverik@gmail.com

***Abstract*—With the rapid advancement of telecommunications, 6G networks are expected to become more intelligent and capable of making autonomous decisions. Artificial Intelligence (AI) will play a crucial role in achieving this, particularly through the use of Large Language Models (LLMs). These models are increasingly being adopted for networking tasks due to their advanced capabilities in coding, reasoning, and language processing. LLMs have significant potential to support the development of autonomous networks by reducing or even eliminating the need for human intervention. However, LLMs are computationally expensive, which necessitates their shared use across different 6G applications, i.e., a single LLM might be required to perform multiple tasks within a 6G network. To this end, routing tasks to the appropriate LLMs presents several challenges: (i) the arrival time of tasks is unpredictable, (ii) tasks must meet specific deadlines, which are part of the Service-Level Objectives (SLOs), and (iii) each LLM may perform better on different types of tasks, leading to varying task scores. In this paper, we propose a Deep Reinforcement Learning (DRL) approach for routing tasks to a set of LLMs (task scheduling). Our goal is to maximize task scores while ensuring their deadlines are met. Evaluations conducted under real-world conditions show that our DRL-based approach outperforms traditional methods like Round-Robin (RR) and random scheduling.**

***Index Terms*—6G networks, autonomous networks, AI, LLMs, DRL, SLO, task scheduling.**

I. Introduction

6G, the next generation of wireless communication technology, is anticipated to be more intelligent and cognitive, incorporating cutting-edge innovations to meet future connectivity demands. A key pillar of 6G, as emphasized by standardization bodies like 3GPP and ETSI, is the integration of Artificial Intelligence (AI) to enable autonomous and self-evolving networks [1]. This is crucial for managing advanced 6G use cases that conventional approaches cannot handle. To this end, multiple 6G concepts will rely on AI, including Zero-touch network and Service Management (ZSM), which facilitates autonomous anomaly detection and resolution, and Intent-Based Networking (IBN), which enables intent-driven autonomous networking [2]. In this context, various AI approaches are being investigated for different 6G challenges (e.g., anomaly detection, resource allocation, and task scheduling), all aimed at realizing these advanced 6G concepts.

Recently, with the rapid expansion of Generative AI (GenAI), Large Language Models (LLMs), advanced AI systems capable of understanding and generating human-like text, have attracted significant attention for their applications in networking problems [3]. These models, such as OpenAI's closed-source GPT series and the open-source Llama and Mistral series [4], excel in various tasks, including coding, reasoning, and language processing. This makes them particularly well-suited for networking tasks that involve, for example: (i) coding, for generating network code from 3GPP standards [5]; (ii) reasoning, for autonomous anomaly resolution based on 6G Key Performance Indicators (KPIs) to enable ZSM [6]; and (iii) language processing, for intent-based human-network interactions to enable IBN [7]. However, LLMs face challenges due to their large size and high computational demands, which complicate their deployment in 6G networks. To address this, LLMs must be shared across multiple 6G tasks. By leveraging in-context learning, where different prompts (inputs) are used without altering the LLMs' weights, these models can handle diverse tasks concurrently [3].

In a shared set of LLMs across various 6G applications, each task must be completed with a high score within a specified deadline. This deadline is defined as part of the Service-Level Objectives (SLOs). Since some LLMs excel, for example, in reasoning while others are better at language tasks, selecting the most suitable LLM for each task, while considering task deadlines and the load on each model, is essential. This presents a multi-objective optimization problem aimed at maximizing task scores and minimizing task latencies. The challenge is further complicated by the unpredictable tasks arrival times, varying task types, and the uncertainty of score calculation (which are only possible once the task response is generated), making this optimization problem difficult to solve. To address this, we propose a Deep Reinforcement Learning (DRL)-based approach for LLM task scheduling that seeks to jointly maximize task scores and minimize task latencies. The DRL agent observes the load on the LLM queues, as well as the type and deadline of arriving tasks within a given time window, and then routes each task to an appropriate LLM. To the best of our knowledge, this is the first work to leverage DRL for SLO-aware LLM scheduling, taking into account different types of tasks in a 6G network setup.

