future internet

MDPI

Article

The Age of Generative AI Model for Fresh Industrial AIGC Services: A Hybrid-Action Multi-Agent DRL Approach

Wenjing Li [1], Ni Tian [1,*] and Long Zhang [1,2]

[1] School of Information and Electrical Engineering, Hebei University of Engineering, Handan 056038, China
[2] Chongqing Key Laboratory of Mobile Communications Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
* Correspondence: tianni@hebeu.edu.cn

Abstract

To meet the growing demand for autonomous decision-making and real-time optimization in industrial manufacturing, integrating Artificial Intelligence-Generated Content (AIGC) services with Industry 5.0 can enable real-time industrial intelligence. The effectiveness of a generative model is closely related to the current state of the production environment. However, existing studies often ignore the dynamic temporal relationship between generative models and production environments, especially in industrial scenarios with large model transmission delays and random AIGC task arrivals. Therefore, we define a novel metric, namely the Age of Model (AoM), to measure the freshness of generative models with respect to current industrial tasks. We then formulate an average-AoM-minimization problem that jointly considers LoRA-based fine-tuning, wireless transmission and resource allocation. To solve this problem, we propose a Hybrid-Action Multi-Agent Proximal Policy Optimization (HA-MAPPO) algorithm. The proposed algorithm follows the centralized training and decentralized execution (CTDE) paradigm and introduces a Main-Agent Priority State Strategy to support coordinated training and independent execution. In addition, a multi-head output structure is designed to handle the hybrid-action space, which includes discrete fine-tuning association decisions and continuous transmission resource allocation. Simulation results show that the proposed scheme outperforms all benchmark methods. Specifically, the cumulative rewards are improved by approximately 11.13%, 20.32%, 36.61%, and 38.78% compared with the four benchmark algorithms, respectively. These results demonstrate that the proposed scheme can significantly reduce the average AoM while providing high-quality and timely industrial AIGC services.

Keywords: AI-Generated Content (AIGC); Age of Model (AoM); AIGC service; Industry 5.0; hybrid-action space; multi-agent deep reinforcement learning

Academic Editor: Ivan Serina

Received: 12 February 2026
Revised: 11 March 2026
Accepted: 16 March 2026
Published: 23 March 2026

1. Introduction

The emergence of Industry 5.0 has driven a major shift from traditional automation-centric factories to intelligent manufacturing ecosystems [1,2]. In this context, close collaboration between smart machines (SMs) and edge servers (ESs) enables real-time industrial intelligence and supports advanced human-centric interaction [3–5]. A key enabler in this evolving paradigm is Generative Artificial Intelligence (GAI), which has become a transformative technology for delivering Artificial Intelligence-Generated Content (AIGC) services. By leveraging advanced generative architectures, such as Generative Adversarial Networks

(GANs), Transformers, diffusion models, and large language models (LLMs), GAI can autonomously generate diverse multimodal content, including text, high-fidelity images, and predictive video sequences [6–8]. In industrial environments, GAI-driven AIGC services enable edge devices to generate on-demand operational guidance, optimized maintenance strategies, and context-aware production content [9]. These capabilities improve situational awareness and enhance workflow efficiency in complex production environments.

In industrial environments, a model's ability to generate relevant content depends not only on the quality of its initial training but also on the current production environment [7]. However, providing high-quality and timely industrial AIGC services at the edge remains challenging. The environment and task arrivals vary over time. Moreover, foundation models are typically large, and model transmission may introduce substantial latency, sometimes even exceeding the fine-tuning time. In addition, spectrum resources in factories are limited, and co-channel interference affects both uplink and downlink transmissions. These factors tightly couple model association, transmission, and computation queues, and the resulting delays directly degrade the quality of AIGC services.

1.1. Related Works

1.1.1. GAI for Industrial Systems

Recent studies have begun to explore the potential of GAI for industrial systems. In [4], the authors proposed a framework for industrial defect detection and visual question answering based on multimodal large language models. A multi-layer network architecture was developed in [10] to support industrial AIGC applications by integrating deterministic communication, computing, and control functions. To improve execution efficiency, the authors in [11] designed a collaborative edge offloading framework that jointly optimizes communication and computation resources for industrial AIGC tasks. Although these studies lay the foundation for industrial AIGC services, they do not adequately address real-time model updating and adaptive evolution in dynamic industrial environments.

1.1.2. AIGC Services in Wireless Networks

AIGC service provisioning in wireless networks has attracted increasing attention, with most studies focusing on performance optimization through resource allocation. In [12], the authors proposed a cloud-edge collaborative framework that combines large-scale cloud-based AI models with lightweight local GAI models at the edge, and developed distributed training and task-oriented deployment schemes. In [13], a dynamic AIGC service provisioning method based on diffusion models was introduced to balance quality of experience and resource efficiency by jointly optimizing inference iterations and transmission power. In [14], the authors presented a framework that is aware of both communication and computation resources, and studied the trade-off between service latency and generation quality. However, most existing studies focus mainly on resource allocation during the inference stage of AIGC tasks, while overlooking the critical role of adaptive fine-tuning throughout the full AIGC service lifecycle.

1.1.3. Multi-Agent Reinforcement Learning for Hybrid-Action Spaces

Deep reinforcement learning has been widely used to solve complex optimization problems with high-dimensional states and coupled decisions, many of which involve hybrid discrete–continuous action spaces. To address such mixed action spaces, Ref. [15] developed an H-SAC framework for service provisioning in industrial edge networks. Moreover, multi-agent reinforcement learning is well-suited to distributed collaborative decision-making and can reduce reliance on centralized control. For example, Ref. [16] applied a multi-agent PPO framework to coordinated trajectory design for unmanned aerial vehicles, while Ref. [17] used an AoI-aware MAPPO method to achieve coordinated search

and data collection for autonomous underwater vehicle groups. These studies provide useful methodological support for our work.

1.2. Contributions and Organization

Motivated by these limitations, this paper proposes a novel metric, termed Age of Model (AoM), to quantify the freshness of generative models for industrial tasks. Based on this metric, we formulate a joint optimization problem that jointly considers model fine-tuning association and transmission resource allocation. To solve this problem, we further propose a Hybrid-Action Multi-Agent Proximal Policy Optimization (HA-MAPPO) framework, which effectively handles the coupled decision-making process involving model fine-tuning, task scheduling, and communication resource allocation. The specific contributions of this work are summarized as follows:

- We propose a novel freshness metric, namely Age of Model (AoM), to characterize the timeliness of generative models in dynamic industrial environments. Based on this metric, we formulate an average-AoM-minimization problem that jointly considers LoRA-based model fine-tuning, wireless transmission, and communication resource allocation.
- To solve the formulated problem, we propose a Hybrid-Action Multi-Agent Proximal Policy Optimization (HA-MAPPO) framework. This framework effectively addresses the coupled decision-making process involving model fine-tuning, task scheduling, and transmission resource allocation in industrial AIGC services.
- Within the centralized training and decentralized execution (CTDE) paradigm, we further design a Main-Agent Priority State Strategy to generate isomorphic observations for homogeneous agents. In addition, a multi-head output structure is introduced to model the hybrid-action space consisting of discrete fine-tuning association decisions and continuous power allocation.
- Simulation results demonstrate that the proposed HA-MAPPO algorithm significantly outperforms benchmark schemes in terms of cumulative reward and AoM performance, thereby validating its effectiveness in providing high-quality and timely industrial AIGC services.

This paper is structured as follows. In Section 2, we presents the system model for AIGC services and the operational workflow of LoRA-based fine-tuning. In Section 3, we formulate the AoM-minimization problem as an MDP. Section 4 describes the proposed HA-MAPPO algorithm to optimize AoM across the system. Section 5 provides the simulation results, followed by conclusions in Section 6.

2. System Model

2.1. System Overview

To meet the demands of intelligent industrial production and provide real-time and high-quality AIGC services, we consider that $\mathcal{K} = \{1, 2, \cdots, K\}$ ESs and $\mathcal{N} = \{1, 2, \cdots, N\}$ SMs are randomly distributed across the factory. Each SM is equipped with multiple AIGC models trained from a predefined model set $\mathcal{M} = \{1, 2, \cdots, M\}$, enabling local generation of content tailored to dynamic production needs. This assumption is consistent with the actual deployment form of the industrial field. SM is responsible for real-time task reasoning and execution, and ES is responsible for computing power-intensive model fine-tuning. This approach can not only effectively reduce the computing load of SM, but also enable centralized and standardized operation and maintenance management of the system.

Let the state of the model type set $S_n \subseteq \mathcal{M}$ be deployed on SM n as $\chi_n = \{(m, t_{n,m}) | m \in S_n\}$, where $t_{n,m}$ represents the last update time of model m on SM n.

Similarly to [15,18,19], a time-slot frame structure is adopted to evenly divide the entire AIGC service period T into multiple time slots, indexed by $t \in \mathcal{T} = \{1, 2, \cdots, T\}$, with each having a duration of τ. Within a single time slot, the channel state and system load are assumed to be approximately constant. While in different time slots, the production environment, business arrival patterns, and link conditions can change dynamically over time. At the beginning of each time slot t, SMs perform the AIGC task scheduling according to the needs of the smart manufacturing system. The set of all AIGC tasks generated at time t is denoted as

$$\mathcal{G}(t) = \{(n, m, j) | n \in \mathcal{N}, m \in S_n, j \in \{1, \cdots, \mathrm{Num}_{n,m}(t)\}\}, \tag{1}$$

where (n, m, j) denotes the j-th task generated by SM n that requires model m and $\mathrm{Num}_{n,m}(t)$ denotes the number of tasks generated by SM n that requires model m during that time slot.

2.2. AoM Metric for Model Freshness

The traditional freshness indicators, such as Age of Information (AoI), focus on the timeliness of the original industrial data. It only considers data transmission and cache latency, but does not involve the model adaptation and fine-tuning process [20]. In contrast, in the evaluation metrics for generative models, such as Inception Score (IS), Fréchet Inception Distance (FID), and R-Precision [21–23], only the quality and diversity of the generated content were evaluated, and the temporal correlation between the model and the dynamic industrial tasks was not captured. However, in an industrial environment, the value of generative models is not static but is associated with the temporal state of the production environment. To address this limitation, we propose a novel metric, Age of Model (AoM), which is used to measure the timeliness of the model when processing AIGC tasks in dynamic environments.

Definition 1. *$AoM_{(n,m,j)}(t)$ is defined as the freshness of model m when serving an AIGC task $(n, m, j) \in \mathcal{G}(t)$ at time slot t. It is quantified as the time difference between the current execution time t and the most recent fine-tuning timestamp $t_{n,m}$ of the model, expressed as*

$$\mathrm{AoM}_{(n,m,j)}(t) = t - t_{n,m}, \ (n, m, j) \in \mathcal{G}(t). \tag{2}$$

A larger AoM indicates that the model is more outdated, potentially resulting in generated content that poorly reflects the current production environment.

