Deploying On-Device AIGC Inference Services in 6G via Optimal MEC-Device Offloading

Changshi Zhou, *Graduate Student Member, IEEE*, Weiqi Liu, *Member, IEEE*, Tao Han, *Senior Member, IEEE*, and Nirwan Ansari

***Abstract*—From AI-assisted art creation to large language model (LLM)-powered ChatGPT, AI-generated contents and services are becoming a transforming force. It calls for the telecom industry to embrace the prospects of AIGC services and face the unique challenges posed by incorporating generative model services into the AI-native 6G wireless network paradigm. We propose enabling AIGC inference services on mobile devices by optimizing MEC-device computing offloading, through which AIGC task latency is minimized by reinforcement learning based policy agent in a computing resource constrained and bandwidth limited wireless environment. Simulation results are presented to demonstrate the performance advantage.**

***Index Terms*—LLM, AIGC, ChatGPT, MEC, on-device computing, 6G, constrained reinforcement learning.**

I. Introduction

THE FAR-REACHING impact of AI-generated content (AIGC) [2] is increasingly apparent. From stable diffusion based art creation to large language model (LLM)-enabled ChatGPT, AIGC techniques and services are exploding into different business fields and diverse application domains, poised to be the new engine of the digital economy. The emergence of AI-native 6G wireless networks is fostering a convergence of computing centred 5G/6G technology and AIGC. The wireless community is assessing the capabilities that AIGC can bring to the wireless domain [3], [6]. The widespread use of interconnected devices and sensors, powered by wireless connectivity, generates an immense volume of data. This data can be harnessed to train AIGC models tailored to the wireless network domain, significantly enhancing network intelligence to better serve mobile users. Additionally, wireless network operators can provision personalized and dynamic AIGC content by deploying self-trained AIGC models or customized lightweight models derived from large models like ChatGPT.

Despite the great potential that AIGC holds, leveraging the power of AIGC services in mobile networks and deploying them on mobile devices poses considerable challenges. The quality of AIGC models relies on large-scale pretrained models with a large number of parameters, such as GPT-3 with up to 175 billion parameters, and requires fine-tuning to fit various tasks [4]. The constraints of mobile devices including computing resources, power consumption, and privacy concerns, must be carefully considered in the context of training, fine-tuning, personalizing, and deploying of AIGC models. These challenges are further complicated by the unique demands of mobile users for AIGC services. Furthermore, new AIGC use cases, ranging from best effort request/response transactions to realtime ChatBot interactions and industrial robotic manipulations, require network design adaptations to meet AIGC specific quality of experience (QoE) metrics, which have yet to be specified. This letter focuses specifically on the deployment of AIGC, where pretrained models are prepared to handle inference requests originating from mobile users.

On-device inference has been proposed to provision AIGC services by running AIGC models on standalone mobile device [5]. The advantage of this approach lies in its ability to protect privacy and provide realtime on-device full access, but it usually requires lightweight models trained by techniques such as model parameter pruning or low bit quantization. This comes at the cost of degraded model accuracy and is only feasible on devices equipped with sufficient memory storage, battery capacity, and CPU/GPU computing power. Another approach is to leverage Mobile Edge Computing (MEC) [7]. MEC allows resource constrained mobile devices to offload computing tasks to resource rich edge servers, thereby fully utilizing the network computing resources. To apply MEC for AIGC deployment, resource demanding AIGC models can be hosted on edge servers, inference requests can be served either by lightweight models preinstalled on device or by large models hosted on edge servers with offloading collaboration. While MEC has been extensively studied in application domains such as IOT [1], [12], the incorporation to support AIGC inferences–a newly emerging issue–has yet to be investigated. Here, we propose a MEC offloading scheme to provision AIGC in mobile networks. We formulate an optimization problem to optimize offloading decisions by considering factors such as traffic generation, wireless channel conditions, and computing constraints. Our contributions include:

- Proposing a MEC-device collaborative scheme to provision AIGC inference, in which decisions to execute the AIGC tasks on edge servers or locally on device are optimized.

