A Diffusion-based DRL Flow Scheduling for AIGC Services in Edge Network

Huachen Jiang[1], Zhe Wang[2], Xuening Shang[1], Zhiwen Yu[1], Deyun Gao[1]*

[1] School of Electronic and Information Engineering, Beijing Jiaotong University, China
[2] The China Academy of Information and Communications Technology, China
huachenjiang@bjtu.edu.cn, wangzhe1@caict.ac.cn, xueningshang@bjtu.edu.cn, zhiwenyu@bjtu.edu.cn, gaody@bjtu.edu.cn

Abstract

With the convergence of computing and networking, Artificial Intelligence Generated Content (AIGC) services are increasingly deployed at the network edge to support low-latency and bandwidth-intensive applications. However, constrained edge bandwidth and dynamically varying traffic demands present significant challenges for efficient flow scheduling. To address these issues, this paper proposes a diffusion-based deep reinforcement learning (DRL) scheduling framework that jointly accounts for flow latency sensitivity and bandwidth constraints. By leveraging the generative exploration capability of the diffusion model into the policy optimization process, the proposed method enhances exploration efficiency and mitigates the risk of convergence to local optima. Extensive experiments demonstrate that the proposed approach significantly reduces average flow latency and improves flow completion rates compared with baseline algorithms, validating its effectiveness for adaptive and intelligent flow scheduling in edge AIGC environments.

Keywords: Artificial Intelligence Generated Content, Flow scheduling, Diffusion model, Deep reinforcement learning

1 Introduction

The rapid progress of generative AI technologies has accelerated the adoption of artificial intelligence generated content (AIGC) services in various domains, including industrial Internet of Things (IoT) and smart manufacturing [1]. Modern AIGC applications—such as image generation, multimodal interaction, and virtual scene construction—produce rich and high-volume content. With the rise of advanced generative models including ChatGPT and other AIGC models like Imagen 3 and Sora, user expectations for immediacy and interactive responsiveness have grown significantly. However, existing AIGC services often rely on centralized cloud servers, resulting in long network paths and poor responsiveness, as illustrated in Figure 1. Such limitations make cloud-only AIGC insufficient for latency-sensitive industrial applications.

*Corresponding Author: Zhe Wang; Email: wangzhe1@caict.ac.cn
DOI: https://doi.org/10.70003/160792642026052703005

To address this issue, edge computing has emerged as a promising paradigm for reducing service latency by bringing computation closer to end devices. Recent studies have explored edge–cloud collaborative AIGC task processing to improve user quality of experience (QoE) [2]. In industrial environments, AIGC services play increasingly important roles in real-time decision-making and human–machine collaboration, making low-latency, high-reliability data delivery crucial for operational performance [3]. Consequently, industrial edge networks must handle large, heterogeneous, and bursty AIGC traffic flows while operating under constrained bandwidth and dynamically varying network loads.

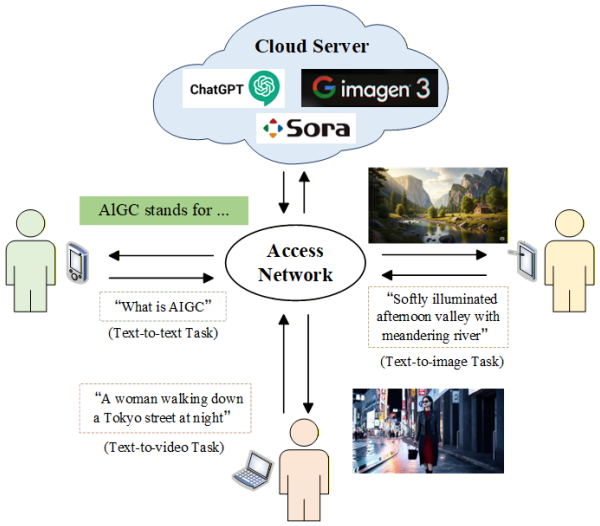


Figure 1. An illustration of existing AIGC applications which rely on centralized cloud servers

Unlike traditional IoT workloads with small and sporadic tasks, AIGC inference generates continuous, data-intensive flows spanning prompt upload, multi-stage inference, and large output delivery. Prior work shows that AIGC workloads generate new types of compute-bound and bandwidth-coupled traffic patterns [4], where latency is jointly affected by computation, transmission, and queueing. Consequently, user-perceived QoE is largely determined by end-to-end flow latency, making flow-level scheduling critical in resource-constrained edge networks.

However, efficient and reliable AIGC flow delivery in

industrial edge networks faces several challenges. First, AIGC flows show high variability in arrival rate, data volume, and latency sensitivity, leading to congestion under limited gateway bandwidth. Second, AIGC services commonly involve flows with different priorities, requiring fine-grained scheduling to satisfy multi-level QoE requirements [5]. Third, resource scarcity at the edge exacerbates queuing delays, making dynamic flow scheduling critical to prevent performance degradation.

To address the challenges of efficient flow scheduling in edge environments for AIGC, we propose an intelligent diffusion-based deep reinforcement learning (DRL) flow scheduling algorithm for edge AIGC services. The proposed method embeds a diffusion-based stochastic policy generator into the DRL actor, enabling structured exploration of scheduling actions, including flow assignment, bandwidth allocation, and resource prioritization. By combining diffusion-based exploration with long-term reward optimization, the framework adapts to bursty AIGC traffic and mitigates the risk of local optima. The proposed method enables efficient and adaptive scheduling of edge AIGC flow, bridging the last mile gap in delivering AI services to end users [6]. The key contributions of this paper are summarized as follows.

