Tsinghua Science and Technology
Doi: 10.26599/TST.2026.9010056

Research Article

LLM-enabled agentic DRL for generic-specialized visual GenAI collaboration in edge–cloud computing networks

Moxia Li, Jiangtian Nie, Jianhang Tang (✉), Guoquan Wu, Yang Zhang, Celimuge Wu, Jiawen Kang, Zehui Xiong

Abstract Visual Generative Artificial Intelligence (GenAI) has emerged as a promising solution to deliver visually stunning content. To achieve seamless synergy between the generic and specialized GenAI models, edge-cloud collaborative networks require an adaptive framework that integrates real-time perception, continual learning, and autonomous optimization. In this work, we develop an Agentic Deep Reinforcement Learning (DRL) framework, where a Large Language Model (LLM)-enabled agent perceives network states and allocates resources contextually. Next, we formulate a joint optimization problem of inference scheme selection and resource orchestration, aiming to minimize time-average inference latency subject to the inference task queue stability constraint. Based on Lyapunov optimization, we first transform the original long-term optimization problem into several deterministic sub-problems. Then, a DRL-based Inference Task Scheduling (DRL-ITS) algorithm is developed to solve the sub-problems in each time slot, where an LLM-enabled agent provides rich prior knowledge and accurate semantic interpretation for resource allocation. Finally, we provide theoretical and simulation evaluations to demonstrate that the DRL-ITS algorithm can obtain faster convergence and reduce inference latency by comparing it with other benchmark schemes.

Keywords Visual GenAI, Edge-cloud collaboration, LLM, Agentic DRL

• Moxia Li and Jianhang Tang are with the State Key Laboratory of Public Big Data, Guizhou University, Guiyang, China. E-mails: gs.limx24@gzu.edu.cn, jhtang@gzu.edu.cn.
• Jiangtian Nie is with the School of Natural and Computing Sciences, University of Aberdeen, UK. E-mail: jiangtian.nie@abdn.ac.uk.
• Guoquan Wu is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. E-mail: wugq2024@mail.sustech.edu.cn.
• Yang Zhang is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. E-mail: yangzhang@nuaa.edu.cn.
• Celimuge Wu is with the Meta-NetworkingResearch Center, The University of Electro-Communications, Tokyo, Japan. E-mail: celimuge@uec.ac.jp.
• Jiawen Kang is with the School of Automation, Guangdong University of Technology, Guangzhou, China. E-mail: kavinkang@gdut.edu.cn.
• Zehui Xiong is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK. E-mail: z.xiong@qub.ac.uk.
* To whom correspondence should be addressed.
Manuscript received: 2026-02-25; revised: 2026-04-22; accepted: 2026-05-25

1 Introduction

Visual Generative Artificial Intelligence (GenAI) has advanced visual creation by enabling the generation of visually stunning images and facilitating wider applications through precise control [1]. The visual GenAI models (e.g., Stable Diffusion and DALL-E 3) demonstrate exceptional generalization capabilities across diverse domains by leveraging deep and robust encoding structures. However, relying solely on generic large generative models poses a critical challenge to ensuring task-oriented adaptability, as these centralized architectures struggle to incorporate private, timely, and personalized edge knowledge required for diverse scenarios [2]. To address this, specialized small generative models provide customized intelligence that complements the global generalization of the core cloud. Therefore, establishing a collaborative synergetic mechanism between generic large generative models and specialized small generative models is of paramount importance.

The collaboration of edge and cloud computing emerges as the pivotal paradigm to achieve hierarchical GenAI collaboration, where resource-abundant cloud servers host generic large GenAI models, whereas the resource-constrained edge accommodates specialized small GenAI models closer to end-users [3]. By integrating the global generative capabilities of large models with the specialized expertise of small models, the edge-cloud collaborative framework facilitates the seamless transition of GenAI from generic generalization to customized service provision. Nevertheless, cloud-edge collaboration necessitates navigating the intricate dependencies inherent to generative inference, particularly within the highly dynamic and heterogeneous nature of network environments.

The traditional cloud-edge collaborative networks typically rely on static models with limited cognitive autonomy, which struggle to adapt to the varying resource requirements of GenAI inference tasks [4]. Via continuous perception–reasoning–action loops, Agentic AI is becoming a promising paradigm for the resource orchestration of cloud-edge collaborative networks with dynamic GenAI service

Just Accepted

requests [5]. The continuous cognitive cycles empower AI agents to actively perceive multimodal task requirements and reason contextually about how to allocate edge and cloud computing resources for the collaborative inference of generic large and specialized small models.

To implement such autonomous loops, Agentic Deep Reinforcement Learning (DRL) has been introduced as a key enabling technology, allowing agents to optimize complex resource orchestration via continuous and iterative interactions with the dynamic environments. Via trial and error, the DRL scheme can enable the agent to learn optimal resource allocation policies. However, the intrinsic trial-and-error mechanism of DRL leads to poor sample efficiency and a lack of semantic awareness [6], making it difficult for agents to interpret high-level task requirements or leverage prior knowledge. Moreover, without sophisticated cognitive guidance, Agentic DRL often converges slowly and struggles to maintain queue stability under the diverse, multi-modal constraints of visual GenAI tasks.

Addressing the semantic deficits of pure trial-and-error DRL necessitates the integration of an Large Language Model (LLM). The LLM serves as a sophisticated cognitive engine that bridges the gap between low-level environmental states and high-level semantic intents. In particular, by serving as an advanced semantic parser, the LLM is capable of distilling intricate task constraints and visual GenAI requirements, which fundamentally augments the contextual representation for the DRL agent [7]. Such context-aware supervision explicitly compensates for the lack of prior knowledge and significantly accelerates policy convergence. Additionally, the LLM's multi-step reasoning empowers the agent to interpret complex system constraints and devise proactive orchestration strategies [8]. Consequently, the LLM acts as an indispensable cognitive core, upgrading traditional reactive DRL into a self-evolving, semantic-aware Agentic AI tailored for dynamic GenAI workloads.

In this work, we develop an LLM-enabled Agentic DRL structure tailored for generic-specialized generative visual model collaboration in edge computing. In contrast to traditional static schemes, the proposed framework empowers an intelligent agent with LLM-driven autonomous reasoning and powerful self-evolving capabilities, which can proactively coordinate the collaboration between large generic models and specialized small models with significantly improved GenAI inference performance. Our contributions can be summarized as follows.

- We develop a novel LLM-enabled Agentic DRL framework that establishes autonomous perception-reasoning-action loops for the collaboration of generic-specialized visual GenAI models. The LLM-enabled DRL agent can achieve a dynamic serial-parallel collaborative inference scheme, allowing the network to autonomously select the optimal inference modes and resource allocation schemes according to GenAI task requirements and available resource states.
- We formulate the inference mode selection and resource orchestration problem as a time-averaged stochastic optimization problem, aiming to minimize execution latency subject to task queue stability constraints. By applying Lyapunov optimization technique, the original stochastic optimization problem is decomposed into deterministic per-slot Mixed Integer Non-Linear Programming (MINLP) sub-problems.
- A DRL-based Inference Task Scheduling (DRL-ITS) algorithm is developed to solve the per-slot MINLP problem. An LLM-enhanced Twin Delayed Deep Deterministic Policy Gradient (TD3) method is investigated to derive the optimal inference mode and aggregation location selection decisions, leveraging the LLM knowledge base for context-aware reasoning, where the closed-form numerical solutions reduce the action space dimension to accelerate the learning convergence. Meanwhile, the optimal allocation solutions for computation and communication resources are derived based on Karush-Kuhn-Tucker (KKT) conditions.
- Extensive simulation experiments and theoretical analyses are provided to evaluate the superiority of our DRL-ITS algorithm. Simulation results indicate that, relative to the benchmark schemes, the DRL-ITS algorithm attains faster convergence, higher rewards, and reduced steady-state latency.

In Section 3, we discuss the previous works most relevant to this work. In Section 4, we present the LLM-enabled Agentic DRL architecture and propose the long-term resource allocation problem. In Section 4, our DRL-ITS algorithm is proposed in detail. We provide several simulation results and experimental discussions in Section 5. Our work is concluded in the last section.

2 Related Work

2.1 Collaborative Inference of Large and Small Models

Relying exclusively on massive, monolithic LLMs for high-throughput applications often incurs unsustainable financial costs and significant energy consumption, necessitating efficient orchestration mechanisms among models of varying

Just Accepted

sizes. To address the prohibitive inference latency and financial overheads inherent in Large Language Models (LLMs), Chen et al. [9] investigated a resource management framework, aiming at minimizing financial costs under the constraints of model accuracy requirements, thereby facilitating an optimal trade-off between performance and expenditure. Li et al. [10] designed a training-free Contrastive Decoding (CD) strategy, where a large expert model is coordinated with a small amateur model. By leveraging the maximization of log-probability differences subject to plausibility constraints, the proposed method effectively mitigates failure modes, i.e., repetition and incoherence, thereby improving inference efficiency and achieving superior performance compared to the expert model alone. Fu et al. [11] proposed a Collaborative Decoding via Speculation (CoS) strategy, aiming to significantly accelerate token generation while maintaining output quality. In the proposed framework, an alternate proposal mechanism is adopted to coordinate a small proposer model and a large verifier model, thereby improving the inference efficiency. Zimmer et al. [12] investigated a 'Mixture of Attentions' architecture where activations from a server-hosted large model are used to guide a local small model. Owing to the transmission of compressed activation tensors rather than raw data, this approach attains state-of-the-art latencies across varying network environments (e.g., 4G and 5G), which verifies the effectiveness of splitting inference workloads based on resource constraints. Fang et al. [13] developed a unified training framework based on DRL to facilitate autonomous decision-making for on-device LLMs regarding query offloading. By leveraging the DRL agent to adaptively decide the task execution locations, the proposed approach achieves a balance between local efficiency and global capability.

Recently, acceleration strategies have been investigated to obtain efficient solutions for specific modalities, which pave the way for optimizing models ranging from text-based LLMs to image-based diffusion models via speculative decoding or feature caching. However, these algorithms generally treat collaboration as static offloading mechanisms, which are hard to achieve an adaptive synergy between broad-spectrum intelligence and domain-specific expertise. In this work, a generic-specialized GenAI collaboration framework is proposed in edge-cloud computing networks. The cloud-hosted Generic Generative Visual Model (gGVM) is in charge of providing foundational knowledge, whereas the Specialized Generative Visual Models (sGVMs) deployed at the edge execute industry-specific refinements. By leveraging this hierarchical collaboration, the proposed framework can bridge general capabilities with domain precision to optimize inference efficiency.

2.2 Agentic AI for Resource Allocation

Recent studies have leveraged agentic AI to optimize resource management and task scheduling in dynamic network environments. Dev et al. [14] developed an advanced framework for next-generation wireless networks integrated with Agentic AI. By deploying intelligent agents across the Control and User planes for real-time learning, this framework achieves efficient allocation of computational resources and energy while reducing operational expenditures. Addressing the constraints of edge computing, Zhang et al. [5] explored a framework for Edge General Intelligence (EGI) enabled by agentification, which utilizes decentralized task allocation and multi-agent collaboration mechanisms to effectively adapt to heterogeneous and dynamic edge resource environments. Mei et al. [15] designed AIOS, an LLM-enabled agent operation framework that uses a specialized kernel to isolate and manage agent resources. The kernel employs an agent scheduler and memory manager to resolve resource contention issues during concurrent agent execution for enhanced global execution efficiency. Amayuelas et al. [16] investigated the capabilities of LLMs in self-resource allocation, focusing on internal task distribution within multi-agent systems. A Planner-based approach outperforms a single Orchestrator in handling concurrent operations, enabling more efficient task assignment and improved agent utilization. Tong et al. [17] introduced the WirelessAgent framework. WirelessAgent simulates human cognitive processes, including perception, memory, planning, and action, to autonomously interpret user intent and optimize slice resource allocation for higher bandwidth utilization compared to traditional prompt-based methods.

The aforementioned literature demonstrates the potential of Agentic AI in automating complex network operations. However, prevalent Agentic AI frameworks primarily rely on standard DRL-based decision engines where the intrinsic trial-and-error mechanism often results in poor sample efficiency and a profound lack of semantic awareness. To address these issues, we study an LLM-enabled Agentic DRL framework tailored for generic-specialized generative visual model collaboration in edge computing, where a LLM-based agent with powerful context-aware reasoning capacity is used to eliminate semantic ambiguity and facilitate intelligent synergy between generic cloud-based and specialized edge-based models by a knowledge base. A detailed comparison highlighting the differences between our approach and the main related schemes is provided in Table 1.

Just Accepted

Table 1 Differences Between Our Approach and The Main Related Schemes

+ Related Works	Visual GenAI Context	Generic-Specialized Model Synergy	LLM-enabled Semantic Awareness	Resource Orchestration	Long-Term Stability
[5]	×	✓	✓	✓	×
[9]	×	✓	×	×	×
[11]	×	✓	✓	×	×
[13]	×	✓	×	✓	✓
[14]	×	✓	✓	✓	×
[16]	×	×	✓	×	×
[17]	×	×	✓	✓	×
Our Approach	✓	✓	✓	✓	✓

3 System Model and Problem Formulation

3.1 Systemic Architecture

As depicted in Fig. 1, we develop an LLM-enabled Agentic DRL framework for generic-specialized GenAI collaboration in edge computing. By establishing a loop integrating perception, memory, cognition, and action modules, the proposed framework empowers the collaborative computing systems with autonomous decision-making capabilities, enabling it to proactively orchestrate generic-specialized model collaboration and optimize resource allocation in dynamic environments. Based on the closed loop, the agent first establishes situational awareness by continuously monitoring environmental states and interpreting task semantics via LLM. Subsequently, the LLM-enabled agent integrates Deep Reinforcement Learning (DRL) policies with numerical solvers to execute stability-aware decisions. Agentic DRL transcends the static nature of conventional models, enabling the system to autonomously adapt to stochastic generative inference demands through self-reflection and proactive reasoning facilitated by the internal LLM knowledge base.