From Equation (2), it can be seen that AoM is directly determined by the latest fine-tuning time slot $t_{n,m}$. Once a fine-tuning update is completed, $t_{n,m}$ is refreshed to the new update completion time. Therefore, increasing the rate of completed model updates tends to reduce $t - t_{n,m}$ and thus lower the AoM used for AIGC tasks. However, the limited availability of real-time industrial data and being constrained on SM computational capacity often make local fine-tuning infeasible. In contrast, ESs possess significantly stronger computational resources and can also collect real-time production data from the factory floor, making them naturally suited for computing fine-tuning tasks.

Therefore, as shown in Figure 1, model fine-tuning is offloaded to ESs, and the fine-tuning decision process is executed sequentially through the following three phases:

(i) Model Association and Upload: The fine-tuning task is associated by selecting model m from SM n and assigning it to ES k; once the association is determined, the selected model is uploaded from the SM to the corresponding ES.

(ii) Fine-Tuning Training: The associated model is fine-tuned on the ES using real-time industrial data; all fine-tuning tasks are processed in a First-In–First-Out (FIFO) manner according to the computation queue of the ES.

(iii) Model Download and Update: Upon completion of LoRA fine-tuning, the learned parameters are transmitted back to the originating SM via the downlink and applied to update the corresponding model. The local model state is then refreshed to record the latest update time.

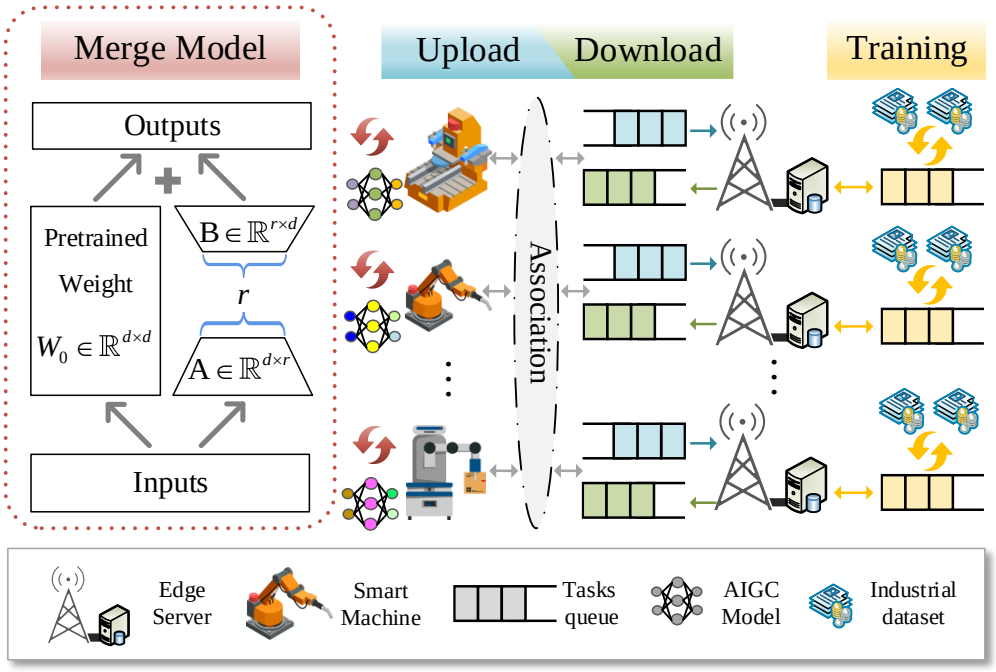


Figure 1. System architecture and workflow of LoRA-based model fine-tuning. Different colors are used to distinguish functional modules and workflow stages: pink denotes model merging, blue and green denote upload and download processes, respectively, and yellow denotes the training stage. The blue and green task queues represent uplink and downlink transmission queues, respectively.

2.3. Fine-Tuning Process with LoRA

At the end of the previous section, we outlined the three sequential stages involved in model fine-tuning. However, due to the large scale of AIGC models, both transmission and computation introduce significant latency overhead in resource-constrained industrial environments. To address this issue and improve the overall efficiency of model updates, we adopt the LoRA technique.

LoRA enables fine-tuning by freezing the majority of the model parameters and introducing a small number of trainable low-rank matrices. This approach significantly reduces the amount of data that must be transmitted during both the upload and return stages, while also reducing the computational load required during training. In the following, we provide a detailed description of the model fine-tuning process with LoRA.

2.3.1. Model Association and Upload

Given limited computing and transmission resources, we define a binary association variable $\alpha_{n,m}^{k}(t) \in \{0,1\}$, where $\alpha_{n,m}^{k}(t) = 1$ indicates that model m on SM n is selected for fine-tuning on ES k at time slot t. Upon successful association, the full model parameters $D_{n,m}^{\text{full}}$ are transmitted to ES k, and the task is added to the uplink transmission queue $Q_k^{\text{UL}}(t)$. The total uplink delay is given by

$$T_{n,m,k}^{\text{UL}} = T_{n,m,k}^{\text{wait,UL}} + \left\lceil \frac{D_{n,m}^{\text{full}}}{R_{n,k}^{\text{UL}}(t) \cdot \tau} \right\rceil, \tag{3}$$

where $T_{n,m,k}^{\text{wait,UL}}$ denotes the waiting delay in the uplink transmission queue, $R_{n,k}^{\text{UL}}(t)$ denotes the uplink transmission rate at time slot t and $\lceil \cdot \rceil$ represents the ceiling function used to round up to the nearest integer.

2.3.2. Fine-Tuning Training

Once the full model is uploaded, it enters the fine-tuning task queue $Q_k^{\text{FT}}(t)$ of ES k. To ensure fairness and handle resource contention, we also adopt a FIFO scheduling strategy. Unlike full-parameter training, LoRA freezes the original weight matrix $W_0 \in \mathbb{R}^{d\times d}$ and injects trainable low-rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$, where $r \ll d$. Since only the matrices A and B are optimized, the number of trainable parameters is significantly reduced from $\mathcal{O}(d^2)$ to $\mathcal{O}(2dr)$, making the fine-tuning process highly efficient.

Consequently, the computational latency required for fine-tuning the model m on SM n on ES k is

$$T_{n,m,k}^{\text{FT}} = T_{n,m,k}^{\text{wait,FT}} + \left\lceil \frac{E \cdot S_n^m \cdot C_n^{\text{LoRA}}}{f_k} \right\rceil, \tag{4}$$

where $T_{n,m,k}^{\text{wait,FT}}$ denotes the waiting delay incurred before the fine-tuning process begins, S_n^m is the number of industrial data samples used, E is the number of training epochs (typically 1), C_n^{LoRA} denotes the CPU cycles required for LoRA training per sample, and f_k is the computational capacity of ES k.

2.3.3. Model Download and Update

After fine-tuning, only the LoRA adaptation parameters (A, B) need to be transmitted back to the SM. Hence, the size of model m from SM n to be transmitted over the downlink is given by

$$D_{n,m}^{\text{LoRA}} = ||A|| + ||B|| \ll D_{n,m}^{\text{full}}, \tag{5}$$

and the downlink delay is given by

$$T_{n,m,k}^{\text{DL}} = T_{n,m,k}^{\text{wait,DL}} + \left\lceil \frac{D_{n,m}^{\text{LoRA}}}{R_{k,n}^{\text{DL}}(t) \cdot \tau} \right\rceil, \tag{6}$$

where $T_{n,m,k}^{\text{wait,DL}}$ denotes the waiting delay in the downlink transmission queue $Q_k^{\text{DL}}(t)$ and $R_{k,n}^{\text{DL}}(t)$ denotes the downlink transmission rate at time slot t.

Upon successful reception, SM n applies (A, B) to update the corresponding model m by merging the LoRA-induced weight increment δW into the pretrained weights W_0. The effective weight used for subsequent inference is

$$W = W_0 + \Delta W = W_0 + BA, \tag{7}$$

where $\Delta W = BA$ represents the LoRA-induced weight update.

Finally, the local model state is refreshed by recording the latest update time. Specifically, when model m of SM n is selected for fine-tuning on ES k at time slot t, the completion time of the update process, based on Equations (3), (4) and (6) can be expressed as

$$t_{n,m,k}^{\text{update}} = t + T_{n,m,k}^{\text{UL}} + T_{n,m,k}^{\text{FT}} + T_{n,m,k}^{\text{DL}}. \tag{8}$$

Accordingly, SM n updates the local model state $(m, t_{n,m}) \in \chi_n$ by setting the latest fine-tuning timestamp to $t_{n,m} = t_{n,m,k}^{\text{update}}$ for the corresponding model m.

2.4. Communication Model

Due to the large size of AIGC models and limited wireless resources, communication delay becomes a major bottleneck in the fine-tuning process. To address this, we adopt a fixed-band wireless system with predefined uplink and downlink resources.

Specifically, the total bandwidth is partitioned into $2K$ orthogonal sub-bands, including K uplink sub-bands and K downlink sub-bands. Each ES is assigned one uplink and one downlink sub-band. Uplink and downlink transmissions are orthogonal, while co-directional transmissions (e.g., multiple uplinks) may suffer from mutual interference due to spectrum reuse. This design conforms to the common concurrent transmission and reuse patterns found within the factory environment [24].

Let W^{UL} and W^{DL} denote the fixed bandwidths of the uplink and downlink sub-bands allocated to ES, respectively. Then, the uplink transmission rate from SM n to ES k at time t is given by

$$R_{n,k}^{\mathrm{UL}}(t) = \frac{W^{\mathrm{UL}}}{K} \cdot \log_2\left(1 + \frac{P_n(t)h_{n,k}}{\sigma^2 + I_{n,k}^{\mathrm{UL}}(t)}\right), \tag{9}$$

similarly, the downlink transmission rate from ES k to SM n is

$$R_{k,n}^{\mathrm{DL}}(t) = \frac{W^{\mathrm{DL}}}{K} \cdot \log_2\left(1 + \frac{P_k(t)h_{k,n}}{\sigma^2 + I_{n,k}^{\mathrm{DL}}(t)}\right), \tag{10}$$

where $P_n(t)$ and $P_k(t)$ are the transmit powers of SM n and ES k, $h_{n,k}$ and $h_{k,n}$ are the channel gains, σ^2 is the noise power, and $I_{n,k}^{\mathrm{UL}}$ and $I_{n,k}^{\mathrm{DL}}$ denote the co-directional interference within uplink and downlink sub-bands, respectively.

As introduced in Section 2.3, both $R_{n,k}^{\mathrm{UL}}(t)$ and $R_{k,n}^{\mathrm{DL}}(t)$ are critical for computing the model update delay, which in turn affects the AoM of the AIGC tasks.