The main contributions of this work are manifold:

- We model the multi-objective optimization problem and the objective function to maximize task scores and minimize task latencies.

- We introduce a DRL-based solution to address the aforementioned objective function of the optimization problem. The state space includes the task load on each LLM's queue, as well as the type and deadline of the arriving task. The action involves routing the arriving task to one of the LLMs, and the reward is formulated according to the objective function.
- The proposed DRL solution is designed to handle multiple tasks with varying types and arrival patterns, i.e., the DRL agent is trained only once and then deployed, regardless of the number of tasks and their arrival patterns.
- We evaluate the solution in a realistic setting by deploying four open-source LLMs on two Nvidia A100 GPUs with 40GB of vRAM. The realistic tasks and their evaluations, i.e., score computation, are sourced from [8], and task arrivals are simulated using a Poisson distribution.

The rest of the paper is organized as follows: Section II describes background on LLMs and DRL and related works. Section III introduces the problem formulation. Our proposed solution is presented in Section IV and evaluated in Section V. Finally, we conclude the paper in Section VI.

II. Background and related works

In this section, we present background on LLMs and DRL. We then review existing literature on LLMs serving.

A. Large language models

LLMs are token generators based on the transformer architecture, i.e., given a sequence of text, they predict the next token, enabling them to generate responses to questions, produce code, and perform various other tasks. In 2023, OpenAI launched ChatGPT, a closed-source LLM designed for public use. Since then, interest in LLMs has increased, driving significant efforts from the open-source community to narrow the gap between the GPT series and open-source alternatives. Existing open-source LLMs differ in their response mechanisms, i.e., each LLM has a token generation speed depending on its architecture, and each excels at certain types of tasks over others, depending on their training methodologies. Today, powerful open-source LLMs, both large and small, such as the Llama and Mistral series [4], are widely available. These LLMs can perform numerous tasks depending on the context provided in the input, utilizing a technique known as in-context learning [3]. This approach allows LLMs to adapt to new tasks using context alone (prompt), without the need for fine-tuning, facilitating the sharing of LLMs across different use cases. In this context, LLMs are gaining significant attention in the telecommunications sector, where they are being applied to various use cases [5, 6, 7]. As a result, managing LLM serving, especially task scheduling across multiple LLMs, has become an important optimization problem that needs to be investigated.

B. Deep reinforcement learning

RL is one of the most successful AI methods, closely resembling human learning. It has been successfully applied in various areas, such as continuous control systems and games [9]. The core idea of RL involves an agent interacting with an environment to make informed decisions, receiving rewards after each action. The agent's goal is to learn a policy that maximizes cumulative rewards over time as it transitions between different states. However, traditional RL methods have faced significant challenges when dealing with large state and/or action spaces. To address these challenges, Deep Learning (DL) was combined with RL in 2013, leading to the development of Deep Q-Learning (DQN) [9], a type of DRL. DQN uses a neural network to approximate the Q-function, which maps state-action pairs to the expected return of taking a specific action in a given state. This allows the agent to select actions that maximize expected rewards, even in complex environments with vast state spaces. DRL, particularly DQN, has shown great promise in overcoming the limitations of traditional RL methods, making it a crucial approach for tasks requiring efficient decision-making in complex environments [10]. In this context, DRL has produced very promising results in task scheduling optimization problems [11]. In our work, the state includes the load of tasks on each LLM queue, as well as the type and deadline of arriving tasks, resulting in a very large state space. Therefore, traditional RL alone would not be sufficient in this case.

