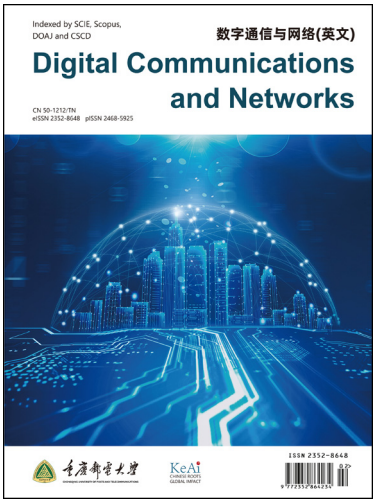
Journal Pre-proof



EdgeAIGC: Model caching and resource allocation for Edge Artificial Intelligence Generated Content

Wu Wen, Yibin Huang, Xinxin Zhao, Peiying Zhang, Kai Liu et al.

PII: S2352-8648(25)00114-2
DOI: https://doi.org/10.1016/j.dcan.2025.07.003
Reference: DCAN 904

To appear in: *Digital Communications and Networks*

Received date: 26 February 2025
Revised date: 23 June 2025
Accepted date: 1 July 2025

Please cite this article as: W. Wen, Y. Huang, X. Zhao et al., EdgeAIGC: Model caching and resource allocation for Edge Artificial Intelligence Generated Content, *Digital Communications and Networks*, doi: https://doi.org/10.1016/j.dcan.2025.07.003.

Digital Communications and Networks(DCN)

HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan

EdgeAIGC: Model caching and resource allocation for Edge Artificial Intelligence Generated Content

Wu Wen[a], Yibin Huang[a], Xinxin Zhao[b], Peiying Zhang[cd*], Kai Liu[ef], Guowei Shi[g]

[a] School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China
[b] Shandong Inspur Academy of Science and Technology , Inspur Group Co., Ltd., 1036 Gaoxin Inspur Road, Jinan 250101, China.
[c] Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China
[d] Shandong Key Laboratory of Intelligent Oil & Gas Industrial Software, China University of Petroleum (East China), Qingdao 266580, China
[e] Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China
[f] State Key Laboratory of Space Network and Communications, Tsinghua University, Beijing 100084, China
[g] China Academy of Information and Communications Technology, Beijing 100191, China

Abstract

With the rapid development of Generative Artificial Intelligence technology, the traditional cloud-based centralized model training and inference face significant limitations due to high transmission latency and costs, which restrict user-side in-situ Artificial Intelligence Generated Content (AIGC) service requests. To this end, we propose the Edge Artificial Intelligence Generated Content (EdgeAIGC) framework, which can effectively solve the problems brought by cloud computing by implementing in-situ processing of services close to the data source through edge computing. However, AIGC models usually have a large parameter scale and complex computing requirements, which poses a huge challenge to the storage and computing resources of edge devices. This paper focuses on the edge intelligence model caching and resource allocation problems in the EdgeAIGC framework, aiming to improve the cache hit rate and resource utilization of edge devices for models by optimizing the model caching strategy and resource allocation scheme, and realize in-situ AIGC service processing. With the optimization objectives of minimizing service request response time and execution cost in resource-constrained environments, we employ the Twin Delayed Deep Deterministic Policy Gradient algorithm for optimization. Experimental results show that, compared with other methods, our model caching and resource allocation strategies can effectively improve the cache hit rate by at least 41.06% and reduce the response cost.

KEYWORDS:
Generative AI, Edge Model Caching, Resource Allocation, Edge Intelligence

1. Introduction

Artificial Intelligence Generated Content (AIGC) is a significant area in Artificial Intelligence (AI), aimed at autonomously generating creative content—such as speech, text, images, and videos—through technologies like machine learning [1, 2]. Unlike traditional AI applications that focus on pattern recognition and data analysis, AIGC generates new data based on input conditions and learned patterns and structures, simulating human creativity. Core technologies of AIGC include Generative Adversarial Networks (GANs), the Transformer architecture, and diffusion models. For instance, GANs generate realistic images through adversarial training between generators and discriminators. The Transformer architecture achieves coherent long-text generation through the self-attention mechanism. Diffusion models generate high-resolution content via gradual denoising. With the rapid advancement of AI technology, AIGC models are rapidly scaling and being deployed across various application domains. For example, text-to-speech (TTS) [3, 4], ChatTTS can generate high-quality, emotionally rich speech output based on text input. It also supports multi-language and personalized voice customization, and is widely used in intelligent customer service and audiobook production. Text-to-text [5], such as ChatGPT [6], can generate high-quality answers to user text input. Its core lies in pre-training based on massive data and human feedback-based reinforcement learning, which can complete complex tasks such as code optimization and text polishing. Text-to-image [7, 8], DALL-E-3 [9] can accurately understand complex text prompts and generate images that are highly consistent with the description. Its technical breakthrough lies in multimodal alignment, such as the precise matching of text and image semantics through the CLIP model. Text-to-video [10, 11], Sora [12] is an artificial intelligence video model that can generate realistic video content based on natural language descriptions. It relies on the spatiotemporal diffusion model and physical engine simulation technology to generate coherent video clips up to 60 seconds.

AIGC has made significant breakthroughs in these fields and has accelerated its penetration into vertical scenarios such as healthcare, education, and industry [13, 14]. For example, in the healthcare field, AIGC can generate synthetic medical images to assist in diagnosis [15, 16]. In the education field, it can automatically generate personalized exercises to assist in enriching course teaching [17–19]. Since AIGC models usually have a large parameter scale and complex computing requirements, they require powerful computing resources when running. These high

*Corresponding author.
Email address: wenwu@gzhu.edu.cn(W. Wen), gd.hyb@e.gzhu.edu.cn(Y. Huang), zhaoxxjs@inspur.com(X. Zhao), zhangpeiying@upc.edu.cn(P. Zhang), liukaiv@tsinghua.edu.cn(K. Liu), shigw@tsinghua.edu.cn(G. Shi).

computing requirements make cloud servers the main computing platform [20]. However, cloud servers face challenges such as high transmission latency and expensive service costs, which limit their application in low-latency, high-performance scenarios. Therefore, how to optimize the computing requirements of AIGC models while ensuring service quality has become a difficulty in current technological development.

Edge computing addresses these challenges by offloading computational tasks from traditional centralized cloud servers to network edge devices closer to the data source [21]. Compared to traditional cloud computing, edge computing can effectively reduce data transmission latency, save bandwidth, improve response speed, and decrease reliance on cloud data centers. This makes edge computing particularly advantageous in scenarios requiring real-time data processing and low-latency responses. Traditional AIGC services require transmitting service requests to cloud data centers, resulting in significant transmission delays and resource consumption. By deploying AIGC models closer to users on edge networks [22, 23], low-latency and high-privacy AIGC services can be provided to users.

In this paper, we employ the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm to address the problem of edge intelligence model caching and resource allocation for in-situ AIGC services. Under the constraints of limited edge storage space, bandwidth resources, and computational resources, our goal is to reduce the response time and cost of user model service requests. The main contributions of this paper are as follows:

(1) We propose the Edge Artificial Intelligence Generated Content (EdgeAIGC) framework. Aiming at the model caching and resource allocation problems of edge AIGC services, it implements edge reasoning or cloud reasoning of models supported by AIGC service requests. Compared with traditional cloud reasoning, it can achieve efficient processing of services.
(2) We consider multi-model environments, build reasonable mathematical models, and concretely measure the response time and service cost of AIGC services in edge reasoning and cloud reasoning.
(3) To address this problem, we adopt the TD3 algorithm. Through appropriate mathematical modeling, the algorithm maps actions to optimal model caching and resource allocation decisions.
(4) Through extensive experiments, we validate the effectiveness of our approach. Compared to benchmark methods, our method achieves lower latency and cost for user service requests, along with a higher edge model hit rate.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the system model and problem formulation. The TD3 algorithm is introduced in Section 4. Section 5 provides all experimental results. Finally, in Section 6, we summarize our work.