Let $|\mathcal{G}(t)|$ denote the total number of tasks generated at time slot t, which is given by

$$|\mathcal{G}(t)| = \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{S}_n} \mathrm{Num}_{n,m}(t). \tag{11}$$

Therefore, according to Equation (2), the average AoM of the tasks generated at time slot t can be expressed as

$$\bar{A}(t) = \frac{1}{|\mathcal{G}(t)|} \sum_{(n,m,j)\in\mathcal{G}(t)} \mathrm{AoM}_{n,m,j}(t). \tag{12}$$

We aim to minimize the average AoM of all AIGC tasks by jointly fine-tuning task scheduling and transmission resource allocation (i.e., transmit power at both SMs and ESs), subject to the constraints of model fine-tuning requirements, bandwidth allocation, and transmit power limits. However, as multiple SMs dynamically generate AIGC tasks and compete for shared wireless and ES computing resources, the coupling among model fine-tuning, model transmission, and bandwidth allocation gives rise to a high-dimensional combinatorial decision space that is intractable for conventional optimization methods. Therefore, we adopt a DRL-based solution.

3. MDP Problem Formulation

The fine-tuning process of the AIGC model involves a series of decision-making processes among multiple ESs. Each ES dynamically selects the model for fine-tuning and allocates resources based on the decision. Since each ES adopts a unified agent architecture, they are all modeled as independent agents, interacting with the environment and making decentralized decisions. Additionally, the random generation of tasks is influenced by the constantly changing industrial environment. That is, the future system state depends only

on the current state and the actions taken by the ES, thus satisfying the Markov property. Therefore, the problem of minimizing the total AoM of all AIGC tasks can be expressed as a Markov Decision Process (MDP). In the MDP representation, the definitions of the state space, action space and reward function are as follows:

3.1. State Space

At each time slot, the state observed by agent k consists of a public state and a private state, denoted by $\mathbf{o}^{\text{pub}}(t)$ and $\mathbf{o}^{\text{pri}}(t)$, respectively.

3.1.1. Public State

Each ES can observe the current model states $\chi_n = \{(m, t_{n,m})\}$ for all SMs $n \in \mathcal{N}$. Accordingly, the public state at time slot t can be defined as

$$\mathbf{o}^{\text{pub}}(t) = \{(m, t_{n,m})\}, \forall m \in S_n, n \in \mathcal{N}. \tag{13}$$

3.1.2. Private State

Each ES k can only access its own internal state, including the uplink transmission queue $Q_k^{\text{UL}}(t)$, fine-tuning computation queue $Q_k^{\text{FT}}(t)$, and downlink transmission queue $Q_k^{\text{DL}}(t)$. Thus, the private state of ES k at time slot t is formulated as

$$\mathbf{o}_k^{\text{pri}}(t) = \left\{ Q_k^{\text{UL}}(t), Q_k^{\text{FT}}(t), Q_k^{\text{DL}}(t) \right\}. \tag{14}$$

Therefore, the environment state observable at time slot t can be expressed as

$$\mathbf{o}(t) = \left[\mathbf{o}^{\text{pub}}(t), \mathbf{o}_1^{\text{pri}}(t), \mathbf{o}_2^{\text{pri}}(t), \ldots, \mathbf{o}_k^{\text{pri}}(t) \right]. \tag{15}$$

3.2. Action Space

At each time slot t, each agent k selects a hybrid action consisting of (i) the binary model association decisions $\alpha_{n,m}^k(t) \in \{0, 1\}$ and (ii) the continuous transmit power control including the downlink power $P_k(t)$ of ES k and the uplink transmit powers of the SMs associated with ES k, denoted by $\dot{P}_k(t)$, which is defined as

$$\dot{P}_k(t) \triangleq \left\{ P_n(t) \middle| \exists m \in S_n,\ \alpha_{n,m}^k(t) = 1 \right\}. \tag{16}$$

Therefore, the action of agent k at time slot t is defined as

$$\mathbf{a}_k(t) = \left[\left\{ \alpha_{n,m}^k(t) \right\}, \{P_k(t)\}, \{\dot{P}_k(t)\} \right], \forall m \in S_n, n \in \mathcal{N}. \tag{17}$$

3.3. Reward Function

To drive the learning process toward minimizing the AoM, we design the reward to be inversely proportional to the total AoM of the tasks generated at time slot t. Specifically, the reward of agent k is defined as

$$r_k(t) = \zeta \frac{|\mathcal{G}(t)|}{\sum_{(n,m,j) \in \mathcal{G}(t)} e\text{AoM}_{n,m,j}(t) + \epsilon} = \zeta \frac{1}{\bar{A}(t) + \epsilon / |\mathcal{G}(t)|}, \tag{18}$$

where $\zeta > 0$ is a reward scaling coefficient and $\epsilon > 0$ is a small constant to avoid numerical instability. Although the reward is expressed in terms of AoM, it essentially balances the competitive factors within the system. Discrete association decisions influence queue congestion and waiting delays by selecting fine-tuning tasks. Continuous power-control decisions change the transmission duration by affecting the transmission link rate. These two actions jointly affect the average change in AoM of the tasks. Therefore, designing

the reward in this way can encourage the agents to reduce AoM and make a principled trade-off between updating enthusiasm and resource contention.

Given the per-slot reward defined in Equation (18), our objective is to learn an optimal policy π^* that maximizes the expected discounted cumulative reward

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \sum_{k\in\mathcal{K}} r_k(t)\right], \tag{19}$$

where $\gamma \in [0, 1]$ is the discount factor.

As the number of ES agents increases, the dimensions of observable states and joint actions both increase. If enumeration is used, the dimensions of the state space and action space will grow linearly. High-dimensional state space and action space will affect the learning effect of the agent. Therefore, we adopt a multi-agent reinforcement learning method to decouple the action space. At the same time, we propose the Main-Agent Priority State Strategy to reduce the impact of state space growth. In addition, we use CTDE and shared strategies to improve the scalability and operability of MDP modeling. The specific content will be introduced in Section 4.

4. Hybrid-Action Multi-Agent Proximal Policy Optimization

Traditional reinforcement learning typically follows a single-agent paradigm, in which a single policy outputs the joint system action based on the global state [15,25]. However, this approach has clear limitations in the considered scenario. First, it relies on a central controller to interact with all ESs and collect both public and private state information. In industrial environments, this is difficult to implement reliably due to communication overhead, system heterogeneity, and data compliance requirements. Second, centralized decision-making exposes the private information of ESs, which weakens privacy protection and local autonomy. Therefore, we adopt a multi-agent reinforcement learning framework in which each ES acts as an independent agent and makes decisions based on local information. This design reduces reliance on the central controller and improves data privacy. Moreover, as the number of agents increases, the dimensions of both the state space and the hybrid-action space grow linearly. As a result, centralized PPO becomes increasingly difficult to optimize, which limits scalability.

To solve the AoM-minimization MDP formulated in Section 3, we propose a Hybrid-Action Multi-Agent Proximal Policy Optimization (HA-MAPPO) algorithm. In the proposed framework, each ES outputs both discrete model-association decisions and continuous power-control actions, which jointly affect the model fine-tuning process. However, most existing multi-agent PPO variants assume purely continuous action spaces [26]. In contrast, the proposed HA-MAPPO has three main advantages: (i) it jointly parameterizes discrete and continuous actions within a unified policy; (ii) it supports coordination among homogeneous agents through parameter sharing under the CTDE paradigm; and (iii) it introduces a Main-Agent Priority State Strategy to remove ambiguity in action attribution. These designs make the proposed method scalable and well-suited to the problem considered in this paper.

4.1. CTDE-Based Hybrid-Action Framework

As shown in Figure 2, we propose a hybrid-action, homogeneity multi-agent PPO framework to solve the above reward-maximization problem. All ESs are modeled as homogeneity agents (identical decision structure and action semantics), hence we adopt a parameter-sharing scheme and learn a single policy network $\pi(\mathbf{a}|\mathbf{o})$ shared by all agents.

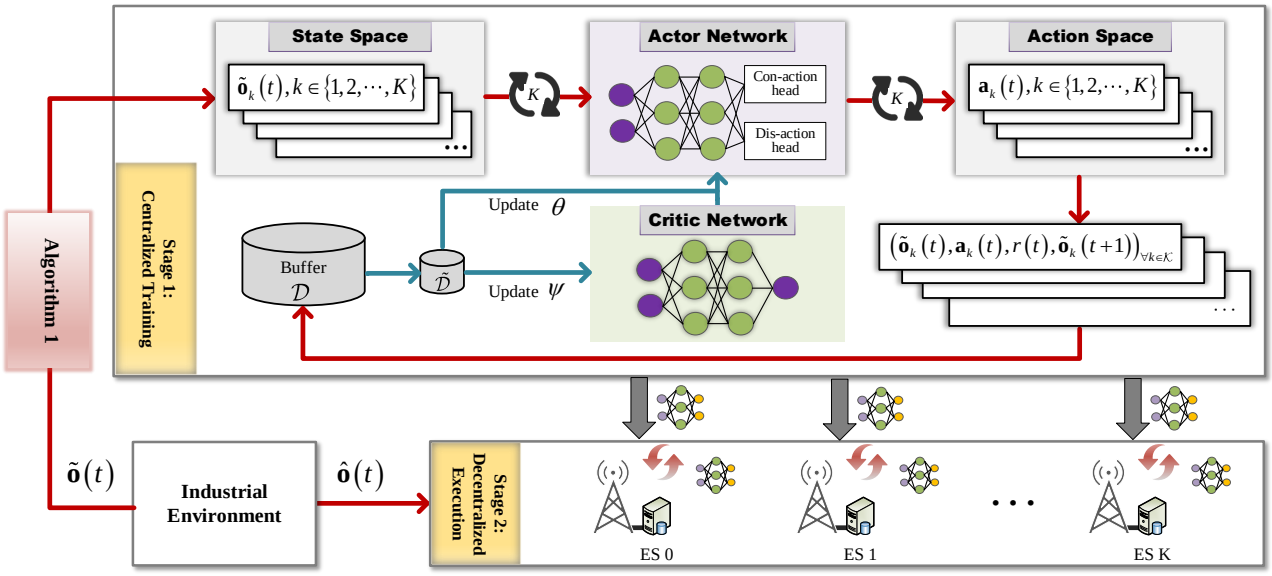


Figure 2. Architecture of the HA-MAPPO framework for industrial AIGC service provision. The red lines indicate the actor exploration process, while the blue lines indicate the learning optimization process.

To improve training stability under multi-agent non-stationarity while preserving decentralized decision making at runtime, we employ the CTDE paradigm. As can be seen in Figure 2 that the proposed HA-MAPPO follows a two-stage CTDE workflow. In the centralized training phase, the learning control center observes the public state $\mathbf{o}^{\text{pub}}(t)$ and all ES private states $\{\mathbf{o}_k^{\text{pri}}(t)|\forall k \in \mathcal{K}\}$. It constructs the environmental state input $\tilde{\mathbf{o}}_k(t)$ and generates hybrid actions $\mathbf{a}_k(t)$ through the forward pass of the shared actor. This process is repeated at each time slot t to obtain the executable joint action $\mathbf{a}(t) = \mathbf{a}_k(t), k \in \mathcal{K}$, which is applied to the industrial environment. In the distributed execution phase, each ES makes independent decisions only based on its local observable information.