C. Related works

The topic of LLM serving is relatively new, with recent research exploring various aspects of the field. For instance, *Liu et al.* in [12] address the assignment of tasks to appropriate closed-source LLMs as a multi-objective optimization problem, aiming to minimize costs while maximizing performance. They employ a heuristic approach to route tasks to specific LLMs; however, their reliance on closed-source LLMs and focus on minimizing API costs presents a notable limitation in the context of SLO-aware 6G networks, as they do not consider task deadlines. In another study, researchers in [13] propose Aladdin, a heuristic-based scheduler designed to route tasks to a set of LLMs to meet task SLOs. However, this method depends on predicting the output length of LLMs before scheduling, which is challenging in fast-changing 6G networks. Generally, heuristic-based schedulers struggle with dense task arrivals, making them less suitable for the high-density environments of 6G networks. As an alternative, DRL has shown promise in rapidly changing environments. In [14], DRL is used to route tasks to a set of LLMs. However, this approach faces difficulties when applied to LLM serving in 6G networks, as it requires an estimate of the task arrival rate in the state space, an estimation that is particularly challenging in the demanding and heterogeneous applications of 6G networks. Additionally, their reward computation is based on model accuracy, which is not an effective metric for evaluating LLMs. In contrast, our DRL approach differs from the aforementioned methods by considering different types of tasks and remaining agnostic to task arrival rates. Moreover, we adopt a more realistic reward computation methodology for each type of task based on the framework proposed in [8].

III. System model and problem formulation

We consider a system composed of a set of m LLMs, denoted as $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$. Each LLM ℓ_j has an associated queue with a capacity $\mathcal{C}_j$ and a token generation speed δ_j (tokens per second). The system receives a set of n tasks, denoted by $\mathcal{K} = \{k_1, k_2, \ldots, k_n\}$. Each task k_i arrives at time t_i, has a specific deadline τ_i, and can generate different output token lengths $\mathcal{P}_{i,j}$ while achieving varying scores $\Phi_{i,j}$ depending on which LLM ℓ_j processes it. The assignment of tasks to LLMs is represented by the binary decision variable $x_{i,j}$, where $x_{i,j} = 1$ if task k_i is assigned to LLM ℓ_j, and $x_{i,j} = 0$ otherwise.

The objective of the system is to maximize the total score obtained from processing the tasks while minimizing penalties associated with tasks that miss their deadlines. The objective function is initially formulated as follows:

$$\max \sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j}\Phi_{i,j} - \sigma \sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j} \max\left(0, t_i + \frac{\mathcal{P}_{i,j}}{\delta_j} - \tau_i\right) \tag{1}$$

Here, the first term maximizes the total score achieved by the system, while the second term imposes a penalty for tasks that exceed their deadlines, with σ being the penalty factor.

To eliminate the use of the max function, we introduce an auxiliary variable $z_{i,j}$. The new objective function is formulated as follows:

$$\max \sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j}\Phi_{i,j} - \sigma \sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j} z_{i,j} \tag{2}$$

Here, the auxiliary variable $z_{i,j}$ is defined through the following constraints:

$$\begin{aligned} &z_{i,j} \geq t_i + \frac{\mathcal{P}_{i,j}}{\delta_j} - \tau_i \quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, m\}, \\ &z_{i,j} \geq 0 \quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, m\} \end{aligned} \tag{3}$$

This optimization problem is subject to several constraints: First, the binary assignment constraint enforces the nature of the decision variable $x_{i,j}$, ensuring that it takes only binary values:

$$x_{i,j} \in \{0, 1\} \quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, m\} \tag{4}$$

This constraint ensures that each task is either assigned to a specific LLM or not, preventing fractional assignments and ensuring clear decision-making in the task assignment process.

Second, the task assignment constraint ensures that each task is assigned to exactly one LLM. This is mathematically expressed as:

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i \in \{1, 2, \ldots, n\} \tag{5}$$

This constraint guarantees that no task is left unassigned or assigned to multiple LLMs, ensuring that each task has a clear and unique processing route.