Received 6 September 2024; revised 14 October 2024; accepted 30 October 2024. Date of publication 4 November 2024; date of current version 30 January 2025. The associate editor coordinating the review of this article and approving it for publication was H. Chergui. *(Corresponding author: Changshi Zhou.)*

Changshi Zhou, Tao Han, and Nirwan Ansari are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: czhou@njit.edu; tao.han@njit.edu; ansari@njit.edu).

Weiqi Liu is with the Department of Computer Science and Computer Information Systems, Auburn University at Montgomery, Montgomery, AL 36117 USA (e-mail: wliu4@aum.edu).

Digital Object Identifier 10.1109/LNET.2024.3490954

- Formulating the optimal decision by minimizing the average task execution latency, considering both the randomness of AIGC service requests and the complexity of the wireless channels. The problem is solved using a reinforcement learning based algorithm with a service latency constraint.
- Demonstrating the performance of the proposed scheme through simulations.

The remainder of this letter is organized as follows: In Section II, related works are reviewed. Section III presents our proposed AIGC offloading scheme and formulates the offloading optimization problem. Section IV describes the simulation results and the performance analysis. The conclusion is made in the final section.

II. Related Works

This section outlines related works into three categories.

AIGC to Telecom: LLM and AIGC are drawing increasing interests in the telecom industry and academia. Recent work by Piovesan et al. [8] conducted an evaluation of small language model Phi-2, comparing its accuracy with LLM in its understanding of the telecom domain and presenting problem-solving scenarios within the telecom sector. Zhou et al. [6] presented a comprehensive review of the potentials of LLM to the telecom fields, describing benefits and challenges of network edge deployment and on-device deployment.

AIGC to Device: Research in this category focuses on enabling on-device AIGC by customizing smaller task-specific models or by optimizing GPU usage for faster inference. Iyer et al. [9] introduced an on-device inference solution to sustain performance by dynamically allocating a combination of CPU and GPU backends per model. However, these allocations involve a trade-off between the faster execution of GPUs and reduced physical memory, forming a Pareto front in a multi-objective space, which is computationally non-trivial to determine. As a scalable easy-to-share solution for task-specific models, particularly in low-resource scenarios, Pfeiffer et al. [10] introduced a framework for adapting transformers. This framework allows the dynamic "stitching-in" of pre-trained adapters for different tasks and languages. While these adapters can be used for inference, they can only support the tasks they were trained on.

MEC-device Offloading Computing: Offloading is a key technique in collaborative MEC-device computing. To apply MEC offloading in AIGC, Du et al. [11] proposed a collaborative framework to execute diffusion-based AIGC denoising steps across edge servers and mobile devices, demonstrating its feasibility experimentally with several mobile devices. However, the approach is specific to diffusion models and faces challenges in optimally splitting the denoising process for collaborative execution on devices with varying computing capacities. Wang et al. [13] presented an optimization scheme for offloading decisions, computation time, and diffusion steps during the reverse diffusion stage. However, this solution is also dedicated to diffusion models and its modeling is limited to a single edge server. In addition, both designs lack consideration of wireless channel conditions.

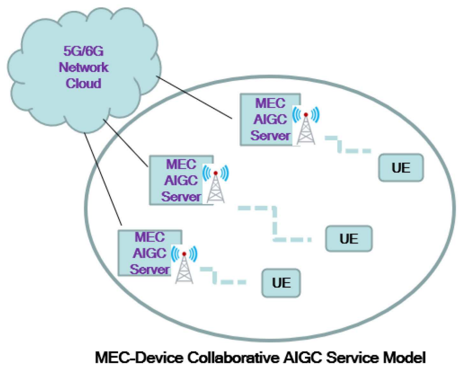

Fig. 1. MEC-device AIGC service system model.

The computing limitation of mobile devices is one main hurdle to provisioning AIGC for telecom. Works on this issue remain scarce, which is the motivation for this letter.