- An AIGC flow scheduling mechanism is formulated as an MDP to balance latency sensitivity and limited edge bandwidth.
- A diffusion-based DRL algorithm is proposed to improve exploration efficiency under dynamic traffic conditions.
- Simulation results with representative AIGC workloads demonstrate improved flow completion rate and reduced average latency over baselines.

The remainder of this paper is organized as follows. Section 2 reviews the related work on AIGC scheduling and edge intelligence. Section 3 presents the system model and problem formulation. Section 4 introduces the proposed diffusion-based DRL scheduling framework and details its algorithmic design. Section 5 evaluates the proposed method through extensive simulations based on representative AIGC workloads and discusses the performance improvements. Finally, Section 6 concludes the paper and summarizes the main contributions.

2 Related Work

In this section, we review four areas relevant to our study: the characteristics of AIGC services and their traffic patterns, task offloading and flow scheduling in edge computing, DRL-based network optimization methods, and the generative optimization capabilities of diffusion models. These works collectively motivate our diffusion-based DRL approach for AIGC flow scheduling.

AIGC Flows Characteristics. With the rapid advancement of large-scale generative models, AIGC services introduce new workload characteristics and traffic patterns that differ significantly from traditional applications. According to [4], AIGC services introduce a new form of compute-bound traffic pattern, where the overall latency is dominated by model transmission, preloading I/O, and inference, rather than by network bandwidth as in traditional video streaming. The study characterizes significant heterogeneity in model resource requirements leading to highly dynamic and location-dependent AIGC traffic behaviors at the network edge.

Conventional AIGC applications rely on clouds for intensive inference and content generation, but in latency-sensitive, reliability-critical edge scenarios, centralized clouds show inherent limitations such as higher communication overhead and response latency from geographic and network constraints [7]. To address these bottlenecks, an AIGC-as-a-Service paradigm was proposed, wherein application service providers deploy AI models on edge servers to offer on-demand services to end users over wireless networks, thus reducing latency and improving service flexibility [8]. Similarly, a collaborative offloading mechanism for edge terminals that supports elastic scheduling and low-latency deployment was developed, further bridging the gap between the cloud and the edge [9].

Scheduling for AIGC. In edge-based AIGC task scheduling, it is demonstrated that the actor-critic-based algorithm outperforms heuristic baselines such as random and greedy policies by improving average generated content quality while satisfying due-time requirements, leveraging deep reinforcement learning's ability to capture global scheduling features [10]. A distributed diffusion framework with inference sharing is proposed, where similar requests are clustered to share intermediate inference steps, reducing resource consumption while ensuring generation quality via a refined contrastive language-image pre-training Score metric [11]. Targeting multi-objective AIGC task scheduling in global cloud systems, the problem is modeled as a multi-agent MDP and integrated with soft actor-critic to balance policy exploration and exploitation, enabling agents to decide whether to process, postpone, or migrate jobs based on real-time energy prices and carbon intensities [12].

DRL-Based Scheduling. Due to its adaptability to optimize decision-making in complex and dynamic environments, DRL has been extensively applied to resource scheduling in edge computing. For example, a dynamic offloading framework for AIGC flows based on DRL was developed, enabling real-time adaptation of offloading policies in response to device conditions and network fluctuations [13]. These works demonstrate that RL methods significantly improve the efficiency and robustness of AIGC flow scheduling at the edge, enabling the system to cope with volatile operational environments.

Diffusion Model Optimization. Meanwhile, diffusion models, as known for their generative fidelity, pose challenges for edge deployment due to computational complexity. Recent studies have integrated diffusion models with RL to enhance AIGC flow scheduling at the edge. The collaborative multi–unmanned aerial vehicle (UAV) AIGC offloading problem was addressed by introducing a diffusion-based RL scheduling framework capable of adapting to task heterogeneity and

bandwidth variability [14]. A two-stage temporal control mechanism (T2DRL) was presented to optimize the caching and distribution of generative models, offering a novel paradigm for deployment in highly dynamic edge environments [15]. Furthermore, a latent action diffusion scheduling strategy was proposed to accelerate decision-making for AIGC service tasks, enhancing task prioritization and execution efficiency at edge nodes, and its feasibility was demonstrated through a prototype system [16]. Collectively, these efforts establish the feasibility of deploying diffusion-based AIGC models efficiently within edge computing frameworks.

3 System Design

In this section, we discuss the system model, delay model, and problem formulation of the framework.

3.1 System Model

We use artificial intelligence of things (AIoT) as a representative generative AIGC service scenario due to its large-scale sensing data and low-latency requirements [17]. We propose an edge-enabled industrial AIGC edge application framework, where heterogeneous IoT devices such as industrial monitors, cameras, AR/VR headsets and others generate text, image, and 3D scene content, as shown in Figure 2. These requests are transmitted through a gateway to edge servers hosting AIGC models, enabling low-latency and bandwidth-efficient AI processing at the network edge.

The system is a three-layer edge AIGC scheduling framework consisting of workshops (flow sources), gateways, an SDN controller and an edge server. The time is divided into a series of time slots, denoted as $\mathcal{T} = \{1,2,...,T\}$. We adopt the following notation and assumptions.

Workshops $\mathcal{W} = \{1,2,...,W\}$ denote the set of workshops. Each workshop $w \in \mathcal{W}$ generates at most one AIGC flow per time slot. A flow from workshop w at time t is characterized by the tuple:

$$f_w(t) = \left\{ D_w(t), c_w(t), p_w(t), T_w^{req}(t) \right\} \tag{1}$$

where $D_w(t)$ is data size (bits), $c_w(t)$ is required compute workload, $p_w(t)$ is priority level, and $T_w^{req}(t)$ is the latency requirement.