Thrid, the queue capacity constraint restricts the number of tasks assigned to any LLM at any given time to be within its queue capacity:

$$\sum_{i=1}^{n} x_{i,j} \leq \mathcal{C}_j \quad \forall j \in \{1, 2, \ldots, m\} \tag{6}$$

This constraint ensures that the number of tasks allocated to a particular LLM does not exceed that LLM's queue capacity.

Finally, the deadline constraint requires that the time taken to complete each task respects its deadline. This can be expressed as follows:

$$\begin{cases} t_i + \frac{\mathcal{P}_{i,j}}{\delta_j} \leq \tau_i & \text{if } x_{i,j} = 1, \\ \text{No constraint imposed} & \text{if } x_{i,j} = 0. \end{cases} \tag{7}$$

To eliminate the explicit conditional logic, we introduce a large constant $\mathcal{M}$. The updated constraint becomes:

$$\begin{aligned} t_i + \frac{\mathcal{P}_{i,j}}{\delta_j} \leq \tau_i + (1 - x_{i,j})\mathcal{M}, \\ \forall i \in \{1, 2, \ldots, n\}, \forall j \in \{1, 2, \ldots, m\} \end{aligned} \tag{8}$$

Here, $\mathcal{M}$ is a large constant that effectively deactivates the constraint when a task is not assigned to an LLM ($x_{i,j} = 0$).

From this formalization, we can classify the task scheduling optimization problem for LLM serving as a Mixed-Integer Linear Programming (MILP) problem. However, predicting key variables such as output token length $\mathcal{P}_{i,j}$, task arrival times t_i, deadlines τ_i, and scores $\Phi_{i,j}$, is particularly challenging in fast-varying 6G networks. This unpredictability highlights the need for adaptive scheduling strategies that can respond to dynamic conditions rather than relying solely on static optimization techniques.

IV. DRL-enabled LLMs task scheduling

As mentioned earlier, solving the optimization problem efficiently without prior knowledge of task information and output token length is challenging. For this reason, we propose a DRL approach as it abstracts the complexity and stochastic nature of the environment, allowing for efficient and quick decision-making that adapts to task arrival patterns and changes in the set of LLMs. Furthermore, DRL develops the ability to learn over time and adapt to various unseen situations. In the remainder of this section, we will describe the key components of RL, i.e., state space, action space, and reward function, followed by an explanation of the DRL approach employed, i.e., DQN.

A. RL key parts

As seen in Fig. 1, the DRL agent interacts with the environment to identify the appropriate task scheduling tactics. At each time t, the agent observes a state s_t and executes the appropriate action a_t, allocating task k to LLM ℓ according to the policy π. The state-action function, also known as the Q-function $Q(s_t, a_t)$, which a deep neural network can approximate, can decide this policy. The environment's state is transited to s_{t+1}, and each agent receives a reward of r_t. In our situation, the reward function is established by maximising each task score while satisfying tasks deadlines.

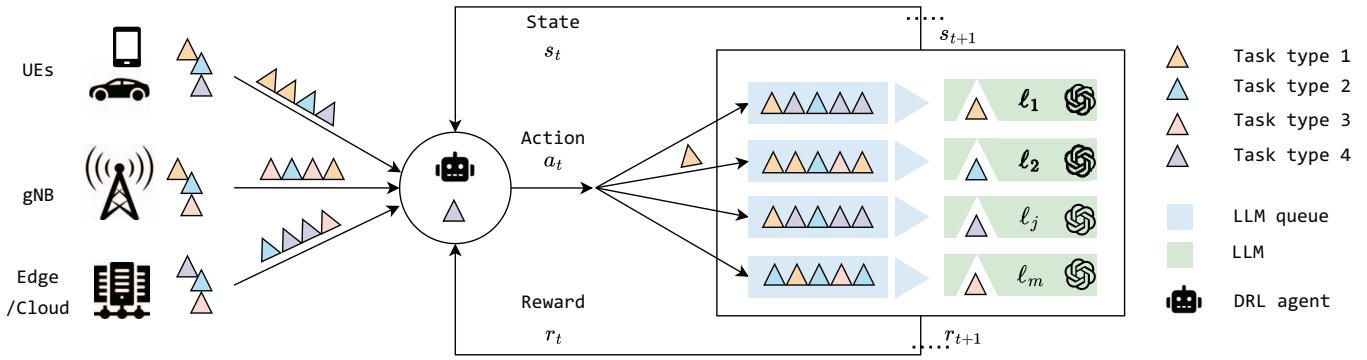


Fig. 1: DRL-enabled task scheduling for LLMs serving in 6G networks design.