The LLM-enabled Agentic DRL framework comprises four core modules working in concert to achieve the closed-loop optimization:

- **Perception Module**: Perception module establishes environmental awareness by continuously monitoring stochastic system conditions via the Perceive Resource State interface. To support situational analysis, the module synthesizes dynamic metrics, including task queue backlogs, generated content size, and real-time computational capacities, into a comprehensive state vector.
- **Memory Module**: Memory module maintains the Experience Memory reservoir to record historical transitions, including states, actions, and rewards, using a replay buffer mechanism. This component breaks the temporal correlation of training data, thereby stabilizing the learning process and enabling policy refinement based on long-term feedback.
- **Cognition Module**: Cognition module serves as the central Cognitive Core supported by an LLM Knowledge Base. Specifically, the Cognitive Core leverages the LLM's contextual reasoning capabilities to interpret the prompt context and provide prior knowledge to the DRL agent. To minimize computational overhead, we deploy a lightweight LLM as the cognitive module. Typically exhibiting fewer than 7 billion parameters, lightweight LLMs optimize structure and computational efficiency. The optimized architecture allows for maintaining performance levels while significantly reducing the demand for computing resources and storage space. Consequently, lightweight LLMs are highly effective in resource-constrained applications, ensuring the system meets the strict real-time and low-power constraints of dynamic networks. Recent frameworks demonstrate the feasibility of using lightweight models for semantic extraction and task scheduling [18]. Since the cognitive engine is exclusively tasked with processing short textual prompts to extract semantic embeddings, the execution latency and resource consumption incurred by the dedicated semantic extractor are negligible compared to visual GenAI inference tasks. By combining the derived semantic understanding with resource states, the Cognitive Core dynamically reasons to select between parallel or serial inference schemes and determines the optimal aggregation location.
- **Action Module**: This module performs resource orchestration by translating cognitive decisions into precise operations via a Numerical Solver. Leveraging KKT-based closed-form solutions, the solver optimizes transmission rates and frequency allocations, subsequently performing GenAI inference on physical hardware to close the

Just Accepted

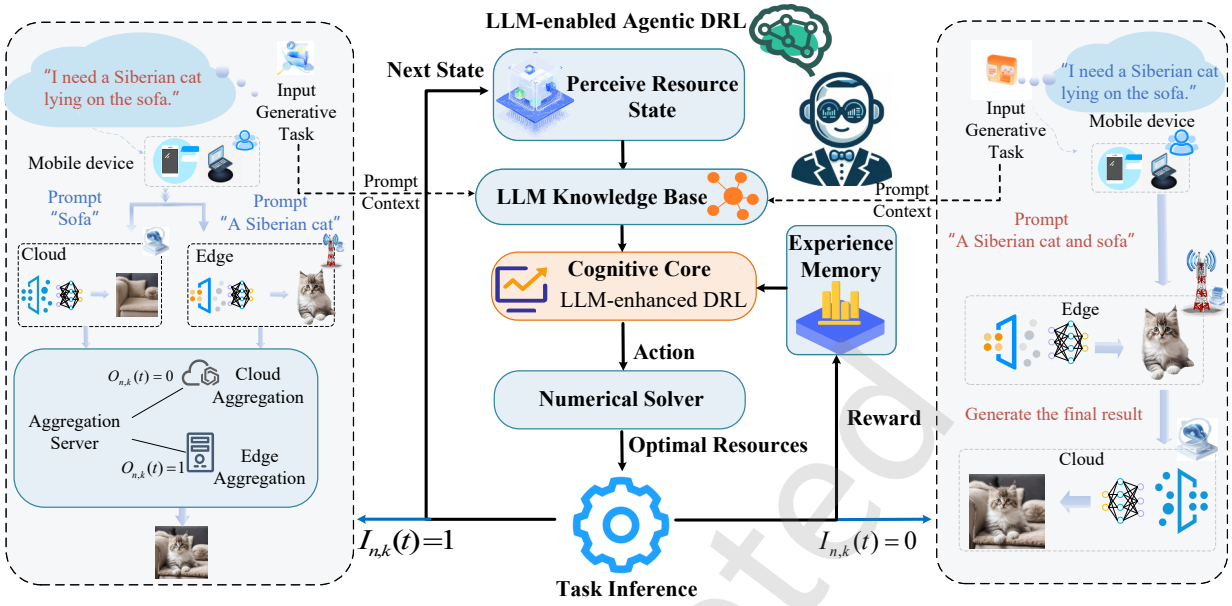


Fig. 1 System architecture of the Agentic DRL framework for generic-specialized GenAI collaboration in edge computing.

autonomous loop.

In this work, we consider an edge-cloud collaborative inference network including a set of edge servers and a central cloud, where Specialized Generative Visual Models (sGVMs) are deployed on edge servers and a generic Generative Visual Model (gGVM) is deployed in the central cloud. The set of edge servers is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$. Without loss of generality, edge-cloud collaborative inference network operates in discrete time slots $T = \{0, 1, \ldots, \mathcal{T}-1\}$. At the start of each time slot, $\mathcal{A}_n(t)$ denotes the set of newly arrived inference tasks at edge server n. For GenAI inference task $k \in \mathcal{A}_n(t)$, the Agent selects a parallel scheme by setting the binary variable $I_{n,k}(t) = 1$, or a serial scheme by setting $I_{n,k}(t) = 0$, as specifically detailed in the task execution flow in Fig. 1. In the parallel mode, the Agent additionally determines the aggregation location via $O_{n,k}(t)$, where $O_{n,k}(t) = 1$ denotes edge aggregation and $O_{n,k}(t) = 0$ denotes cloud aggregation, constrained by $I_{n,k}(t) \geqslant O_{n,k}(t)$. The Agent's objective is to minimize inference latency while ensuring service continuity. To facilitate the subsequent problem formulation, the key notations used in this paper are summarized in Table 2.

3.2 GenAI Task Model

The visual GenAI inference tasks uploaded by users can be offloaded to the edge layer or the cloud layer. In time slot t, the data size of the generated image of GenAI inference task k on edge server n can be represented as,

$$\varsigma_{n,k}(t) = \beta \cdot g^2_{n,k}(t), \tag{1}$$

where β denotes the bit depth of the generated image (i.e., the storage space occupied by each pixel, measured in bits/pixel) [19] [20] [21], and $g_{n,k}(t)$ (in pixels) is the computational resolution of the image generated by GenAI inference task k on edge server n, and the total number of pixels of the image is denoted as $g^2_{n,k}(t)$.

When the GenAI application specifies the image quality, the data size of the intermediate generation results transmitted between the edge and cloud layers is essentially the same as that of the final generation results, which can be expressed as $\varsigma_{n,k}(t)$.

3.3 Communication model

The maximum transmission rate of the wired link connecting edge server n and the central cloud is represented by $R_{n,1}$. In time slot t, let $r_{n,k,1}(t)$ denote the edge-to-cloud transmission rate allocated to GenAI inference task k arriving at edge server n. Thus, we can calculate the transmission time for sending the generation result of GenAI inference task k from edge server n to the cloud as,

$$\mathcal{T}_{n,k,1}(t) = \frac{\varsigma_{n,k}(t)}{r_{n,k,1}(t)}. \tag{2}$$

The transmission resources of edge server n allocated to GenAI inference task k are subject to the following constraints,

$$0 \leqslant r_{n,k,1}(t) \leqslant R_{n,1}, \tag{3}$$

Just Accepted

Table 2 Summary of Key Notations

Notation	Description	Unit
$\mathcal{N}=\{1,2,\ldots,N\}$	Index set of edge servers	N/A
$\mathcal{T}$	Total number of discrete time slots	N/A
$T=\{0,1,\ldots\mathcal{T}-1\}$	Index set of discrete time slots	N/A
$\mathcal{A}_n(t)$	Set of newly arrived GenAI inference tasks at edge server n in time slot t	N/A
$I_{n,k}(t)$	Binary indicator for inference scheme (1 for parallel, 0 for serial)	N/A
$O_{n,k}(t)$	Binary indicator for aggregation location (1 for edge, 0 for cloud)	N/A
$\varsigma_{n,k}(t)$	Data size of the generated image of task k on edge server n	bits
β	The bit depth of the generated image	bits/pixel
$g_{n,k}(t)$	Computational resolution of the generated image	pixels
$R_{n,1}, R_{n,2}$	Maximum transmission rates of the edge-to-cloud and cloud-to-edge links	bps
$r_{n,k,1}(t), r_{n,k,2}(t)$	Allocated transmission rate for task k (edge-to-cloud and cloud-to-edge)	bps
$\mathcal{T}_{n,k,1}(t), \mathcal{T}_{n,k,2}(t)$	Transmission delay for task k (edge-to-cloud and cloud-to-edge)	s
$\mathcal{C}_{n,k}(t)$	Computation requirement for generating task k	FLOPs
U_0	Computation intensity (FLOPs required per bit)	FLOPs/bit
$f_{n,k,c1}(t), f_{n,k,g1}(t)$	CPU and GPU capacities allocated to task k on edge server n	FLOPS
$f_{n,k,c2}(t), f_{n,k,g2}(t)$	CPU and GPU capacities allocated to task k on the cloud server	FLOPS
$\mathcal{F}_{n,c}, \mathcal{F}_{n,g}$	Total available CPU and GPU capacities of edge server n	FLOPS
$\mathcal{F}_c, \mathcal{F}_g$	Total available CPU and GPU capacities of the cloud server	FLOPS
$\varkappa_{n,k,1}(t), \varkappa_{n,k,2}(t)$	Fraction of workload assigned to the CPU on edge server n and the cloud	N/A
$\mathcal{D}_{n,k,1}(t), \mathcal{D}_{n,k,2}(t)$	Computation time of executing task k on edge server n and the cloud	s
$Q_n(t), Q_c(t)$	Computational backlog of the task queue at edge server n and the cloud	bits
$\psi_n(t), \psi_c(t)$	Task workload entering the edge and cloud task buffers in time slot t	bits
$\mu_n(t), \mu_c(t)$	Service capacity of the edge and cloud servers in time slot t	bits
τ	Time slot duration	s
χ_{agg}	Aggregation cost ratio relative to single-image generation	N/A
$\mathcal{D}^{agg}_{n,k,1}(t), \mathcal{D}^{agg}_{n,k,2}(t)$	Total response time for the aggregation process on the edge server and the core cloud	s
$\mathcal{T}_{n,k,p}(t), \mathcal{T}_{n,k,s}(t)$	Computation delay of the parallel and serial inference schemes	s
$\mathcal{M}_{n,k}(t)$	Overall inference time of task k	s
V	Lyapunov control parameter balancing delay and queue stability	bit^2/s
$s(t)$	Network state in time slot t	N/A
$\varpi(t)$	Discrete scheduling action in time slot t	N/A
$\zeta(t)$	Reward function evaluated by the LLM-enabled Agent	N/A
$\theta^P_\pi, \theta^P_{Q_1}, \theta^P_{Q_2}$	Network parameters of the primary actor and twin critic networks	N/A
$\theta^T_\pi, \theta^T_{Q_1}, \theta^T_{Q_2}$	Network parameters of the target actor and twin critic networks	N/A
γ, ω, ξ	Discount factor, soft update weight, and delayed update interval for DRL-ITS	N/A
$\mathcal{R}, J$	Experience replay buffer and mini-batch size for sampling	N/A

and

$$\sum_{k\in\mathcal{A}_n(t)} r_{n,k,1}(t) \leqslant R_{n,1}. \tag{4}$$

Similarly, the maximum transmission capacity from the central cloud to edge server n is denoted by $R_{n,2}$. Based on the cloud-to-edge transmission rate $r_{n,k,2}(t)$ assigned to generation task k at edge server n, we can determine the following transmission delay,

$$\mathcal{T}_{n,k,2}(t) = \frac{\varsigma_{n,k}(t)}{r_{n,k,2}(t)}. \tag{5}$$

It is worth noting that $r_{n,k,2}(t)$ has the following constraints,

$$0 \leqslant r_{n,k,2}(t) \leqslant R_{n,2}, \tag{6}$$

and

$$\sum_{k\in\mathcal{A}_n(t)} r_{n,k,2}(t) \leqslant R_{n,2}, \forall n \in \mathcal{N}, t \in T. \tag{7}$$

3.4 Computation Model

Leveraging data parallelism [19],[22],[23], we distribute the image generation workload of the edge and cloud servers across their CPU and GPU resources to enable concurrent

Just Accepted

processing. The computation requirement for generating task k of edge server n is formulated as,

$$\mathcal{C}_{n,k}(t) = U_0 \times \varsigma_{n,k}(t), \tag{8}$$

where $U_0(t)$ denotes the computation intensity floating-point operations (FLOPs) per bit on edge server n.

The GenAI inference tasks are performed locally on edge servers or transmitted to the central cloud for remote processing. The former involves the execution of generative tasks on edge servers, whereas the latter involves execution on cloud server.

1) Generative Task Computing Delay on Edge Server: Edge server n divides the image-generation workload of task k between the CPU and GPU. The parameter $\varkappa_{n,k,1}(t)$ represents the fraction of the workload assigned to the CPU, while the remaining portion is executed on the GPU. This allocation strategy is motivated by the observation that the CPU demonstrates enhanced performance in logical reasoning tasks. Accordingly, the CPU and GPU are assigned workloads of size $\varkappa_{n,k,1}(t) \times \varsigma_{n,k}(t)$ and $(1 - \varkappa_{n,k,1}(t)) \times \varsigma_{n,k}(t)$, respectively. In time slot t, let $f_{n,k,c1}(t)$ and $f_{n,k,g1}(t)$ represent the computation capacities of CPUs and GPUs allocated to GenAI inference task k on edge server n. The computation time of executing inference task k on edge server n is given by the maximum of CPU and GPU parallel processing times,

$$\mathcal{D}_{n,k,1}(t) = \max\left\{\frac{\varkappa_{n,k,1}(t)\mathcal{C}_{n,k}(t)}{f_{n,k,c1}(t)}, \frac{(1-\varkappa_{n,k,1}(t))\mathcal{C}_{n,k}(t)}{f_{n,k,g1}(t)}\right\}. \tag{9}$$

By distributing the image-generation workload across CPU and GPU resources for concurrent processing, the total inference latency is determined by the maximum execution time of the two parallel paths [24].

The total CPU and GPU computational capacities of edge server n are denoted as $\mathcal{F}_{n,c}$ and $\mathcal{F}_{n,g}$. Constrained by the total computational capacities of edge server n, the CPU and GPU allocations $f_{n,k,c1}(t)$ and $f_{n,k,g1}(t)$ must satisfy the following conditions,

$$0 \leqslant f_{n,k,c1}(t) \leqslant \mathcal{F}_{n,c}, n \in \mathcal{N}, k \in \mathcal{A}_n(t), t \in T, \tag{10}$$

$$\sum_{k\in\mathcal{A}_n(t)} f_{n,k,c1}(t) \leqslant \mathcal{F}_{n,c}, \forall n \in \mathcal{N}, t \in T, \tag{11}$$

and

$$0 \leqslant f_{n,k,g1}(t) \leqslant \mathcal{F}_{n,g}, \forall n \in \mathcal{N}, k \in \mathcal{A}_n(t), t \in T, \tag{12}$$

$$\sum_{k\in\mathcal{A}_n(t)} f_{n,k,g1}(t) \leqslant \mathcal{F}_{n,g}. \tag{13}$$

2) GenAI Inference Task Execution Time on Cloud Server: During time slot t, the cloud server partitions the image generation workload of task k into two components for parallel execution. The allocation ratio $\varkappa_{n,k,2}(t)$ specifies the portion of the workload handled by the CPU, while the remaining $(1 - \varkappa_{n,k,2}(t))$ is offloaded to the GPU. As a result, the CPU and GPU are assigned workload amounts of $\varkappa_{n,k,2}(t) \times \varsigma_{n,k}(t)$ and $(1-\varkappa_{n,k,2}(t)) \times \varsigma_{n,k}(t)$, respectively.

Let $f_{n,k,c2}(t)$ and $f_{n,k,g2}(t)$ represent the CPU and GPU computational capacities, measured in floating-point operations per second (FLOPS) [24], allocated by the cloud server in time slot t to task k from edge server n. Similar to the edge computation model, the cloud partitions the workload into CPU and GPU components based on the allocation ratio $\varkappa_{n,k,2}(t)$. Then, the total computation time of executing inference task k on cloud server n can be obtained from,

$$\mathcal{D}_{n,k,2}(t) = \max\left\{\frac{\varkappa_{n,k,2}(t)\mathcal{C}_{n,k}(t)}{f_{n,k,c2}(t)}, \frac{(1-\varkappa_{n,k,2}(t))\mathcal{C}_{n,k}(t)}{f_{n,k,g2}(t)}\right\}. \tag{14}$$

Here, $\mathcal{F}c$ and $\mathcal{F}g$ denote the total available CPU and GPU computational capacities (in FLOPS) of the cloud server. The allocated CPU and GPU capacities are subject to the following resource constraints,

$$0 \leqslant f_{n,k,c2}(t) \leqslant \mathcal{F}_c, \tag{15}$$

$$\sum_{k\in\mathcal{A}_n(t)} f_{n,k,c2}(t) \leqslant \mathcal{F}_c, \tag{16}$$

$$0 \leqslant f_{n,k,g2}(t) \leqslant \mathcal{F}_g, \tag{17}$$

$$\sum_{k\in\mathcal{A}_n(t)} f_{n,k,g2}(t) \leqslant \mathcal{F}_g. \tag{18}$$

3.5 Task Queueing Model

Based on the task queuing model in [25], we establish a task buffer at each edge server to manage the stochastic requests. Let $Q_n(t)$ denote the computational backlog of the task queue at edge server n, measured in terms of the equivalent data size (in bits) required to be processed. Initialized with $Q_n(0) = 0$, the queue dynamics follow the standard evolution equation,

$$Q_n(t+1) = \max\{Q_n(t) - \mu_n(t), 0\} + \psi_n(t), \tag{19}$$

where $\psi_n(t) = \sum_{k\in\mathcal{A}_n(t)}[\tau \cdot O_{n,k}(t)r_{n,k,2}(t)) + I_{n,k}(t)(\varsigma_{n,k}(t)]$ denotes the task workload entering the edge-side buffer in time slot t. The service capacity can be achieved as $\mu_n(t) = \tau \cdot \sum_{k\in\mathcal{A}_n(t)}[(f_{n,k,c1}(t) + f_{n,k,g1}(t))/U_0 + (1 - I_{n,k}(t)O_{n,k}(t))r_{n,k,1}(t)]$, where τ represents the length of the time slot t. As a result, $(1 - I_{n,k}(t)O_{n,k}(t))r_{n,k,1}(t)$ indicates the edge-to-cloud uplink transmission that migrates

Just Accepted

workload from the edge task buffer to the cloud in the parallel mode with cloud-side aggregation.