Let $\mathcal{G} = \{1,2,...,G\}$ denote the set of gateways. Each gateway $g \in \mathcal{G}$ provides an independent communication interface with total bandwidth capacity $R_g^{total}(t)$ and available bandwidth $R_g^{avail}(t)$ at time t. The queue of data waiting to be transmitted on the gateway is denoted as $Q_g(t)$. Each gateway periodically reports $\{R_g^{avail}(t), Q_g(t)\}$ to the controller, which uses this information for scheduling and bandwidth allocation decisions.

Edge server has available compute capacity $F^{avail}(t)$. The edge executes scheduled flows and returns completion rate and delay to the controller.

Controller collects state from gateway and issues scheduling actions via the SDN southbound interface. A scheduling action may assign flow $f_w(t)$ to a gateway g with a priority, and determine resource allocations including bandwidth portion and compute share subject to capacity constraints.

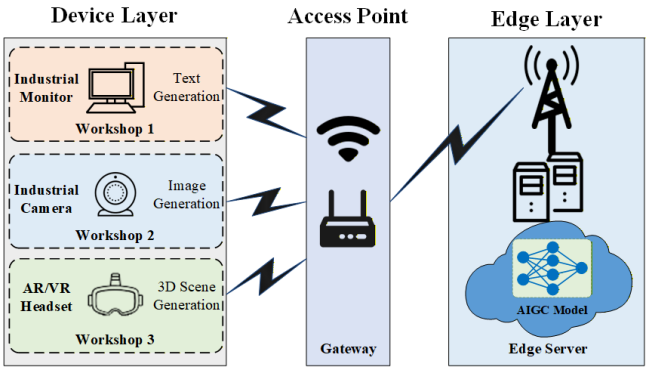


Figure 2. Overview of the proposed industrial AIGC edge application scenario

3.2 Delay Model

We decompose the actual end-to-end delay $T_w^{act}(t)$ of flow $f_w(t)$ into transmission, queuing, processing, and minor propagation components. Consequently, the total delay can be expressed as

$$T_w^{act}(t) = T_w^{tx}(t) + T_w^{q}(t) + T_w^{proc}(t) + T_w^{prop}(t) \tag{2}$$

The transmission delay $T_w^{tx}(t)$ is determined by the flow data size and the allocated bandwidth, expressed as

$$T_w^{tx}(t) = \frac{D_w(t)}{r_{w,b}(t)} \tag{3}$$

where $r_{w,b}(t)$ is the bandwidth actually allocated to flow $f_w(t)$ on gateway g. Allocation must satisfy

$$\sum\nolimits_{f \in f_w(t)} r_{w,b}(t) \le R_b^{avail}(t) \tag{4}$$

where $w_b(t)$ is the set of tasks assigned to b.

The queuing delay $T_w^{q}(t)$ represents the waiting time, given by

$$T_w^{q}(t) = \frac{Q_b(t)}{R_b^{avail}(t)} \tag{5}$$

which is a FIFO-style approximation: the new flow waits for the current backlog to be served at the gateway's available rate.

The processing delay $T_w^{proc}(t)$ denotes the time required for inference or content generation at the edge server, formulated as

$$T_w^{proc}(t) = \frac{f_w(t)}{c_w(t)} \tag{6}$$

where $c_w(t)$ is the compute resource allocated to flow $f_w(t)$ by the edge, subject to

$$\sum\nolimits_w c_w(t) \le F^{avail}(t). \tag{7}$$

The propagation delay T_w^{prop} is typically negligible in the targeted industrial edge setting and can be treated as a small constant if necessary.

To ensure the successful completion of the flow, during the process of flow transmission and processing, the actual total delay of the flow for workshops needs to satisfy the latency constraint, that is

$$T_w^{act}(t) = \frac{D_w(t)}{r_{w,b}(t)} + \frac{Q_b(t)}{R_b^{avail}(t)} + \frac{f_w(t)}{c_w(t)} \le T_w^{req}(t). \tag{8}$$

3.3 Problem Formulation

We jointly consider network bandwidth and computing load in the AIGC edge scheduling framework, aiming to minimize the overall latency violation of multi-priority AIGC flows under dynamic resource constraints.

The objective function is defined as:

$$O(t) = \alpha H_d(t) + \beta H_r(t) \tag{9}$$

where $H_d(t) = T_w^{act}(t) - T_w^{req}(t)$ denotes the difference between the actual delay and the required delay of the AIGC flows and $H_r(t) = R_b^{max}(t) - R_b^{min}(t)$ represents the maximum and minimum bandwidth utilization across gateways at time t.

The objective $O(t)$ minimizes system cost while balancing latency satisfaction and bandwidth fairness, enabling adaptive scheduling of multi-priority AIGC flows. The problem is formulated as follows:

$$\mathcal{P}\ 0{:}\min\nolimits_{\forall w,\forall b,\forall c} O(t) \tag{10}$$

$$s.t.\ T_w^{\text{act}}(t) \le T_w^{\text{req}}(t), \qquad \forall w \in W \tag{10a}$$

$$\sum\nolimits_{w \in W} a_{w,b}(t) r_{w,b}(t) \le R_b^{max}(t), \forall b \in B \tag{10b}$$

$$\sum\nolimits_{w \in W} c_w(t) \le F_c^{max}(t), \qquad \forall c \in C \tag{10c}$$

$$a_{w,b}(t) \in \{0,1\}, r_{w,b}(t) \ge 0, c_w(t) \ge 0 \tag{10d}$$

The constraints jointly ensure that flow latency requirements are satisfied while bandwidth and computing resource allocations remain within their respective capacity limits.

4 Diffusion-based Deep Reinforcement Learning Algorithm

To cope with dynamic bandwidth variations and priority-aware flow contention in edge AIGC scheduling, we develop a diffusion-based DRL under the MDP formulation. The algorithm jointly optimizes flow assignment, bandwidth allocation, and computing resource scheduling for multi-priority AIGC flows, and is executed within the controller for offline training and online policy inference.