Moreover, since the action space is hybrid, we design the shared policy network with a shared backbone and two output heads (As shown in Figure 2 of the actor network). Specifically, the discrete head is responsible for association-related decisions, while the continuous head outputs the power-control variables. This architecture enables the policy to jointly generate the discrete model-association variables and the continuous transmit power-control action. The detailed design is presented in Section 4.3.

4.2. Multi-Agent Observation to Isomorphic Input

In conventional multi-agent state processing, the ordering of agent states in the network input is often stochastic or fixed arbitrarily [16,17]. However, under the CTDE paradigm, in each forward pass, the shared policy outputs the action for only one agent (termed the main agent). Therefore, an appropriate input structure is needed, such that the output actions can be clearly associated with the intended agent.

To this end, we adopt a Main-Agent Priority State Strategy, as shown in Algorithm 1. For any selected main agent k, we construct the network input by placing its private state at a fixed position, and then appending the private states of the remaining agents. Specifically, the processed input for agent k at time slot t is given by

$$\tilde{\mathbf{o}}_k(t) = \left[\mathbf{o}^{\text{pub}}(t),\ \mathbf{o}_k^{\text{pri}}(t),\ \mathbf{o}_{\sigma_k(2)}^{\text{pri}}(t),\ \ldots,\ \mathbf{o}_{\sigma_k(|\mathcal{K}|)}^{\text{pri}}(t)\right], \tag{20}$$

where $\sigma_k(\cdot)$ denotes a permutation of agent indices with $\sigma_k(1) = k$. With this construction, the policy network always interprets the first private block as the main agent's local information, which facilitates parameter sharing and avoids ambiguity in action attribution.

Algorithm 1: Main-Agent Priority State Strategy.

Initialize: Main-agent state $\mathbf{o}_{\text{main}} = \emptyset$ and other agent state $\mathbf{o}_{\text{other}} = \emptyset$.
Input: The set of all agents $\mathcal{K} = 1, 2, ..., K$, the current main agent $k \in \mathcal{K}$, the public state $\mathbf{o}^{\text{pub}}(t)$ and the private states set of all agents $\mathbf{o}^{\text{pri}}(t) = \mathbf{o}_i^{\text{pri}}(t), \forall i \in \mathcal{K}$ at time step t.
for *each agent* $i \in \mathcal{K}$ **do**
    The learning control center obtains initial state $\mathbf{s}_0$ from the GAI-enabled CPM environment.
    if $i = k$ **then**
        Record main agent's state, $\mathbf{o}_{\text{main}} \leftarrow \mathbf{o}_i^{\text{pri}}(t)$.
    else
        Record other agents' states, $\mathbf{o}_{\text{other}} \leftarrow \mathbf{o}_{\text{other}} \cup \mathbf{o}_i^{\text{pri}}(t)$.
Construct the final state set, $\tilde{\mathbf{o}}_k(t) = \left[\mathbf{o}^{\text{pri}}(t), \mathbf{o}_{\text{main}}, \mathbf{o}_{\text{other}}\right]$.
return Main-Agent, k, Priority State $\tilde{\mathbf{o}}_k(t)$.
Output: The processed state set $\tilde{\mathbf{o}}_k(t)$ for the main agent k.

During training, each agent is selected in turn as the main agent to generate training samples using the same shared policy, which improves sample efficiency and promotes homogeneous behavior learning. During decentralized execution, agent k only has access to its own private state. To maintain the same input dimension as in training, we keep the public state unchanged and set the unavailable private blocks (corresponding to other agents) to zeros, i.e.,

$$\hat{\mathbf{o}}_k(t) = \left[\mathbf{o}^{\text{pub}}(t),\ \mathbf{o}_k^{\text{pri}}(t),\ \mathbf{0},\ \ldots,\ \mathbf{0}\right]. \tag{21}$$

This design ensures consistency between centralized training and decentralized execution, while enabling a single shared policy to be deployed across all ESs.

4.3. Hybrid-Action Decoding via Multi-Head Outputs

To address the hybrid-action space, we parameterize the shared policy network with (i) a shared backbone for feature extraction and (ii) two parallel output heads for discrete and continuous action components, respectively. Let $\mathbf{x}$ denote the input state vector (during centralized training, $\mathbf{x}$ corresponds to the constructed input state $\tilde{\mathbf{o}}_k(t)$; during execution, $\mathbf{x}$ corresponds to the local state input $\hat{\mathbf{o}}_k(t)$ of the acting ES), thus we have

$$\mathbf{x}_k(t) \triangleq \begin{cases} \tilde{\mathbf{o}}_k(t), \text{Centralized Training stage} \\ \hat{\mathbf{o}}_k(t), \text{Decentralized Execution stage} \end{cases}, \tag{22}$$

and the policy produces the main agent's hybrid action $\mathbf{a}_k(t) = (\mathbf{a}_k^d(t), \mathbf{a}_k^c(t))$.

(i) Shared backbone feature: Given $\mathbf{x}_k(t)$, the shared backbone extracts a feature embedding $f_k(t)$ as

$$\begin{aligned} f_k(t) &= \text{Backbone}(\mathbf{x}_k(t); \theta_{\text{backbone}}) \\ &= \text{ReLU}(W_{\text{backbone}}\mathbf{x}_k(t) + b_{\text{backbone}})' \end{aligned} \tag{23}$$

where $\theta_{\text{backbone}} = \{W_{\text{backbone}}, b_{\text{backbone}}\}$ is shared by all homogeneous ES agents under the parameter-sharing scheme.

(ii) Parallel heads for hybrid actions: Based on the feature $f_k(t)$, two parallel heads output the discrete action a with association decision and continuous power-control variables for agent k.

- Discrete-action head: The discrete head outputs a categorical distribution over $S = M * N$ association-related choices for the main agent

$$\pi_{\theta_d}\left(\mathbf{a}_k^d(t) = s \mid \mathbf{x}_k(t)\right) = \frac{\exp(W_{d,s} f_k(t) + b_{d,s})}{\sum_{j=1}^{S} \exp\left(W_{d,j} f_k(t) + b_{d,j}\right)}, \tag{24}$$

where $\theta_d = \{W_d, b_d\}$.

- Continuous-action head: The continuous head outputs the parameters of a diagonal Gaussian policy for the main agent's continuous controls

$$\begin{cases} \mu_k(t) = \mathbf{W}_{c,\mu} \boldsymbol{f}_k(t) + \boldsymbol{b}_{c,\mu} \\ \log \sigma_k(t) = \mathbf{W}_{c,\sigma} \boldsymbol{f}_k(t) + \boldsymbol{b}_{c,\sigma} \end{cases}. \tag{25}$$

Thus, we have

$$\pi_{\theta_c}(\mathbf{a}_k^c(t) \mid \mathbf{x}_k(t)) = \prod_{d=1}^{D} \mathcal{N}\left(\mathbf{a}_{k,d}^c(t); \mu_{k,d}(t), \exp(2 \log \sigma_{k,d}(t))\right), \tag{26}$$

where $\theta_c = \{W_c, b_c\}$.

(iii) Overall policy parameters: According to Equations (23), (24) and (26), the overall policy parameters are given by

$$\theta = \theta_{\text{backbone}} \cup \theta_d \cup \theta_c. \tag{27}$$

with parameter sharing, the same θ is used for all ESs.

This multi-head design enables the policy to simultaneously produce association-related discrete decisions and continuous power-control commands, providing a unified and tractable method for hybrid-action.

4.4. Training Process

This subsection describes the implementation of the proposed HA-MAPPO under the CTDE paradigm. During training, the policy and value networks are allowed to exploit the public state shared by all agents and the private states of all agents to mitigate non-stationarity in multi-agent learning. During execution, each ES performs decision-making independently using only its locally observable state, while keeping the network input dimension consistent via zero-padding.

4.4.1. Sample Collection

Under CTDE, we collect training data by building the joint action of all ESs at every time slot in the GAI-enabled industrial environment, rather than using an offline industrial dataset. Specifically, at time slot t, the learning control center constructs the main-agent input $\mathbf{x}_k(t)$ using the main-agent priority rule, and it performs a forward pass of the shared hybrid policy network to obtain the hybrid action $\mathbf{a}_k(t)$. It repeats this for all $k \in \mathcal{K}$ to obtain an executable joint action for the environment.

Let $\pi_\theta(\cdot \mid \mathbf{x})$ denote the shared action hybrid policy with two heads, as described in Section 4.3. The per-slot joint action is formed as

$$\mathbf{a}(t) = \{\mathbf{a}_k(t)\}, \forall k \in \mathcal{K}, \tag{28}$$

$$\mathbf{a}_k(t) \sim \pi_\theta(\cdot \mid \mathbf{x}_k(t)), k \in \mathcal{K}. \tag{29}$$

After executing $\mathbf{a}(t)$, the environment advances one slot following the FIFO upload–LoRA fine-tuning–download pipeline, yielding the next state $\mathbf{x}_k(t+1)$ and the reward $r(t)$.

To support HA-MAPPO updates, we store a replay buffer $\mathcal{D}$ that is continuously populated with transition tuples generated online during training. Each record corresponds to $\{\mathbf{x}_k(t), \mathbf{a}_k(t), r(t), \mathbf{x}_k(t+1)\}$, which is produced from the real-time interaction between the agents and the simulated industrial environment

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{x}_k(t), \mathbf{a}_k(t), r(t), \mathbf{x}_k(t+1)\}_{\forall k \in \mathcal{K}}. \tag{30}$$

4.4.2. Advantage Estimation

To obtain a low-variance policy-gradient estimate for the shared hybrid policy, HA-MAPPO performs actor updates using an advantage function. We randomly sampled a mini-batch $\tilde{\mathcal{D}}$ from replay buffer $\mathcal{D}$. For each sampled tuple $(\mathbf{x}_k(\tilde{t}), \mathbf{a}_k(\tilde{t}), r(\tilde{t}), \mathbf{x}_k(\tilde{t}+1)) \in \tilde{\mathcal{D}}$, the temporal-difference (TD) residual based on the centralized critic $V_\psi(\cdot)$ is computed as

$$\delta_{\tilde{t}} = r(\tilde{t}) + \gamma V_\psi(\mathbf{x}_k(\tilde{t}+1)) - V_\psi(\mathbf{x}_k(\tilde{t})), \tag{31}$$

where $r(\tilde{t})$ is the per-slot reward defined in Equation (18) and $\gamma \in [0,1]$ is the discount factor in Equation (19). To balance bias and variance, we adopt generalized advantage estimation (GAE) and compute the advantage as

$$A_{\tilde{t}} = \sum_{l=0}^{T-\tilde{t}-1} (\gamma\lambda)^l \delta_{\tilde{t}+l}, \tag{32}$$

where $\lambda \in [0,1]$ is the GAE smoothing coefficient. Correspondingly, the target action value for critic learning is given by

$$\hat{R}_{\tilde{t}} = A_{\tilde{t}} + V_\psi(\mathbf{x}_k(\tilde{t})). \tag{33}$$

4.4.3. Centralized Critic Network Update

Under CTDE, the centralized critic $V_\psi(\cdot)$ takes the training-time input $\mathbf{x}_k(\tilde{t})$ and outputs the scalar value estimate $V_\psi(\mathbf{x}_k(\tilde{t}))$. The critic parameters ψ are optimized by minimizing the mean-squared error (MSE) loss

$$\mathcal{L}_V(\psi) = \mathbb{E}_{\tilde{t}\sim\tilde{\mathcal{D}}}\left[\left(V_\psi(\mathbf{x}_k(\tilde{t})) - \hat{R}_{\tilde{t}}\right)^2\right], \tag{34}$$

where $\hat{R}_{\mathbf{t}}$ is the target action value defined in Equation (33).