Analogously, the evolution of the cloud-side task backlog is captured by $Q_c(t)$. With the initial condition $Q_c(0) = 0$, the update rule for $Q_c(t)$ is formulated as follows,

$$Q_c(t+1) = \max\{Q_c(t) - \mu_c(t), 0\} + \psi_c(t), \tag{20}$$

where $\psi_c(t) = \sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} [I_{n,k}(t)\varsigma_{n,k}(t) + \tau \cdot I_{n,k}(t)(1 - O_{n,k}(t))r_{n,k,1}(t) + \tau \cdot (1 - I_{n,k}(t))r_{n,k,1}(t)]$ represents the cloud-side workload volume admitted in time slot t. Moreover, $\mu_c(t) = \tau \cdot \sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} [(f_{n,k,c2}(t) + f_{n,k,g2}(t))/U_0 + I_{n,k}(t)O_{n,k}(t)r_{n,k,2}(t)]$ denotes the the computation capacity of the core cloud in terms of data bits per time slot.

Based on Equation (19) and (20), to ensure the stability of the edge-cloud collaborative network and keep the GenAI inference task backlogs bounded, we enforce the following strong stability constraint on all task queues,

$$\limsup_{\mathcal{T} \to \infty} \frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}\Big[\sum_{n=1}^{\mathcal{N}} Q_n(t) + Q_c(t)\Big] < \infty. \tag{21}$$

This constraint requires the Agentic Scheduler to learn a policy where the average service rate is sufficient to handle the average arrival rate of inference requests. Consistent with recent studies on edge-cloud collaborative scheduling [26], enforcing this Lyapunov-based queue stability constraint ensures that the GenAI inference task backlog remains bounded over time, thereby preventing system congestion and guaranteeing service continuity [27].

3.6 Problem Formulation

Execution delay is an important metric for the inference of GenAI inference tasks and will be adopted to optimize the inference scheme of the edge–cloud collaborative system. Owing to the distinct execution characteristics of parallel and serial inference schemes, the corresponding latency must be analyzed for each scheme individually. Accordingly, the processing delay of task k under each inference scheme can be derived.

1) Computation Delay of Parallel Inference Scheme: When $I_{n,k}(t) = 1$, the inference of GenAI inference task k follows a parallel scheme, where both the edge server n and the core cloud receive the textual prompts and independently generate intermediate results. The computation delay is determined by the maximum processing time between the edge and the core cloud. Following this, the intermediate results are aggregated either at the cloud or the edge, depending on the aggregation indicator $O_{n,k}(t)$.

- If $O_{n,k}(t) = 0$, aggregation is completed on the core cloud. Initially, the edge server n sends its generated intermediate result to the core cloud. Since aggregation mainly involves lightweight feature fusion or joint reasoning, its computational cost is significantly smaller than that of single-image generation [28]. Let χ_{agg} denote the aggregation cost ratio. Accordingly, the total computation delay for this aggregation process is expressed as,

$$\mathcal{D}^{agg}_{n,k,2}(t) = \chi_{agg} \cdot \mathcal{D}_{n,k,2}(t), \tag{22}$$

  where $0 < \chi_{agg} \ll 1$. Eventually, the final aggregated result is transmitted back from the cloud to the edge server n.
- If $O_{n,k}(t) = 1$, edge server n performs the aggregation process.Then, the core cloud should send the intermediate results of GenAI inference tasks to edge server n. The total response time for the aggregation process is given by,

$$\mathcal{D}^{agg}_{n,k,1}(t) = \chi_{agg} \cdot \mathcal{D}_{n,k,1}(t). \tag{23}$$

In the considered generative tasks, the size of the initial textual prompt or control signal is significantly smaller than the generated visual content and intermediate representations. Consequently, the transmission latency for dispatching the prompt from the user to the servers is considered negligible. As a result, the computation delay of the parallel inference scheme can be obtained by,

$$\begin{aligned}\mathcal{T}_{n,k,p}(t) = &\max\{\mathcal{D}_{n,k,1}(t), \mathcal{D}_{n,k,2}(t)\} \\ &+ O_{n,k}(t)[\mathcal{T}_{n,k,2}(t) + \mathcal{D}^{agg}_{n,k,1}(t)] \\ &+ [1 - O_{n,k}(t)][\mathcal{T}_{n,k,1}(t) + \mathcal{D}^{agg}_{n,k,2}(t) + \mathcal{T}_{n,k,2}(t)],\end{aligned} \tag{24}$$

where the $\max(\cdot)$ operator identifies the concurrent execution bottleneck as the parallel latency is governed by the slower of the two processing paths, while the remaining terms account for the location-dependent transmission and lightweight aggregation overheads determined by $O_{n,k}(t)$.

2) The Computation Delay of Serial Inference Scheme: In time slot t, when $I_{n,k}(t) = 0$, the edge server n first generates an intermediate result upon receiving the textual prompt. Subsequently, edge server n uploads the intermediate results of GenAI inference tasks to the core cloud. Finally, the final generated result is transmitted back to the edge server n. The computation time of executing GenAI inference tasks on edge server n can be given by

$$\mathcal{T}_{n,k,s}(t) = \mathcal{D}_{n,k,1}(t) + \mathcal{T}_{n,k,1}(t) + \mathcal{D}_{n,k,2}(t) + \mathcal{T}_{n,k,2}(t). \tag{25}$$

In summary, the overall inference time of GenAI inference

Just Accepted

tasks during the time slot t can be achieved by,

$$\mathcal{M}_{n,k}(t) = I_{n,k}(t)\mathcal{T}_{n,k,p}(t) + (1 - I_{n,k}(t))\mathcal{T}_{n,k,s}(t). \tag{26}$$

Finally, to minimize the total response time of GenAI inference tasks, problem $\mathcal{P}_1$ can be obtained by,

$$\mathcal{P}_1 : \min \lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}-1} \sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathbb{E}[\mathcal{M}_{n,k}(t)] \tag{27}$$

$$\text{s.t. } (3), (4), (6), (7), (10), (11), (12), (13), (15), (16), (17), (18)$$

$$I_{n,k}(t), O_{n,k}(t) \in \{0, 1\}, \tag{28}$$

$$O_{n,k}(t) \leqslant I_{n,k}(t), \forall n \in \mathcal{N}, k \in \mathcal{A}_n(t). \tag{29}$$

Constraint (3) ensures that the allocated transmission rate $r_{n,k,1}(t)$ does not exceed the total available transmission rate. Constraint (4) ensures that the aggregate of allocated transmission rates does not exceed the total available edge-to-cloud link capacity $R_{n,1}(t)$. Constraint (6) represents the value range of $r_{n,k,2}(t)$. Constraint (7) restricts the total rate allocation such that it does not exceed the aggregate cloud-to-edge link capacity $R_{n,2}(t)$. Constraint (10) defines the admissible CPU resource range for the edge server. Constraint (11) states that, during the time slot t, the aggregate computational demands of all tasks on edge server n do not exceed the server's maximum CPU capacity. Constraint (12) specifies the corresponding GPU resource range. Constraint (13) imposes a CPU-like requirement on the cumulative GPU computational resources, with these cycles being less than or equal to the edge server's total GPU capacity. The value range of available computing resources for the cloud server's CPU and GPU is constrained. Specifically, it is restricted by Constraints (15) and (17). Constraint (16) mandates that the computation capacities allocated to all tasks on the cloud server can not exceed the total available CPU frequencies of the central cloud. Constraint (18) dictates that the GPU frequencies shared by all tasks operating on the central cloud do not exceed the total available GPU frequency. In addition, the zero-one indicator constraints are defined by Constraint (28), while Constraint (29) imposes stricter limitations to narrow the feasible region of the optimization problem.

3.7 Problem Transformation

Equations (19) and (20) facilitate the definition of a Lyapunov function, expressed as

$$\mathcal{L}(\Theta(t)) = \frac{1}{2}\sum_{n=1}^{\mathcal{N}} Q_n^2(t) + \frac{1}{2}Q_c^2(t), \tag{30}$$

where $\Theta(t)$ is the vector of physical task queues tracking the accumulated data backlog.

To characterize the variation in the queue accumulation over two consecutive time slots, we have the following Lyapunov drift function,

$$\Delta(\Theta(t)) = \mathbb{E}[\mathcal{L}(\Theta(t+1)) - \mathcal{L}(\Theta(t))|\Theta(t)]. \tag{31}$$

Specifically, the Lyapunov drift $\Delta(\Theta(t))$ represents the expected growth of task backlogs. By minimizing this drift, the system effectively suppresses queue accumulation, which, according to Little's Law, reduces the long-term cumulative queuing delay.

To strike an optimal balance between the cumulative queuing delay and the immediate inference latency, we formulate the drift-plus-penalty function $\mathcal{GK}(t)$ as

$$\mathcal{GK}(t) = \Delta(\Theta(t)) + V \cdot \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)|\Theta(t)], \tag{32}$$

where $V \geqslant 0$ is a non-negative weight constant that balances the trade-off between decreasing the response time of GenAI inference tasks and stabilizing the physical task queues. Then, we have the following lemma and theorem with respect to Laypunov parameter V.

Lemma 1. *The upper bound of $\mathcal{GK}(t)$ can be expressed as*

$$\mathcal{GK}(t) \leqslant B + V \cdot \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)|\Theta(t)] + \sum_{n=1}^{\mathcal{N}} Q_n(t)[\psi_n(t) - \mu_n(t)] + Q_c(t)[\psi_c(t) - \mu_c(t)], \tag{33}$$

where B is a constant,

$$B = \sum_{n=1}^{\mathcal{N}} \frac{\psi_{n,max}^2 + \mu_{n,max}^2}{2} + \frac{\psi_{c,max}^2 + \mu_{c,max}^2}{2}. \tag{34}$$

Proof. See Appendix A.

Theorem 1. *The time-averaged expected inference delay is bounded by*

$$\lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)] \leqslant C^* + \frac{B}{V}, \tag{35}$$

where C^* is the theoretical minimum inference delay.

Proof. See Appendix B.

Theorem 2. The time-averaged expected total physical queue backlog is upper bounded by a value proportional to V, satisfying

$$\lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}\left[\sum_{n=1}^{N} Q_n(t) + Q_c(t)\right] \leqslant \frac{B + V\mathcal{M}^{max}}{\epsilon}, \tag{36}$$

where $\epsilon > 0$ is a strictly positive constant representing the system stability margin (Slater condition), and $\mathcal{M}^{max}$ is the

Just Accepted

maximum possible total inference delay in a time slot.

Proof. See Appendix C.

The performance bounds in Theorems 1–2 are derived for an ideal opportunistic controller that performs per-slot minimization of the Lyapunov drift-plus-penalty upper bound (equivalently, minimizes $\mathcal{Z}(t)$ in (37) given the queue state $Q(t)$), which is standard in Lyapunov optimization. In implementation, our proposed algorithm uses a trained DRL policy to approximate this per-slot minimization for the inference-scheme selection subproblem. Therefore, Theorems 1–2 should be interpreted as theoretical benchmarks/design guidance for the proposed framework, rather than strict guarantees for every learned policy realization. Note that B can be determined at the beginning of time slot t and remains unaffected by the inference scheme selection within the same time slot. Consequently, we aim to minimize the following equation,

$$\begin{aligned}\mathcal{Z}(t) = &\sum_{n=1}^{\mathcal{N}} Q_n(t)[\psi_n(t) - \mu_n(t)] + Q_c(t)[\psi_c(t) - \mu_c(t)] \\ &+ V \cdot \sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} \mathcal{M}_{n,k}(t).\end{aligned} \tag{37}$$

To maximize per-slot inference efficiency, Problem $\mathcal{P}_1$ is equivalently converted into the subsequent optimization model,

$$\begin{aligned}\mathcal{P}_2 : &\min \mathcal{Z}(t) \\ &\text{s.t. } (21), (3), (4), (6), (7), (10), (11), (12), (13), \\ &\quad (15), (16), (17), (18)\end{aligned}$$

Constraint (21) ensures the stability of the physical task data queue, thereby guaranteeing that all arriving inference requests are processed within a finite time delay.

Here, the queuing delay refers to the waiting time induced by the computational backlogs $Q_n(t)$ and $Q_c(t)$, while the per-slot service time $M_{n,k}(t)$ accounts for computation and transmission delays under the selected inference scheme. Specifically, the drift term $\Delta(\Theta(t))$ minimizes queue backlogs to implicitly reduce the queuing delay based on Little's Law, whereas the penalty term $V \cdot \sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)$ explicitly minimizes the execution delay by optimizing resource allocation and inference scheme selection.

4 Problem Transformation and Design of Resources Allocation

Based on the constructed DRL agent, we design a DRL-based Inference Task Scheduling (DRL-ITS) algorithm to address the problem $\mathcal{P}_2$. First, problem $\mathcal{P}_2$ is decomposed into multiple sub-problems, where numerical methods are applied to solve the resource allocation sub-problems. Subsequently, a DRL approach is employed to address the inference scheme selection sub-problem. Specifically, the DRL agent is trained with the per-slot objective $Z(t)$ in (37) as the reward surrogate, aiming to approximate the ideal opportunistic minimization policy.

4.1 Design of Resource Allocation

Problem $\mathcal{P}_2$ is characterized by non-convexity and mixed-integer nonlinear structures. Due to this complexity, solving it optimally within polynomial time is extremely challenging. Therefore, we decompose problem $\mathcal{P}_2$ into several sub-problems to better handle its complexity, where DRL is employed to determine the GenAI inference scheme, and fast numerical methods are incorporated to optimally solve the remaining components.