4.1 Markov Decision Process

State Space: The system state at time t is defined to capture the environment conditions as

$$s(t) = \left\{ f_w(t), Q(t), R^{avail}(t), F^{avail}(t) \right\} \tag{11}$$

where $f_w(t)$ includes flow attributes (data size, delay requirement, priority), $R^{avail}(t)$ and $Q(t)$ denote the available bandwidth and queue state of each gateway, and $F^{avail}(t)$ represents the available computing resources at the edge server. This state captures both network and computing conditions for real-time decision-making.

Action Space: At each time slot, the agent determines the optimal scheduling action

$$a(t) = \{A^{assign}(t), R^{alloc}(t), C^{alloc}(t)\} \tag{12}$$

where $A^{assign}(t)$ specifies the gateway selection for each AIGC flow, $R^{alloc}(t)$ is the allocated bandwidth, and $C^{alloc}(t)$ represents the computing resource allocation at the selected edge gateway. This joint action design enables adaptive resource management under dynamic constraints.

Reward Function: The reward aligns with the optimization objective and penalizes delay violations or imbalance:

$$r(t) = -\left(\alpha H_d(t) + \beta H_r(t) + \kappa \phi(t)\right) \tag{13}$$

where $H_d(t)$ represents the delay deviation of AIGC flows, $H_r(t)$ denotes the bandwidth imbalance, $\phi(t)$ indicates the number of violated constraints, and κ is the penalty factor. By maximizing cumulative rewards, the agent learns policies that minimize delay and balance resource utilization.

The diffusion model introduces stochastic exploration to capture temporal variations in bandwidth and queue states, while the DRL agent generates deterministic actions guided by policy $\pi^*(s(t))$. At each step, the agent observes $s(t)$, selects $a(t)$, receives $r(t)$, and updates its value function:

$$Q\left(s\left(t\right),a\left(t\right)\right)=\mathbb{E}\left[r\left(t\right)+\gamma\max_{a'}Q\left(s\left(t+1\right),a'\right)\right] \quad (14)$$

The diffusion-based DRL algorithm adopts an actor-critic framework with double Q-networks to improve training stability and reduce overestimation bias. Each critic network $Q_{\psi_k}(s, a)$, $k\in\{1, 2\}$, estimates the expected cumulative reward of taking action a in state s.

Algorithm 1 provides the detailed procedure of the diffusion-based DRL algorithm. In Algorithm 1, s_t denotes the system state at time slot t, a_t represents the scheduling action generated by the diffusion policy, and r_t is the immediate reward. K indicates the number of diffusion denoising steps, and ϵ_k denotes Gaussian noise sampled at the k-th diffusion step.

The policy network $\pi_\theta(s)$ generates deterministic actions, while the diffusion model $D_\phi(s)$ introduces stochastic exploration to improve adaptability.

The target value for critic updates is computed as:

$$y_i=r_i+\gamma\min_{k=1,2}Q_{\psi_k'}\left(s_i',\pi_\theta\left(s_i'\right)\right) \quad (15)$$

where the smaller of the two target critics mitigates Q-value overestimation, and γ is the discount factor.

The critic parameters are optimized by minimizing the temporal-difference loss:

$$L_Q=\frac{1}{B}\sum_i\left(Q_{\psi_k}\left(s_i,a_i\right)-y_i\right)^2 \quad (16)$$

The actor network is updated via the policy gradient using the main critic Q_{ψ_1}, and target networks are softly updated for stability:

$$\psi_k'\leftarrow\tau\psi_k+\left(1-\tau\right)\psi_k',k=1,2 \quad (17)$$

By combining double Q-learning and diffusion-guided exploration, the algorithm achieves more stable value estimation and robust policy optimization under dynamic AIGC edge environments.

However, even with double critics and entropy regularization, the deterministic policy generated by the actor network may fail to explore beneficial scheduling actions when network conditions change abruptly. This motivates the incorporation of a diffusion-based stochastic policy generator, which refines the actor's output through multi-step denoising and produces diverse yet structured action candidates.