1) State: The state s_t at time t includes: (i) The queue sizes q_j for each LLM ℓ_j; (ii) The information about the newly arriving task, i.e., its type κ_i; and (iii) The task deadline τ_i. Formally:

$$s_t = (q, \kappa_i, \tau_i) \mid q = (q_1, q_2, \ldots, q_m) \tag{9}$$

This state design enables the DRL agent to remain agnostic to variations in task arrival rates and the set of LLMs in the system (token generation speeds). This flexibility ensures that task routing remains efficient for realistic scenarios in 6G networks, where LLMs can be updated or replaced, and task arrival rates can change rapidly.

2) Action: At time t, the action a_t involves assigning the incoming task k_i to one of the LLMs ℓ_j. The action is defined as:

$$a_t = j \quad \text{for } j \in \{1, 2, \ldots, m\} \tag{10}$$

Where $a_t = j$ indicates that the task k_i is assigned to LLM ℓ_j, meaning $x_{i,j} = 1$ for the selected LLM ℓ_j, and $x_{i,k} = 0$ for all other LLMs ℓ_v where $v \neq j$.

3) Reward: The reward r_t at time t is computed based on the completion of tasks assigned at previous time steps that finish exactly at time t. It accounts only for tasks that complete at this time, excluding those whose rewards were computed in earlier steps. The reward can be expressed as:

$$r_t = \sum_{i=1}^{n} \sum_{j=1}^{m} \Phi_{i,j} \mathbb{I}\left(t = t_i + \frac{\mathcal{P}_{i,j}}{\delta_j}\right) x_{i,j} \tag{11}$$

Here, $\mathbb{I}\left(t = t_i + \frac{\mathcal{P}_{i,j}}{\delta_j}\right)$ is an indicator function that equals 1 if the task k_i assigned at time t_i to LLM ℓ_j finishes at time t and 0 otherwise. The variable $x_{i,j}$ indicates that task k_i was assigned to LLM ℓ_j at time t_i. Thus, r_t ensures that only tasks completing at the current time contribute to the reward. Once a task's reward has been computed at t, it is excluded from future time step rewards. This reward design enables the DRL agent to adaptively learn and optimize its routing strategy in response to the dynamic conditions of 6G networks, where task completion times and scores may vary significantly based on the LLM used.

B. DRL approach

DQN is ideal in the context of this problem due to its ability to effectively manage large state and action spaces while ensuring stability and convergence through experience replay and target networks [9]. In DQN, the objective is to find the policy π^* that maximizes cumulative rewards, represented as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \tag{12}$$

where γ is the discount factor, starting from any initial state s_0. The Q-function $Q(s, a; \theta)$ estimates the expected cumulative reward for taking action a in state s and subsequently following the optimal policy. DQN addresses this by approximating the Q-function using a neural network with parameters θ, which is trained by minimizing the loss function:

$$\begin{aligned} L(\theta) &= \mathbb{E}\left[\left(y_t - Q(s_t, a_t; \theta)\right)^2\right], \\ y_t &= r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-). \end{aligned} \tag{13}$$

Here, the target y_t includes the immediate reward r_t and the discounted maximum future reward predicted by a target network with parameters θ^-. This target network, which is periodically updated, helps stabilize training and enables the Q-function to converge more effectively.