Provided that the inference scheme has been determined, we can derive an expanded form of the objective function in $\mathcal{P}_2$ by substituting the relevant formulations from Sections III and IV,

$$\begin{aligned}&V \cdot \sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} \mathcal{M}_{n,k}(t) + \sum_{n=1}^{\mathcal{N}} Q_n(t)[\psi_n(t) - \mu_n(t)] \\ &+ Q_c(t)[\psi_c(t) - \mu_c(t)] \\ =&V \cdot \Bigg\{ \underbrace{\Big\{ \sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} \Big[(1 - I_{n,k}(t) O_{n,k}(t)) \frac{\varsigma_{n,k}(t)}{r_{n,k,1}(t)} \Big]}_{\text{part I}} \\ &+ \underbrace{\sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} \frac{\varsigma_{n,k}(t)}{r_{n,k,2}(t)}}_{\text{part II}} + \underbrace{\sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} [y_{n,k}(t) \mathcal{D}_{n,k,1}(t)]}_{\text{part III}} \\ &+ \underbrace{\sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} [z_{n,k}(t) \mathcal{D}_{n,k,2}(t)]}_{\text{part IV}} \\ &+ \sum_{n=1}^{\mathcal{N}} \sum_{k \in \mathcal{A}_n(t)} [I_{n,k}(t)(\max\{\mathcal{D}_{n,k,1}(t), \mathcal{D}_{n,k,2}(t)\})] \Big\} \Bigg\} \\ &+ \sum_{n=1}^{\mathcal{N}} Q_n(t)[\psi_n(t) - \mu_n(t)] + Q_c(t)[\psi_c(t) - \mu_c(t)],\end{aligned} \tag{38}$$

where $y_{n,k}(t) = \chi_{agg} I_{n,k}(t) O_{n,k}(t) - I_{n,k}(t) + 1$ and $z_{n,k}(t) = \chi_{agg} I_{n,k}(t)(1 - O_{n,k}(t)) + 1 - I_{n,k}(t)$. It can be observed that problem $\mathcal{P}_2$ inherently has a decomposable nature with respect to the variables $r_{n,k,1}(t)$, $r_{n,k,2}(t)$, $\mathcal{D}_{n,k,1}(t)$ and $\mathcal{D}_{n,k,2}(t)$. In particular, P2 can be broken down further into 4 parts (labeled part I to IV in (38)), with each part corresponding to a separate sub-problem.

Just Accepted

1) Edge-to-cloud Transfer Rate Allocation: Based on part I of Equation (38), we can construct the transmission resource allocation problem of edge server n as follows,

$$\begin{aligned}\mathcal{P}_{2.1}: \min & \sum_{k\in\mathcal{A}_n(t)} x_{n,k}(t)\frac{\varsigma_{n,k}(t)}{r_{n,k,1}(t)}\\ & \text{s.t. } (4),\\ & 0 \leqslant r_{n,k,1}(t) \leqslant x_{n,k}(t)R_{n,1}, \forall k \in \mathcal{A}_n(t), \end{aligned} \tag{39}$$

where $x_{n,k}(t) = 1 - I_{n,k}(t)O_{n,k}(t)$. In this context, Constraint (39) dictates that the assigned transmission rate $r_{n,k,1}(t)$ must stay within the available limit and is based on Constraint (3). Meanwhile, Constraint (4) further specifies that the sum of all such rates cannot exceed the total capacity of the edge-to-cloud link. Note that when $x_{n,k}(t) = 0$, constraint (39) enforces $r_{n,k,1}(t) = 0$, and the corresponding term $x_{n,k}(t)\varsigma_{n,k}(t)/r_{n,k,1}(t)$ is defined as zero.

Next, **Lemma 2** derives and proves the optimal transmission rate $r^*_{n,k,1}(t)$.

Lemma 2. Under the specified constraints, i.e., (4) and (39), the optimal $r_{n,k,1}(t)$ in sub-problem $\mathcal{P}_{2.1}$ can be explicitly formulated as,

$$r^*_{n,k,1}(t) = \frac{R_{n,1}\sqrt{x_{n,k}(t)\varsigma_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{x_{n,j}(t)\varsigma_{n,j}(t)}}. \tag{40}$$

Moreover, if $\sum_{j\in\mathcal{A}_n(t)}\sqrt{x_{n,j}(t)\varsigma_{n,j}(t)} > 0$, the capacity constraint is tight, i.e., $\sum_{k\in\mathcal{A}_n(t)} r^*_{n,k,1}(t) = R_{n,1}$.

Proof. See Appendix D.

2) Cloud-to-edge Transfer Rate Allocation: Moreover, the transmission resource allocation problem from the core cloud to edge server n is derived from part II of Equation (38) as follows,

$$\begin{aligned}\mathcal{P}_{2.2}: \min & \sum_{k\in\mathcal{A}_n(t)} \frac{\varsigma_{n,k}(t)}{r_{n,k,2}(t)}\\ & \text{s.t. } (6), (7).\end{aligned}$$

Similar to the derivation of $r^*_{n,k,1}(t)$ in **Lemma 2**, by exploiting the convex structure of problem $\mathcal{P}_{2.2}$ and satisfying the KKT conditions, the analytical expression of the optimal transmission rate $r^*_{n,k,2}(t)$ is derived as,

$$r^*_{n,k,2}(t) = \frac{R_{n,2}\sqrt{\varsigma_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{\varsigma_{n,j}(t)}}. \tag{41}$$

3) Edge Computing Resource Allocation: Drawing upon the analysis in part III of Equation (38), we formulate computation resource allocation problem of edge server n as follows,

$$\begin{aligned}\mathcal{P}_{2.3}: \min & \sum_{k\in\mathcal{A}_n(t)} y_{n,k}(t)\mathcal{D}_{n,k,1}(t)\\ & \text{s.t. } (10), (11), (12), (13).\end{aligned}$$

To avoid resource under-utilization, the computation time allocated to CPU and GPU for each task should be balanced [19]. Thus, we introduce the following constraint to ensure that their execution durations are approximately synchronized,

$$\frac{\varkappa_{n,k,1}(t)}{f_{n,k,c1}(t)} = \frac{1-\varkappa_{n,k,1}(t)}{f_{n,k,g1}(t)}, \tag{42}$$

where

$$\varkappa_{n,k,1}(t) = \frac{f_{n,k,c1}(t)}{f_{n,k,c1}(t) + f_{n,k,g1}(t)}. \tag{43}$$

Substitute (43) back into (9),

$$\mathcal{D}_{n,k,1}(t) = \frac{\mathcal{C}_{n,k}(t)}{f_{n,k,c1}(t) + f_{n,k,g1}(t)}. \tag{44}$$

The optimal frequency allocation for CPU and GPU under the delay-only objective is derived in Appendix E using KKT conditions, where the total computing frequency is allocated proportional to the square root of the weighted workload of each task. Accordingly, the optimal resource allocation on the edge server n in time slot t can be written as,

$$f^*_{n,k,c1}(t) = \frac{F_{n,c}\sqrt{y_{n,k}(t)\mathcal{C}_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{y_{n,j}(t)\mathcal{C}_{n,j}(t)}}, \tag{45}$$

and

$$f^*_{n,k,g1}(t) = \frac{F_{n,g}\sqrt{y_{n,k}(t)\mathcal{C}_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{y_{n,j}(t)\mathcal{C}_{n,j}(t)}}. \tag{46}$$

4) Cloud Computing Resource Allocation: Guided by the analysis in (38), Part IV, the edge resource allocation task for the cloud server is characterized as follows,

$$\begin{aligned}\mathcal{P}_{2.4}: \min & \sum_{k\in\mathcal{A}_n(t)} z_{n,k}(t)\mathcal{D}_{n,k,2}(t)\\ & \text{s.t. } (15), (16), (17), (18).\end{aligned}$$

Constraints analogous to those of edge servers are applied to prevent resource underutilization and to align their execution durations,

$$\frac{\varkappa_{n,k,2}(t)}{f_{n,k,c2}(t)} = \frac{1-\varkappa_{n,k,2}(t)}{f_{n,k,g2}(t)}. \tag{47}$$

Similarly, we have,

$$\mathcal{D}_{n,k,2}(t) = \frac{\mathcal{C}_{n,k}(t)}{f_{n,k,c2}(t) + f_{n,k,g2}(t)}. \tag{48}$$

Similar to the proof in Appendix E, the optimal CPU and GPU frequency allocation under the delay-only objective is obtained via the KKT conditions, with the total computing frequency allocated dynamically based on the square root of the weighted task workload. In time slot t, the optimal allocation results of the core cloud can be described by,

$$f^*_{n,k,c2}(t) = \frac{F_c\sqrt{z_{n,k}(t)\mathcal{C}_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{z_{n,j}(t)\mathcal{C}_{n,j}(t)}}, \tag{49}$$

Just Accepted

and

$$f^*_{n,k,g2}(t) = \frac{F_g\sqrt{z_{n,k}(t)\mathcal{C}_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{z_{n,j}(t)\mathcal{C}_{n,j}(t)}}. \quad (50)$$

4.2 DRL-based Inference Task Scheduling Algorithm

To solve problem $\mathcal{P}_2$ with high computation complexity, our DRL-ITS algorithm leverages an LLM-enabled Agentic DRL framework, built upon an extended Twin Delayed Deep Deterministic Policy Gradient (TD3) [29]structure. Specifically, the agent transcends passive execution by integrating an LLM-based knowledge base to interpret complex task semantics and environmental context, providing informed priors for decision-making. To maintain differentiability in the discrete action space for inference scheme selection, we integrate the Gumbel-Softmax reparameterization into the actor's output layer. Furthermore, to ensure fast convergence and avoid local optima, orthogonal initialization is applied to the weights of actor and critic networks. This approach ensures the Agentic Scheduler effectively learns the optimal serial-parallel inference mode and aggregation location decisions, while the remaining resource orchestration sub-problems are solved by efficient KKT-based numerical methods. An overview of the proposed DRL-ITS algorithm is provided in Fig. 2. It highlights how the LLM-enabled agent synergizes semantic reasoning with deep reinforcement learning to achieve autonomous coordination.

Task Inference Scheme Selector: Solving the rate and resource allocation sub-problems through numerical approaches yields the optimal $\mathcal{T}^*_{n,k,1}(t)$, $\mathcal{T}^*_{n,k,2}(t)$, $\mathcal{D}^*_{n,k,1}(t)$, and $\mathcal{D}^*_{n,k,2}(t)$. By substituting these variables into $\mathcal{P}_2$, the residual decision problem reduces to the GenAI inference scheme optimization, expressed as,

$$\mathcal{P}_{2.5}: \min \mathcal{Z}(t)$$
$$\text{s.t. } (21), (28), (29)$$

This sub-problem is modeled as a Markov Decision Process (MDP) [30] and solved by TD3. The MDP formulation is given as follows,

1) State: The network state in each time slot is represented by an observation vector describing its features and conditions. In each time slot, we denote the network state as $s(t) = \Big\{\{Q_n(t)\}, \{Q_c(t)\}, \{\varsigma_{n,k}(t)\}, R_{n,1}, R_{n,2}, \mathcal{F}_{n,c}, \mathcal{F}_{n,g}, \mathcal{F}_c, \mathcal{F}_g\Big\}$.

2) Action: Since the transmission rates and computing resource allocations are already obtained via numerical methods, the action set only includes GenAI inference scheme selection, denoted as

$$\varpi(t) = \Big\{I_{n,k}(t), O_{n,k}(t)\Big\}. \quad (51)$$

3) Reward: The reward $\zeta(t)$ is defined based on the drift-plus-penalty objective in (37),

$$\zeta(t) = -\mathcal{Z}(t). \quad (52)$$

Maximizing the reward function is equivalent to minimizing the $\mathcal{P}_2$ objective. Here $\mathcal{Z}(t)$ jointly accounts for the system latency cost and the Lyapunov drift associated with physical queue stability. Consequently, the reward function intrinsically incorporates system stability constraints through the drift-plus-penalty framework, rendering additional heuristic penalty shaping unnecessary.

To address the MDP formulation, the TD3 algorithm utilizes a primary actor network $\pi(s(t) \mid \theta^P_\pi)$ to generate the optimal action $\varpi(t)$ based on the environmental state $s(t)$ for determining the GenAI inference scheme. A core mechanism of this framework involves the use of two critic networks to independently evaluate Q-values, where the system selects the minimum estimate as the update target to reduce bias. These critic networks are parameterized by $\theta^P_{Q_1}$ and $\theta^P_{Q_2}$ respectively, providing estimates for the Q-value $y(t)$ based on the current state and action. To enhance training stability, the algorithm incorporates target networks $\pi^*(s(t) \mid \theta^T_\pi)$, $Q^*_1(s(t), \varpi(t) \mid \theta^T_{Q_1})$, and $Q^*_2(s(t), \varpi(t) \mid \theta^T_{Q_2})$ corresponding to the primary models. Furthermore, an experience replay buffer is implemented to archive historical transition tuples $(s(t), \varpi(t), \zeta(t), s(t+1))$ at each time slot t for subsequent learning.

The primary actor network π outputs the action $\varpi(t)$ from the state $s(t)$, where exploration noise σ^p sampled from a truncated normal distribution $\mathcal{N}(0, \eta^P)$ is added to enhance scheme selection. The perturbed policy $\tilde{\pi}(s(t))$ is expressed as,

$$\tilde{\pi}(s(t)) = \pi(s(t) \mid \theta^P_\pi) + \sigma^p. \quad (53)$$

For each training step, the parameters $\{\theta^P_\pi, \theta^P_{Q_1}, \theta^P_{Q_2}\}$ for the actor and two critics are adjusted using J transitions randomly sampled from $\mathcal{R}$. Following the methodology in [31], the sampled policy gradient update rules are formulated as follows,

$$\theta^P_\pi = \theta^P_\pi - \frac{\rho_\pi}{J}\sum_{i=1}^{J}\Big(\nabla_a Q^P_1\big(s(i), \pi(s(i)) \mid \theta^P_{Q_1}\big) \nabla_{\theta^P_\pi}\pi\big(s(i) \mid \theta^P_\pi\big)\Big), \quad (54)$$

and

$$\theta^P_{Q_i} = \theta^P_{Q_i} - \frac{\rho_Q}{J}\sum_{j=1}^{J}\Big(2(v(j) - Q_i(s(j), a(j) \mid \theta^P_{Q_i})) \nabla_{\theta^P_{Q_i}} Q_i(s(j), a(j) \mid \theta^P_{Q_i})\Big), i \in \{1, 2\}, \quad (55)$$

Just Accepted

where ρ_π and ρ_Q denote the learning rates associated with θ_π^P and the critic networks Q_1 and Q_2, respectively. To derive the target Q-value $v(t)$, the lower estimate of Q_1^* and Q_2^* is adopted, with additional perturbation introduced by noise drawn from a truncated normal distribution σ^T. The resulting expression of $v(t)$ is,

$$\begin{aligned} &v(t) \\ &= y(t) + \gamma \min_{i=1,2} Q_i^*(s(t+1), \pi^*(s(t+1)) + \sigma^T \mid \theta_{Q_i}^T), \end{aligned} \tag{56}$$

where γ denotes the discount factor, and Q_i^* represents the Q-value estimated by the target critic network. Similarly, the policy output $\pi^*(s(t+1))$ is perturbed by σ^T, sampled from a truncated normal distribution $\mathcal{N}(0, \eta^T)$.

The parameters $\theta_{Q_1}^P$ and $\theta_{Q_2}^P$ are optimized by minimizing the Temporal-Difference (TD) error [32] loss functions $\mathcal{L}(\Theta_{Q_1}^P)$ and $\mathcal{L}(\Theta_{Q_2}^P)$, expressed as,

$$\mathcal{L}(\Theta_{Q_i}^P) = \mathbb{E}\Big(\big(v(t) - Q_i(s(t), \varpi(t) \mid \theta_{Q_i}^P)\big)^2\Big). \tag{57}$$

The values of parameters θ_π^T, $\theta_{Q_1}^T$, and $\theta_{Q_2}^T$ are updated via the following equations [33],

$$\theta_\pi^T = \omega\theta_\pi^T + (1-\omega)\theta_\pi^P, \tag{58}$$

$$\theta_{Q_i}^T = \omega\theta_{Q_i}^T + (1-\omega)\theta_{Q_i}^P, \tag{59}$$

where $\omega \in (0, 1)$ represents the soft update weight. With ω very close to 1. The target network parameters are updated using a weighted average of their past values and the corresponding primary network parameters.

4.3 Algorithm Description

We outline the procedure of the DRL-ITS algorithm as follows.