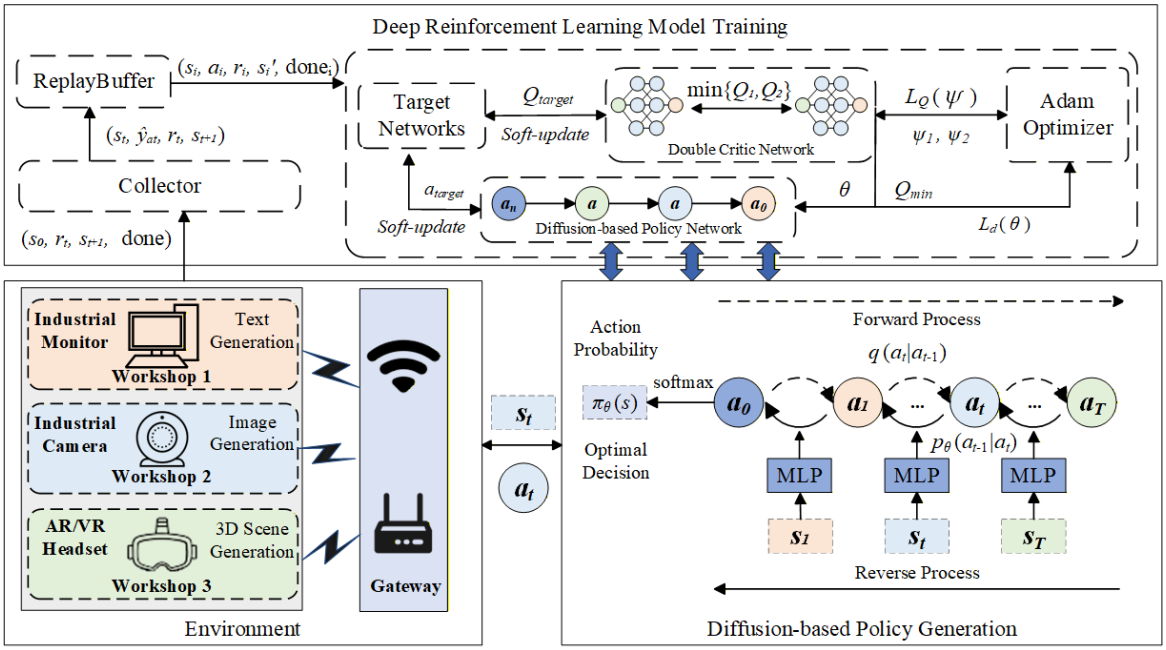


Figure 3. Diffusion-based Deep Reinforcement Learning model training framework

4.2 Diffusion-based Policy Generation

To enhance exploration and robustness in dynamic AIGC edge environments, we integrate a diffusion-based policy generation mechanism into the DRL framework, as shown in Figure 3. Instead of directly producing deterministic actions from the actor network, the diffusion model replaces the policy network and generates the policy distribution in a multi-step denoising process conditioned on the observed system state $s(t)$. This design enables the policy to model complex, multi-modal action distributions, which is essential for handling diverse service demands and fluctuating network conditions.

Forward Diffusion Process. The forward diffusion process gradually perturbs the latent action variable a_0 into a noisy representation a_t through a Markovian corruption process:

$$q\left(a_i|a_{i-1}\right)=\mathcal{N}\left(a_i;\sqrt{\alpha_i}a_{i-1},\left(1-\alpha_i\right)I\right) \tag{18}$$

where α_i controls the noise level at step i. This process ensures that after sufficiently large N, the variable a_N approaches pure Gaussian noise.

Reverse Policy Generation Process. During action generation, the diffusion-based policy replaces the traditional actor network. Starting from an initial noise vector $a_N \sim \mathcal{N}(0, I)$, the model performs N denoising steps to gradually produce a structured policy representation a_0. The overall reverse generative process follows:

$$p_\theta\left(a_{0:N}\mid s_t\right)=\mathcal{N}\left(a_N;0,I\right)\prod_{i=1}^{N}p_\theta\left(a_{i-1}\mid a_i,s_i\right) \tag{19}$$

where a_0 is the final policy logits used for action selection.

Each reverse step $p_\theta(a_{i-1}|a_i, s_i)$ follows a Gaussian transition model:

$$p_\theta\left(a_{i-1}\mid a_i,s_i\right)=\mathcal{N}\left(a_{i-1};\mu_\theta\left(a_i,s_i,i\right),\beta_i I\right) \tag{20}$$

where β_i is the variance schedule controlling the denoising strength at step i, and the mean is parameterized as:

$$\mu_\theta\left(a_i,s_i,i\right)=\frac{1}{\sqrt{\alpha_i}}\left(a_i-\frac{\beta_i}{\sqrt{1-\overline{\alpha_i}}}\epsilon_\theta\left(a_i,s_i,i\right)\right) \tag{21}$$

with $\alpha_i = 1 - \beta_i$ and $\overline{\alpha_i}=\prod_{j=1}^{i}\alpha_j$. The conditional noise estimator $\epsilon_\theta(a_i, s_i, i)$ is implemented by a multilayer perceptron (MLP), which predicts the noise component given the current state s_i and action a_i.

The variance parameter β_i is scheduled between β_{min} and β_{max}:

$$\beta_i=1-exp\left(-\frac{\beta_{min}}{N}-0.5\cdot\left(\beta_{max}-\beta_{min}\right)\cdot\frac{\left(2i-1\right)}{N^2}\right) \tag{22}$$

ensuring smooth diffusion steps across the denoising process.

At each diffusion step, the policy vector is recursively updated as:

$$a_{i-1}=\mu_\theta\left(a_i,s_i,i\right)+\sqrt{\beta_i}\epsilon,\epsilon\sim\mathcal{N}\left(0,I\right) \tag{23}$$

which can be expanded into a recursive formulation:

$$a_{i-1}=\frac{1}{\sqrt{\alpha_i}}a_i-\frac{\beta_i}{\alpha_i}\sqrt{\overline{\alpha_i}}\,\epsilon_\theta\left(a_i,s_i,i\right)+\sqrt{\beta_i}\,\epsilon \tag{24}$$

After N denoising iterations, the diffusion model outputs a_0, which represents the learned policy logits conditioned on the global state $s(t)$.

Training Objective of the Diffusion Module. Following standard diffusion models, the reverse process is trained using a noise-prediction objective:

$$L_d=\mathbb{E}\left[\left\|\epsilon-\epsilon_\theta(a_i,s_i,i)\right\|^2\right] \tag{25}$$

Minimizing this loss ensures that the model accurately reconstructs clean latent actions from noisy inputs. In the proposed framework, minimizing L_d enables the diffusion model to generate exploratory yet feasible policy candidates, which enhances policy diversity while maintaining decision consistency under dynamic bandwidth and priority constraints.

Policy Distribution and Action Sampling. The final logits a_0 are transformed into a discrete policy distribution using a SoftMax function:

$$\pi_\theta\left(s_t\right)=\frac{\exp\left(a_0^{(k)}\right)}{\sum_{k'=1}^{K}\exp\left(a_0^{(k')}\right)},k\in\left\{1,\ldots,K\right\} \tag{26}$$

where K denotes the number of possible discrete actions, corresponding to the combinations of gateway selection and priority level. An action a_t is then sampled from this distribution and executed for flow scheduling and bandwidth allocation.