Accordingly, the gradient of the critic loss is

$$\nabla_\psi \mathcal{L}_V(\psi) = \mathbb{E}_{\tilde{t}\sim\tilde{\mathcal{D}}}\left[2\left(V_\psi(\mathbf{x}_k(\tilde{t})) - \hat{R}_{\tilde{t}}\right)\nabla_\psi V_\psi(\mathbf{x}_k(\tilde{t}))\right], \tag{35}$$

and the critic is updated via stochastic gradient descent as

$$\psi \leftarrow \psi - \eta_\psi \nabla_\psi \mathcal{L}_V(\psi), \tag{36}$$

where η_ψ denotes the critic learning rate.

4.4.4. Shared Actor Network Update

All ESs share one hybrid policy parameter set θ, and each forward pass is agent-indexed by the main-agent input $\mathbf{x}_k(\tilde{t})$.

For HA-MAPPO, we form the hybrid importance ratio $\rho_{\tilde{t}}(\theta)$ as the product of the discrete and continuous ratios

$$\rho_{\tilde{t}}(\theta) = \underbrace{\frac{\pi_{\theta^{\mathrm{d}}}\left(\mathbf{a}_k^{\mathrm{d}}(\tilde{t}) \mid \mathbf{x}_k(\tilde{t})\right)}{\pi_{\theta^{\mathrm{d}}_{\mathrm{old}}}\left(\mathbf{a}_k^{\mathrm{d}}(\tilde{t}) \mid \mathbf{x}_k(\tilde{t})\right)}}_{\text{discrete action ratios}} \times \underbrace{\frac{\pi_{\theta^{\mathrm{c}}}\left(\mathbf{a}_k^{\mathrm{c}}(\tilde{t}) \mid \mathbf{x}_k(\tilde{t})\right)}{\pi_{\theta^{\mathrm{c}}_{\mathrm{old}}}\left(\mathbf{a}_k^{\mathrm{c}}(\tilde{t}) \mid \mathbf{x}_k(\tilde{t})\right)}}_{\text{continuous action ratios}}. \tag{37}$$

Then, the shared actor network is updated by maximizing the clipped surrogate

$$\mathcal{L}_{ppo}(\theta) = \mathbb{E}_{\tilde{t}\sim\tilde{\mathcal{D}}}[\min(\rho_{\tilde{t}}(\theta)A_{\tilde{t}}, \mathrm{clip}(\rho_{\tilde{t}}(\theta), 1-\epsilon, 1+\epsilon)A_{\tilde{t}})], \tag{38}$$

here $\epsilon > 0$ is the clipping threshold and $A_{\tilde{t}}$ is the advantage given by Equation (32).

The policy-gradient update is then obtained by ascending the gradient of the surrogate objective

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\mathrm{PPO}}(\theta) = \mathbb{E}_{\tilde{t}\sim\tilde{\mathcal{D}}}[\nabla_\theta \min(&\rho_{\tilde{t}}(\theta)A_{\tilde{t}}, \\ &\mathrm{clip}(\rho_{\tilde{t}}(\theta), 1-\epsilon, 1+\epsilon)A_{\tilde{t}})], \end{aligned} \tag{39}$$

$$\theta \leftarrow \theta + \eta_\theta \nabla_\theta \mathcal{L}_{\mathrm{PPO}}(\theta), \tag{40}$$

where η_ψ is the actor learning rate.

This joint update matches our AoM-minimization objective: the discrete association determines which SM-model update requests enter an ES's three-stage pipeline, while the continuous power control affects the wireless transmission durations, which directly impacts subsequent AoM and the reward in Equation (18).

4.5. Algorithm Implementation

In Section 4, we develop a CTDE-based HA-MAPPO algorithm. Specifically, Section 4.1 presents the overall architecture and network design. Section 4.2 proposes the Main-Agent Priority State Strategy to restructure the state input and improve scalability. Section 4.3 details the unified treatment of the hybrid-action space. Section 4.4 further describes the training procedure and derives the update rules for the actor and critic networks. These components together constitute the complete HA-MAPPO framework.

Based on the above, Algorithm 2 summarizes the end-to-end implementation of the proposed method. The overall procedure can be described in two stages: experience exploration and learning optimization. The former interacts with the environment to collect experience samples and store them in the replay buffer. The latter samples mini-batches to compute advantage-related quantities and importance ratios, and then updates the critic and actor parameters to achieve stable policy improvement.

4.5.1. Experience Exploration

At each time slot t, the learning control center first observes the public state $\mathbf{o}^{\mathrm{pub}}(t)$ and private states $\left\{\mathbf{o}_i^{\mathrm{pri}}(t) \mid i \in \mathcal{K}\right\}$ (lines 6). Then, each ES is selected in turn as the main agent to construct its network input $\mathbf{x}_k(t)$ using the main-agent priority rule (lines 7–8). Based on $\mathbf{x}_k(t)$, the shared actor performs a forward pass to generate the hybrid action $\mathbf{a}_k(t) = \left(\mathbf{a}_k^d(t), \mathbf{a}_k^c(t)\right)$ (lines 9). After repeating this process for all $k \in \mathcal{K}$, an executable joint action $\mathbf{a}(t) = \{\mathbf{a}_k(t)\}_{k\in\mathcal{K}}$ is formed and applied to the industrial environment (lines 10). The environment then returns the per-slot reward $r(t)$ and transitions to the next state (lines 11). The corresponding transition tuples $\{\mathbf{x}_k(t), \mathbf{a}_k(t), r(t), \mathbf{x}_k(t+1)\}$ are stored in the replay buffer $\mathcal{D}$ for subsequent updates (lines 12).

Algorithm 2: Training procedure of the proposed HA-MAPPO under CTDE.

1 **Initialize:** Discount factor γ, GAE factor λ, PPO clipping ϵ, and replay buffer $\mathcal{D} = \emptyset$.
2 **Initialize:** The shared actor parameters θ and the centralized critic parameters ψ.
3 **for** *each episode* **do**
4 Reset the GAI enabled industry environment and obtain the initial public state.
5 **for** *each step* $t = 1$ *to* T **do**
6 Observe public state $\mathbf{o}^{\text{pub}}(t)$ and private states $\left\{\mathbf{o}_i^{\text{pri}}(t) \mid i \in \mathcal{K}\right\}$.
7 **for** *each ES* $k \in \mathcal{K}$ **do**
8 Construct main-agent k input $\mathbf{x}_k(t)$ using the priority rule in Equations (20)–(22).
9 According to input $\mathbf{x}_k(t)$ generate hybrid action $\mathbf{a}_k(t) \sim \pi_\theta(\cdot \mid \mathbf{x}_k(t))$.
10 Based on the above, the joint action $\mathbf{a}(t)$ is formed according to Equations (28) and (29).
11 Execute joint action $\mathbf{a}(t)$, calculate reward $r(t)$, and transit to next state.
12 Store transition tuple $\{\mathbf{x}_k(t), \mathbf{a}_k(t), r(t), \mathbf{x}_k(t+1)\}$ into replay buffer $\mathcal{D}$ and update $\mathcal{D}$ as in Equation (30).
13 Randomly sample a mini-batch $\tilde{\mathcal{D}}$ from $\mathcal{D}$.
14 **for** *each* $\tilde{\mathcal{D}} \in \mathcal{D}$ **do**
15 Calculate the TD residual $\delta_{\tilde{t}}$, the GAE advantage $A_{\tilde{t}}$, and the target action value $\hat{R}_{\tilde{t}}$ based on Equations (31)–(33).
16 Update the critic parameter ψ using gradient descent on the loss $\mathcal{L}_V(\psi)$ in Equation (34) according to Equations (35) and (36).
17 Calculate the hybrid importance ratio $\rho_{\tilde{t}}(\theta)$ based on Equation (37).
18 Update the actor parameter θ using gradient ascent on the clipped surrogate $\mathcal{L}_{\text{PPO}}(\theta)$ in Equation (38) according to Equations (39) and (40).
19 **return** Trained shared hybrid policy $\pi_\theta(\cdot \mid \mathbf{x})$ and centralized critic $V_\psi(\cdot)$.

4.5.2. Learning Optimization

At the learning stage, the algorithm updates both the centralized critic and the shared hybrid actor by randomly sampling a mini-batch $\tilde{\mathcal{D}}$ from the replay buffer $\mathcal{D}$ (lines 13). For each sampled transition in $\tilde{\mathcal{D}}$, we first compute the TD residual $\delta_{\tilde{t}}$, the GAE advantage $A_{\tilde{t}}$, and the target return $\hat{R}_{\tilde{t}}$ to get the value regression loss $\mathcal{L}_V(\psi)$, then we update the centralized critic parameters ψ via gradient descent to improve value estimation (lines 15–16). Next, we compute the hybrid importance ratio $\rho_{\tilde{t}}(\theta)$ to get the PPO clipped surrogate objective $\mathcal{L}_{\text{PPO}}(\psi)$, and then we update the shared actor parameters θ via gradient ascent to obtain a more stable policy improvement under the hybrid-action space (lines 17–18).

4.6. Complexity Analysis

Let d_{pub} and d_{pri} denote the dimensions of the public state and each private state, respectively. The centralized-training input dimension for each main-agent sample is $d_{\text{in}} = d_{\text{pub}} + |K| d_{\text{pri}}$. Consider an L-layer MLP backbone with hidden width H, a discrete action space of size $S = M * N$, and continuous action dimension D_{c}. Since the shared policy outputs the action for only one main agent per forward pass, assembling the joint action at each slot requires $|k|$ forward evaluations, leading to a rollout-time per-slot complexity of the order of $\mathcal{O}(|k|(d_{\text{in}} H + H(S + D_{\text{c}})))$, where the first term accounts for feature extraction and the second term accounts for the two output heads.