At each time step t, the action a_t is chosen based on an ϵ-greedy strategy. With a probability ϵ, a random action is selected to encourage exploration, while with a probability $1 - \epsilon$, the action maximizing the Q-value is chosen to exploit learned knowledge. ϵ will decrease over time during the learning pushing the agent to explore the environment at the beginning of the training and driving it to exploitation over time. After executing the action, the reward r_t is obtained, and the transition (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer. This buffer allows the agent to sample previous transitions randomly during training, helping to break correlations between samples and stabilize learning. This overall approach promotes convergence and enables DQN to learn optimal policies in complex environments.

V. Performance evaluation

The section is structured into three subsections: *Experiment setup,* which details the experimental setup; *Experiment results,* which presents and analyzes the performance of the DRL approach; and *Experiment conclusion,* which offers additional insights related to the experiment.

A. Experiment setup

Our experimental setup includes three machines hosting the DRL-DQN agent and the LLMs. The first machine, equipped with a 12th Gen Intel(R) Core(TM) i7-12700, is dedicated to the training and inference of the DRL agent. The two remaining machines, equipped with an Intel(R) Xeon(R) Gold 6240R CPU and an NVIDIA A100 GPU (40GB vRAM), are dedicated to the LLMs environment. Each GPU machine is used to deploy two LLMs, resulting in a total of four LLMs for evaluation. These four LLMs use the Instruct versions of: *phi3-3.8b*, *gemma2-9b*, *qwen2-7b*, and *llama3-8b*. The DRL-DQN agent interacts with the LLMs using API calls. This interaction is facilitated by the Ollama[1] framework, which serves the LLMs, while the Langchain[2] library is used for the interaction. Meanwhile, the DRL-DQN agent is trained using the Stable Baselines library [15]. Realistic tasks are derived from [8] and their arrival pattern is simulated using the Poisson distribution with a parameter λ. This latter indicates the mean number of tasks arriving per second. From [8], five types of tasks were considered: 'reasoning', 'coding', 'math', 'data_analysis', and 'language'. Table I present the parameters regarding DRL-DQN, LLMs and tasks information.

TABLE I: Setup parameters.

Parameter	Value
Replay buffer size	$50,000$
Minibatch size	32
Hidden layers	2
No. of neurons	[128, 128]
Activation function	[ReLU, ReLU]
Optimizer	Adam
Learning rate	0.001
Discount factor γ	0.99
Epsilon decay	0.1 (initial) to 0.05 (final)
Step per Episode	30
Number of total steps	$10,000$
Time step	1s
LLM temperature	0
LLM top_p	0.95
LLM repeat_penalty	1.15
LLM queue size	10
Task deadlines	10s
Training Task arrival rate (λ)	1
Testing Task arrival rate (λ)	$0.5, 1, 2, 4, 8$

[1]https://ollama.com
[2]https://www.langchain.com

B. Experiment results

Fig. 2 presents the convergence evaluation of the DRL-DQN agent throughout the training process. The x-axis denotes the training steps corresponding to various episodes, with each episode consisting of 50 steps, resulting in a total of 334 episodes. The y-axis represents the score, defined as the cumulative rewards obtained during each episode. Analysis of the reward curve indicates a gradual increase over time, converging after approximately 5,000 training steps. These results show that the DRL-DQN agent successfully identified a policy that outperforms earlier iterations in training, ultimately achieving a high reward score.