By integrating the diffusion-based policy generator into the DRL, the agent benefits from stochastic exploration in the policy space, effectively capturing temporal dependencies and mitigating local optima during training. This hybrid diffusion-based DRL mechanism thus enhances robustness and adaptability in multi-priority AIGC flow scheduling across dynamic edge environments.

Algorithm 1. Diffusion-based DRL

Initialize: Policy network π_θ, Diffusion model $D\phi$, Double critic networks Q_{ψ_1}, Q_{ψ_2}, Target critics $Q_{\psi'_1}$, $Q_{\psi'_2}$, Replay buffer $\mathcal{B}$, Discount factor γ, Soft update coefficient τ, Diffusion steps T_d, Batch size B

for *episode* = 1 **to** M **do**
  Initialize environment and observe state s_0
  Sample initial action $a_0 \sim \pi_\theta(s_0)$
  Store (s_0, a_0) in trajectory τ
  while not terminal **do**
    Generate exploratory action $\widetilde{a_t} \sim D_\phi\left(s_t\right)$
    Execute $\widetilde{a_t}$, observe (r_t, s_{t+1})
    Store transition $\left(s_t,\widetilde{a_t},r_t,s_{t+1}\right)$ into $\mathcal{B}$
    if $|\mathcal{B}| \geq B$ **then**
      Sample batch $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^{B}$ from
      Compute target values

12: $y_i = r_i + \gamma \min_{k=1,2} Q_{\psi_k'}\left(s_i', \pi_\theta\left(s_i'\right)\right)$

13: Update critics

14: $\psi_k \leftarrow \psi_k - \alpha_Q \nabla_{\psi_k} \frac{1}{B} \sum_i \left(Q_{\psi_k}\left(s_i, a_i\right) - y_i\right)^2$

15: Update actor via policy gradient with diffusion-guided exploration

16: $\theta \leftarrow \theta + \alpha_\pi \nabla_\theta J(\pi_\theta)$

17: Update diffusion model parameters ϕ by minimizing diffusion loss L_d

18: Soft update target critics

19: $\psi'_k \leftarrow \tau\psi_k + (1-\tau)\,\psi'_k$, $k = 1, 2$

20: **end if**

21: $s_t \leftarrow s_{t+1}$

22: **end while**

23: Periodically evaluate π_θ and save best policy

24: **end for**

25: **return** Optimized policy π_θ^*

4.3 Computational Complexity Analysis

The computational complexity of the proposed diffusion-based DRL framework mainly consists of two parts: the diffusion-based policy generation process and the underlying actor–critic learning procedure.

For each decision step, the diffusion model performs K denoising iterations to progressively generate the action vector. Each iteration involves a forward pass of the diffusion network with complexity $O(P_d)$, where P_d denotes the number of parameters in the diffusion model. Therefore, the overall inference complexity of the diffusion-based policy generation is $O(K \cdot P_d)$.

The actor–critic learning component follows a standard DRL structure, where the actor and critic networks incur complexities of $O(P_a)$ and $O(P_c)$, respectively, with P_a and P_c representing the number of parameters of the actor and critic networks. During training, the total complexity per update step is given by

$$O\left(K \cdot P_d + P_a + P_c\right). \tag{27}$$

Compared with conventional DRL methods, the additional overhead introduced by the diffusion model mainly lies in the iterative denoising process. However, this overhead is incurred only during policy inference and can be efficiently parallelized on edge computing platforms equipped with GPUs or NPUs. Moreover, the improved exploration capability and robustness provided by the diffusion model significantly reduce performance degradation under dynamic traffic conditions, which justifies the moderate increase in computational cost.

5 Performance Evaluation

This section introduces the simulation setup, followed by a performance evaluation of the proposed diffusion-based DRL algorithm.

5.1 Simulation Setup

The key parameters for the diffusion-based design are summarized in Table 1.

Table 1. Key notations

Symbol	Value	Description
γ	0.95	Discount factor
α	0.05	Entropy temperature
τ	0.005	Soft update rate
B	512	Batch size
$\mathcal{B}$	1,000,000	Replay buffer size
T_d	10	Diffusion timesteps
h	256	Hidden layer size
M	1000	Training epochs
$N_{\text{step/epoch}}$	100	Steps per epoch
$N_{\text{step/collect}}$	1000	Steps per collection
η_{actor}	1×10^{-4}	Actor learning rate
η_{critic}	1×10^{-4}	Critic learning rate
$n-step$	3	n -step TD learning
λ_{wd}	1×10^{-4}	Weight decay

The simulation environment is implemented using the PyTorch framework and Gym–Tianshou for the reinforcement learning interaction. The experimental setup models an edge AIGC scheduling scenario consisting of eight service gateways with bandwidth capacities ranging from 50 to 200 Mbps. User flow arrivals follow a Poisson distribution with a rate parameter $\lambda = 0.001$. Each flow is characterized by three parameters: bandwidth requirement, delay requirement, and priority level. The bandwidth requirement is uniformly distributed from 1 to 20 Mbps, and the delay requirement ranges from 100 ms to 1000 ms. A total of 1000 user nodes are randomly distributed within a 100×100 area, covering three AIGC flow types. The simulation runs for 1000 time slots, each lasting 10 ms, with all random events initialized using a fixed seed for reproducibility.

5.2 Results Analysis

To evaluate the performance of the proposed scheduling strategies, we analyze the learning behaviors of four representative algorithms under identical training conditions. The following results illustrate how each algorithm adapts to dynamic AIGC flow environments over the course of training.