2. Related works

2.1. Edge content caching

In the era of 6G digitalization, with the widespread adoption of smart devices, user demands have gradually shifted toward pursuing high-quality services. Cloud computing, due to its high resource availability and computational power, has been widely adopted. However, its inherent high latency issues have made it unable to meet the growing demand for efficient services. Edge content caching, enabling in-situ services for users, has become a trend [24, 25]. Xu et al. [26] modeled the edge caching problem as a Markov Decision Process (MDP) and proposed an end-to-end edge caching strategy using a Double Deep Recurrent Q-Network (DDRQN) algorithm, effectively improving system rewards and cache hit rates. Yasir et al. [27] proposed a content popularity and user preferences aware content caching method in mobile edge computing by considering user preferences and content popularity, significantly enhancing edge caching performance, cache hit rates, and reducing response times. Li et al. [28] proposed a cache strategy based on user mobility and content popularity, a content delivery strategy based on marginal benefits, and a data placement and content replacement algorithm. The cache strategy models user mobility based on the Markov process, and uses a multivariate linear regression model data placement algorithm to predict popularity. Tang et al. [29] introduced an intelligent edge caching strategy, EICache, for mobile edge computing, which improved cache hit rates by considering user mobility and interest preferences. Qiao et al. [30] proposed an Adaptive Federated learning-based Proactive Content caching (AFPC). This scheme combines deep reinforcement learning algorithms to solve the problems of Non-Independent and Identically Distributed (Non-IID) data and resource constraints in distributed federated learning, effectively improving cache efficiency and reducing resource consumption. Zhou et al. [31] designed a caching decision method based on the simplex algorithm, enabling edge content caching for computational offloading, which significantly reduced task response latency. Xu et al. [32] proposed a smart city edge content caching method (E-Cache) with service demand prediction function. Minimize the execution time and energy consumption of in-vehicle services. Zhao et al. [33] considered a UAV-supported mobile edge computing network, addressing content caching, service placement optimization, and task offloading optimization problems. They proposed an algorithm based on Gibbs sampling and matching games, effectively improving network quality of experience.

Regarding edge model caching, Xu et al. [34] addressed the problem of edge caching and inference for Pretrained Foundation Models (PFMs) in metaverse edge intelligence. They proposed a joint model caching and inference framework to manage models and allocate resources effectively to meet user requests, thereby reducing system latency and energy consumption. Chen et al. [35] investigated the Deep Neural Network (DNN) Model Caching and Processor Allocation (DMCPA) problem and proposed an online algorithm, DMCPA-GS-Online, to reduce user-perceived latency and energy consumption. However, these works do not fully consider the optimization of cost-related factors and edge cache hit rate.

2.2. Edge computing resource allocation

In edge network environments, resource allocation is crucial for meeting the performance requirements of various applications. Reasonable resource allocation strategies can balance loads, avoiding situations where some servers are overloaded while others remain idle, thereby improving overall system performance [36]. Jiang et al. [37] proposed an online Joint Offloading and Resource Allocation (JORA) framework under energy constraints in mobile edge computing and designed online JORA methods for both centralized and distributed scenarios, achieving near-optimal performance. Mahmood et al. [38] designed a three-layer network architecture for IoT-enabled smart cities and proposed an auction-based edge resource allocation scheme to optimize energy consumption and computational latency. Liu et al. [39] proposed a Digital Twin (DT)-supported collaborative edge intelligence scheme, fully considering communication, computation, and caching resource allocation to minimize response latency. Cang et al. [40] addressed the optimization problem of semantic-aware communication and computational resource allocation, proposing a low-complexity online algorithm based on Lyapunov optimization to minimize system energy consumption while satisfying latency constraints. Liu et al. [41] introduced a Multi-Objective Resource Allocation Method (MRAM) based on the Pareto Archived Evolution Strategy (PAES) to optimize load balancing in edge servers.

In summary, extensive research has been conducted on edge content caching and computational resource allocation. However, as shown in Table 1, studies specifically focusing on in-situ AIGC services and model caching at the edge remain limited, particularly with respect to cost optimization and cache hit rate improvement. Due to the typically

Table 1
Comparison of edge content caching literature.

Literature	Optimization objective				Methods	Considerations
	Response latency	Energy consumption	Cost	Hit rate		
[26]	✗	✗	✓	✓	DDRQN algorithm	Content popularity
[27]	✓	✗	✗	✓	CoPUP content caching method	User preferences and content popularity
[28]	✓	✗	✓	✓	Caching strategies based on user mobility and content popularity	User mobility and content popularity
[29]	✗	✗	✗	✓	EICache content caching decisions	User mobility and user interest
[30]	✗	✓	✗	✓	AFPC active content caching scheme	Non-independent and uniformly distributed data across clients
[31]	✓	✗	✗	✗	STGNN and the Simplex algorithm	In-vehicle Services in Internet of Vehicles (IoV)
[32]	✓	✓	✗	✗	E-Cache content caching method	In-vehicle services in the Internet of Vehicles
[33]	✓	✗	✗	✓	An algorithm based on Gibbs sampling and matching games	UAV edge content services
[34]	✓	✓	✗	✗	Least-Context (LC) algorithm	Pre-training Foundation Model
[35]	✓	✓	✗	✗	Online algorithm DMCPA-GS-Online	Deep Neural Networks
Ours	✓	✗	✓	✓	TD3 algorithm	Model service request popularity and AIGC models

large parameter scales and complex computational demands of AIGC models, achieving efficient model caching and resource allocation in edge environments faces numerous challenges. Efficient edge resource allocation for AIGC model caching is crucial for meeting user Quality of Service (QoS) requirements.

3. System model and problem formulation

As shown in Fig. 1, we consider an EdgeAIGC network architecture that supports in-situ AIGC, consisting of a cloud service center, an edge service layer, and a user layer. The cloud service center consists of a Cloud Server (CS) with rich computing resources and storage space, which stores many pre-trained AIGC models, such as text-to-speech, text-to-text, text-to-image, and text-to-video models. It can meet all user inference service requests, but the services provided will also take a higher time and cost. We denote the set of AIGC models in the CS as $\mathcal{M} = \{1, \ldots, M\}$. The edge service layer consists of E Edge Servers (ES), represented by the set $\mathcal{E} = \{1, \ldots, E\}$. Each edge server has limited model storage space, transmission bandwidth resources, and computational resources, enabling it to cache a subset of models and provide high-quality, low-latency, and low-cost in-situ AIGC services to users. The user layer comprises U Users (UD), represented by the set $\mathcal{U} = \{1, \ldots, U\}$. Each user generates a model service request in each time frame, denoted by the set $\mathcal{T} = \{1, \ldots, T\}$. Each user can arbitrarily choose an edge server in the current time frame as a node for edge execution or a transit transmission node for cloud execution. If the model requested by a user in the current time frame is not cached on the corresponding edge server, the service request is forwarded to the cloud server for processing.

The EdgeAIGC network architecture consists of $1 + E + U$ nodes, where the coordinates of each node are represented by two-dimensional coordinates (X, Y). The computational resources of the ES and CS are represented by the number of A800 GPUs. The model service request generated by the i_{th} UD ($i \in \mathcal{U}$) in each time frame t ($t \in \mathcal{T}$) is represented as $\{Z_i(t), d_i^{in}(t), d_i^{out}(t)\}$. Here, $Z_i(t)$ ($Z_i(t) \in \mathcal{M}$) denotes the type of model requested by the i_{th} UD in the t_{th} time frame. $d_i^{in}(t)$ represents the input size of the service request from the i_{th} UD in the t_{th} time frame, and $d_i^{out}(t)$ represents the expected output size of the model requested by the i_{th} UD in the t_{th} time frame.