5. Simulation Results

5.1. Simulation Settings

To evaluate the performance of the proposed HA-MAPPO framework, we conduct extensive simulations under an industrial edge computing scenario. The simulated system

with $K = 5$ uniformly deployed ESs and $N = 20$ randomly distributed SMs, which are confined to a 400 m $\times$ 400 m square manufacturing area. The system supports $M = 5$ types of foundation models for industrial AIGC services. The total duration of AIGC services is divided into $T = 1000$ time slots, and the duration of each time slot is $\tau = 0.2$ s. Each ES acts as an independent agent and makes model association and power-control decisions within each time slot [11,15].

Due to the abundant metallic surfaces of SMs, they typically induce rich scattering. In non-line-of-sight (NLoS) regions, the amplitudes of the generated multipath components are well described by a Rayleigh distribution. Thus, we adopt Rayleigh fading to model the channel gain from ES to SM. Similarly to [15,27], Rayleigh fading incorporates the elevated non-line-of-sight path loss (dB) of $140.7 + 36.7 \log_{10} d_n^m$ [km], the shadowing standard deviation of 10 dB, and the Gaussian noise with a noise figure of 9 dB to simulate the noisy industry environments.

For RL training, the total number of episodes is set to 10,000, and the mini-batch size for network updates is 128. Under the parameter-sharing scheme, all agents share the same actor network parameters. We implement the neural networks using fully connected layers. The actor backbone uses hidden sizes $[512, 256]$, and the output heads use hidden sizes $[256, 128]$. The critic uses hidden sizes $[64, 32]$. ReLU is applied in all hidden layers to introduce nonlinearity. We train the networks using the Adam optimizer. Other simulation parameters are followed as in Table 1 [11,15,28].

Table 1. Simulation parameters.

Parameters	Value
Model fine-tuning	$S_n^m = 1000$, $C_{n,m}^{\mathrm{LoRA}} = 0.22$ Gigacycles, $E = [1, 2]$, $f_k = [50, 100]$ Gigacycles/s
Model transmission	$D_{n,m}^{\mathrm{full}} = [100, 1000]$ MB, $D_{n,m}^{\mathrm{LoRA}} = [1, 10]$ MB, $W^{\mathrm{UL}} = 20$ MHz, $W^{\mathrm{DL}} = 100$ MHz, $\sigma_n^2 = -95$ dBm
Problem formulation	$\zeta = 200$, $\epsilon = 0.01$
HA-MAPPO, algorithm hyperparameters	$\eta_\theta = 0.001$, $\eta_\psi = 0.0005$, $\gamma = 0.95$, $\sigma = 0.002$, $\lvert\mathcal{D}\rvert = 100,000$, $N_{\mathrm{b}} = 128$,

For our experimental environment, the hardware configuration includes an AMD EPYC 9654 96-Core Processor CPU with 48 cores and an NVIDIA GeForce RTX 4090 GPU with 24 GB of memory, while the software setup includes CUDA-12.8.1 and Python-3.9. In addition, the simulation environment is implemented in Python using the TensorFlow deep learning framework (the code is available at https://github.com/lwj6633/AoM_python (accessed on 15 March 2026)).

For comparison, the following special cases and HA-MAPPO algorithms are used as the benchmark schemes:

- Random fine-tuning (Random FT): The fine-tuning strategy is randomly configured. The agent performs resource allocation and task scheduling based on randomly selected fine-tuning strategies, without optimizing the fine-tuning process.
- Fixed transmit power (Fixed Power): The transmit power of both ESs and SMs is fixed during task execution. No power optimization is performed, and the Fixed Power configuration is applied uniformly for all tasks involving ESs and SMs.
- Proximal Policy Optimization (PPO): The standard PPO algorithm is adopted for agent policy optimization. The neural network only takes the common state and the private state of a single agent as inputs, and optimizes the policy by clipping the

surrogate objective to ensure stable training in decision-making scenarios such as resource allocation and task scheduling.

- Deep Q-Network (DQN): The DQN algorithm is employed to handle decision-making tasks such as resource allocation and task scheduling, where continuous action spaces are discretized into discrete values for training and inference.

5.2. Convergence Analysis

In Figure 3, the average rewards of all DRL-based algorithms show an upward trend during the early stage of training. This trend indicates that the agents gradually learn effective policies for service association and resource allocation through continuous interaction with the industrial environment. The simulation results show that the cumulative reward of the proposed algorithm is approximately 11.13%, 20.32%, 36.61%, and 38.78% higher than those of the other four benchmark algorithms, respectively. Compared with single-agent baselines such as PPO and DQN, multi-agent reinforcement learning algorithms exhibit more noticeable fluctuations during training. However, they eventually achieve significantly higher steady-state rewards. This advantage comes from their effective coordination mechanism, which enhances collaboration among agents, improves the global system utility, and alleviates resource contention. Moreover, the proposed algorithm significantly outperforms the Random FT and Fixed Power benchmarks. This gain is attributed to its joint optimization of model association and power control, which enables the agents to make more effective decisions and utilize limited communication and computation resources more efficiently. Overall, these results demonstrate that jointly optimizing fine-tuning service association and resource allocation is both necessary and effective for maintaining model freshness in complex industrial AIGC environments.

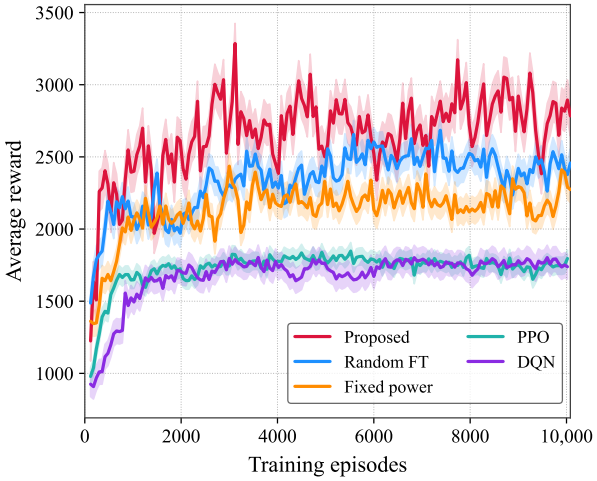
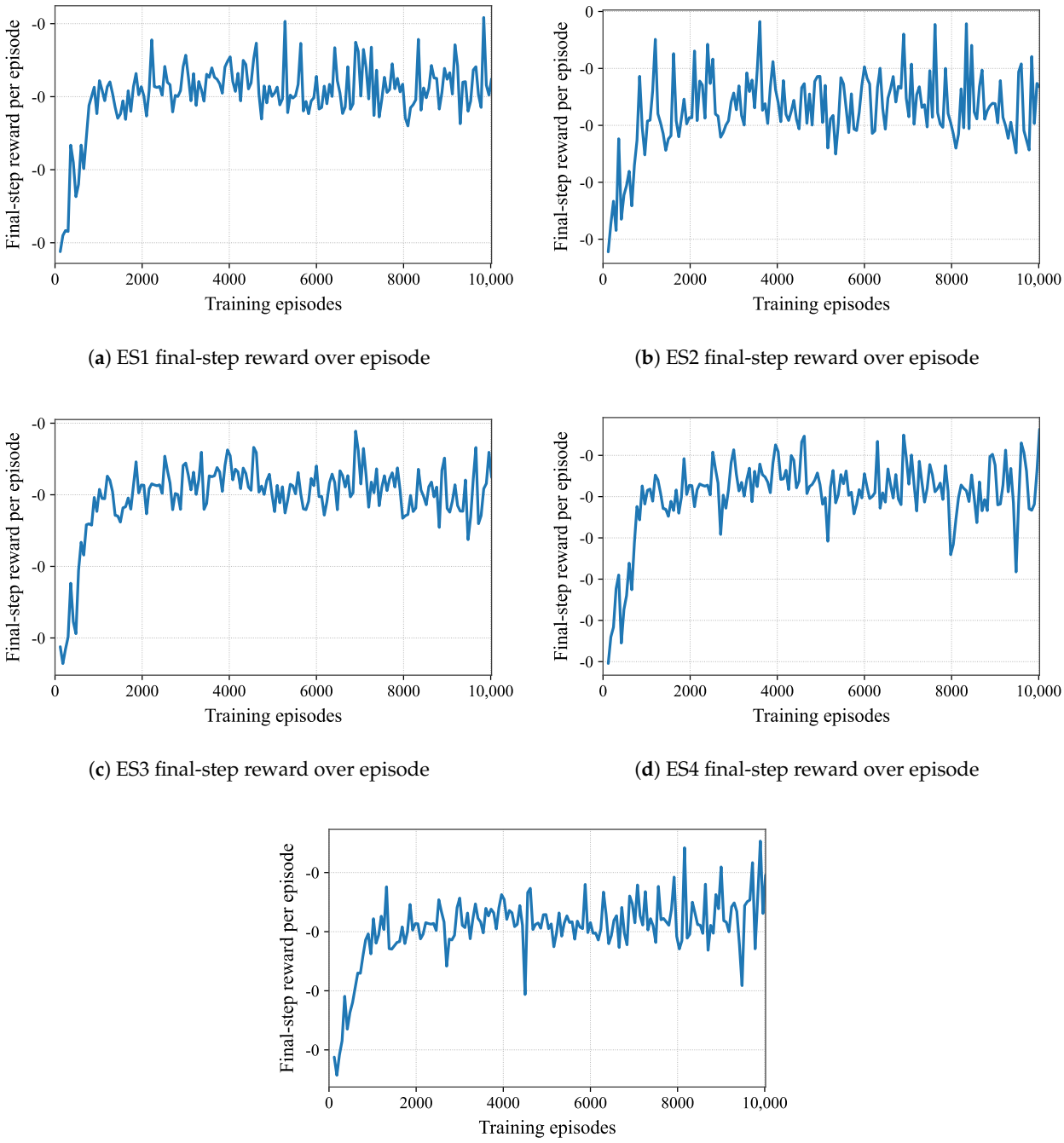


Figure 3. The convergence performance of the proposed algorithm.

To further evaluate the learning efficiency of individual agents in a competitive industrial environment, we analyze the reward obtained at the final step of each episode. This reward reflects the cumulative effect of the model fine-tuning process within an episode. As shown in Figure 4, the final-step rewards of all ESs gradually evolve during training. Unlike the average cumulative reward, the reward at the terminal step of an episode directly reflects the utility of the final system state, which results from a sequence of decisions on fine-tuning association and resource allocation. Figure 4 also shows that different ESs may obtain different rewards in the same episode. Nevertheless, their rewards exhibit a consistent upward trend and eventually converge to stable levels. This observation demonstrates that each ES, as an independent agent, can successfully learn a cooperative policy

under the HA-MAPPO framework. Such coordination enables the system to efficiently handle randomly arriving AIGC tasks from SMs. Meanwhile, it helps reduce the terminal AoM of the industrial system.



(**a**) ES1 final-step reward over episode

(**b**) ES2 final-step reward over episode

(**c**) ES3 final-step reward over episode

(**d**) ES4 final-step reward over episode

(**e**) ES5 final-step reward over episode

Figure 4. The convergence of the final-step reward per episode for each individual ES agent.