Figure 4 shows the average reward of four algorithms during training. Initially, diffusion-based DRL rapidly increases, surpassing others, while DQN rises moderately with fluctuations. PPO shows slower growth and larger oscillations, and Round Robin stays consistently low. As training continues, diffusion-based DRL achieves high, stable rewards, demonstrating superior learning efficiency and convergence. DQN gradually reaches a relatively high reward, whereas PPO remains lower, and Round Robin

consistently performs worst due to its static scheduling approach.

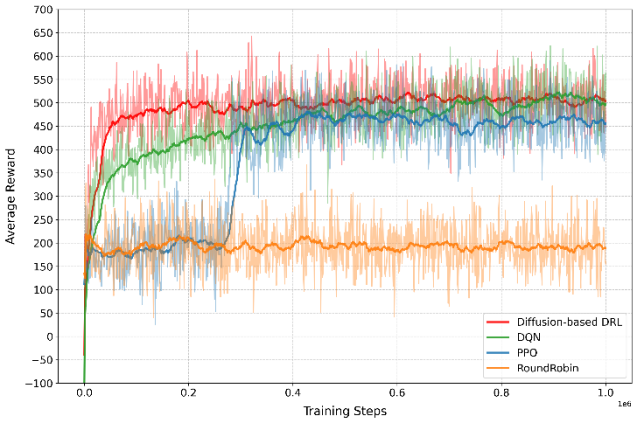


Figure 4. Comparison of learning curves among diffusion-based DRL and benchmarks

Figure 5 depicts the flow completion rate of four algorithms as total flows increase. The diffusion-based DRL rises rapidly and stabilizes at a high level. This is because its diffusion-powered strategy adapts flexibly to growing flow volumes, enabling efficient resource allocation for better completion. DQN and PPO also increase steadily and converge to high rates. Their improvement comes from reinforcement learning's policy optimization over time accumulated experience refines how they schedule flows. Round Robin stays low and even declines. Its fixed, rule-based scheduling lacks flexibility, so it cannot adapt to changing flow scales, leading to poor completion as flows pile up.

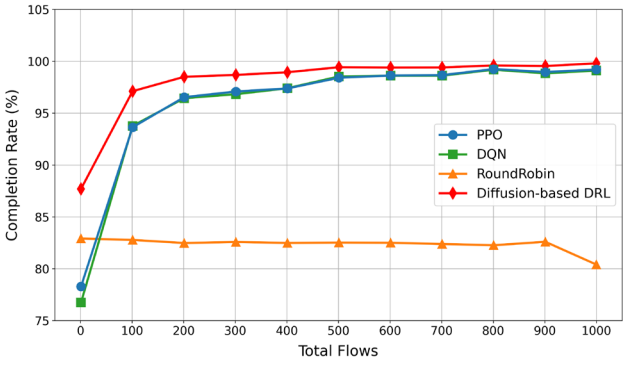


Figure 5. Comparison of flow completion rate among diffusion-based DRL and benchmarks

Figure 6 shows the average latency of four algorithms under different task loads. The Diffusion-based DRL consistently achieves the lowest latency. In the early stages, it rapidly reduces latency compared with DQN and PPO, while Round Robin maintains much higher latency due to its static scheduling strategy. As the number of tasks increases, the latency of Diffusion-based DRL continues to decrease and remains stable, showing strong adaptability to increasing workloads. DQN and PPO also improve gradually but remain inferior to the proposed method. In contrast, Round Robin exhibits consistently high latency with fluctuations, indicating limited capability in handling dynamic scheduling scenarios.

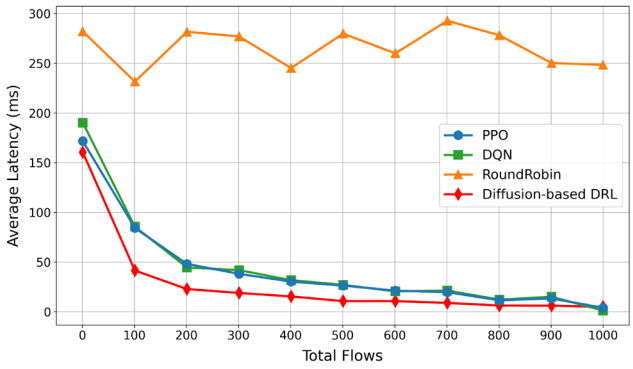


Figure 6. Comparison of average latency among diffusion-based DRL and benchmarks

6 Conclusion

This paper presents a diffusion-based DRL framework for adaptive AIGC flow scheduling in edge network. By integrating diffusion models with reinforcement learning, it jointly optimizes flow priority and bandwidth allocation under dynamic resource conditions. The diffusion model refines Gaussian noise into an optimal policy distribution, enabling stable and adaptive scheduling. Simulation results show that diffusion-based DRL effectively reduces latency and improves flow completion compared with existing DRL baselines, highlighting its potential for intelligent edge resource management.

7 Acknowledgements

This work is supported by the National Science and Technology Major Project (Grant No. 2025ZD1601100), and the National Natural Science Foundation of China (Grant No. 62394323).