Since each user is mainly concerned about the efficiency and cost of task completion, while the service provider focuses on cost profit,

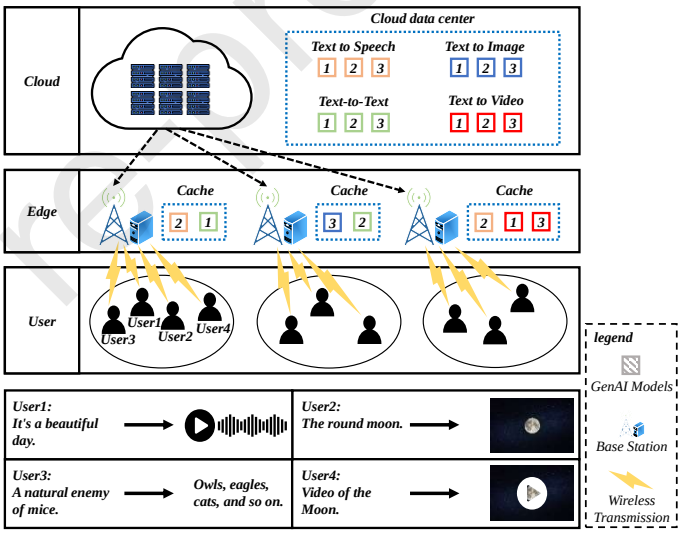


Fig. 1. EdgeAIGC framework.

here we will consider optimizing the average response time and average cost of all user request model service executions. The ES serves both as a service processing node and a relay transmission node for the CS. We focus on the storage capacity constraints, bandwidth resource constraints, and computational resource constraints of the edge servers. Table 2 summarizes the key notations used in this paper. Next, we introduce the single-time-scale user AIGC model service request, model caching, response time model, and cost model.

3.1. Single-time-scale user AIGC model service requests

As shown in Fig. 2, the type of model requested by each user dynamically changes according to popularity trends. A finite Markov state transition model with three states, γ_1, γ_2, and γ_3, is used to model the popularity of AIGC services, with parameters $\gamma_1 = 0.2$, $\gamma_2 = 0.5$, and $\gamma_3 = 0.8$. The transition probabilities between these popularity states are defined as shown in Equation 1 [42]. For example, $P_{13} = 0.2$ indicates that the probability of transitioning from popularity state γ_1 to γ_3 is 0.2.

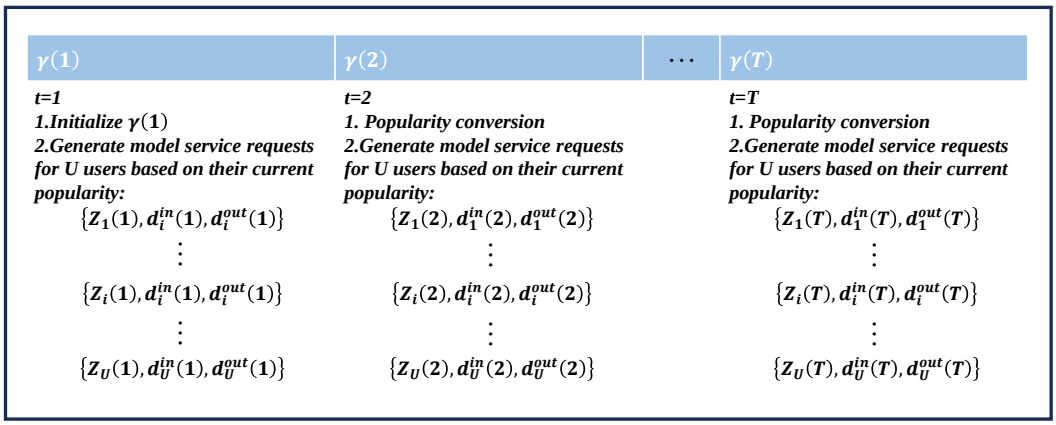


Fig. 2. User model service request.

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.1 & 0.7 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} \tag{1}$$

In each new time frame, each user generates a model service request based on the current popularity state. Assuming that the probability of i_{th} UD requesting service from model m ($m \in \mathcal{M}$) in time slot t follows a Zipf distribution, it is defined as shown in Equation 2.

$$P\{Z_i(t) = m\} = \frac{m^{-\gamma(t)}}{\sum_{i \in M} i^{-\gamma(t)}} \tag{2}$$

3.2. Model caching

For the models in the cloud, we consider four types, including text-to-speech, text-to-image, text-to-text, and text-to-video. The model type influences the size of the generated output. Additionally, for each type, models are further distinguished by their scale, where different scales correspond to different floating-point operations F and model sizes S. Each model m is characterized by two features: (S_m, F_m). Due to the limited storage space in the ES, only a subset of AIGC models from the CS can be cached. Therefore, each ES must make reasonable model caching decisions in each time frame t based on the current model requests from all users, thereby improving the overall edge cache hit rate of the system. This is represented in Equation 3. For example, $\beta_{21} = 1$ indicates that edge server 2 caches model 1, and $\beta_{32} = 1$ indicates that edge server 3 caches model 2.

$$\beta(t) = \begin{bmatrix} \beta_{11}(t) & \beta_{12}(t) & \cdots & \beta_{1M}(t) \\ \beta_{21}(t) & \beta_{22}(t) & \cdots & \beta_{2M}(t) \\ \beta_{31}(t) & \beta_{32}(t) & \cdots & \beta_{3M}(t) \\ \cdots & \cdots & \cdots & \cdots \\ \beta_{E1}(t) & \beta_{E2}(t) & \cdots & \beta_{EM}(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 1 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & \cdots & 1 \end{bmatrix} \tag{3}$$

3.3. Response time model

3.3.1. Edge computing model

The edge computing mode refers to the scenario where a user's model service request is executed on an edge server. This requires that the edge server has cached the AIGC model requested by the user. Assume that the service request characteristics of the i_{th} UD are $\{Z_i(t), d_i^{in}(t), d_i^{out}(t)\}$, where $Z_i(t) = m$, and the request is assigned to the j_{th} ES ($j \in E$) with $\beta_{jm} = 1$. Then, the service request of the i_{th} UD is executed on the j_{th} ES. For the downlink bandwidth resources of the ES, an exclusive resource allocation strategy is adopted for transmitting the output results. Therefore, the downlink transmission waiting time from the ES to the UD must be considered. The response time of the model service request is the sum of the uplink transmission time from the UD to the ES, the model inference time on the ES, the downlink transmission waiting time from the ES to the UD, and the downlink transmission time. When the downlink bandwidth resources of the ES are not occupied, the downlink transmission waiting time is 0.

The uplink transmission rate $R^{up}_{U_i,E_j}$ between the i_{th} UD and the j_{th} ES is defined as:

$$D_{i,j} = \sqrt{(X_{U_i} - X_{E_j}) + (Y_{U_i} - Y_{E_j})} \tag{4}$$

$$g_i(t) = -128.1 - 37.6 \log_{10} D_{i,j} \tag{5}$$

$$h_{i,j}(t) = g_{i,j} |\delta_i(t)|^2 \tag{6}$$

$$R^{up}_{U_i,E_j}(t) = \alpha_{j,i}(t) W^{up}_j \log_2 \left(1 + \frac{p^U_i h_{i,j}(t)}{N_0 \alpha_{j,i}(t) W^{up}_j}\right) \tag{7}$$

Where $\delta_i(t) \sim CN(0, 1)$ represents the Rayleigh fading component of the i_{th} UD.

The downlink transmission rate $R^{down}_{U_i,E_j}$ between the j_{th} ES and the i_{th} UD is defined as:

$$R^{down}_{U_i,E_j}(t) = W^{down}_j \log_2 \left(1 + \frac{p^E_j h_{i,j}(t)}{N_0 W^{down}_j}\right) \tag{8}$$

The response time of the service request in edge computing mode is defined as:

$$T^{to}_i(t) = T^{tw}_i(t) + \frac{d^{out}_i(t)}{R^{down}_{U_i,E_j}(t)} \tag{9}$$

$$T^{res}_i(t) = \frac{d^{in}_i(t)}{R^{up}_{U_i,E_j}(t)} + \frac{F_m}{\mathcal{E}_{j,i}(t) K_j A_8} + T^{to}_i(t) \tag{10}$$

Where T^{tw}_i represents the downlink transmission waiting time of the output results for the i_{th} UD service request, and T^{to}_i represents the downlink transmission completion time of the output results for the i_{th} UD service request.

3.3.2. Cloud computing model

The cloud computing mode refers to the scenario where a user's model service request is transmitted to the cloud server for execution. First, the user sends the service request to the edge server. If the edge server has not cached the AIGC model requested by the user, the request is forwarded to the cloud server for execution. Assume that the characteristics of the i_{th} UD are $Z_i(t) = m$, and the request is assigned to the j_{th} ES with $\beta_{jm} = 0$. Then, the service request of the i_{th} UD is executed on the CS. Since the downlink bandwidth resources of the ES adopt an exclusive resource allocation strategy, when the cloud server completes processing the request and transmits it to the edge server, it competes

Table 2
Commonly used terms in edge intelligence model.