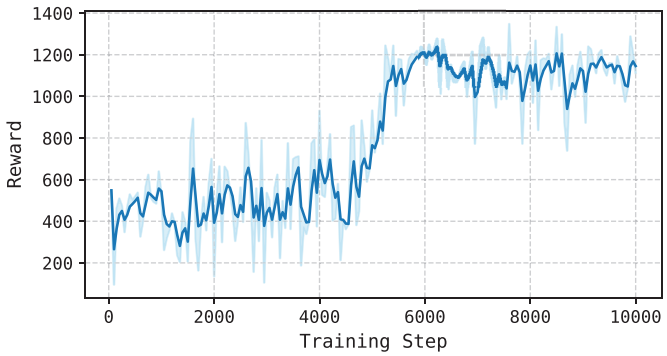
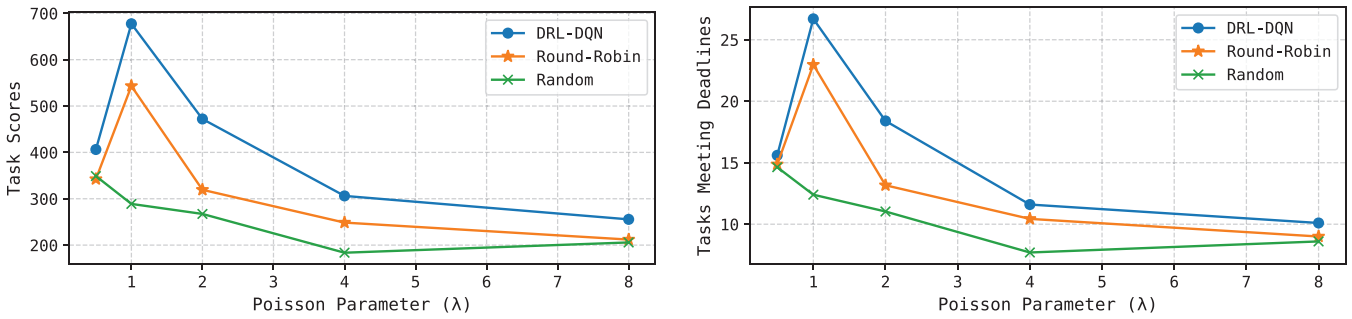


Fig. 2: Convergence evaluation of DRL-DQN during training.

The resulting policy from the aforementioned training is evaluated against two baselines: *(i) Round-Robin (RR),* a well-known and powerful scheduling baseline that optimizes resource utilization by allocating tasks in a cyclic manner. This approach ensures that all LLMs receive equitable task allocation; and *(ii) Random,* which selects LLMs randomly without any strategic consideration. We conduct experiments with the three approaches over 30 episodes while varying the task arrival rate over time using five different λ values: 0.5, 1, 2, 4, and 8. We record the average reward (sum of the task scores) across all 30 episodes. Additionally, we track the average number of tasks for which deadlines were satisfied across all 30 episodes. The results are shown in Fig. 3.

In subfigure 3a, we present the average rewards achieved by each approach over the 30 episodes. The DRL-DQN agent consistently outperforms both the RR and Random methods, demonstrating its effectiveness in learning a superior scheduling policy. While the RR method shows stable performance, it does not reach the reward levels of the DRL-DQN agent. This is because the RR method ensures equitable routing among LLMs, but the varying generation times of the LLMs limit overall performance. The DRL-DQN agent surpasses RR by learning the token generation speeds of the LLMs and identifying faster models based on its experience, even when the set of LLMs is updated. In contrast, the Random method exhibits variability in performance due to its lack of strategic task allocation. On the other hand, subfigure 3b illustrates the average number of tasks for which deadlines were satisfied across the 30 episodes. Here, the DRL-DQN



(a) Task scores for different task arrival rates (λ).

(b) Satisfied deadlines for different task arrival rates (λ).

Fig. 3: Performance comparison among DRL-DQN, Round-Robin, and Random scheduling methods.

agent significantly outperforms the baselines in deadline satisfaction. For instance, for $\lambda = 1$, it achieves 89%, compared to 76% (RR) and 41% (Random). Similarly, for $\lambda = 2$, it achieves 61%, versus 43% (RR) and 36% (Random). The RR method performs reasonably well; however, it does not reach the performance level of the DRL-DQN agent. The Random method, in contrast, results in a notably lower rate of deadline satisfaction, highlighting its inefficiency in task scheduling.