- **Step 1: Initialization.**
  The initialization phase begins by assigning random weights to the primary actor and twin critic networks within the DRL-ITS framework. Subsequently, the target networks are instantiated as replicas of their primary counterparts to stabilize training. We also establish an empty experience replay buffer to archive transition data. The training regime is organized into a series of episodes, where the simulation environment is re-initialized to its starting configuration at the onset of each episode.
- **Step 2: Action Generation and Reward Evaluation.**
  In time slot t, the interaction between the LLM and the TD3 core operates as a closed-loop pipeline. Specifically, the LLM parses inference task prompts into semantic embeddings, which are concatenated with raw environmental states to obtain network state $s(t)$. Guided by context-aware network state, the TD3's primary actor network produces a discrete scheduling action $\varpi(t)$, which determines the GenAI inference scheme selection for all tasks in the system. Based on the action output, the closed-form solvers are invoked to obtain the transmission rates and the CPU/GPU frequency allocation for both edge and cloud servers. Then, substituting these decisions into the system model, the LLM-enabled agent evaluates the reward $\zeta(t)$ following the drift-plus-penalty design.
- **Step 3: Experience Storage.**
  Following the action execution, the system transitions to a new state $s(t+1)$, and the corresponding tuple $(s(t), \varpi(t), \zeta(t), s(t+1))$ is archived in the replay buffer. To maintain a fixed buffer size, the algorithm evicts the most antecedent data entries whenever the storage limit is saturated.
- **Step 4: Network Optimization.**
  For network optimization, a mini-batch of experiences is randomly sampled. The critic networks function to minimize the temporal-difference error, while the actor network is updated periodically (every ξ steps) using the deterministic policy gradient. Concurrently, the target networks undergo a soft update process to ensure convergence stability.
- **Step 5: Real-time Execution.**
  After the learning process finalizes, the actor network acts as a real-time decision-maker to select the appropriate inference schemes. Substituting this decision into the closed-form solvers yields the optimal transmission rates, CPU/GPU resource allocations, and overall task scheduling policy.

4.4 Complexity Analysis of the DRL-ITS Algorithm

As aforementioned in Section IV-C, the proposed DRL-ITS algorithm is composed of three cascaded functional modules. Based on the execution flow in Algorithm 1, the complexity analysis is detailed as follows:

1) **Cognition Module (Step 5):** At the beginning of the time slot (line 1-5), the LLM processes the textual prompts of arriving tasks to extract semantic context. Let L_{seq} denote the average sequence length and d_{model} be the model dimension. Therefore, processing all GenAI inference tasks incurs a complexity of $\mathcal{O}_{LLM} = \mathcal{O}(K_{tot} \cdot L_{seq}^2 \cdot d_{model})$

2) **Task Inference Scheme Selector (Step 6):** The environment state vector $s(t)$ aggregates the queue states and semantic embeddings of all tasks. Consequently, the input dimension of the primary actor network is scaled linearly with the total number of tasks, i.e., input size $\propto K_{tot}$. Assuming

Just Accepted

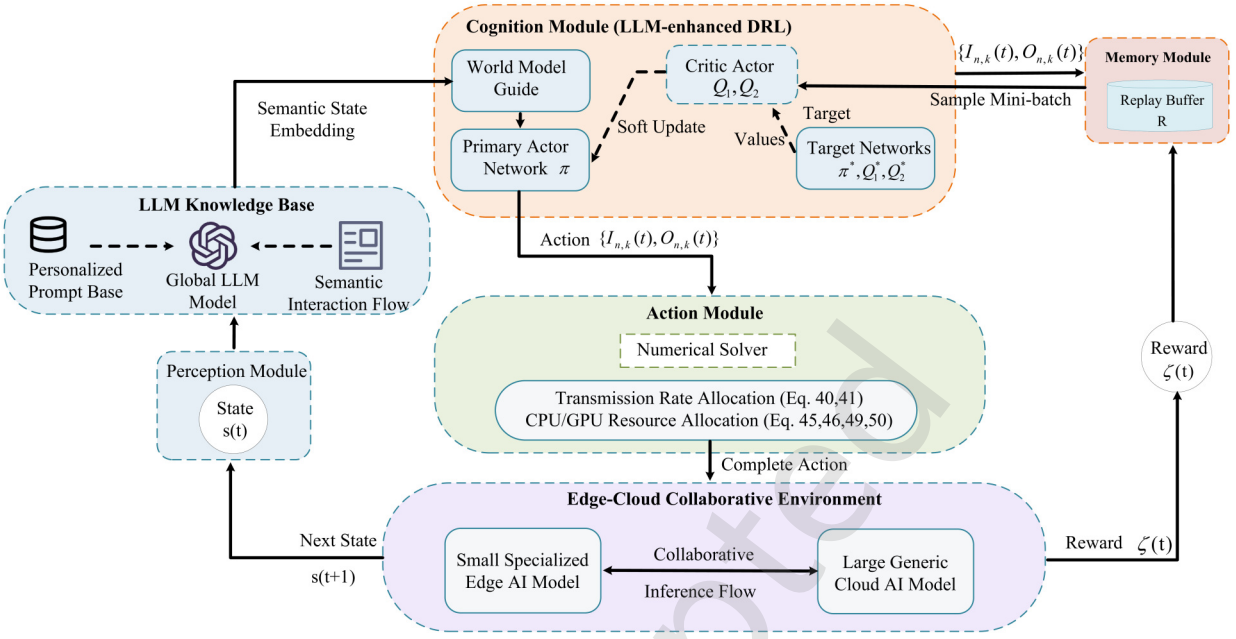


Fig. 2 Structure of LLM-enabled Agentic DRL Algorithm.

the actor network consists of Λ fully connected layers with H hidden units, the complexity of the first layer is $\mathcal{O}(K_{tot} \cdot H)$, while subsequent layers incur $\mathcal{O}(H^2)$. Thus, the complexity for the DRL inference is $\mathcal{O}_{DRL} = \mathcal{O}(K_{tot} \cdot H + \Lambda \cdot H^2)$.

3) **Resource Orchestration (Steps 7-9):** Once the inference schemes are determined, the transmission rates and computing resources are computed using the closed-form solutions derived in Section IV-B. These calculations, along with the reward evaluation, involve linear arithmetic operations over the task set. Therefore, the complexity is $\mathcal{O}_{Num} = \mathcal{O}(K_{tot}) = \mathcal{O}(\mathcal{N}K)$.

4) **Overall Complexity:** As summarized, the total per-slot computational overhead of the DRL-ITS algorithm is $\mathcal{O}_{total} = \mathcal{O}\big(\mathcal{N}K \cdot (L_{seq}^2 d_{model} + H + 1) + \Lambda H^2\big)$, which scales linearly with the number of tasks ($\mathcal{N}K$) and ensures high scalability. Although the LLM-based cognition module introduces a quadratic term L_{seq}^2, its practical impact is marginal as average prompt lengths are significantly smaller than high-intensity visual payloads. Given the fixed DRL architecture (Λ, H) during inference, the algorithm maintains a constant execution overhead per task, satisfying the real-time requirements for edge-cloud resource orchestration. Beyond computational efficiency, the LLM-enabled agent provides context-aware prior knowledge that prunes the action search space, thereby accelerating policy convergence. Such synergy allows the DRL-ITS algorithm to reach optimal states more rapidly, effectively offsetting initial reasoning costs and minimizing the long-term average inference latency.

5 Performance and Discussion

This section presents a comprehensive set of simulations to assess how effectively the proposed DRL-ITS algorithm performs. We first describe the parameter settings, followed by convergence analysis, and then present the simulation results and corresponding discussions.

5.1 Parameter Settings

To investigate the effectiveness of our DRL-ITS algorithm, extensive numerical simulations are carried out. In the experimental setup, and based on [19],[30],[24], the simulation parameters are set as follows. A set of 40 inference tasks is produced and then randomly dispatched to $N = 4$ edge servers. To better reflect real-world application scenarios, the resolution $g_{n,k}(t)$ of the generated images is directly specified by the user and set to 600 [34]. Then the size of the generated image $\varsigma_{n,k}(t) = 32 \times 600^2$ bits. Note that for clarity of presentation in the simulation results, data-related metrics such as queue lengths and the Lyapunov parameter V are scaled to Megabits (Mbit) and Mbit^2/s, respectively. We set the cloud–edge transmission rate to 100Mbps. Each edge server is equipped with a total CPU capacity of $\mathcal{F}_{n,c} = 1 \times 10^{11}$ FLOPS and a GPU capacity of $\mathcal{F}_{n,g} = 100 \times 3 \times 10^9$ FLOPS, respectively. The cloud server is provisioned with substantially higher computational resources, including a total CPU capacity of $\mathcal{F}_c = 16 \times 10^{11}$ FLOPS and a GPU capacity of

Just Accepted

Algorithm 1 LLM-Enhanced Agentic DRL for Inference Task Scheduling

Input: Initial primary networks $\theta_\pi^P, \theta_{Q_1}^P, \theta_{Q_2}^P$, target networks $\theta_\pi^T, \theta_{Q_1}^T, \theta_{Q_2}^T$, replay buffer $\mathcal{R}$, batch size J, discount factor γ, soft update weight ω, learning rates ρ_π, ρ_Q, exploration/smoothing noises η^P, η^T, and the update interval ξ.

Output: Trained policy $\pi(s|\theta_\pi^P)$ and optimal decisions $\{\varpi^*, r^*, f^*\}$.

Initialize $\mathcal{R}$ and synchronize target networks: $\theta^* \leftarrow \theta$;
for each episode $e = 1, \ldots, E$ **do**
  Initialize task queue $Q(0) = 0$ and observe initial system state;
  for $t = 1, \ldots, \mathcal{T}$ **do**
    Invoke the Cognitive Module to extract semantic embeddings and combine them with resource states to construct $s(t)$;
    Execute inference scheme $\varpi(t)$ via $\pi(s(t))$ using Gumbel-Softmax for discrete action mapping;
    for each selected task k **do**
      Compute optimal transmission power r^* and CPU frequency f^* using closed-form solvers derived in **Eq. (40)–(50)**;
    end for
    Apply $\{\varpi(t), r^*, f^*\}$ to induce a state transition to $s(t+1)$, and compute reward $\zeta(t) = -\mathcal{Z}(t)$;
    Update physical task queues $Q(t+1)$ based on actual processed data and arrivals via **Eq. (19)–(20)**;
    Store transition $(s(t), \varpi(t), \zeta(t), s(t+1))$ in $\mathcal{R}$;
    A sub-batch of J transitions is randomly sampled from $\mathcal{R}$;
    Update Critic θ_{Q_i} by minimizing the TD error via **Eq. (55)** and **Eq. (57)**;
    if $t \pmod{\xi} == 0$ **then**
      Update Actor θ_π via deterministic policy gradient via **Eq. (54)**;
      Perform soft update of target networks via **Eq. (58)–(59)**;
    end if
  end for
end for
return Optimized policy π_θ and final scheduling decisions $\{\varpi^*, r^*, f^*\}$ for execution.

$\mathcal{F}_g = 18576 \times 10^9$ FLOPS, to accommodate more intensive computation during task transmission. Based on the computational load formulation in [19] [35] and the reference EfficientNet-B7 model, the per-bit computation cost is set to $U_0 = 6.4 \times 10^3$ FLOPs/bit, which is adopted in our experiments. As for the aggregation latency, since there is no prior work explicitly quantifying it at the system level, we adopt a conservative range for the aggregation cost ratio χ_{agg} (e.g., $\chi_{agg} \in [0.01, 0.05]$). This setting is based on the fact that aggregation is a lightweight operation compared to full image inference. Notably, we have verified that our experimental conclusions remain unchanged across this range.

In this work, it holds that $(\rho_\pi = 1 \times 10^{-6})$ and $(\rho_Q = 1 \times 10^{-5})$. The noise coefficients η^P and η^T are both configured to 0.1, and the discounting factor is fixed at $\gamma = 0.99$. We set the soft update coefficient ω to 0.995 and the update interval ξ for both the primary actor and target networks to 3. The actor produces logits for the discrete scheduling actions $\{I_{n,k}(t), O_{n,k}(t)\}$. The critic adopts the standard TD3 structure and outputs two Q-values through a linear, non-activated final layer. The training procedure runs for 500 episodes, with 300 time slots per episode. The replay buffer capacity is 10000, and the batch size is fixed at 256.

We select four representative algorithms as baselines for performance evaluation. The implementation details of all baseline schemes are summarized as follows.

- Stochastic Inference Algorithm (SIA): As a baseline strategy, the system selects serial inference or parallel inference with equal probability for each task, without considering network conditions, computational resources, or inference quality.
- Deep Deterministic Policy Gradient (DDPG) [36]: The DDPG-based scheme aims to minimize the long-term inference latency by modeling the edge–cloud cooperative inference as a continuous-action MDP. A model-free actor–critic agent observes task queues, system resource states, and virtual queues, and outputs continuous decisions on inference splitting ratios and computation/communication resources, which are updated via deterministic policy gradients.
- Double Deep Q-Network (DDQN) [37]: The DDQN-based scheme discretizes the joint decision space of inference modes and resource levels and formulates a finite-action MDP. A dueling Double DQN with online and target networks, together with an experience replay buffer, is employed to estimate the Q-values of all candidate scheduling actions, where the task offloading resource allocation decisions are determined by selecting

Just Accepted

the candidate action with the highest Q-value, thereby reducing the long-term inference latency under dynamic task arrivals and time-varying system conditions.

The evaluation metrics employed to assess the proposed Agentic DRL framework include the average reward, GenAI inference time, and GenAI inference task queue length. The average reward serves as a comprehensive indicator of the Agent's long-term performance, which can be quantified by Equation (52), which jointly reflects the optimization of execution latency and the stability of physical queues. The total GenAI inference delay is determined by the sum of computation and transmission latencies for all generative tasks within an episode, as formulated in Equation (27), directly measuring the timeliness of edge-cloud collaborative generation. Additionally, the task queue length is utilized to quantify the stability of the GenAI inference task queues within the edge-cloud system. A bounded and stable queue backlog indicates that the service rate is sufficient to handle the stochastic arrival rate of inference requests, illustrating the Agent's capability to ensure service continuity and satisfy the long-term stability constraint defined in Equation (21).

5.2 Theoretical Verification and Convergence Analysis

First of all, Fig. 3 plots the convergence performance of the cumulative reward. As shown in Fig. 3(a), the proposed DRL-ITS algorithm converges fast after approximately 150 episodes to the highest stable reward, significantly outperforming the pure DRL-based baselines. This is because DRL-ITS leverages a hybrid architecture to decouple resource allocation via efficient numerical methods, which reduces the DRL action space and enhances learning efficiency. In contrast, the stochastic SIA scheme fluctuates at a low level without effective optimization. As observed from Fig. 3(b), the total delay achieved by the DRL-ITS algorithm decreases sharply as training progresses and stabilizes at a minimum of approximately 140s. The reason is that the DRL-ITS algorithm obtains optimal resource allocation through closed-form solutions, avoiding sub-optimal configurations caused by continuous exploration in DDPG or discretization errors in DDQN. Consequently, the proposed algorithm achieves a lower steady-state delay compared with DDPG and DDQN. Finally, Fig. 3(c) illustrates that the DRL-ITS algorithm maintains a low and stable task queue backlog of approximately 600 Mbit, validating its ability to satisfy long-term stability constraints. Conversely, SIA results in consistently high backlogs. DDPG and DDQN yield higher average queue lengths than that of the DRL-ITS algorithm due to less efficient policy learning. The proposed DRL-ITS algorithm outperforms all benchmark policies as it efficiently generates optimal decisions to improve resource utilization.