Notation	Description
U	Number of UDs
E	Number of ESs
M	Number of models
T	Number of time slots
N	Number of algorithm iterations
p_i^U	Transmission power of the i_{th} UD
p_j^E	Transmission power of the j_{th} ES
$Z_i(t)$	Model requested by the i_{th} UD in time slot t
$d_i^{in}(t)$	Input size from the i_{th} UD in time slot t
$d_i^{out}(t)$	Output size of the model from the i_{th} UD in time slot t
S_j^E	Storage capacity of the j_{th} ES
W_j^{up}	Uplink bandwidth resources of the j_{th} ES
W_j^{down}	Downlink bandwidth resources of the j_{th} ES
K_j	Computational resources of the j_{th} ES
$R_{E,CS}^{up}$	Uplink transmission rate between ES and CS
$R_{E,CS}^{down}$	Downlink transmission rate between ES and CS
$R_{U_i,E_j}^{up}(t)$	Uplink transmission rate between the i_{th} UD and the j_{th} ES in time slot t
$R_{U_i,E_j}^{down}(t)$	Downlink transmission rate between the i_{th} UD and the j_{th} ES in time slot t
$D_{i,j}$	Euclidean distance between the i_{th} UD and the j_{th} ES
$g_{i,j}$	Path loss between the i_{th} UD and the j_{th} ES
N_0	Noise power spectral density
$C_{E,CS}^{tran}$	Transmission cost between ES and CS
$C_{U,E}^{tran}$	Transmission cost between UD and ES
C_{A10}^{com}	Computational cost of A800
C_E^{st}	Storage cost of ES
A_8	Computing power of A800
$T_i^{res}(t)$	Response time of the i_{th} UD service request in time slot t
$C_i(t)$	Cost of the i_{th} UD service request in time slot t
$\gamma(t)$	Popularity in time slot t
S_m	Size of the m_{th} AIGC model
F_m	Floating-point operations required for the m_{th} AIGC model
$\beta(t)$	Caching decision matrix in time slot t
$\alpha(t)$	Bandwidth resource allocation matrix in time slot t
$\mathcal{L}$	Computational resource allocation matrix in time slot t

for resources with all other service requests undergoing downlink transmission through the same edge server. Therefore, the response time of the model service request is the sum of the uplink transmission time from the UD to the ES, the uplink transmission time from the ES to the CS, the model inference time on the CS, the downlink transmission time from the CS to the ES, the downlink transmission waiting time from the ES to the UD, and the downlink transmission time.

The response time of the service request in cloud computing mode is defined as:

$$T_i^{res}(t) = \frac{d_i^{in}(t)}{R_{U_i,E_j}^{up}(t)} + \frac{d_i^{in}(t)}{R_{E,CS}^{up}} + \frac{F_m}{K_{CS}A_8} + \frac{d_i^{out}(t)}{R_{E,CS}^{down}} + T_i^{to}(t) \tag{11}$$

3.4. Cost model

3.4.1. Edge computing model

The edge computing model needs to consider the cost of data transmission. Therefore, the cost of the model service request in edge computing mode is the sum of the uplink transmission cost from the UD to the ES, the GPU rental cost for model inference on the ES, and the downlink transmission cost from the ES to the UD.

The cost in edge computing mode is defined as:

$$C_i(t) = d_i^{in}(t)C_{U,E}^{tran} + C_{A8}^{com}\frac{F_m}{A_8} + d_i^{out}(t)C_{U,E}^{tran} \tag{12}$$

3.4.2. Cloud computing model

The cloud computing model, in addition to the edge computing mode, also needs to consider the transmission cost between the edge server and the cloud server. Therefore, the cost of the model service request in cloud computing mode is the sum of the uplink transmission cost from the UD to the ES, the uplink transmission cost from the ES to the CS, the GPU rental cost for model inference on the CS, the downlink transmission cost from the CS to the ES, and the downlink transmission cost from the ES to the UD.

The cost in cloud computing mode is defined as:

$$\begin{aligned} C_i(t) = {} & d_i^{in}(t)C_{U,E}^{tran} + d_i^{in}(t)C_{E,CS}^{tran} + C_{A8}^{com}\frac{F_m}{A_8} \\ & + d_i^{out}(t)C_{U,E}^{tran} + d_i^{out}(t)C_{E,CS}^{tran} \end{aligned} \tag{13}$$

3.5. Optimization problem

Under the constraints of limited storage space, bandwidth resources, and computational resources of edge servers, we formulate the problem of edge model caching and resource allocation for AIGC as a dynamic optimization problem. Our goal is to minimize the total response time and total cost of all user service requests across all time slots. Equations 10, 11, 12, and 13 describe the response time and cost of service requests in the two computing modes, respectively. Since caching multiple AIGC models on edge servers occupies significant storage space, we also consider the caching cost of edge servers in the total cost, as shown in Equation 14.

$$C^{st}(t) = \sum_{j\in E} S_{M,j}C_E^{st} \tag{14}$$

Where $S_{M,j}$ represents the storage space occupied by all models cached on the j_{th} ES.

Therefore, the weighted sum of response time and cost for all users in time slot t is:

$$G(t) = \omega_1 \sum_{i\in\mathcal{U}} T_i^{res}(t) + \omega_2 \left(\sum_{i\in\mathcal{U}} C_i(t) + C^{st}(t) \right) \tag{15}$$

Where ω_1 and ω_2 represent the weights of response time and cost, respectively.

To provide high-quality, low-latency, and low-cost AIGC services, it is necessary to optimize the edge caching of models $\beta(t)$, bandwidth allocation $\alpha(t)$, and computational resource allocation $\mathcal{L}(t)$, as shown in Equations 16 and 17.

$$\alpha(t) = \begin{bmatrix} \alpha_{11}(t) & \alpha_{12}(t) & \cdots & \alpha_{1U}(t) \\ \alpha_{21}(t) & \alpha_{22}(t) & \cdots & \alpha_{2U}(t) \\ \alpha_{31}(t) & \alpha_{32}(t) & \cdots & \alpha_{3U}(t) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{E1}(t) & \alpha_{E2}(t) & \cdots & \alpha_{EU}(t) \end{bmatrix} \tag{16}$$

$$\mathcal{L}(t) = \begin{bmatrix} \mathcal{L}_{11}(t) & \mathcal{L}_{12}(t) & \cdots & \mathcal{L}_{1U}(t) \\ \mathcal{L}_{21}(t) & \mathcal{L}_{22}(t) & \cdots & \mathcal{L}_{2U}(t) \\ \mathcal{L}_{31}(t) & \mathcal{L}_{32}(t) & \cdots & \mathcal{L}_{3U}(t) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{L}_{E1}(t) & \mathcal{L}_{E2}(t) & \cdots & \mathcal{L}_{EU}(t) \end{bmatrix} \tag{17}$$

Among them, bandwidth resource allocation $\alpha(t)$ and computing resource allocation $\mathcal{L}(t)$ are both represented by matrices, and each specific value represents the percentage of resource allocation. $\alpha_{22} = 0.2$ means that the bandwidth resources allocated by edge server 2 to user 2 account for 20% of the total bandwidth resources of edge server 2.

Similarly, $\mathcal{L}_{34} = 0.4$ means that the computing resources allocated by edge server 3 to user 4 account for 40% of the total computing resources of edge server 3.