III. System Model and Problem Formulation

A. System Model

We consider a system model in the 5G/6G wireless network setting as shown in Fig. 1, consisting of AIGC large models deployed on a set of MEC edge servers collocated with base stations (BSs) and lightweight AIGC models preinstalled on user devices (UEs). For simplicity, the system is modeled for a single AIGC service type, such as stable diffusion or ChatGPT. A service task originated by a UE can be executed either on-device or by the MEC server, controlled by an AIGC service scheduler. Denote the set of servers by $\mathcal{M} = \{1, 2, 3, \dots, M\}$ and the set of UEs by $\mathcal{N} = \{1, 2, 3, \dots, N\}$. Assume each device and each server can execute c_m and c_n FLOPS, respectively; the large model requires χ_{ser} FLOPS, while the UE needs χ_{dev} FLOPS, potentially with a degraded model accuracy. To analyze AIGC tasks, we discretize the task arrival process into T time slots $\mathcal{T} = \{1, 2, 3, \dots, T\}$. The AIGC service scheduler makes scheduling decisions on a per-time-slot basis. At the beginning of each slot, each user generates a new task, forming a task set $\mathcal{U} = \{1, 2, 3, \dots, U\}$. Each task $u \in \mathcal{U}$ is identified by a tuple $(\mu_u^{dev}, \mu_u^{pkt}, t_u)$, corresponding to the task device, task data size, and time slot of the task, respectively. The task data sizes are of random lengths.

B. Communication Channel Model.

The wireless channel path loss between UE n and MEC server m can be expressed as:

$$\psi_{n,m} = pl_0 + \gamma \log\Big(dis\Big(l_n^{dev}, l_m^{ser}\Big)\Big)$$

where pl_0 and γ are the path loss and path exponent, respectively; $dis(l_n^{dev}, l_m^{ser})$ denotes the distance between UE n and server m. Then, the signal to noise and interference ratio (SINR) from UE n to server m can be derived as:

$$e_{n,m} = \frac{P_n 10^{\frac{-\psi_{n,m}}{10}}}{\sigma^2 + \sum_{j \in \mathcal{M}, j \neq m} \sum_{i \in \mathcal{N}, i \neq n} P_i 10^{\frac{-\psi_{i,j}}{10}}}$$

where P_n denotes the transmission power of UE n, and σ^2 denotes the Gaussian white noise power. The denominator is the interference plus white noise. Then, the data rate from UE n to server m is

$$r_{n,m} = \eta_{n,m} B \log_2\left(1 + e_{n,m}\right)$$

where B is the total available spectrum bandwidth, and $\eta_{n,m}$ denotes the percentage of spectrum allocated to UE n.

C. Computing Model

We use $y_n^u \in \{0,1\}$ to indicate whether AIGC task u is executed locally, i.e., if the task is computed locally, $y_n^u = 1$; otherwise, it is offloaded to the edge server. In the case that the task is offloaded, $z_{n,m}^u \in \{0,1\}$ indicates if it is executed on server m ($z_{n,m}^u = 1$ if true, and zero otherwise). The computing time to process a task u locally is given as:

$$\tau_u^{local} = \sum_{n=1}^{N} y_n^u \cdot \frac{\chi_{dev}}{c_n}$$

The computing time to process a task on the server can be calculated as:

$$\tau_u^{server} = \sum_{m=1}^{M} \sum_{n=1}^{N} z_{n,m}^u \cdot \frac{\chi_{ser}}{c_m^u}$$

and the communication link latency for an offloaded task can be calculated as:

$$\tau_u^{comm} = \sum_{m=1}^{M} \sum_{n=1}^{N} z_{n,m}^u \cdot \frac{\mu_u^{pkt}}{r_{n,m}}$$