Fig. 4 illustrates how varying the control parameter (V) balances the objectives of minimizing GenAI inference time and maintaining GenAI task queue stability. Specifically, the experimental results reveal that, as the value of Lyapunov parameter V increases, the total GenAI inference delay of the DRL-ITS algorithm decreases approximately in inverse proportion to (V) and eventually converges to the optimal value. In contrast, the GenAI inference task queue backlog increases linearly with the value of V. Consequently, Fig. 4 verifies the $[O(1/V), O(V)]$ trade-off theoretical results in **Theorem 1** and **Theorem 2**. Accordingly, the value of Lyapunov parameter V can be adjusted to balance the achievable GenAI inference time and the task queue backlog.

5.3 Performance Evaluation

Fig. 5 evaluates the impacts of different GenAI task numbers on performance metrics. As shown in Fig. 5(a), the total inference delay rises with task load due to limited resources. Nevertheless, the DRL-ITS algorithm outperforms all benchmarks by leveraging agentic DRL and numerical methods for optimal resource orchestration. Specifically, under the peak workload of 60 tasks, the proposed DRL-ITS algorithm reduces the total GenAI inference delay by approximately 9.43%, 17.35%, and 31.45% compared with DDPG, DDQN, and SIA schemes, respectively. A similar increasing trend in queue backlog with growing traffic is shown in Fig. 5(b). Notably, the DRL-ITS algorithm maintains a substantially lower backlog than the other schemes. For instance, at the same peak load, reductions of 10.42%, 24.51%, and 38.21% in task queue length are achieved by the DRL-ITS algorithm compared to DDPG, DDQN, and SIA, respectively. This demonstrates that the agentic DRL agent effectively learns to balance the trade-off between minimizing delay and stabilizing queues, even under heavy traffic conditions.

In Fig. 6, we evaluate the impacts of varying generated image resolutions on system performance. As shown in Fig. 6(a), the total inference delay rises sharply as resolution increases. This is because the data volume grows quadratically with resolution, increasing transmission and computation overheads. Notably, at 800 pixels, the proposed DRL-ITS algorithm reduces delay by approximately 8.24%, 9.43%, and 25.17% compared with DDPG, DDQN, and SIA, respectively, owing to its optimal resource allocation strategy. Fig. 6(b) illustrates that task queue length increases linearly with resolution. The reason is that heavier computational workloads saturate limited edge resources. In contrast, the DRL-ITS algorithm

Just Accepted

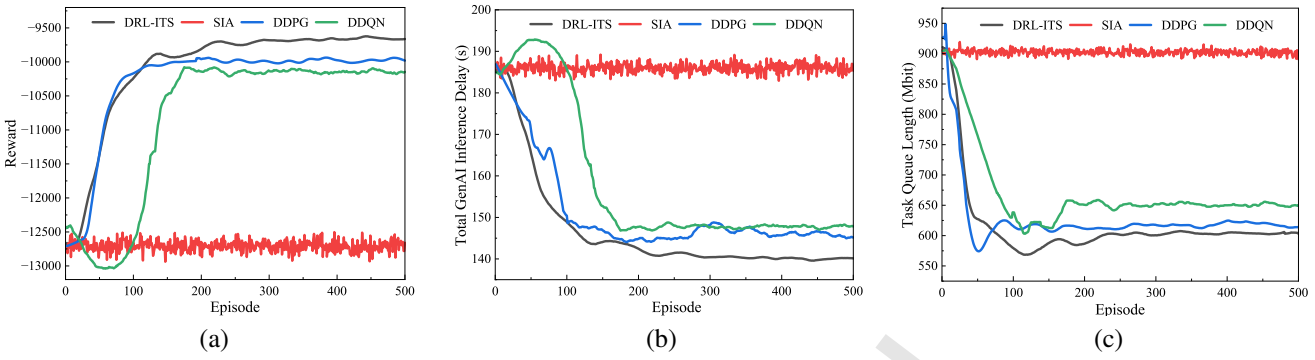


Fig. 3 Convergence performance comparison among different algorithms. (a) Cumulative reward per episode; (b) Total GenAI inference delay per episode. (c) Task queue length per episode.

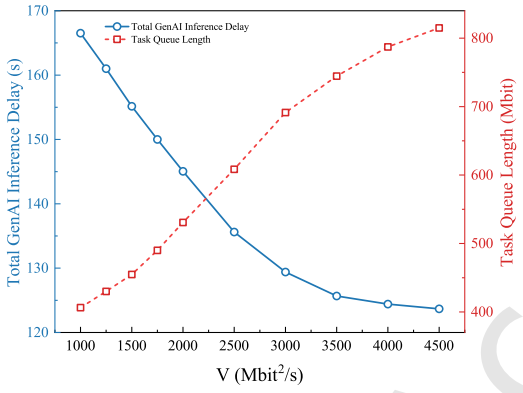


Fig. 4 Impacts of Lyapunov parameter V on total GenAI inference delay and task queue length.

effectively mitigates this backlog. Specifically, at 800 pixels, it reduces the queue backlog by roughly 9.53%, 17.61%, and 42.16% compared to DDPG, DDQN, and SIA. This demonstrates the agent's effectiveness in stabilizing queues under high data-intensity conditions.

Finally, we investigate the impacts of available edge computing resources on system performance. Intuitively, Fig. 7(a) plots the downward trend of total GenAI inference delay as the edge computing capability scaling factor rises from 20% to 100%. The reason is that enhanced computational capacities at the network edge allow for accelerated execution of generative tasks, thereby significantly reducing processing latency. Specifically, when the edge computing capability scaling factor is set to 20%, the proposed DRL-ITS algorithm can reduce the total GenAI inference delay by approximately 8.84%, 17.97%, and 50.12% by comparing it with DDPG, DDQN, and SIA schemes. Similarly, the task queue backlogs exhibit a monotonic decrease with the augmentation of computing capabilities, as observed from Fig. 7(b). In this case, the sub-optimal solutions cannot fully explore the benefits of increased computing power to clear accumulated backlogs rapidly, resulting in slower queue dissipation. For instance, with the scaling factor of 20%, the DRL-ITS algorithm reduces the task queue length by 5.75%, 14.93%, and 33.25% compared with DDPG, DDQN, and SIA algorithms. Consequently, the DRL-ITS algorithm achieves the lowest queue occupancy and outperforms all benchmark policies, validating its robustness in effectively orchestrating dynamic resources to maintain system stability.

6 Conclusion

In this work, we investigate an LLM-enabled Agentic DRL framework in edge-cloud computing networks, where a powerful LLM-enabled agent orchestrates the synergy between cloud-based generic and edge-based specialized visual GenAI models for efficient inference. Consequently, a long-term execution latency minimization problem involving serial-parallel inference scheme selection and resource orchestration is formulated and subsequently transformed into deterministic per-slot sub-problems by leveraging Lyapunov optimization. Then, a novel DRL-ITS algorithm is proposed to obtain optimal resource allocation solutions in each time slot, where a TD3 approach is used to generate discrete inference scheme selection decisions with the aid of KKT-based closed-form numerical solvers for continuous resource allocation. Finally, we provide extensive theoretical and simulation evaluations to investigate the effectiveness of the DRL-ITS algorithm. To further support multi-agent 6G scenarios, our framework can be extended by integrating decentralized coordination paradigms and hierarchical topologies. Such an architecture ensures that inter-agent negotiation remains localized and computationally efficient, providing robustly scalable performance in dynamic 6G environments.

Just Accepted

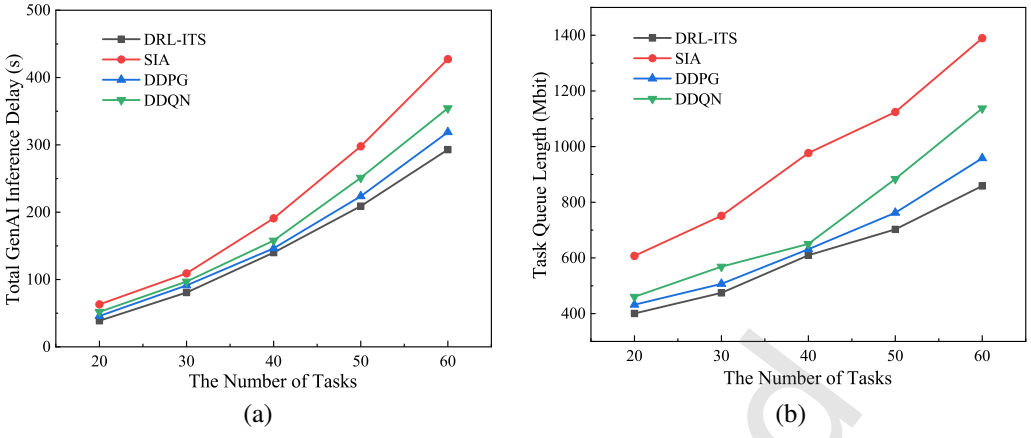


Fig. 5 Impact of the number of GenAI tasks on system performance. (a) Total GenAI inference delay; (b) Task queue length.

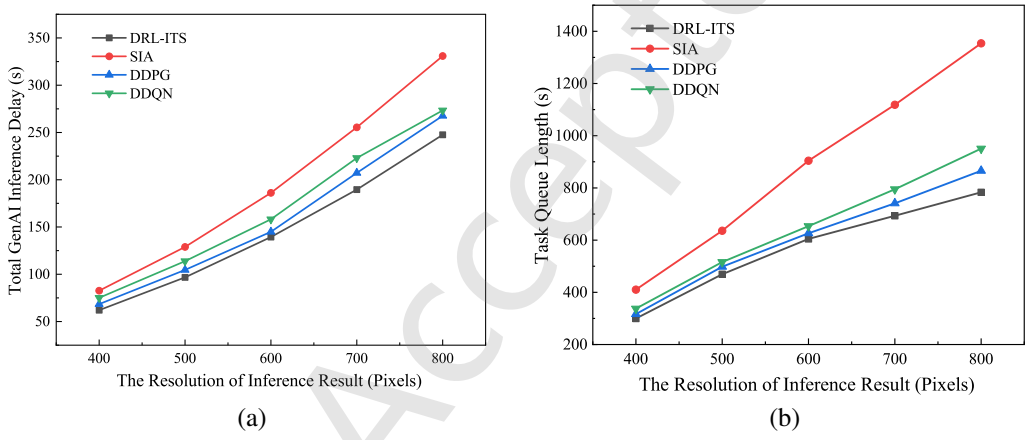


Fig. 6 Impact of generated image resolution on system performance. (a) Average total GenAI inference delay; (b) Task queue length.

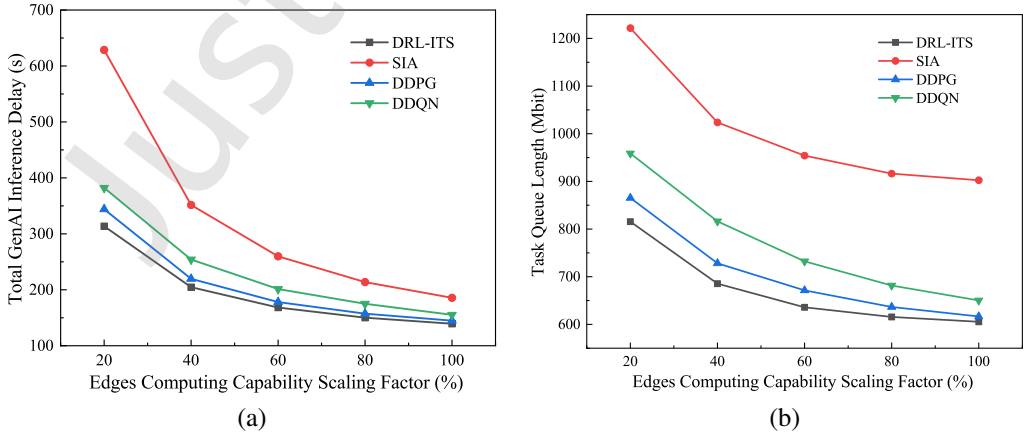


Fig. 7 Impact of edge server computing capability on system performance. (a) Total GenAI inference delay; (b) Task queue length.

Appendix A: Proof of Lemma 1

Based on the generic structure $\max\{l-p,0\}+q$ observed in Equations (19) and (20) for all queues within $\Theta(t)$, we leverage the inequality $(\max\{l-p,0\}+q)^2 \leqslant l^2+p^2+q^2+2l(p-q)$

Just Accepted

to derive the following bound,

$$\begin{aligned} Q_n^2(t+1) \leqslant & Q_n^2(t) + \psi_n^2(t) + \mu_n^2(t) \\ & + 2Q_n(t)[\psi_n(t) - \mu_n(t)], \end{aligned} \tag{A1}$$

$$\begin{aligned} Q_c^2(t+1) \leqslant & Q_c^2(t) + \psi_c^2(t) + \mu_c^2(t) \\ & + 2Q_c(t)[\psi_c(t) - \mu_c(t)]. \end{aligned} \tag{A2}$$

Then, according to Equation (31), we have

$$\begin{aligned} & \mathcal{L}(\Theta(t+1)) - \mathcal{L}(\Theta(t)) \\ & = \frac{1}{2}\left\{ \sum_{n=1}^{\mathcal{N}} Q_n^2(t+1) - \sum_{n=1}^{\mathcal{N}} Q_n^2(t) + Q_c^2(t+1) - Q_c^2(t) \right\} \\ & \leqslant \sum_{n=1}^{\mathcal{N}} Q_n(t)[\psi_n(t) - \mu_n(t)] + Q_c(t)[\psi_c(t) - \mu_c(t)] \\ & + \sum_{n=1}^{\mathcal{N}} \frac{\psi_n^2(t) + \mu_n^2(t)}{2} + \frac{\psi_c^2(t) + \mu_c^2(t)}{2} \\ & \leqslant \sum_{n=1}^{\mathcal{N}} Q_n(t)[\psi_n(t) - \mu_n(t)] + Q_c(t)[\psi_c(t) - \mu_c(t)] \\ & + \sum_{n=1}^{\mathcal{N}} \frac{\psi_{n,max}^2 + \mu_{n,max}^2}{2} + \frac{\psi_{c,max}^2 + \mu_{c,max}^2}{2}, \end{aligned} \tag{A3}$$

where $\psi_{n,max}, \mu_{n,max}, \psi_{c,max}$, and $\mu_{c,max}$ are the upper bounds of the new inference data arrival and the service capacity for edge server n and the cloud server, respectively. Specifically, $\psi_{c,max}$ is the maximum total data size transmitted from all edge servers, while $\mu_{c,max}$ is the maximum processing rate of the cloud server. The term B is a constant independent of the current queue vector $\mathbf{Q}(t)$, which is derived as,

$$B = \sum_{n=1}^{\mathcal{N}} \frac{\psi_{n,max}^2 + \mu_{n,max}^2}{2} + \frac{\psi_{c,max}^2 + \mu_{c,max}^2}{2}. \tag{A4}$$

Applying conditional expectation to both sides of Equation (A3) and incorporating the penalty term, we can derive,

$$\begin{aligned} \Delta(\Theta(t)) \leqslant & B + \\ & \sum_{n=1}^{\mathcal{N}} Q_n(t)[\psi_n(t) - \mu_n(t)] + Q_c(t)[\psi_c(t) - \mu_c(t)]. \end{aligned} \tag{A5}$$

The proof is completed.

Appendix B: Proof of Theorem 1

Based on the drift-plus-penalty upper bound derived in **Lemma 1**, for any feasible control policy, we have:

$$\begin{aligned} & \Delta(\Theta(t)) + V\mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)|\Theta(t)] \\ & \leqslant B + V\mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)|\Theta(t)] \\ & + \sum_{n=1}^{\mathcal{N}} Q_n(t)[\psi_n(t) - \mu_n(t)] + Q_c(t)[\psi_c(t) - \mu_c(t)]. \end{aligned} \tag{B1}$$

Define C^* as the optimal value of problem $\mathcal{P}_1$, i.e., the infimum time-average expected total processing delay over all stabilizing policies:

$$C^* \triangleq \inf_{\pi\in\Pi} \limsup_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}\left[\sum_{n=1}^{N} \sum_{k\in\mathcal{A}_n(t)} M_{n,k}^{\pi}(t) \right], \tag{B2}$$

where Π denotes the set of all admissible control policies that keep all queues strongly stable.