References

[1] J. Zhang, D. Tao, Empowering Things with Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things, *IEEE Internet of Things Journal*, Vol. 8, No. 10, pp. 7789–7817, May, 2021. https://doi.org/10.1109/JIOT.2020.3039359

[2] R. Zhang, R. Zhou, Y. Wang, H. Tan, K. He, Incentive Mechanisms for Online Task Offloading with Privacy-preserving in UAV-assisted Mobile Edge Computing, *IEEE/ACM Transactions on Networking*, Vol. 32, No. 3, pp. 2646–2661, June, 2024. https://doi.org/10.1109/TNET.2024.3364141

[3] M. Xu, H. Du, D. Niyato, J. Kang, Z. Xiong, S. Mao, Unleashing the Power of Edge-Cloud Generative AI in Mobile Networks: A Survey of AIGC Services, *IEEE Communications Surveys & Tutorials*, Vol. 26, No. 2, pp. 1127–1170, Secondquarter, 2024. https://doi.org/10.1109/COMST.2024.3353265

[4] Y. Liang, P. Yang, Y. He, F. Lyu, Resource-Efficient

Generative AI Model Deployment in Mobile Edge Networks, *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, Cape Town, South Africa, 2024, pp. 2647–2652. https://doi.org/10.1109/GLOBECOM52923.2024.10901571

[5] X. Wang, C. Hou, C. Qiu, X. Ren, Z. Xiong, H. Yao, A Resource Management Strategy for Fluid Equilibrium in Edge-Cloud Market Supporting AIGC Services, *IEEE Transactions on Services Computing*, Vol. 18, No. 4, pp. 1922–1937, July-August, 2025. https://doi.org/10.1109/TSC.2025.3583154

[6] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing, *Proceedings of the IEEE*, Vol. 107, No. 8, pp. 1738–1762, August, 2019. https://doi.org/10.1109/JPROC.2019.2918951

[7] Z. Xu, Z. Tang, J. Lou, Z. Yao, X. Xie, T. Wang, Y. Wang, W. Jia, EAT: QoS-Aware Edge-Collaborative AIGC Task Scheduling via Attention-Guided Diffusion Reinforcement Learning, *arXiv preprint*, arXiv: 2507.10026, July, 2025. https://arxiv.org/abs/2507.10026

[8] H. Du, Z. Li, D. Niyato, J. Kang, Z. Xiong, H. Huang, Diffusion-based Reinforcement Learning for Edge-enabled AI-generated Content Services, *IEEE Transactions on Mobile Computing*, Vol. 23, No. 9, pp. 8902–8918, September, 2024. https://doi.org/10.1109/TMC.2024.3356178

[9] J. Wang, Y. Peng, X. Zhang, L. Liu, S. Mumtaz, M. Guizani, S. Dustdar, Efficient Seamless Task Offloading based on Edge-Terminal Collaborative for AIoT Elastic Computing Services, *IEEE Transactions on Services Computing*, Vol. 18, No. 5, pp. 2794–2807, September-October, 2025. https://doi.org/10.1109/TSC.2025.3592386

[10] C. Feng, Y. Zheng, Y. Xu, Online AI-Generated Content Request Scheduling with Deep Reinforcement Learning, *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Vancouver, BC, Canada, 2024, pp. 1–6. https://doi.org/10.1109/INFOCOMWKSHPS61880.2024.10620786

[11] H. Yang, Y. Hou, Y. Zheng, Z. Li, Diffusion-Type AIGC Request Scheduling with Inference Sharing, *2025 IEEE/ACM 33rd International Symposium on Quality of Service (IWQoS)*, Gold Coast, Australia, 2025, pp. 1–10. https://doi.org/10.1109/IWQoS65803.2025.11143263

[12] S. Zhang, M. Xu, W. Y. B. Lim, D. Niyato, Sustainable AIGC Workload Scheduling of Geo-Distributed Data Centers: A Multi-Agent Reinforcement Learning Approach, *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, Kuala Lumpur, Malaysia, 2023, pp. 3500–3505. https://doi.org/10.1109/GLOBECOM54140.2023.10437617

[13] X. Li, R. Deng, J. Wei, X. Wu, J. Chen, C. Yi, J. Cai, D. Niyato, X. Shen, AIGC-Driven Real-Time Interactive 4D Traffic Scene Generation in Vehicular Networks, *IEEE Network*, Vol. 39, No. 6, pp. 261–269, November, 2025. https://doi.org/10.1109/MNET.2025.3545609

[14] H. Wang, Z. Liu, X. Wang, C. Qiu, C. Zhang, W. Wang, Q. Ye, SC-TSDRL: A Cloud-Edge Collaboration Framework for Diffusion Model Inference Acceleration, *IFIP International Conference on Network and Parallel Computing*, Haikou, China, 2024, pp. 75–87. https://doi.org/10.1007/978-981-96-2864-3_7

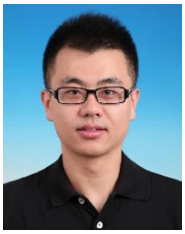
[15] Z. Liu, H. Du, X. Hou, L. Huang, S. Hosseinalipour, D. Niyato, K. B. Letaief, Two-timescale Model Caching and Resource Allocation for Edge-enabled AI-generated Content Services, *arXiv preprint*, arXiv: 2411.01458, November, 2024. https://arxiv.org/abs/2411.01458

[16] C. Xu, J. Guo, W. Lin, H. Zou, W. Fan, T. Wang, X. Chu, J. Cao, Accelerating AIGC Services with Latent Action Diffusion Scheduling in Edge Networks, *arXiv preprint*, arXiv: 2412.18212, December, 2024. https://arxiv.org/abs/2412.18212

[17] 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*, Vol. 4, pp. 2952–2971, 2023. https://doi.org/10.1109/OJCOMS.2023.3320646

Biographies

Huachen Jiang is a Master's student at the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. His research interests include network and computing convergence, and flow scheduling.



Zhe Wang is a Senior Engineer with the China Academy of Information and Communications Technology, Beijing, China. His current research interests include edge computing, Industrial Internet.

Xuening Shang is a Ph.D. student with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. His research interests include network and computing convergence, and the next-generation network technology.

Zhiwen Yu is a Ph.D. student with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. His research interests include Internet of agents, generative AI, and resource allocation.

Deyun Gao is a Full Professor with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. His research interests include Internet of Things, vehicular networks, and next-generation Internet.