D. Problem Formulation

The optimal offloading decision, whether to execute each service task on-device locally or offload it to a server, is formulated by minimizing the average AIGC task execution latency:

$$\begin{aligned}
&\min_{y_n^u, z_{n,m}^u, \eta_{n,m}, c_m^u} \frac{1}{U} \sum_{u \in \mathcal{U}} \left[\tau_u^{local} + \tau_u^{server} + \tau_u^{comm}\right] \\
&s.t.: \; C1: \sum_{n=1}^{N} y_n^u + \sum_{n=1}^{N} \sum_{m=1}^{M} z_{n,m}^u = 1 \\
&\quad C2: \sum_{n=1}^{N} \eta_{n,m} B \leq B \\
&\quad C3: \sum_{u=1}^{U} \sum_{n=1}^{N} z_{n,m}^u c_m^u \leq c_m \\
&\quad C4: \left[\tau_u^{local} + \tau_u^{server} + \tau_u^{comm} \leq T^u\right]
\end{aligned} \tag{1}$$

where U is the number of task requests, and the optimization variables represent a collective set of decisions for all tasks (i.e., $\forall u \in \mathcal{U}$, $\forall n \in \mathcal{N}$, $\forall m \in \mathcal{M}$). Among the constraints: $C1$ imposes the exclusiveness of task execution; $C2$ and $C3$ ensure the channel bandwidth B and computing capacity of the edge servers, respectively; $C4$ defines a hard time limit T^u within which the execution latency of each task is bounded.

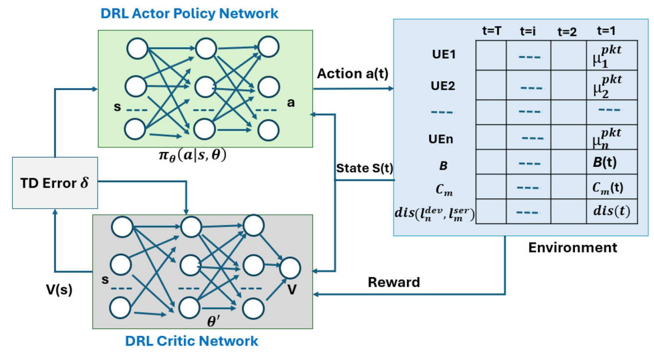

Fig. 2. Actor-Critic based DRL design.

Algorithm 1 Actor-Critic Offloading Algorithm

```
INPUT: policy $\pi_\theta(a|s,\theta)$, state-value $\vee(s,\theta')$
OUTPUT: RL policy $\pi_\theta$
INITIALIZE: RL network weight $\theta$, $\theta'$, learning rate $\lambda^\theta$,$\lambda^{\theta'}$
 For each episode e=1 to E do
    Initialize s (first state of e)
    For training step t=1 to $T_{train}$ do
        $a \leftarrow \pi_\theta(a|s,\theta)$
        take action a, observe $s'$,R
        $\delta \leftarrow R + \vee(s',\theta') - \vee(s,\theta')$
        $\theta \leftarrow \theta + \lambda^\theta \delta \nabla \ln(\pi_\theta(a|s,\theta))$
        $\theta' \leftarrow \theta' + \lambda^{\theta'} \delta \nabla \vee (s,\theta')$
        $s \leftarrow s'$
 Return $\pi_\theta$
```

E. DRL Algorithm

This problem is a mixed-integer nonlinear programming (MINLP) due to the interference, resource allocation, and offloading decisions, making it NP-hard. Given the complexity of characterizing AIGC tasks, deriving a tractable solution is generally challenging. Therefore, we resort to deep reinforcement learning (DRL) to solve this problem [14].