5.3. Performance Comparisons

Figure 5 shows the average AoM under different maximum step lengths, denoted by $T_{\max}$. To improve the reliability of the results, each bar reports the average over five independent runs, and the error bars represent the standard deviation. As $T_{\max}$ increases from 100 to 2000, the average AoM of all methods increases steadily. For the proposed method, the average AoM rises from 69 to 220, 549, and 976, respectively. This trend shows that AoM accumulates over time when transmission and computation resources are limited.

At $T_{\max} = 1000$, the proposed method achieves an average AoM of 549, which is 3.2% lower than Random FT and 12.2% lower than Fixed Power. At $T_{\max} = 2000$, the corresponding improvements are 6.6% and 14.3%, respectively. In addition, when $T_{\max}$ increases from 1000 to 2000, the average AoM of the proposed method increases by 77.8%, which further shows the growing impact of model aging over longer operation horizons. These results confirm that joint optimization of fine-tuning and resource allocation is necessary to maintain model freshness in industrial AIGC systems.

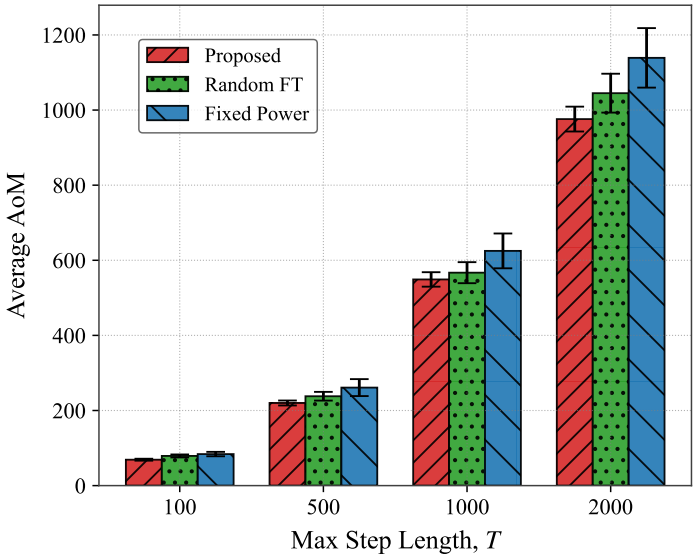
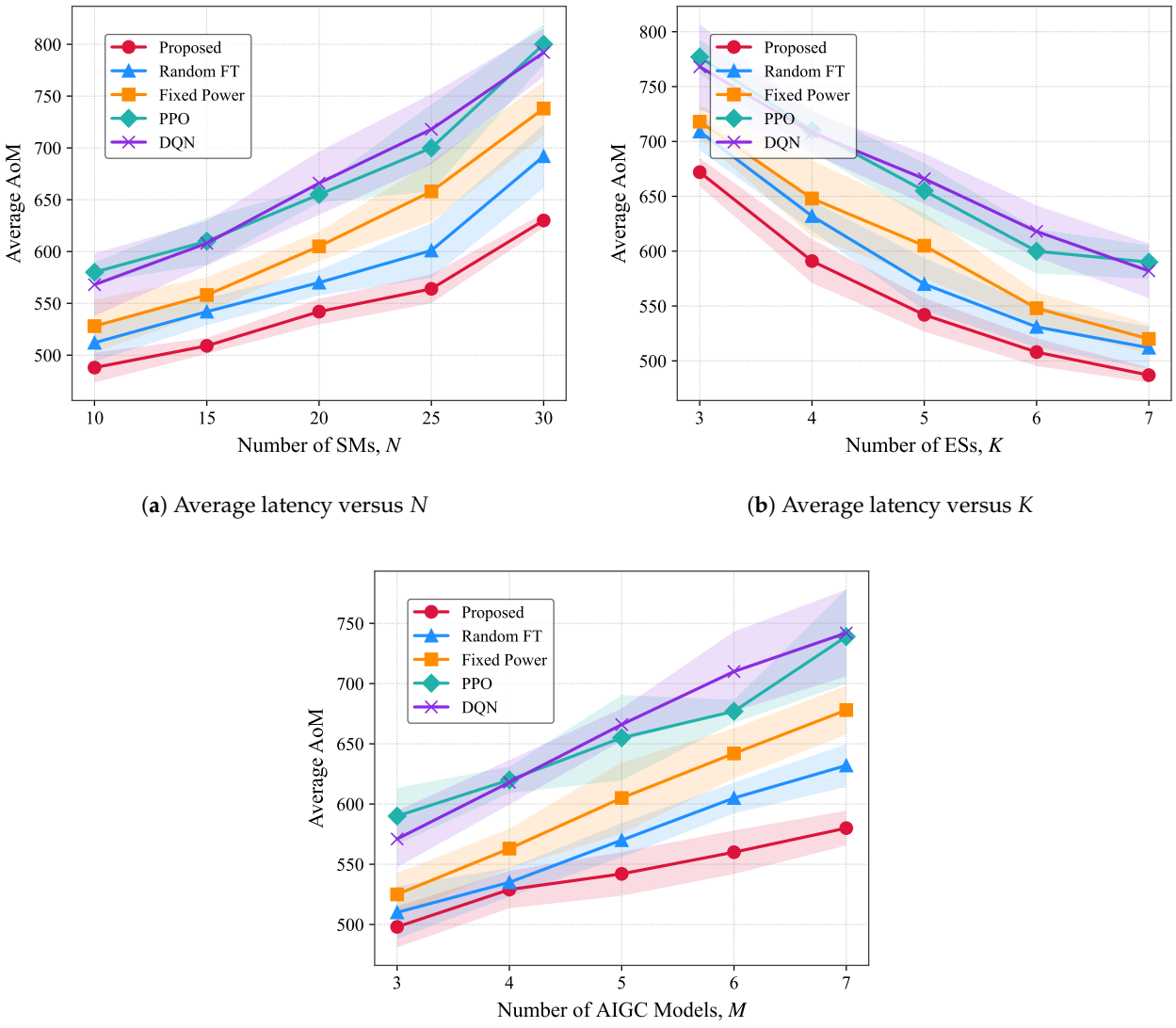


Figure 5. Impact of maximum step length on average AoM.

To evaluate the scalability of the proposed scheme, we conduct simulations by varying the numbers of ESs, SMs, and AIGC models. Figure 6 shows the corresponding average AoM results. Each curve reports the average of five independent runs, and the shaded area represents the standard deviation. As shown in Figure 6a,c, the average AoM increases as the number of SMs or AIGC models increases. This is because more devices and model candidates create stronger competition for the limited wireless and edge computing resources. For example, when the number of AIGC models increases from $M = 4$ to $M = 6$, the average AoM of the proposed method increases from 529 to 560, which is a 5.9% increase. Under the same setting, the AoM increases from 535 to 605 for Random FT and from 563 to 642 for Fixed Power, corresponding to increases of 13.1% and 14.0%, respectively. A similar trend can be observed when the number of SMs increases. When N increases from 20 to 30, the average AoM of the proposed method increases from 542 to 630, while the increases are 21.4% for Random FT and 22.0% for Fixed Power. In contrast, as shown in Figure 6b, increasing the number of ESs effectively reduces the average AoM. Specifically, when K increases from 3 to 7, the average AoM of the proposed method decreases from 672 to 487, which is a 27.5% reduction. Over the same range, Random FT decreases from 709 to 512 and Fixed Power decreases from 718 to 520, corresponding to reductions of 27.8% and 27.6%, respectively. These results show that the proposed scheme maintains lower AoM across different system scales and has good scalability in industrial environments.



(**a**) Average latency versus N

(**b**) Average latency versus K

(**c**) Average latency versus M

Figure 6. The impact of the numbers of SMs, ESs, and AIGC models on the average AoM, respectively.

Bandwidth is a critical communication resource that directly determines the transmission latency of fine-tuning GAI models. Figure 7 illustrates the improvement in AoM performance as the bandwidth allocated to each ES and SM increases. As the bandwidth increases, the average AoM decreases clearly. This is because higher bandwidth shortens the transmission time of model parameters. When the ES bandwidth is fixed at $W^{\mathrm{DL}} = 100$ MHz, increasing the SM bandwidth from $W^{\mathrm{UL}} = 10$ MHz reduces the average AoM from 745.76 to 452.72, which is a 39.3% reduction. In contrast, when the SM bandwidth is fixed at $W^{\mathrm{UL}} = 20$ MHz, increasing the ES bandwidth from $W^{\mathrm{DL}} = 50$ MHz to 200 MHz reduces the average AoM from 575.76 to 522.72, corresponding to a 9.2% reduction. These results show that increasing the uplink bandwidth brings a larger AoM reduction than increasing the downlink bandwidth. The main reason is the asymmetric data volume in the fine-tuning process. After LoRA-based fine-tuning, the downlink data size is much smaller than the uplink data size. Therefore, the uplink link becomes the main bottleneck for timely model updates.

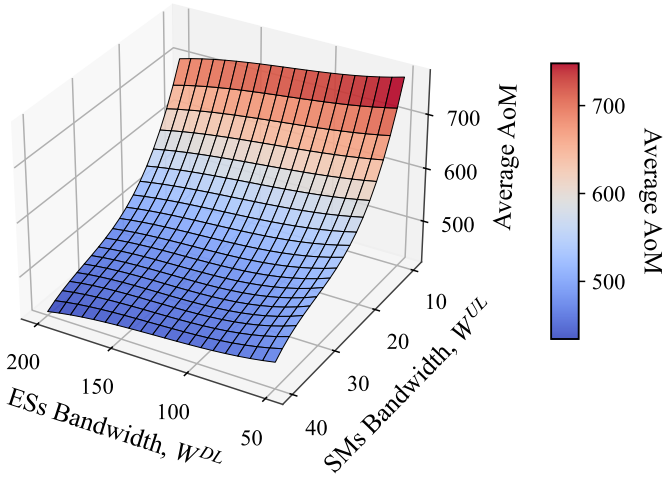


Figure 7. Average AoM versus the available bandwidth for ESs and SMs.

6. Conclusions

In this paper, we investigated AIGC service provisioning in Industry 5.0 from the perspective of model freshness. We introduced AoM, a metric specifically designed to evaluate the temporal relevance of generative models in time-varying industrial environments. We also developed a three-stage LoRA-based fine-tuning workflow model, which captures the uplink transmission, edge-side fine-tuning computation, and downlink model delivery processes. To address the resulting problem, we proposed a CTDE-based hybrid-action multi-agent reinforcement learning framework, namely HA-MAPPO. The proposed method enables each ES to make decisions based on local observations while achieving inter-agent coordination through centralized training. It also supports the joint optimization of discrete model association and continuous wireless resource allocation. Simulation results show that the AoM-centered design can maintain model freshness under limited communication and computation resources, while significantly improving the timeliness and reliability of industrial AIGC services. By linking model freshness with fine-tuning scheduling and resource allocation, this work lays a foundation for providing time-sensitive AIGC services in future smart manufacturing systems.