We define the optimization problem as follows:

$$p : \min_{\beta,\alpha,\mathcal{L}} \frac{1}{T} \sum_{t\in\mathcal{T}} G(t) \tag{18a}$$

s.t.:

$$\beta_{j,m}(t) \in \{0,1\}, \quad \forall t \in \mathcal{T}, j \in \mathcal{E}, m \in \mathcal{M} \tag{18b}$$

$$\alpha_{j,i}(t) \in [0,1], \quad \forall t \in \mathcal{T}, i \in \mathcal{U}, j \in \mathcal{E} \tag{18c}$$

$$\mathcal{L}_{j,i}(t) \in [0,1], \quad \forall t \in \mathcal{T}, i \in \mathcal{U}, j \in \mathcal{E} \tag{18d}$$

$$\sum_{m\in M} \beta_{j,m}(t) S_m \le S_j^E, \quad \forall j \in \mathcal{E}, t \in \mathcal{T} \tag{18e}$$

$$\sum_{i\in U} \alpha_{j,i}(t) \le 1, \quad \forall j \in \mathcal{E}, t \in \mathcal{T} \tag{18f}$$

$$\sum_{i\in U} \mathcal{L}_{j,i}(t) \le 1, \quad \forall j \in \mathcal{E}, t \in \mathcal{T} \tag{18g}$$

Where Equation 18c represents the proportion of bandwidth resources allocated by the j_{th} ES to the i_{th} UD in time slot t.Equation 18d represents the proportion of computational resources allocated by the j_{th} ES to the i_{th} UD in time slot t.Equation 18e represents the storage capacity constraint of the j_{th} ES, ensuring that the total storage occupied by cached models does not exceed the server's capacity.Equation 18f represents the bandwidth resource allocation constraint of the edge server.Equation 18g represents the computational resource allocation constraint of the edge server.

4. Model caching and resource allocation method based on TD3

4.1. Motivation for adopting TD3 algorithm

In the resource allocation scenario, the action space is continuous, bandwidth resources, computing resource allocation, etc. are all continuous variables, and the state space presents high-dimensional characteristics, covering many factors such as task characteristics and user status. The TD3 algorithm can effectively solve the problem of continuous action space, and it also has good adaptability to the problem of high-dimensional state space. The algorithm determines the best cache decision and resource allocation scheme by learning the optimal continuous action strategy, which is an advantage that most discrete action space algorithms do not have. At the same time, the algorithm introduces a mechanism for delayed updating the policy network, making the update of the policy network more stable. It can prevent the policy network from falling into the local optimum due to frequent updates during the training process, allowing the algorithm to better explore better solutions. Therefore, we use the TD3 algorithm to solve the resource allocation problem, helping the system to efficiently and accurately find the optimal model cache and resource allocation strategy.

4.2. Markov decision process (MDP) elements

The problem P is formulated as a MDP, which consists of a state space, an action space, and a reward. Since we model the dynamic AIGC model requests of users as single-time-scale dynamic changes, the three elements are defined as follows:

- **State Space:** At the beginning of each time slot, the CS observes the current model service requests of all users to provide a good policy based on reinforcement learning. The state space is defined as:

$$s(t) = \{\gamma(t), \delta(t), Z(t), d^{in}(t), d^{out}(t)\} \tag{19}$$

  Where $\gamma(t)$ represents the popularity trend of user requests for AIGC models in different time slots. $\delta(t)$ represents the Rayleigh fading component between different users, used to measure the channel gain between users and edge servers. $Z(t)$ represents the types of models requested by all users in the current time slot. $\{d^{in}(t), d^{out}(t)\}$ correspond to the input data size and the expected output data size of the requested model $Z(t)$.

- **Action Space:** Our goal is to minimize the response time and cost over T time slots. To achieve this, we need to determine the specific models cached on the edge servers, allocate bandwidth resources, and allocate computational resources to users. The action space is defined as:

$$a(t) = \{\beta(t), \alpha(t), \mathcal{L}(t)\} \tag{20}$$

  where $\beta(t)$ represents the model caching matrix. $\alpha(t)$ represents the bandwidth resource allocation matrix. $\mathcal{L}(t)$ represents the computational resource allocation matrix.

- **Reward:** To minimize the objective function, the environment provides a specific reward value based on the current state $s(t)$ and action $a(t)$ in each time slot. The reward is defined as:

$$r(t) = -\frac{1}{U} G(t) \tag{21}$$

4.3. Architecture of the TD3 algorithm

Fig. 3 describes the architecture of the TD3 algorithm, which includes a total of 6 neural networks.

- **Online Actor Network (π_Φ, Parameters Φ):** The online actor network is used to generate deterministic actions a based on the current environment state s. The goal is to learn an optimal policy that maximizes the long-term reward by selecting actions in a given state. The network is updated using the policy gradient method, specifically by maximizing the Q-value provided by the critic network. The update formula is:

$$\nabla_\Phi J(\Phi) = \frac{1}{E} \sum_{e=1}^{E} \nabla_a Q_{\theta_1}(s_e(t), a)\big|_{a=\pi_\Phi(s_e(t))} \nabla_\Phi \pi_\Phi(s_e(t)) \tag{22}$$

- **Target Actor Network ($\pi_{\Phi'}$, Parameters Φ'):** The target actor network is used to provide a more stable policy estimate. It has the same structure as the online actor network but updates its parameters more slowly. A soft update is applied, with the formula:

$$\Phi' \leftarrow \tau\Phi + (1-\tau)\Phi' \tag{23}$$

- **Dual Online Critic Networks (Q_{θ_1}, Q_{θ_2}, Parameters θ_1, θ_2):** The dual online critic networks are used to estimate the Q-value for a given state s and action a. The algorithm introduces two critic networks to reduce the overestimation problem in Q-value estimation. Both networks are updated by minimizing the mean squared error between the predicted Q-value and the target Q-value. The update formulas are:

$$\begin{aligned} y'_e(t) = r_e(t) + h \min\Big(&Q_{\theta_1}(s_e(t+1), \pi_{\Phi'}(s_e(t+1))), \\ &Q_{\theta_2}(s_e(t+1), \pi_{\Phi'}(s_e(t+1)))\Big) \end{aligned} \tag{24}$$

$$y_e^i(t) = Q_{\theta_i}(s_e(t), \pi_\Phi(s_e(t))), \quad i \in \{1,2\} \tag{25}$$

$$L_{\theta_i} = \frac{1}{E} \sum_{e=1}^{E} \left(y'_e(t) - y_e^i(t)\right)^2, \quad i \in \{1,2\} \tag{26}$$

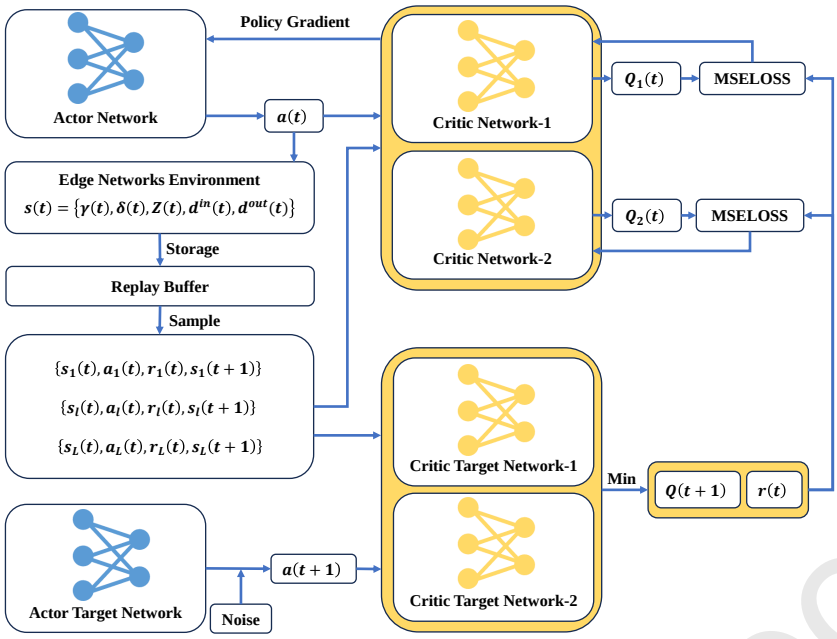


Fig. 3. TD3 architecture.

- **Dual Target Critic Networks ($Q_{\theta'_1}$, $Q_{\theta'_2}$, Parameters θ'_1, θ'_2):** The dual target critic networks are used to provide more stable Q-value estimates. They have the same structure as the online critic networks but update their parameters more slowly. A soft update is applied, with the formula:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i, \quad i \in \{1,2\} \tag{27}$$

Where E represents the number of training samples.

Algorithm 1 TD3 Algorithm Training Process.