In this actor-critic based approach as illustrated in Fig. 2, the DRL agent runs on the AIGC resource scheduler. It takes task requests of each time slot t and the available network resource (channel bandwidth, computing capacity, path info. between UE and BS) as input state $s(t)$ to the DRL policy network $a \sim \pi_\theta(a|\theta, s)$. The DRL agent outputs the offloading actions $a(t)$, where the integers in $a(t)$ represent the task assignment. With action $a(t)$ taken, the DRL agent receives a reward, and the input state to the DRL agent transitions to the next state. Successive tasks in different time slots are treated as independent. The reward function is designed as:

$$R(t) = -\frac{1}{U} \sum_{u \in \mathcal{U}, t_u = t} \left(\tau_u^{local} + \tau_u^{server} + \tau_u^{comm}\right)$$

The actor-critic based algorithm to train the DRL policy network is summarized in Algorithm 1. Note that θ parameterizes the policy network (the actor), and θ' parameterizes the state-value function network (the critic). Each scheduling time slot corresponds to one training episode, during which the actor updates the policy to improve the action reward,

TABLE I
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
χ_{ser}	200MFLOPS	χ_{dev}	50MFLOPS
N	[10, 15,...,40]	M	5
c_n	300MFLOPS	c_m	15GFLOPS
Tx PWR	20 dbm	δ_n^2	-100 dbm
Threshold T^u	[10 20] ms	Packet Size μ_u^{pkt}	[100,200] KB
Channel BW	20 MHz		

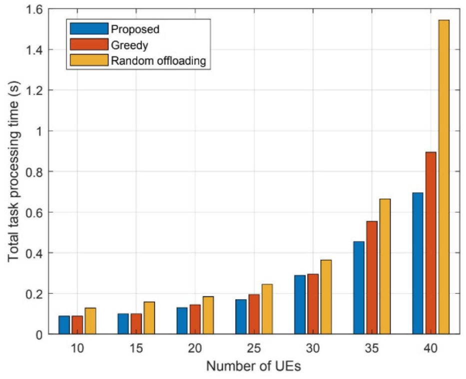

Fig. 3. AIGC latency: Proposed vs. Greedy vs. Random Offloading.

and the critic also updates to better estimate the state-value function $V(s)$; these updates are based on the evaluation of the temporal difference (TD) error δ, which measures the difference between the predicted and actual values of the state-value function. Through iterations, the DRL training maximizes the policy rewards.

IV. Performance Evaluations

To evaluate the proposed MEC-device AIGC task offloading, simulations were conducted to demonstrate its performances. The simulation setup as illustrated in Fig. 1 consists of a number of mobile UEs randomly located within a RF coverage area with a radial distance of 1km, served by multiple MECs for AIGC service under the offloading policy. Table I defines the simulation parameters for our system model.

For the performance evaluation, we first compared the proposed offloading scheme with the greedy algorithm and the random offloading scheme. In the random offloading scheme, tasks originated by each UE are randomly assigned to an edge server or processed locally. In contrast, the greedy algorithm makes decisions based on known information about local processing capability, channel conditions, and edge server computing capabilities. The result is presented in Fig. 3. The plot shows that, for a given set of tasks generated by a fixed number of UEs, as the number of UEs in the set increases from 10 to 40, the processing time increases for all the offloading schemes, but the proposed scheme consistently demonstrates a clear advantage in taking the least time to execute the tasks.

We further examined the effect of the latency bound applied in the DRL policy training. Fig. 4 presents the test result when the latency constraint is set to be 15 ms and 20 ms, respectively. The plot shows that as the number of UEs increases from 10 to 40, few UEs violate the latency bound due

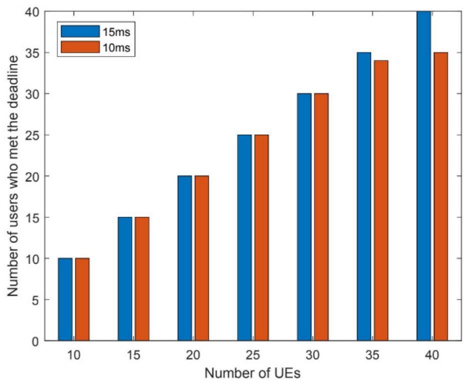

Fig. 4. Number of UEs meeting the latency bound.