Based on the Lyapunov optimization theory [38], for any $\delta > 0$, there exists a stationary randomized policy π^* such that,

$$\mathbb{E}\left[\psi_n^{\pi^*}(t) - \mu_n^{\pi^*}(t)\right] \leqslant 0, \ \forall n, \tag{B3}$$

$$\mathbb{E}\left[\psi_c^{\pi^*}(t) - \mu_c^{\pi^*}(t)\right] \leqslant 0, \tag{B4}$$

$$\mathbb{E}\left[\sum_{n=1}^{N} \sum_{k\in\mathcal{A}_n(t)} M_{n,k}^{\pi^*}(t) \right] \leqslant C^* + \delta. \tag{B5}$$

Since the ideal opportunistic controller chooses the action that minimizes the right-hand side (RHS) of the drift-plus-penalty bound in (B1) for the current state. Therefore, the value achieved by our algorithm must be less than or equal to the value achieved by the stationary policy π^*,

$$\begin{aligned} & \Delta(Q(t)) + V\mathbb{E}\left[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)|\Theta(t) \right] \\ & \leqslant B + V\mathbb{E}\left[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}^{\pi^*}(t)|\Theta(t) \right] \\ & + \sum_{n=1}^{\mathcal{N}} Q_n(t)\mathbb{E}[\psi_n^{\pi^*}(t) - \mu_n^{\pi^*}(t)|\Theta(t)] \\ & + Q_c(t)\mathbb{E}[\psi_c^{\pi^*}(t) - \mu_c^{\pi^*}(t)|\Theta(t)]. \end{aligned} \tag{B6}$$

By substituting the properties of π^* into the above inequality, the terms multiplied by $Q_n(t)$ and $Q_c(t)$ become $\leqslant 0$ and

Just Accepted

can be removed to yield a tighter upper bound. We obtain

$$\Delta(\Theta(t)) + V\mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)] \leqslant B + V(C^* + \delta). \tag{B7}$$

Now, we sum the above inequality over the time slots $t \in \{0, 1, \ldots, \mathcal{T}-1\}$. By applying the law of telescoping sums to the drift term $\Delta(\Theta(t)) = \mathbb{E}[\mathcal{L}(\Theta(t+1))] - \mathbb{E}[\mathcal{L}(\Theta(t))]$, we have

$$\begin{aligned} &\sum_{t=0}^{\mathcal{T}-1} (\mathbb{E}[\mathcal{L}(\Theta(t+1))] - \mathbb{E}[\mathcal{L}(\Theta(t))]) \\ &+ V \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)] \\ &\leqslant \sum_{t=0}^{\mathcal{T}-1} B + \sum_{t=0}^{\mathcal{T}-1} V(C^* + \delta). \end{aligned} \tag{B8}$$

Simplifying the telescoping sum $\sum_{t=0}^{\mathcal{T}-1}(\mathbb{E}[\mathcal{L}(\Theta(t+1))] - \mathbb{E}[\mathcal{L}(\Theta(t))]) = \mathbb{E}[L(\mathcal{T})] - \mathbb{E}[\mathcal{L}(\Theta(0))]$, the inequality becomes

$$\begin{aligned} &\mathbb{E}[\mathcal{L}(\Theta(\mathcal{T}))] - \mathbb{E}[\mathcal{L}(\Theta(0))] \\ &+ V \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)] \\ &\leqslant \mathcal{T}B + \mathcal{T}V(C^* + \delta). \end{aligned} \tag{B9}$$

Rearranging the terms to isolate the inference delay

$$\begin{aligned} &V \cdot \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)] \\ &\leqslant \mathcal{T}B + \mathcal{T}V(C^* + \delta) + \mathbb{E}[\mathcal{L}(\Theta(0))] - \mathbb{E}[\mathcal{L}(\Theta(\mathcal{T}))]. \end{aligned} \tag{B10}$$

Since the Lyapunov function $\mathcal{L}(\Theta(t))$ is non-negative (i.e., $\mathbb{E}[\mathcal{L}(\Theta(\mathcal{T}))] \geqslant 0$) and assuming the initial queues are empty (i.e., $\mathbb{E}[\mathcal{L}(\Theta(0))] = 0$), we can drop these terms for the upper bound:

$$V \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)] \leqslant \mathcal{T}B + \mathcal{T}V(C^* + \delta). \tag{B11}$$

Dividing both sides by $V\mathcal{T}$, we get

$$\frac{1}{\mathcal{T}} \sum_{t=0}^{\mathcal{T}-1} \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}_{n,k}(t)] \leqslant C^* + \delta + \frac{B}{V}. \tag{B12}$$

Since $\delta > 0$ is arbitrary, letting $\delta \to 0$ yields (35), which completes the proof.

The proof is completed.

Appendix C: Proof of Theorem 2

To prove the $O(V)$ bound on the queue length, we utilize the concept of Strong Stability based on the Slater condition.

Assumption 1 (Slater Condition): We assume there exists a stationary randomized policy π that chooses actions independent of the current queue backlogs, satisfying the following conditions:

The system is strictly stable with a slackness parameter $\epsilon > 0$:

$$\begin{aligned} \mathbb{E}[\psi_n^\pi(t) - \mu_n^\pi(t)] &\leqslant -\epsilon, \quad n \in \mathcal{N}, \\ \mathbb{E}[\psi_c^\pi(t) - \mu_c^\pi(t)] &\leqslant -\epsilon, \end{aligned} \tag{C1}$$

The expected inference delay under this policy is bounded by a finite constant $\mathcal{M}^\pi$:

$$\mathbb{E}\left[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}^\pi(t) | \Theta(t)\right] = \mathcal{M}^\pi \leqslant \mathcal{M}^{\max}. \tag{C2}$$

Proof: Recall the drift-plus-penalty upper bound derived in Lemma 1 (Eq. 33). Our proposed algorithm (denoted as alg) minimizes the RHS of the drift-plus-penalty bound in each time slot. Therefore, the value of the drift-plus-penalty expression achieved by our algorithm is less than or equal to that achieved by the stationary policy π:

$$\begin{aligned} &\Delta(\Theta(t)) + V \cdot \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}^{alg}(t) | \Theta(t)] \\ &\leqslant B + V \cdot \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}^{\pi}(t) | \Theta(t)] \\ &+ \sum_{n=1}^{N} Q_n(t)\mathbb{E}[\psi_n^\pi(t) - \mu_n^\pi(t)] + Q_c(t)\mathbb{E}[\psi_c^\pi(t) - \mu_c^\pi(t)]. \end{aligned} \tag{C3}$$

Substituting the Slater conditions from **Assumption 1** into the inequality:

$$\begin{aligned} &\Delta(\Theta(t)) + V \cdot \mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}^{alg}(t) | \Theta(t)] \\ &\leqslant B + V\mathcal{M}^{max} + \sum_{n=1}^{N} Q_n(t)(-\epsilon) + Q_c(t)(-\epsilon). \end{aligned} \tag{C4}$$

Rearranging the terms to isolate the queue backlog term on the left-hand side:

$$\begin{aligned} &\epsilon\mathbb{E}\left[\sum_{n=1}^{N} Q_n(t) + Q_c(t)\right] \\ &\leqslant B + V\mathcal{M}^{max} \\ &\quad - V\mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}^{alg}(t) | \Theta(t)] - \Delta(\Theta(t)). \end{aligned} \tag{C5}$$

Since the inference delay is non-negative (i.e., $\mathcal{M}^{alg}(t) \geqslant 0$), we can simply remove the term $-V\mathbb{E}[\sum_{n=1}^{\mathcal{N}} \sum_{k\in\mathcal{A}_n(t)} \mathcal{M}^{alg}(t) | \Theta(t)]$ to obtain a looser

Just Accepted

upper bound:

$$\epsilon\mathbb{E}\left[\sum_{n=1}^{N}Q_n(t)+Q_c(t)\right] \leqslant B+V\mathcal{M}^{max}-\mathbb{E}[\mathcal{L}(\Theta(t+1))-\mathcal{L}(\Theta(t))|\Theta(t)]. \tag{C6}$$

Next, we sum the above inequality over the time slots $t \in \{0,1,\ldots,\mathcal{T}-1\}$:

$$\epsilon\sum_{t=0}^{\mathcal{T}-1}\mathbb{E}\left[\sum_{n=1}^{N}Q_n(t)+Q_c(t)\right] \leqslant \mathcal{T}(B+V\mathcal{M}^{max})-\sum_{t=0}^{\mathcal{T}-1}(\mathbb{E}[\mathcal{L}(\Theta(t+1))]-\mathbb{E}[\mathcal{L}(\Theta(t))]). \tag{C7}$$

Using the telescoping sum property for the Lyapunov drift term:

$$\epsilon\sum_{t=0}^{\mathcal{T}-1}\mathbb{E}\left[\sum_{n=1}^{N}Q_n(t)+Q_c(t)\right] \leqslant \mathcal{T}(B+V\mathcal{M}^{max})+\mathbb{E}[\mathcal{L}(\Theta(0))]-\mathbb{E}[\mathcal{L}(\Theta(T))]. \tag{C8}$$

Dividing both sides by $T\epsilon$ and assuming finite initial queues (i.e., $\mathbb{E}[\mathcal{L}(\Theta(0))]<\infty$) and non-negative Lyapunov function values (i.e., $\mathbb{E}[\mathcal{L}(\Theta(\mathcal{T}))]\geqslant 0$):

$$\frac{1}{\mathcal{T}}\sum_{t=0}^{\mathcal{T}-1}\mathbb{E}\left[\sum_{n=1}^{N}Q_n(t)+Q_c(t)\right] \leqslant \frac{B+V\mathcal{M}^{max}}{\epsilon}+\frac{\mathbb{E}[\mathcal{L}(\Theta(0))]}{\mathcal{T}\epsilon}. \tag{C9}$$

Finally, taking the limit as $\mathcal{T}\to\infty$, we have $\mathbb{E}[\mathcal{L}(\Theta(0))]/\mathcal{T}\epsilon\to 0$. We obtain the following bound,

$$\lim_{\mathcal{T}\to\infty}\frac{1}{\mathcal{T}}\sum_{t=0}^{\mathcal{T}-1}\mathbb{E}\left[\sum_{n=1}^{N}Q_n(t)+Q_c(t)\right]\leqslant\frac{B+V\mathcal{M}^{max}}{\epsilon}. \tag{C10}$$

Since $B,\mathcal{M}^{max},\epsilon$ are constants, the average queue length is bounded by $O(V)$. The proof is completed.

Appendix D: Proof of Lemma 2

The convexity of Problem $\mathcal{P}_{2.1}$ stems from the convex nature of the objective function relative to $r_{n,k,1}(t)$, combined with an affine and compact feasible set determined by (4) and (39). Consequently, the KKT conditions serve as the unique necessary and sufficient criteria for the optimal solution.

We introduce a Lagrange multiplier $\alpha\geqslant 0$ for the inequality constraint (4). The inequality constraints (39) are considered in the KKT verification later. The Lagrangian function is constructed as,

$$\mathcal{L}_1=\sum_{k\in\mathcal{A}_n(t)}x_{n,k}(t)\frac{\varsigma_{n,k}(t)}{r_{n,k,1}(t)}+\alpha\left(\sum_{k\in\mathcal{A}_n(t)}r_{n,k,1}(t)-R_{n,1}\right). \tag{D1}$$

Taking the partial derivative of $\mathcal{L}_1$ with respect to $r_{n,k,1}(t)$ and setting it to zero,

$$\frac{\partial\mathcal{L}_1}{\partial r_{n,k,1}(t)}=-\frac{x_{n,k}(t)\varsigma_{n,k}(t)}{[r_{n,k,1}(t)]^2}+\alpha=0. \tag{D2}$$

Solving the above gives,

$$r_{n,k,1}(t)=\sqrt{\frac{x_{n,k}(t)\varsigma_{n,k}(t)}{\alpha}}. \tag{D3}$$

When $\sum_{j\in\mathcal{A}_n(t)}\sqrt{x_{n,j}(t)\varsigma_{n,j}(t)}>0$, the objective is strictly decreasing in $r_{n,k,1}(t)$ for those k with $x_{n,k}(t)\varsigma_{n,k}(t)>0$, hence the capacity constraint (4) is tight at optimum, i.e., $\sum_{k\in\mathcal{A}_n(t)}r^*_{n,k,1}(t)=R_{n,1}$. Thus, the complementary slackness condition $\alpha\left(\sum_{k\in\mathcal{A}_n(t)}r^*_{n,k,1}(t)-R_{n,1}\right)=0$ holds, with $\alpha>0$ when the constraint is tight. Substitute the expression into constraint (4),

$$\alpha=\left(\frac{\sum_{j\in\mathcal{A}_n(t)}\sqrt{x_{n,j}(t)\varsigma_{n,j}(t)}}{R_{n,1}}\right)^2. \tag{D4}$$

Substitute α back into (D3),

$$r^*_{n,k,1}(t)=\frac{R_{n,1}\sqrt{x_{n,k}(t)\varsigma_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{x_{n,j}(t)\varsigma_{n,j}(t)}}. \tag{D5}$$

If $\sum_{j\in\mathcal{A}_n(t)}\sqrt{x_{n,j}(t)\varsigma_{n,j}(t)}=0$, a valid optimal solution is $r^*_{n,k,1}(t)=0$ for all $k\in\mathcal{A}_n(t)$, and the capacity constraint is not tight (thus $\alpha=0$).

To confirm the global optimality of the obtained $r^*_{n,k,1}(t)$, we verify its adherence to the KKT conditions. Since problem $\mathcal{P}_{2.1}$ is strictly convex and satisfies Slater's condition [39], the KKT conditions serve as both necessary and sufficient criteria for optimality [40]. It can be readily verified that the derived solution strictly resides within the interior of the feasible region, perfectly satisfying primal feasibility, stationarity, and complementary slackness (with inactive bound multipliers set to zero), thereby guaranteeing its global optimality.
The proof is completed.

Appendix E: Optimal CPU and GPU Frequency Allocation Via KKT Conditions

With the CPU and GPU capacities of edge server n being predetermined, the objective shifts to the minimization of cumulative GenAI inference latency. Based on equation (44),

Just Accepted

sub-problem $\mathcal{P}_{2.3}$ can be further expressed as,

$$\mathcal{P}_{2.3.1}: \min \sum_{k\in\mathcal{A}_n(t)} y_{n,k}(t)\frac{\mathcal{C}_{n,k}(t)}{f_{n,k,c1}(t)+f_{n,k,g1}(t)}$$
$$\text{s.t. } (10),(11),(12),(13), \tag{E1}$$

Note that the objective depends on $(f_{n,k,c1}(t), f_{n,k,g1}(t))$ only through their sum $s_{n,k}(t) \triangleq f_{n,k,c1}(t)+f_{n,k,g1}(t)$. Moreover, by summing constraints (11) and (13), we obtain an aggregated capacity constraint $\sum_{k\in\mathcal{A}_n(t)} s_{n,k}(t) \leqslant F_{n,c}+F_{n,g}$.