Input: Initialize network parameters, number of iterations N, total time slots T, training threshold Min
Output: Trained neural network

```
for episode = 1 to M do do
    Randomly initialize popularity, generate user model service requests according to Equation 2, and generate initial state s(1)
    for t = 1 to T do do
        Input state s(t) into the actor network to obtain action a(t)
        Input a(t) into the network environment to obtain reward r(t), perform popularity transition according to Equation 1, generate user model service requests, and generate next state s(t + 1)
        Store {s(t), a(t), r(t), s(t + 1)} in the experience replay buffer C
        if size(C) > Min then
            Perform neural network training in TD3
        end if
    end for
end for
```

4.4. TD3 algorithm

As shown in Algorithm 1, first perform $\{s(t), a(t), r(t), s(t+1)\}$ data access according to the number of algorithm iterations N and the total time slot T. Each iteration will generate T groups of data, and each data will be stored in the experience pool in turn. When the amount of data in the experience pool accumulates to the training capacity, the network training process is started. During each training, a small batch of data samples are randomly selected from the experience pool, and these samples are used to train the Actor network and the Critic network. Through training, the Actor network has been optimized so that it can output an action strategy that maximizes long-term cumulative returns in a given state; at the same time, the Critic network has also been improved so that under a given state and action combination, it can output a predicted Q value that is as close as possible to the actual Q value, thereby continuously improving the algorithm's ability to learn and optimize model caching and resource allocation strategies.

Table 3
Experimental parameter setting.

Parameters	Value
U	10
E	2
M	20
T	10
N	800
p_i^U (dBm)	23
p_j^E (dBm)	43
$d_i^{\text{in}}(t)$ (MB)	[4, 8]
$d_i^{\text{out}}(t)$ (MB)	[500, 510], [520, 530], [540, 550], [560, 570]
S_j^E (GB)	[30, 50]
W_j^{up} (MHz)	[10, 20]
W_j^{down} (MHz)	[30, 40]
K_j	[25, 30]
$R_{E,CS}^{\text{up}}$ (Kbps)	500
$R_{E,CS}^{\text{down}}$ ((Kbps))	500
N_0 (dBm/Hz)	-174
$C_{E,CS}^{\text{tran}}$	0.6
$C_{U,E}^{\text{tran}}$	0.2
C_{A8}^{com}	4
C_E^{st}	0.4
S_m (GB)	[4, 8]
F_m (Eflops)	[10, 20]

5. Experimental simulation and evaluation

5.1. Environment settings

We consider a network topology environment consisting of a cloud data center, multiple ESs, and multiple UDs. The locations of CS, ESs,

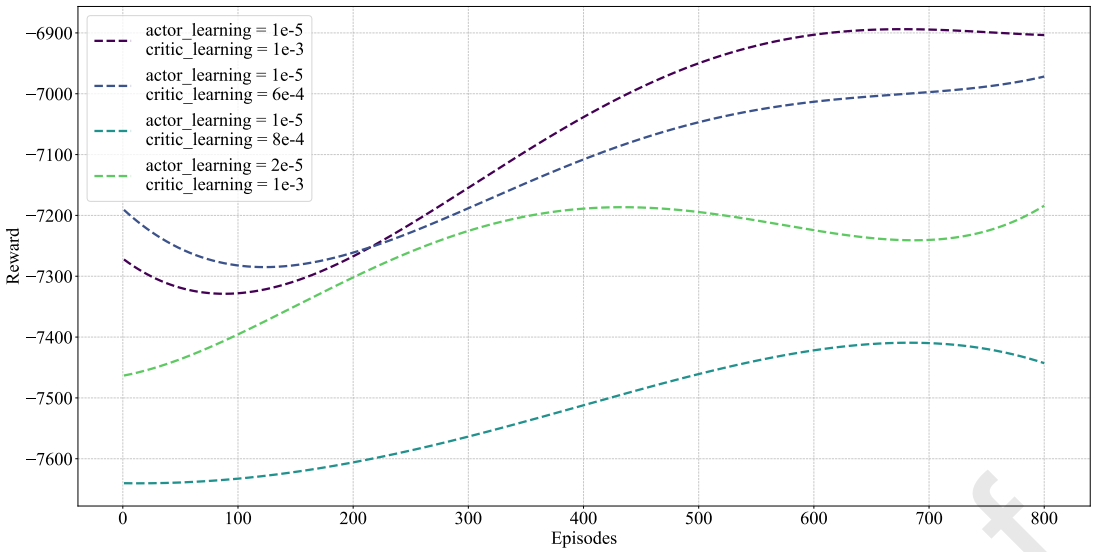


Fig. 4. The impact of different learning rates.

and UDs are determined by two-dimensional coordinates (X, Y). For model service requests that are not hit in ESs, they will be assigned to cloud servers for execution. The cloud server will allocate 5 A800 graphics cards for each service request for model calculation. In addition, in order to simulate the network environment of various AIGC models, we divide the model types into four categories, namely text to text, text to speech, text to image, and text to video. The scale of the output results of different types of model execution is different. The model size and the number of floating-point operations required for execution of different models are also different.

The experimental simulation environment is implemented on the server, which is equipped with two Intel(R) Xeon(R) E5-2690 v2 CPUs (3.00GHz, 10 physical cores), 256G RAM. The servers run Linux kernel 4.4.0-31. The relevant parameters in the simulation experiment are shown in Table 3. The following algorithms are used to compare the performance with the algorithm proposed in this paper: Deep Deterministic Policy Gradient Algorithm (DDPG), PSO-based Cache and Resource Allocation Scheme (PCRAS), GA-based Cache and Resource Allocation Scheme (GCRAS).

5.2. *Performance evaluation*

We evaluate EdgeAIGC from two aspects: performance and scalability.

5.2.1. *Performance*

The data convergence curve is fitted by a 4th-order polynomial function. As shown in Fig. 4, the TD3 algorithm shows a better convergence value of the total average user reward for T time slots when the actor network learning rate is $1e-5$ and the critic network learning rate is $1e-3$. When the actor network learning rate is fixed at $1e-5$, by reducing the critic network learning rate and setting it to $6e-4$ and $8e-4$, the overall training progress is slow from the perspective of the curve fluctuation amplitude. The reduced critic network learning rate causes the critic network to converge slowly on the value function, and it takes a long time to accurately estimate the value of the state and action. This causes the actor network to be unable to obtain accurate value feedback in a timely manner, thereby affecting the effect of the decision. When the critic network learning rate is fixed at $1e-3$, by increasing the actor network learning rate and setting it to $2e-5$, from the perspective of the curve fluctuation amplitude, the curve oscillates in the later stage and it is difficult to achieve convergence. The increase in the actor network learning rate leads to excessive network parameter update amplitude. The output strategy of the actor network changes dramatically during the training process, making it difficult to converge to a stable strategy and easily miss the optimal solution. This proves that the TD3 algorithm has high requirements for the setting of learning rate. As shown in Fig. 5, the TD3 algorithm shows a better convergence effect than the DDPG algorithm. Due to the problem of overestimation of Q value in the DDPG algorithm, it is easy to cause instability in the learning process, resulting in poor learning stability and repeated oscillation. TD3 significantly improves learning efficiency and stability through dual critic networks and delayed update strategies, and improves reward optimization by about 1.72% compared with DDPG.

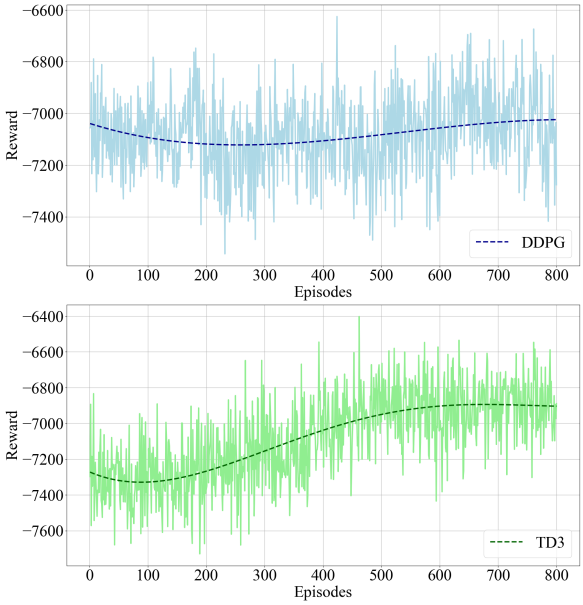


Fig. 5. Comparison of algorithm convergence.