C. Experiment conclusion

Task scheduling for LLMs is a relatively new area that has recently gained attention in research. This is not a simple task that traditional heuristics can easily resolve in the rapidly evolving landscape of 6G networks. Therefore, as experimental results have shown, DRL-based solutions are effective in optimizing task scheduling for LLMs. It can learn the dynamic nature of the environment, such as task arrival patterns and LLMs token generation speed, and adapt to these factors to develop an effective scheduling policy. However, several considerations need to be addressed in future works. These include optimizing energy consumption, as LLMs require significant energy for inference and serving due to their computational demands. Additionally, strategies for balancing the use of Small Language Models (SLMs) at the far edge with larger LLMs at the edge or in the cloud should be considered. Although SLMs may have lower performance scores, they consume less energy and have faster execution times.

VI. Conclusion

As 6G networks evolve to become more intelligent, the integration of AI, particularly LLMs, will be crucial in achieving these capabilities. However, their computational demands necessitate efficient task scheduling strategies to optimize their performance across multiple 6G use cases. To this end, this paper introduced a DRL-based scheduler, aiming to maximize task scores while adhering to latency constraints. Our evaluations, conducted under real-world conditions, demonstrate that our proposed approach outperforms traditional scheduling methods such as RR and Random. These results highlight the potential of DRL in enhancing the efficiency and effectiveness of LLMs within 6G networks. Future research directions will focus on tuning the DRL agent to simultaneously optimize execution time, task scores, and energy consumption. Additionally, the DRL environment will be extended to incorporate SLMs at the far edge, and LLMs at the edge and cloud.

Acknowledgment

This work is supported by the European Union's Horizon Program under the SUNRISE-6G (Grant No. 101139257) and 6G-DALI (Grant No. 101192750) projects.

References

[1] Muhammad K Shehzad et al. "Artificial intelligence for 6G networks: Technology advancement and standardization". In: *IEEE Vehicular Technology Magazine* 17.3 (2022), pp. 16–25.

[2] Estefania Coronado et al. "Zero touch management: A survey of network automation solutions for 5G and 6G networks". In: *IEEE Communications Surveys & Tutorials* 24.4 (2022), pp. 2535–2578.

[3] Hao Zhou et al. "Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities". In: *arXiv preprint arXiv:2405.10825* (2024).

[4] Kilian Carolan, Laura Fennelly, and Alan F Smeaton. "A Review of Multi-Modal Large Language and Vision Models". In: *arXiv preprint arXiv:2404.01322* (2024).

[5] Azzedine Idir Ait Said et al. "5G INSTRUCT Forge: An Advanced Data Engineering Pipeline for Making LLMs Learn 5G". In: *IEEE Transactions on Cognitive Communications and Networking* (2024).

[6] Abdelkader Mekrache et al. "On Combining XAI and LLMs for Trustworthy Zero-Touch Network and Service Management in 6G". In: *IEEE Communications Magazine* (2024).

[7] Abdelkader Mekrache, Adlen Ksentini, and Christos Verikoukis. "Intent-based management of next-generation networks: an LLM-centric approach". In: *IEEE Network* (2024).

[8] Colin White et al. "LiveBench: A Challenging, Contamination-Free LLM Benchmark". In: *arXiv preprint arXiv:2406.19314* (2024).

[9] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[10] Kai Arulkumaran et al. "Deep reinforcement learning: A brief survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[11] Conghao Zhou et al. "Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN". In: *IEEE Transactions on Wireless Communications* 20.2 (2020), pp. 911–925.

[12] Yueyue Liu et al. "OptLLM: Optimal Assignment of Queries to Large Language Models". In: *arXiv preprint arXiv:2405.15130* (2024).

[13] Chengyi Nie, Rodrigo Fonseca, and Zhenhua Liu. "Aladdin: Joint Placement and Scaling for SLO-Aware LLM Serving". In: *arXiv preprint arXiv:2405.06856* (2024).

[14] Siddharth Jha et al. "Learned Best-Effort LLM Serving". In: *arXiv preprint arXiv:2401.07886* (2024).

[15] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.