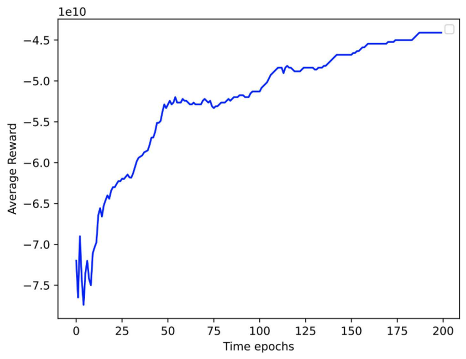

Fig. 5. DRL training convergence.

to increased contention for computing resources. In addition, to evaluate the training convergence, we plotted the average reward of each epoch as the DRL training advances, as shown in Fig. 5.

The simulation results demonstrate the advantage of the proposed AIGC offloading scheme as well as its effectiveness in the latency bound. However, it is worth noting that the AIGC task traffic model is simplified for the convenience of analysis. While it serves the purpose of exploring a solution to support AIGC service in wireless networks, characterizing the AIGC service, defining QoE metrics and seeking an optimal solution remains complex and demands further work.

V. Conclusion

We have proposed a MEC-device collaborative scheme aimed at optimizing the utilization of computing resources across edge servers and mobile devices to support on-device AIGC inference services in emerging 6G wireless networks. A MEC-device offloading optimization problem is formulated to minimize service latency which includes channel communication latency and computing latency, constrained by wireless channel bandwidth, server computing capacity, and desired service completion time. The problem is solved using a deep reinforcement learning-based method. Simulations were conducted, and the performance results demonstrate the advantage of the proposed scheme over other approaches such as greedy offloading or random offloading. While this study, with its simplified models, highlights the benefits of leveraging offloading to enable AIGC in wireless networks, further research is needed to address the uniqueness and complexity of serving AIGC services.

REFERENCES

[1] L. Zhang and N. Ansari, "Latency-aware IoT service provisioning in UAV-aided mobile-edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10573–10580, Oct. 2020.

[2] Y. Cao et al., "A comprehensive survey of AI-generated content (AIGC): A history of generative AI from GAN to ChatGPT," 2023, *arXiv:2303.04226*.

[3] L. Bariah, Q. Zhao, H. Zou, Y. Tian, F. Bader, and M. Debbah, "Large generative AI models for telecom: The next big thing?" *IEEE Commun. Mag.*, early access, Jan. 8, 2024, doi: 10.1109/MCOM.001.2300364.

[4] T. B. Brown et al., "Language models are few-shot learners," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[5] S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup, and M. Shah "A survey of on-device machine learning: An algorithms and learning theory perspective," *ACM Trans. Internet Things*, vol. 2, no. 15, pp. 1–49, 2021.

[6] H. Zhou et al., "Large language model (LLM) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," 2024, *arXiv:2405.10825*.

[7] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1160–1192, 2nd Quart., 2021.

[8] N. Piovesan, A. De Domenico, and F. Ayed, "Telecom language models: Must they be large?" 2024, *arXiv:2403.04666*.

[9] V. Iyer, S. Lee, S. Lee, J. J. Kim, H. Kim, and Y. Shin, "Automated backend allocation for multi-model, on-device AI inference," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 7, no. 3, pp. 1–33, 2023.

[10] J. Pfeiffer et al., "Adapterhub: A framework for adapting transformers," in *Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP)*, Oct. 2020, pp. 46–54.

[11] H. Du et al., "Exploring collaborative distributed diffusion-based AI-generated content in wireless networks," *IEEE Netw.*, vol. 38, no. 3, pp. 178–186, May 2024.

[12] X. Hou, Y. Guan, T. Han, and N. Zhang, "DistrEdge: Speeding up convolutional neural network inference on distributed edge devices," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, Lyon, France, 2022, pp. 1097–1107.

[13] X. Wang, C. Liu, and J. Zhao, "Offloading and quality control for AI generated content services in 6G mobile edge computing networks," in *Proc. IEEE 99th Veh. Technol. Conf.*, 2024, pp. 1–7.

[14] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.