We construct the Lagrangian function with multipliers ν associated with the total CPU and GPU constraints:

$$\mathcal{L}_2 = \sum_{k\in\mathcal{A}n(t)} \frac{y_{n,k}(t)\mathcal{C}_{n,k}(t)}{f_{n,k,c1}(t)+f_{n,k,g1}(t)} + \nu\left(\sum_{k\in\mathcal{A}_n(t)}(f_{n,k,c1}(t)+f_{n,k,g1}(t)) - (\mathcal{F}_{n,c}+\mathcal{F}_{n,g})\right). \tag{E2}$$

Applying the KKT conditions, we compute the partial derivatives with respect to the decision variables,

$$\frac{\partial\mathcal{L}_2}{\partial(f_{n,k,c1}(t)+f_{n,k,g1}(t))} = -\frac{y_{n,k}(t)\mathcal{C}_{n,k}(t)}{(f_{n,k,c1}(t)+f_{n,k,g1}(t))^2}+\nu=0. \tag{E3}$$

Then, we obtain,

$$f_{n,k,c1}(t)+f_{n,k,g1}(t) = \sqrt{\frac{y_{n,k}(t)\mathcal{C}_{n,k}(t)}{\nu}}. \tag{E4}$$

Based on the fact that the aggregated capacity constraint is tight at optimum (otherwise increasing some $s_{n,k}(t)$ would strictly decrease the objective), we have $\sum_{k\in\mathcal{A}_n(t)}(f_{n,k,c1}(t)+f_{n,k,g1}(t)) = \mathcal{F}_{n,c}+\mathcal{F}_{n,g}$. Hence,

$$\frac{1}{\sqrt{\nu}} = \frac{F_{n,c}+F_{n,g}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{y_{n,j}(t)\mathcal{C}_{n,j}(t)}}, \tag{E5}$$

Substituting (E5) into (E4),

$$D^*_{n,k,1}(t) = f^*_{n,k,c1}(t)+f^*_{n,k,g1}(t) = \frac{(\mathcal{F}_{n,c}+\mathcal{F}_{n,g})\sqrt{y_{n,k}(t)\mathcal{C}_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{y_{n,j}(t)\mathcal{C}_{n,j}(t)}}. \tag{E6}$$

Hence, the optimal solution ensures that the total computing capacity allocated to each task is proportional to the square root of the weighted workload.

Existence and Uniqueness of the Optimal Solution:
The objective function in (E1) is convex with respect to the aggregated variables $s_{n,k}(t) = f_{n,k,c1}(t)+f_{n,k,g1}(t)$ over the domain $s_{n,k}(t) > 0$, and the aggregated constraint is affine. Thus, the KKT conditions are necessary and sufficient, and the optimal aggregated allocation $\{s^*_{n,k}(t)\}$ exists and is unique when $y_{n,k}(t)\mathcal{C}_{n,k}(t) > 0$. However, the decomposition of $s^*_{n,k}(t)$ into CPU and GPU parts may be non-unique; any feasible split satisfying (10),(11),(12),(13) is optimal.

Closed-form Optimal Solution:
The optimal CPU and GPU computing capacities for each task $k \in \mathcal{A}_n$ are given by:

$$f^*_{n,k,c1}(t) = \frac{\mathcal{F}_{n,c}\sqrt{y_{n,k}(t)\mathcal{C}_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{y_{n,j}(t)\mathcal{C}_{n,j}(t)}}, \tag{E7}$$

and

$$f^*_{n,k,g1}(t) = \frac{\mathcal{F}_{n,g}\sqrt{y_{n,k}(t)\mathcal{C}_{n,k}(t)}}{\sum_{j\in\mathcal{A}_n(t)}\sqrt{y_{n,j}(t)\mathcal{C}_{n,j}(t)}}. \tag{E8}$$

The proof is completed.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China (Grants No. 62562011 and No. 62466009), in part by Guizhou Provincial Basic Research Program (Natural Science) (Grant No. ZK[2024]YiBan048 and No. ZK[2025]Mianshang627), in part by Youth Science and Technology Talent Growth Project of Guizhou Education Department (Grant No. QianjiaoJi[2024]22), in part by Guizhou Provincial Science and Technology Plan Project (Grants No. QKHRC-KJZY[2025]021 and No. QKHPT-KXJZ[2025]003), in part by Guizhou Provincial Science and Technology Projects (QKHRC-XKBF[2025]014, QKHRC-XKBF[2025]015, QKHZD[2025]zhongda031, and BQW[2024]010).

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Reference

[1] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, October 2023, pp. 3836–3847.

[2] Y. Tian, Z. Zhang, Y. Yang, Z. Chen, Z. Yang, R. Jin, T. Q. S. Quek, and K.-K. Wong, "An edge-cloud collaboration framework for generative AI service provision with synergetic big cloud model and small edge models," *IEEE Network*, vol. 38, no. 5, pp. 37–46, 2024.

[3] R. Morabito and S. Jang, "Smaller, smarter, closer: The edge of collaborative generative AI," *IEEE Internet Comput.*, vol. 29, no. 4, pp. 7 – 15, 2025.

[4] M. Chen, Z. Feng, L. Wang, and Y. Khamayseh, "Agentic AI-empowered conversational embodied intelligence networks in 6G," *arXiv preprint arXiv:2511.19865*, 2025.

Just Accepted

[5] R. Zhang, G. Liu, Y. Liu, C. Zhao, J. Wang, Y. Xu, D. Niyato, J. Kang, Y. Li, S. Mao *et al.*, "Toward edge general intelligence with agentic AI and agentification: Concepts, technologies, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 28, pp. 4285–4318, 2026.

[6] C. Zhao, G. Liu, R. Zhang, Y. Liu, J. Wang, J. Kang, D. Niyato, Z. Li, Z. Han, S. Sun *et al.*, "Edge general intelligence through world models and agentic AI: Fundamentals, solutions, and challenges," *arXiv preprint arXiv:2508.09561*, 2025.

[7] L. Cai, R. Zhang, J. Wang, Y. Zhang, M. Peng, T. Jiang, D. Niyato, W. Ni, A. Jamalipour, and D. I. Kim, "Large language model-enhanced deep reinforcement learning for secure data collection in low-altitude economy networking," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2026.

[8] X. Wu, Y. Wang, J. Farooq, and J. Chen, "LLM-driven agentic AI approach to enhanced O-RAN resilience in next-generation networks," in *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2025, pp. 1–6.

[9] L. Chen, M. Zaharia, and J. Zou, "Frugalgpt: How to use large language models while reducing cost and improving performance," *arXiv preprint arXiv:2305.05176*, 2023.

[10] X. L. Li, A. Holtzman, D. Fried, P. Liang, J. Eisner, T. B. Hashimoto, L. Zettlemoyer, and M. Lewis, "Contrastive decoding: Open-ended text generation as optimization," pp. 12 286–12 312, 2023.

[11] J. Fu, Y. Jiang, J. Chen, J. Fan, X. Geng, and X. Yang, "Fast large language model collaborative decoding via speculation," *arXiv preprint arXiv:2502.01662*, 2025.

[12] M. Zimmer, M. Gritta, G. Lampouras, H. B. Ammar, and J. Wang, "Mixture of attentions for speculative decoding," *arXiv preprint arXiv:2410.03804*, 2024.

[13] W. Fang, D.-J. Han, L. Yuan, and C. Brinton, "Collaborative device-cloud LLM inference through reinforcement learning," *arXiv preprint arXiv:2509.24050*, 2025.

[14] K. Dev, S. A. Khowaja, E. Zeydan, K. Singh, and M. Debbah, "Advanced architectures integrated with agentic AI for next-generation wireless networks," *IEEE Communications Standards Magazine*, pp. 1–8, 2025.

[15] K. Mei, X. Zhu, W. Xu, W. Hua, M. Jin, Z. Li, S. Xu, R. Ye, Y. Ge, and Y. Zhang, "Aios: LLM agent operating system," *arXiv preprint arXiv:2403.16971*, 2024.

[16] A. Amayuelas, J. Yang, S. Agashe, A. Nagarajan, A. Antoniades, X. E. Wang, and W. Wang, "Self-resource allocation in multi-agent LLM systems," *arXiv preprint arXiv:2504.02051*, 2025.

[17] J. Tong, W. Guo, J. Shao, Q. Wu, Z. Li, Z. Lin, and J. Zhang, "Wirelessagent: Large language model agents for intelligent wireless networks," *arXiv preprint arXiv:2505.01074*, 2025.

[18] X. Tang, F. Liu, D. Xu, J. Jiang, Q. Tang, B. Wang, Q. Wu, and C. L. Philip Chen, "LLM-assisted reinforcement learning: Leveraging lightweight large language model capabilities for efficient task scheduling in multi-cloud environment," *IEEE Transactions on Consumer Electronics*, vol. 71, no. 2, pp. 5631–5644, 2025.

[19] J. Feng, X. Huang, L. Liu, M. Yang, Q. Pei, and Y. G. Shee, "Resource allocation for task-oriented generative artificial intelligence in the internet of things," *IEEE Internet of Things Journal*, vol. 12, no. 10, pp. 13 233–13 247, 2025.

[20] J. Du, Y. Lin, D. Yang, X. Li, A. Peng, and C. Peng, "Latent space embedding for bit-depth enhancement: Synergize with super-resolution," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 36, no. 4, pp. 4913–4926, 2026.

[21] T. Suzuki, S. Yukikata, K. Yang, and T. Yoshida, "Lightning-fast dual-layer lossless coding for radiance format high dynamic range images," *IEEE Signal Processing Letters*, vol. 30, pp. 1362–1366, 2023.

[22] Y. Huang, X. Zheng, and Y. Zhu, "Optimized CPU–GPU collaborative acceleration of zero-knowledge proof for confidential transactions," *Journal of Systems Architecture*, vol. 135, p. 102807, 2023.

[23] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7947–7962, 2021.

[24] Z. He, Y. Sun, B. Wang, S. Li, and B. Zhang, "CPU–GPU heterogeneous computation offloading and resource allocation scheme for industrial internet of things," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 11 152–11 164, 2024.

[25] W. Fan, L. Zhao, X. Liu, Y. Su, S. Li, F. Wu, and Y. Liu, "Collaborative service placement, task scheduling, and resource allocation for task offloading with edge-cloud cooperation," *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 238–256, 2024.

[26] J. Chen, X. Li, L. Pan, and S. Liu, "A Lyapunov optimization-based online algorithm for scheduling cloud-edge collaborative real-time video stream analytics tasks," *IEEE Internet of Things Journal*, vol. 12, no. 14, pp. 27 713–27 727, 2025.

[27] J. Tang, M. M. Jalalzai, C. Feng, Z. Xiong, and Y. Zhang, "Latency-aware task scheduling in software-defined edge and cloud computing with erasure-coded storage systems," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1575–1590, 2023.

[28] J. Liu, W. Zhang, Y. Tang, J. Tang, and G. Wu, "Residual feature aggregation network for image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2359–2368.

[29] Y. Zhang, W. Fan, Y. Yu, and Y. Liu, "DRL-based resource orchestration for vehicular edge computing with multi-edge and multi-vehicle assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 6, pp. 8764 – 8779, 2025.

[30] W. Fan, "Blockchain-secured task offloading and resource allocation for cloud-edge-end cooperative networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 8, pp. 8092–8110, 2024.

Just Accepted

[31] S. Khadka and K. Tumer, "Evolution-guided policy gradient in reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[32] J. Guan, S. E. Verch, C. Voelcker, E. C. Jackson, N. Papernot, and W. A. Cunningham, "Temporal-difference learning using distributed error signals," *Advances in Neural Information Processing Systems*, vol. 37, pp. 108 710–108 734, 2024.

[33] Z. Wang, W. Jiang, R. Peng, Q. Kou, L. Wan, and X. Lan, "Improving sample efficiency through stability enhancement in deep-reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 9, pp. 6164–6176, 2025.

[34] Y. Sun, Y. Liu, S. Guo, X. Qiu, J. Chen, J. Hao, and D. Niyato, "Edge large AI model agent-empowered cognitive multimodal semantic communication," *IEEE Transactions on Mobile Computing*, vol. 25, no. 1, pp. 19 – 36, 2026.

[35] J. Tang, J. Nie, Y. Zhang, Z. Xiong, W. Jiang, and M. Guizani, "Multi-UAV-assisted federated learning for energy-aware distributed edge training," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 280–294, 2024.

[36] X. Chen, Z. Li, W. Ni, X. Wang, S. Zhang, Y. Sun, S. Xu, and Q. Pei, "Toward dynamic resource allocation and client scheduling in hierarchical federated learning: A two-phase deep reinforcement learning approach," *IEEE Transactions on Communications*, vol. 72, no. 12, pp. 7798–7813, 2024.

[37] I. Ullah, H.-K. Lim, Y.-J. Seok, and Y.-H. Han, "Optimizing task offloading and resource allocation in edge-cloud networks: A DRL approach," *Journal of Cloud Computing*, vol. 12, no. 1, p. 112, 2023.

[38] Z. Liu, D. Niyato, J. Wang, G. Sun, L. Huang, Z. Gao, and X. Wang, "Generative AI for lyapunov optimization theory in uav-based low-altitude economy networking," *IEEE Network*, pp. 1–9, 2026.

[39] T. Xu, T. Ding, O. Han, Y. Huang, and Y. He, "Counterpart and correction for strong duality of second-order conic program in radial networks," *IEEE Transactions on Power Systems*, vol. 37, no. 5, pp. 4117–4120, 2022.

[40] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

Moxia Li received the B.S. degree from Nanjing Tech University, Nanjing, China, in 2022. She is currently pursuing the M.S. degree in electronic information with the State Key Laboratory of Public Big Data, Guizhou University. Her research interests include AIGC and edge intelligence.

Jiangtian Nie received the Ph.D. degree from the Interdisciplinary Graduate School, Nanyang Technological University, Singapore. She is currently a Lecturer with the Computer Science Department, University of Aberdeen, U.K. Her research interests include edge intelligence, wireless communications, and the Internet of Things.

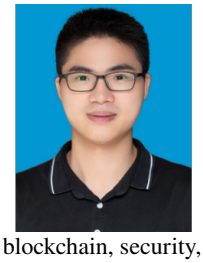
Jianhang Tang is currently a Professor at the State Key Laboratory of Public Big Data, Guizhou University, Guiyang, China. He has published more than 50 research papers in leading journals and flagship conferences, such as IEEE TMC, IEEE TSC, ACM TECS, IEEE TCC, IEEE TNSM, ACM CSUR, and IEEE Network, where two of them are ESI Highly Cited Papers. His research interests include edge intelligence and Communication Networks.

Guoquan Wu is currently pursuing a Ph.D. degree at the Southern University of Science and Technology, Shenzhen, China. His current research interests include wireless communications, wireless sensing, and digital twins.

Yang Zhang received the B.Eng. and M.Eng. degrees from Beihang University in 2008 and 2011, respectively, and the Ph.D. degree in computer engineering from Nanyang Technological University, Singapore, in 2015. He is currently an Associate Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His current research topic is multi-agent unmanned systems. He is an editor of the IEEE Transactions on Machine Learning in Communications and Networking.

Celimuge Wu received the Ph.D. degree from The University of Electro-Communications, Japan, where he is currently a Professor and the Director of the Meta-Networking Research Center. His research interests include edge computing, Internet of Things, and AI-driven wireless networking and computing.

Just Accepted



Jiawen Kang received the Ph.D. degree from Guangdong University of Technology, China, in 2018. He has been a Post-Doctoral Researcher at Nanyang Technological University, Singapore, from 2018 to 2021. He currently is a Full Professor with Guangdong University of Technology. His research interests include blockchain, security, and privacy protection in wireless communications and networking.

Zehui Xiong is currently a Full Professor with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, United Kingdom. Prior to that, he was with the Singapore University of Technology and Design, and Nanyang Technological University (NTU). He received the Ph.D. degree in Computer Science and Engineering at NTU, Singapore. He was a visiting scholar with Department of Electrical Engineering at Princeton University and a visiting scholar with Broadband Communications Research (BBCR) Lab in Department of Electrical and Computer Engineering at University of Waterloo. His research interests include wireless networks, Internet of Things, edge intelligence, semantic communications, generative AI, and Metaverse.

Just Accepted