5.2.2. *Scalability*

As shown in Fig. 6, as the number of users increases, the model hit rate also increases. This is because users' model service requests gradually reflect the probability distribution, and popular model services account for a larger proportion of total requests. The model hit rate of the TD3 algorithm is always better than other benchmark algorithms,

with a maximum improvement of 41.06% compared to DDPG, a maximum improvement of 50.93% compared to PCRAS, and a maximum improvement of 57.85% compared to GCRAS. Fig. 7 shows that due to the bandwidth resource and computing resource constraints of the edge server, as the number of users increases, users compete for fixed resources, and each user will obtain fewer resources, causing the value of the objective optimization function to continue to grow.

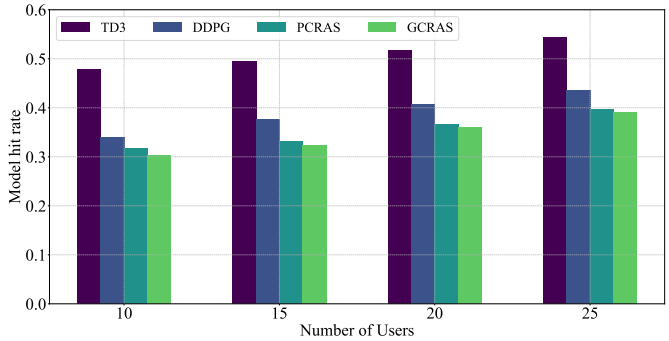


Fig. 6. Comparison of algorithm hit rates.

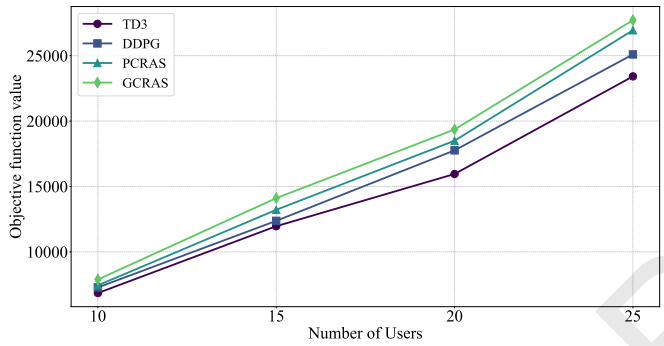


Fig. 7. Changes in objective function values for different numbers of users.

6. Conclusion

Aiming at the high latency and high cost problems faced by AIGC in cloud computing scenarios, this paper proposes a joint optimization framework for edge intelligent model caching and resource allocation based on the TD3 algorithm. By building an EdgeAIGC network architecture that includes cloud service centers, edge servers, and users, combined with dynamic model popularity and storage and computing resource constraint modeling, the model caching decision of the edge server and the coordinated allocation of bandwidth and computing resources are realized. Experiments show that compared with the baseline method, the TD3 algorithm improves the model hit rate by at least approximately 41.06% and can effectively reduce the objective function value, providing new ideas and methods for the future integration of edge computing and AIGC. However, in terms of model edge caching, facing the growing scale of AIGC models and diverse model types, how to further optimize caching strategies to adapt to more complex scenarios remains an unresolved issue. Although the TD3 algorithm showed advantages in this study, there is still room for improvement. Future research can design more efficient algorithms so that they can focus more on key information and improve decision-making efficiency and accuracy when processing a large number of dynamic user requests and complex model environments. At the same time, in view of the privacy sensitivity of AIGC model parameters and user data, future research can explore secure caching technology in edge environments and make caching decisions under the premise of ensuring the security of model intellectual property rights and user data.

CRediT authorship contribution statement

Wu Wen: Conceptualization, Data curation, Formal analysis, Methodology, Visualization, Writing – original draft, Writing – review & editing. **Yibin Huang:** Conceptualization, Formal analysis, Methodology, Visualization, Writing – original draft, Writing – review & editing. **Xinxin Zhao:** Conceptualization, Formal analysis, Methodology, Writing – original draft, Writing – review & editing. **Peiying Zhang:** Conceptualization, Formal analysis, Methodology, Writing – original draft, Writing – review & editing. **Kai Liu:** Data curation, Methodology, Writing – review & editing. **Guowei Shi:** Data curation, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the Shandong Provincial Natural Science Foundation under Grant ZR2023LZH017, ZR2022LZH015 and ZR2024MF066, the National Natural Science Foundation of China under Grant 62471493, the Tertiary Education Scientific research project of Guangzhou Municipal Education Bureau under Grant 2024312246, the Guangzhou Higher Education Teaching Quality and Teaching Reform Project under Grant 2023KCJJD002.

References

[1] E. Cetinic, J. She, Understanding and creating art with ai: Review and outlook, ACM Transactions on Multimedia Computing, Communications, and Applications 18 (2) (2022) 1–22.
[2] G. Liu, H. Du, D. Niyato, J. Kang, Z. Xiong, D. I. Kim, X. Shen, Semantic communications for artificial intelligence generated content (aigc) toward effective content creation, IEEE Network 38 (5) (2024) 295–303.
[3] H. Kim, S. Kim, S. Yoon, Guided-tts: A diffusion model for text-to-speech via classifier guidance, in: International Conference on Machine Learning, PMLR, 2022, pp. 11119–11133.
[4] C. Zhang, C. Zhang, S. Zheng, M. Zhang, M. Qamar, S.-H. Bae, I. S. Kweon, A survey on audio diffusion models: Text to speech synthesis and enhancement in generative ai, arXiv preprint arXiv:2303.13336.
[5] B. An, Ai-generated text detection: challenges and future directions, International Journal of Asian Language Processing 33 (02) (2023) 2330002.
[6] O. AI, Gpt-4 technical report, arXiv preprint arXiv:2303.08774.
[7] Y. X. Tan, C. P. Lee, M. Neo, K. M. Lim, J. Y. Lim, A. Alqahtani, Recent advances in text-to-image synthesis: Approaches, datasets and future research prospects, IEEE Access 11 (2023) 88099–88115.
[8] S. Xu, D. Hou, L. Pang, J. Deng, J. Xu, H. Shen, X. Cheng, Invisible relevance bias: Text-image retrieval models prefer ai-generated images, in: Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval, 2024, pp. 208–217.
[9] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, I. Sutskever, Zero-shot text-to-image generation, in: International conference on machine learning, Pmlr, 2021, pp. 8821–8831.
[10] Z. Xing, Q. Feng, H. Chen, Q. Dai, H. Hu, H. Xu, Z. Wu, Y.-G. Jiang, A survey on video diffusion models, ACM Computing Surveys 57 (2) (2024) 1–42.
[11] B. Qu, X. Liang, S. Sun, W. Gao, Exploring aigc video quality: A focus on visual harmony video-text consistency and domain distribution gap, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 6652–6660.
[12] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao, et al., Sora: A review on background, technology, limitations, and opportunities of large vision models, arXiv preprint arXiv:2402.17177.
[13] S. K. Ahmed, The impact of chatgpt on the nursing profession: revolutionizing patient care and education, Annals of biomedical engineering 51 (11) (2023) 2351–2352.