This study still has several limitations: (i) it adopts a time-slot-based model and assumes that the channel state and system load remain nearly unchanged within each slot. Although this assumption simplifies the analysis and algorithm design, it may not fully capture rapidly changing industrial environments; (ii) although LoRA significantly reduces the size of model updates, the update process may still incur considerable communication overhead and computational cost when the AIGC model is very large or the delay requirement is stringent. This issue may become more severe in dense industrial networks with limited wireless and computing resources; and (iii) the current results are based solely on simulations. The proposed framework has not yet been validated on real industrial devices or hardware platforms. As a result, several practical factors, such as hardware heterogeneity, protocol overhead, and deployment complexity, are not fully considered, which may lead to discrepancies between the simulation results and real-world performance.

Based on the above limitations, future work will mainly focus on the following three aspects: (i) Conducting real-world validation on industrial devices and edge computing platforms. For example, we plan to build a small-scale testbed consisting of smart machines, wireless edge servers, and practical industrial communication links. This will help evaluate the actual performance of model transmission, online LoRA fine-tuning, and scheduling decisions under realistic latency, interference, and hardware constraints. (ii) Extending the current optimization objective from model freshness alone to the joint optimization of model

freshness and AIGC generation quality. This extension can support more refined fine-tuning and scheduling decisions. And (iii) investigating secure model-update transmission, data privacy protection, and defense against malicious attacks in industrial AIGC scenarios. This direction can further improve the security, reliability, and long-term stability of the system.

Author Contributions: Conceptualization, W.L. and N.T.; methodology, W.L. and N.T.; software, W.L.; validation, W.L.; formal analysis, W.L.; investigation, W.L.; resources, N.T.; data curation, W.L.; writing—original draft preparation, W.L.; writing—review & editing, W.L., N.T. and L.Z.; visualization, W.L.; supervision, N.T. and L.Z.; project administration, L.Z.; funding acquisition, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The code used in this study is publicly available at https://github.com/lwj6633/AoM_python (accessed on 15 March 2026).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AIGC	Artificial Intelligence-Generated Content
AoM	Age of Model
LoRA	Low-Rank Adaptation
HA-MAPPO	Hybrid-Action Multi-Agent Proximal Policy Optimization
CTDE	Centralized Training with Decentralized Execution
SM	Smart Machine
ES	Edge Server
GAI	Generative Artificial Intelligence
FIFO	First-In–First-Out

References

1. Noor-A-Rahim, M.; Firyaguna, F.; John, J.; Khyam, M.O.; Pesch, D.; Armstrong, E.; Claussen, H.; Poor, H.V. Toward industry 5.0: Intelligent reflecting surface in smart manufacturing. *IEEE Commun. Mag.* **2022**, *60*, 72–78. [CrossRef]
2. Chi, H.R.; Wu, C.K.; Huang, N.F.; Tsang, K.F.; Radwan, A. A survey of network automation for industrial internet-of-things toward industry 5.0. *IEEE Trans. Ind. Informat.* **2023**, *19*, 2065–2077. [CrossRef]
3. Wen, J.; Kang, J.; Niyato, D.; Zhang, Y.; Mao, S. Sustainable diffusion-based incentive mechanism for generative AI-driven digital twins in industrial cyber-physical systems. *IEEE Trans. Ind. Cyber-Phys. Syst.* **2025**, *3*, 139–149. [CrossRef]
4. Li, S.; Lin, X.; Xu, W.; Li, J. AI-generated content-based edge learning for fast and efficient few-shot defect detection in IIoT. *IEEE Trans. Serv. Comput.* **2024**, *17*, 3140–3153. [CrossRef]
5. Yu, X.; Zhang, S.; Zhang, H.; Song, L. Model collaboration at network edge: Feature-large models for real-time IoT communications. *IEEE Internet Things J.* **2025**, *12*, 13259–13272. [CrossRef]
6. Xu, M.; Niyato, D.; Zhang, H.; Kang, J.; Xiong, Z.; Mao, S.; Han, Z. Sparks of generative pretrained transformers in edge intelligence for the metaverse: Caching and inference for mobile artificial intelligence-generated content services. *IEEE Veh. Technol. Mag.* **2023**, *18*, 35–44. [CrossRef]
7. Xu, M.; Du, H.; Niyato, D.; Kang, J.; Xiong, Z.; Mao, S.; Han, Z.; Jamalipour, A.; Kim, D.I.; Shen, X.; et al. Unleashing the power of edge-cloud generative AI in mobile networks: A survey of AIGC services. *IEEE Commun. Surv. Tutor.* **2024**, *26*, 1127–1170. [CrossRef]
8. Zhang, S.; Liu, Q.; Chen, K.; Di, B.; Zhang, H.; Yang, W.; Niyato, D.; Han, Z.; Poor, H.V. Large models for aerial edges: An edge-cloud model evolution and communication paradigm. *IEEE J. Sel. Areas Commun.* **2025**, *43*, 21–35. [CrossRef]
9. Dimitrakopoulos, G.; Varga, P.; Gutt, T.; Schneider, G.; Ehm, H.; Hoess, A. Industry 5.0: Research areas and challenges with artificial intelligence and human acceptance. *IEEE Ind. Electron. Mag.* **2024**, *18*, 43–54. [CrossRef]
10. Zhang, W.; Sun, T.; Yang, D.; Luan, T.H.; Zhang, H. Dynamic resource scheduling for deterministic communication, computation, and control integration in industrial cyber-physical systems. *IEEE Trans. Cogn. Commun. Netw.* **2025**, *12*, 864–880. [CrossRef]

11. Li, S.; Lin, X.; Xu, H.; Hua, K.; Jin, X.; Li, G.; Li, J. Multi-agent RL-based industrial AIGC service offloading over wireless edge networks. In Proceedings of the IEEE INFOCOM WKSHPS, Vancouver, BC, Canada, 20–23 May 2024.
12. Tian, Y.; Zhang, Z.; Yang, Y.; Chen, Z.; Yang, Z.; Jin, R.; Quek, T.Q.S.; Wong, K.-K. An edge-cloud collaboration framework for generative AI service provision with synergetic big cloud model and small edge models. *IEEE Netw.* **2024**, *38*, 37–46. [CrossRef]
13. Liu, Y.; Zhang, R.; Wang, J.; Niyato, D.; Wang, X.; Kim, D.I.; Du, H. Intelligent mobile AI-generated content services via interactive prompt engineering and dynamic service provisioning. *arXiv* **2025**, arXiv:2502.11386. [CrossRef]
14. Liu, Z.; Du, H.; Huang, L.; Gao, Z.; Niyato, D. Joint model caching and resource allocation in generative AI-enabled wireless edge networks. In Proceedings of the IEEE WCNC, Milan, Italy, 24–27 March 2025.
15. Zhang, L.; Song, D.-A.; Zhang, H.; Tian, N.; Zhuang, Z.; Niyato, D.; Han, Z. Edge-driven industrial computing power networks: Digital twin-empowered service provisioning by hybrid soft actor-critic. *IEEE Trans. Veh. Technol.* **2025**, *74*, 8095–8109. [CrossRef]
16. Guan, Y.; Zou, S.; Peng, H.; Ni, W.; Sun, Y.; Gao, H. Cooperative UAV trajectory design for disaster area emergency communications: A multiagent PPO method. *IEEE Internet Things J.* **2024**, *11*, 8848–8859. [CrossRef]
17. Jiang, B.; Du, J.; Jiang, C.; Han, Z.; Debbah, M. Underwater searching and multiround data collection via AUV swarms: An energy-efficient AoI-aware MAPPO approach. *IEEE Internet Things J.* **2024**, *11*, 12768–12782. [CrossRef]
18. Wu, J.; Zhuang, X.; Tang, M.; Gao, L. QoE-aware offloading and resource allocation for MEC-empowered AIGC services. *IEEE Trans. Mobile Comput.* **2025**, *24*, 9664–9682. [CrossRef]
19. Fan, J.; Xu, M.; Liu, Z.; Ye, H.; Gu, C.; Niyato, D.; Lam, K.-Y. A learning-based incentive mechanism for mobile AIGC service in decentralized internet of vehicles. In Proceedings of the IEEE VTC-Fall, Hong Kong, China, 10–13 December 2023.
20. Wen, J.; Kang, J.; Xu, M.; Du, H.; Xiong, Z.; Zhang, Y.; Niyato, D. Freshness-aware incentive mechanism for mobile AI-generated content (AIGC) networks. In Proceedings of the IEEE/CIC International Conference on Communications in China (ICCC), Dalian, China, 10–12 August 2023; pp. 1–6.
21. Kynkäänniemi, T.; Karras, T.; Laine, S.; Lehtinen, J.; Aila, T. Improved precision and recall metric for assessing generative models. *Proc. Adv. Neural Inf. Process. Syst.* **2019**, *32*, 3927–3936.
22. Wu, C.; Yin, S.; Qi, W.; Wang, X.; Tang, Z.; Duan, N. Visual ChatGPT: Talking, drawing and editing with visual foundation models. *arXiv* **2023**, arXiv:2303.04671. [CrossRef]
23. Benny, Y.; Galanti, T.; Benaim, S.; Wolf, L. Evaluation metrics for conditional image generation. *Int. J. Comput. Vis.* **2021**, *129*, 1712–1731. [CrossRef]
24. Shi, T.; Zhang, T.; Loo, J.; Huang, R.; Wang, Y. Joint task offloading and channel allocation in spatial-temporal dynamic for MEC networks. *IEEE Trans. Ind. Inform.* **2025**, *21*, 6240–6250. [CrossRef]
25. Wang, C.; Han, Y.; Zhang, L.; Jia, Z.; Zhang, H.; Hong, C.S.; Han, Z. Computing power in the sky: Digital twin-assisted collaborative computing with multi-UAV networks. *IEEE Trans. Veh. Technol.* **2025**, *74*, 14466–14482. [CrossRef]
26. Xiong, X.; Li, M.; Yu, F.R.; Zhang, H.; Wang, K.; Si, P. Cloud-edge-end collaborative computing-enabled intelligent sharding blockchain for industrial IoT based on PPO approach. *IEEE Trans. Mob. Comput.* **2025**, *24*, 8011–8024. [CrossRef]
27. Zhang, L.; Wang, H.; Xue, H.; Zhang, H.; Liu, Q.; Niyato, D.; Han, Z. Digital twin-assisted edge computation offloading in industrial internet of things with NOMA. *IEEE Trans. Veh. Technol.* **2023**, *72*, 11935–11950. [CrossRef]
28. Okegbile, S.D.; Gao, H.; Talabi, O.; Cai, J.; Niyato, D.; Shen, X. Optimizing federated semantic learning in distributed AIGC-enabled human digital twins: A multi-criteria and multi-shard user selection framework. *IEEE Trans. Mob. Comput.* **2025**, *24*, 5916–5933. [CrossRef]