[14] L. Shao, B. Chen, Z. Zhang, Z. Zhang, X. Chen, Artificial intelligence generated content (aigc) in medicine: A narrative review, Mathematical Biosciences and Engineering 21 (1) (2024) 1672–1711.
[15] H. Li, H. Liu, D. Hu, J. Wang, I. Oguz, Promise: Prompt-driven 3d medical image segmentation using pretrained image foundation models, in: 2024 IEEE International Symposium on Biomedical Imaging, IEEE, 2024, pp. 1–5.
[16] H. Chen, B. Zhao, G. Yue, W. Liu, C. Lv, R. Wang, F. Zhou, Clip-medfake: Synthetic data augmentation with ai-generated content for improved medical image classification, in: 2024 IEEE International Conference on Image Processing, IEEE, 2024, pp. 3854–3860.
[17] X. Chen, Z. Hu, C. Wang, Empowering education development through aigc: A systematic literature review, Education and Information Technologies 29 (13) (2024) 17485–17537.
[18] R. Zhang, D. Zou, G. Cheng, A review of chatbot-assisted learning: pedagogical approaches, implementations, factors leading to effectiveness, theories, and future directions, Interactive Learning Environments 32 (8) (2024) 4529–4557.
[19] X. Deng, Z. Yu, A meta-analysis and systematic review of the effect of chatbot technology use in sustainable education, Sustainability 15 (4) (2023) 2940.
[20] Y.-C. Wang, J. Xue, C. Wei, C.-C. J. Kuo, An overview on generative ai at scale with edge–cloud computing, IEEE Open Journal of the Communications Society 4 (2023) 2952–2971.
[21] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, J. Cao, Edge computing with artificial intelligence: A machine learning perspective, ACM Computing Surveys 55 (9) (2023) 1–35.
[22] M. Xu, H. Du, D. Niyato, J. Kang, Z. Xiong, S. Mao, Z. Han, A. Jamalipour, D. I. Kim, X. Shen, et al., Unleashing the power of edge-cloud generative ai in mobile networks: A survey of aigc services, IEEE Communications Surveys & Tutorials 26 (2) (2024) 1127–1170.
[23] Y. Zhang, J. Zhang, S. Yue, W. Lu, J. Ren, X. Shen, Mobile generative ai: Opportunities and challenges, IEEE Wireless Communications 31 (4) (2024) 58–64.
[24] M. Reiss-Mirzaei, M. Ghobaei-Arani, L. Esmaeili, A review on the edge caching mechanisms in the mobile edge computing: A social-aware perspective, Internet of Things 22 (2023) 100690.
[25] I. Zyrianoff, A. Trotta, L. Sciullo, F. Montori, M. Di Felice, IoT edge caching: taxonomy, use cases and perspectives, IEEE Internet of Things Magazine 5 (3) (2022) 12–18.
[26] H. Xu, Y. Sun, J. Gao, J. Guo, Intelligent edge content caching: A deep recurrent reinforcement learning method, Peer-to-Peer Networking and Applications 15 (6) (2022) 2619–2632.
[27] M. Yasir, S. K. uz Zaman, T. Maqsood, F. Rehman, S. Mustafa, Copup: Content popularity and user preferences aware content caching framework in mobile edge computing, Cluster Computing 26 (1) (2023) 267–281.
[28] C. Li, M. Song, C. Yu, Y. Luo, Mobility and marginal gain based content caching and placement for cooperative edge-cloud computing, Information Sciences 548 (2021) 153–176.
[29] B. Tang, L. Kang, Eicache: A learning-based intelligent caching strategy in mobile edge computing, Peer-to-Peer Networking and Applications 15 (2) (2022) 934–949.
[30] D. Qiao, S. Guo, D. Liu, S. Long, P. Zhou, Z. Li, Adaptive federated deep reinforcement learning for proactive content caching in edge computing, IEEE Transactions on Parallel and Distributed Systems 33 (12) (2022) 4767–4782.
[31] X. Zhou, M. Bilal, R. Dou, J. J. Rodrigues, Q. Zhao, J. Dai, X. Xu, Edge computation offloading with content caching in 6g-enabled iov, IEEE Transactions on Intelligent Transportation Systems 25 (3) (2023) 2733–2747.
[32] X. Xu, Z. Fang, J. Zhang, Q. He, D. Yu, L. Qi, W. Dou, Edge content caching with deep spatiotemporal residual network for IoV in smart city, ACM Transactions on Sensor Networks 17 (3) (2021) 1–33.
[33] Y. Zhao, C. Liu, X. Hu, J. He, M. Peng, D. Wing Kwan Ng, T. Q. S. Quek, Joint content caching, service placement, and task offloading in uav-enabled mobile edge computing networks, IEEE Journal on Selected Areas in Communications 43 (1) (2025) 51–63.
[34] M. Xu, D. Niyato, H. Zhang, J. Kang, Z. Xiong, S. Mao, Z. Han, Sparks of generative pretrained transformers in edge intelligence for the metaverse: Caching and inference for mobile artificial intelligence-generated content services, IEEE Vehicular Technology Magazine 18 (4) (2023) 35–44.
[35] Z. Chen, S. Zhang, Z. Ma, S. Zhang, Z. Qian, M. Xiao, J. Wu, S. Lu, An online approach for dnn model caching and processor allocation in edge computing, in: 2022 IEEE/ACM 30th International Symposium on Quality of Service, IEEE, 2022, pp. 1–10.
[36] H. Djigal, J. Xu, L. Liu, Y. Zhang, Machine and deep learning for resource allocation in multi-access edge computing: A survey, IEEE Communications Surveys & Tutorials 24 (4) (2022) 2449–2494.
[37] H. Jiang, X. Dai, Z. Xiao, A. Iyengar, Joint task offloading and resource allocation for energy-constrained mobile edge computing, IEEE Transactions on Mobile Computing 22 (7) (2022) 4000–4015.
[38] O. A. Mahmood, A. R. Abdellah, A. Muthanna, A. Koucheryavy, Distributed edge computing for resource allocation in smart cities based on the IoT, Information 13 (7) (2022) 328.
[39] T. Liu, L. Tang, W. Wang, X. He, Q. Chen, X. Zeng, H. Jiang, Resource allocation in dt-assisted internet of vehicles via edge intelligent cooperation, IEEE Internet of Things Journal 9 (18) (2022) 17608–17626.
[40] Y. Cang, M. Chen, Z. Yang, Y. Hu, Y. Wang, C. Huang, Z. Zhang, Online resource allocation for semantic-aware edge computing systems, IEEE Internet of Things Journal 11 (17) (2023) 28094–28110.
[41] Q. Liu, R. Mo, X. Xu, X. Ma, Multi-objective resource allocation in mobile edge computing using paes for internet of things, Wireless networks 30 (5) (2024) 3533–3545.
[42] Z. Liu, H. Du, L. Huang, Z. Gao, D. Niyato, Joint model caching and resource allocation in generative ai-enabled wireless edge networks, in: 2025 IEEE Wireless Communications and Networking Conference, IEEE, 2025, pp. 1–6.

Wu Wen received the M.S. degree in communication and information system from Huazhong University of Science and Technology, P. R. China in 2009. He is currently an Associate Professor. He is engaged in teaching and scientific research with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, P. R. China. His main research interests include software defined networks, computer applications, and service computing.

Yibin Huang received his B.S. degree in Internet of Things Engineering from Hefei Normal University, Hefei, China, in 2023. He is currently pursuing a M.S. degree in Computer Technology from Guangzhou University, Guangzhou, China. His current research interests include cloud-edge-end collaboration and mobile edge intelligence.

Xinxin Zhao received his M.S. degree from University of Science and Technology of China, Hefei, China, in 2015. He is a senior engineer at Shandong Inspur Academy of Science and Technology, Jinan, China. His research interests include intelligent computing, edge computing.

Peiying Zhang is currently a Professor with the College of Computer Science and Technology, China University of Petroleum (East China). He received his Ph.D. in the School of Information and Communication Engineering at University of Beijing University of Posts and Telecommunications in 2019. He has published multiple IEEE/ACM Trans./Journal/Magazine papers since 2016, such as IEEE TII, IEEE T-ITS, IEEE TVT, IEEE TNSE, IEEE TNSM, IEEE TETC, IEEE Network and etc. He served as the Technical Program Committee of AAAI'24, AAAI'23, IEEE ICC'23 and IEEE ICC'22. He is the Leading Guest Editor of Drones, Mathematics, Electronics, Wireless Communications and Mobile Computing, and etc. His research interests include semantic computing and future internet architecture.

Kai Liu received the B.S. and M.S. degrees from Xidian University, Xi'an, China, in 2009 and 2012, respectively, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2016. He is currently an Assistant Research Fellow with the Beijing National Research Center for Information Science and Technology, Tsinghua University. His current research interests include space information networks and edge computing.

Guowei Shi received the M.S. and Ph.D. degrees in electrical engineering from Northwestern Polytechnical University, Xi'an, China, in 1998 and 2001, respectively.,He is a Senior Research Staff with the China Academy of Information and Communications Technology, Beijing, China. He has worked on network optimization, network architecture, mesh networking, and M2M communications. His research interests include communication theory, networking, and information theory.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: