Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Task Offloading in Multi-UAV Edge Computing Using an Improved Multi- Agent Proximal Policy Optimization Approach

Changhao Duan
Hubei University of Automotive Technology
Zheng Yao
Hubei University of Automotive Technology
ChangJun Deng
hanzhekj@163.com
Yunnan Hanzhe Technology Co., Ltd.
Rui Li
Yunnan Hanzhe Technology Co., Ltd.

Research Article

Keywords:

Posted Date: November 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10982608/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Task Offloading in Multi-UAV Edge Computing Using an Improved Multi-Agent Proximal Policy Optimization Approach

Changhao Duan [1],Zheng Yao [1],ChangJun Deng [2]*, Rui Li [2,3]

[1] School of Robotics and Automation, Hubei University of Automotive Technology, Shiyan, 442002, China;

[2] Yunnan International Joint Laboratory of Agricultural Remote Sensing and Digital Technology, Yunnan Hanzhe Technology Co., Ltd., Kunming, China;

[3] Key Laboratory of Plateau Agricultural Environment Monitoring and Control, Ministry of Agriculture and Rural Affairs, Kunming, China;

* Correspondence: hanzhekj@163.com (ChangJun Deng)

Abstract

With the rapid growth of UAV-enabled applications in inspection, logistics, and emergency communication, onboard computation and energy resources have become key bottlenecks for large-scale UAV swarms. Mobile edge computing (MEC) provides a practical way to relieve this pressure by offloading computation-intensive tasks to nearby edge and cloud servers. However, task offloading in multi-UAV environments remains challenging because of partial observability, time-varying wireless conditions, resource contention, and the need for coordinated decisions among multiple agents. To address these issues, this paper studies an energy-efficient task offloading problem for a heterogeneous UAV swarm and develops a three-layer computing architecture consisting of a UAV layer, an edge server layer, and a cloud layer. Based on this architecture, we propose an MAPPO-based framework, called SS-MAPPO, which combines a top-K sparse attention mechanism with a Gaussian-based latent policy representation. The problem is formulated as a decentralized partially observable Markov decision process (Dec-POMDP), where each UAV makes offloading decisions according to its local task attributes, queue states, and observable server conditions. Simulation results show that SS-MAPPO reduces energy consumption, accelerates convergence, and improves decision stability compared with the evaluated baselines under the tested settings. These results indicate that the proposed framework is a promising approach for energy-aware collaborative offloading in dynamic multi-UAV edge computing environments.

Keywords: Mobile Edge Computing; Multi-Agent Reinforcement Learning; Sparse Attention; Task Priority Queue

1 Introduction

With the rapid development of UAV technology, UAVs have been widely used in inspection, logistics distribution, environmental monitoring, and emergency communication. Their mobility and flexibility allow them to collect data and execute tasks close to the physical scene, which is valuable for time-sensitive applications. However, as mission complexity increases, a single UAV often faces limited onboard energy and computation capacity. For computation-intensive or latency-sensitive tasks, local processing alone is no longer sufficient to meet both efficiency and energy requirements, which constrains the deployment of larger and more capable UAV systems.

Mobile edge computing (MEC) provides an effective way to relieve this burden by shifting computation from resource-constrained UAVs to nearby edge or cloud servers. In a UAV task offloading system, computational tasks generated by UAVs can be executed locally or offloaded to edge and cloud servers according to current network conditions and resource availability. By leveraging nearby edge nodes with stronger computation capability and lower communication delay, MEC can reduce onboard energy consumption and improve task completion efficiency. In multi-UAV scenarios, the edge architecture also enables collaborative scheduling across UAVs and servers, which helps increase overall system throughput and service continuity.

However, task offloading in multi-UAV environments remains difficult because of dynamic topology, partial observability, and contention for wireless and computing resources. UAV mobility causes frequent changes in link quality and server accessibility, while concurrent access from multiple agents can lead to channel interference and queue congestion. Traditional optimization methods are often too expensive to solve in real time for such high-dimensional and time-varying systems. Although deep reinforcement learning improves adaptability, single-agent methods do not scale well to large swarms, and existing multi-agent approaches still face challenges such as inefficient interaction modeling and unstable policy learning. Therefore, designing a scalable and stable collaborative offloading strategy for large multi-UAV systems remains an open problem.

The main contributions of this paper are summarized as follows:

1. We construct a heterogeneous three-layer UAV-edge-cloud offloading architecture for large-

scale UAV swarms, and formulate the multi-UAV offloading problem as a Dec-POMDP with queue-aware observations and an energy-oriented reward function. The architecture incorporates a shared edge server pool together with a task priority queue to capture resource contention and queue dynamics, so that each UAV learns from local information while the training process still exploits centralized value estimation.

2. We propose SS-MAPPO, an improved MAPPO framework for decentralized offloading decisions, where a top-K sparse attention mechanism is introduced into the centralized critic to reduce redundant inter-agent coupling, and a Gaussian-based latent policy representation is used in the actor to smooth discrete action selection.

3. We validate the proposed method through simulation and compare it with representative baselines. The results show that SS-MAPPO achieves lower energy consumption, faster convergence, and more stable decisions under different swarm sizes and task loads.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 presents the system model and problem formulation; Section 4 describes the MAPPO-based framework and the proposed improvements; Section 5 presents the simulation results; and Section 6 concludes the paper and discusses future work.

2 Related Work

Current research primarily focuses on three dimensions: traditional optimization methods, single-agent deep reinforcement learning (DRL), and multi-agent reinforcement learning (MARL).

Traditional task offloading optimization methods often employ heuristic algorithms. Wei et al.[5] proposed a multi-resource computing offloading particle swarm optimization task scheduling algorithm for energy consumption optimization in mobile edge computing, effectively reducing the energy consumption of mobile terminals. Bozorgchenani et al.[6] modeled task offloading in MEC as a constrained multi-objective optimization problem and designed an evolutionary algorithm to effectively find the best balance between energy consumption and task processing delay. Zhou et al.[7] considered the optimization problem in edge-computing-based smart grids, aiming to minimize task cost, a weighted sum of time and energy consumption. Wang et al.[8] studied the battery and computing resource limitations of UAV systems based on fog computing, maximizing the efficiency of collaboration among UAVs using an iterative algorithm. Chen et al.[9] studied the joint optimization problem of computation offloading, channel allocation, and location deployment in

coalition-based UAV mobile edge computing networks, formulating a discrete multi-leader multi-follower energy minimization Stackelberg game and achieving equilibrium through a hierarchical iterative learning algorithm based on spatial adaptive play best response, obtaining the minimum network energy consumption. Ning et al.[10] considered multi-UAV-assisted MEC networks, addressing multi-user computation offloading and edge server deployment, using an asynchronous update algorithm to minimize system computation cost in dynamic environments. Han et al.[11] studied multi-UAV-assisted MEC networks, jointly optimizing user association and UAV deployment using optimal transport theory and particle swarm algorithm to minimize average task delay. Pang et al.[12] studied a multi-UAV-assisted cloud wireless charging system with cloud server and UAV collaboration, jointly optimizing resources and trajectory using a block coordinate descent algorithm assisted by a Cauchy mutation-based chaotic adaptive beetle swarm optimization algorithm to enhance system energy efficiency. Chen et al.[13] proposed a task offloading strategy and an online UAV scheduling algorithm based on the shortest distance algorithm for multi-UAV-assisted MEC networks, aiming to minimize UAV energy consumption and achieve higher resource utilization. Zhu et al.[14] employed a combination of particle swarm optimization and genetic algorithm operators to optimize UAV deployment for minimal delay. These methods obtain optimal solutions for task offloading, resource allocation, or UAV trajectories through iterative solving. However, UAV networks often exhibit high dynamics; the continuous movement of UAVs leads to rapid changes in network topology and communication channels, along with the stochastic generation of tasks. This results in an extremely complex system state space, where traditional methods often suffer from high computational complexity, making real-time efficient decision-making difficult, and their scalability performs poorly when facing large-scale UAV swarms.

Deep reinforcement learning (DRL) offers new solutions for dynamic decision-making problems. Single-agent DRL algorithms, such as Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), and Proximal Policy Optimization (PPO), are applied to optimize mobile edge offloading scenarios for a single UAV. Liu et al.[15] proposed a centralized Deep Deterministic Policy Gradient (DDPG) algorithm based on a Markov Decision Process (MDP) combining short-term task offloading and bandwidth resource allocation to reduce frequent cache switching and system overhead at edge nodes, experimentally verifying the model's efficiency. Wang et al.[16] proposed the TD3-TDCTM algorithm for UAV-assisted MEC systems to optimize UAV trajectories

and minimize data collection time. Adnan et al.[17] utilized a reinforcement learning (RL) strategy framework for path planning optimization in multi-UAV-assisted mobile edge computing (MEC) networks. Lin et al.[18] integrated DRL with linear programming, proposing the PDDQNLP algorithm to maximize UAV energy efficiency while considering offloading fairness. Liu et al.[19] modeled environments like mountainous and disaster areas, proposing the MODRL-COP algorithm to effectively alleviate the conflict between consumption and delay. Wang et al.[20] proposed a hybrid trajectory planning algorithm combining PF and DRL for feature extraction and reward shaping, enhancing the agent's environmental perception and decision-making capabilities. Huo et al.[21] proposed a joint scheduling strategy based on an improved MATD3 algorithm to obtain robust solutions for UAVs in highly dynamic environments with channel fading, dynamic computing loads, and energy constraints. Zhang et al.[22] proposed an end-to-end model based on Deep Q-Networks to jointly optimize computation offloading and UAV trajectory control, reducing the overall system delay and energy consumption. Liu et al.[23] employed a Double Deep Q-Network to optimize UAV flight trajectories and associated users, maximizing system throughput in a real-time user mobility environment. However, in multi-UAV systems, employing single-agent reinforcement learning algorithms leads to a dimension explosion in state and action spaces, i.e., the "curse of dimensionality," making training difficult and convergence challenging.

In contrast, multi-agent reinforcement learning (MARL) is more suitable for multi-UAV collaborative systems. It optimizes global objectives through cooperation and competition among agents. For multi-UAV edge computing, Wang et al.[24] effectively addressed the load-balancing problem using MARL methods. Wang et al.[25] utilized the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) method to optimize multi-UAV trajectories in real-time, maximizing the fairness of computing services. Nie et al.[26] targeted privacy issues in multi-UAV networks, using multi-agent federated learning for distributed training, achieving results close to MARL. Tao et al.[27] employed the MADDPG algorithm to find optimal task offloading strategies and UAV flight trajectories in dynamic air-ground network environments, reducing overall energy consumption and improving computational performance. Li et al.[28] adopted a multi-agent deep reinforcement learning scheme to balance task offloading fairness and energy efficiency for multiple UAVs in dynamic environments. Hu et al.[29] proposed an offloading optimization and resource allocation algorithm based on the multi-agent deep deterministic policy gradient to determine offloading

strategies for MEC servers and terminal devices. Qin et al.[30] modeled users and UAVs as heterogeneous agents and used the multi-agent deep deterministic policy gradient algorithm to handle high-dimensional hybrid action space problems. SEID et al.[31] designed a multi-UAV-based IoT edge network model and proposed a multi-agent deep reinforcement learning approach for collaborative computation offloading and resource allocation.

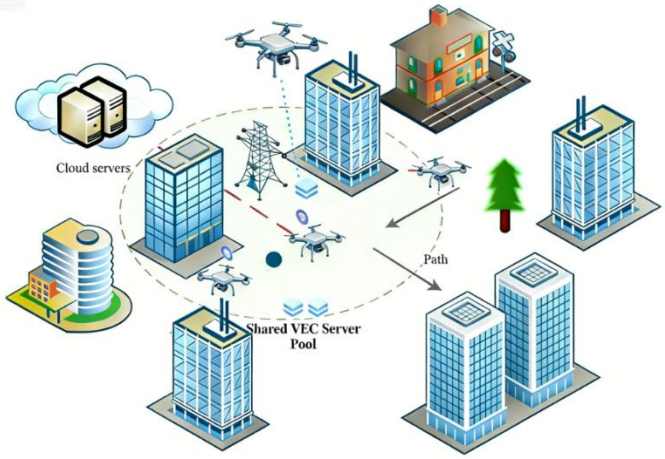
Although the existing studies have provided useful insights into UAV task offloading and multi-agent decision-making, several gaps remain in large-scale heterogeneous UAV edge computing environments. In particular, many existing methods rely on centralized optimization or simplified system assumptions, which limits their applicability under partial observability and dynamic network conditions. In addition, some MARL-based approaches do not explicitly account for the computational overhead introduced by dense inter-agent interactions, which can reduce scalability as the number of UAVs increases. The stability of policy learning under discrete offloading decisions is also still insufficiently explored, especially when queue states and resource availability vary over time. Motivated by these limitations, this paper investigates an energy-efficient collaborative offloading strategy for large UAV swarms and develops an MAPPO-based framework with a top-K sparse attention mechanism and a Gaussian-based latent policy representation to improve scalability and training stability.

3. System Model And Problem Formulation

3.1 System Architecture

This paper considers a heterogeneous UAV edge offloading system and constructs a three-layer architecture consisting of a UAV layer, an edge server layer, and a cloud layer. The system is designed to reflect the practical characteristics of multi-UAV task execution, including heterogeneous UAV capabilities, dynamic wireless access, queue congestion, and resource contention among multiple computing nodes.

Figure 1 UAV Edge Computing Architecture for Multi-Zone Aerial Coverage



In the considered scenario, UAVs generate computation tasks over time and must decide, for each task, whether to execute it locally, offload it to an edge server, or offload it to the cloud. The architecture is intended to capture the trade-off between energy consumption and processing delay under limited onboard resources. The UAV layer is responsible for task generation and local execution; the edge server layer provides nearby computing services with relatively low communication delay; and the cloud layer offers stronger computing capability but typically incurs larger transmission delay.

(1) The UAV layer consists of a heterogeneous set of UAVs, denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$, each equipped with computing resources. UAVs are categorized into three types: light, medium, and heavy, with the primary distinction lying in their power consumption. The UAVs move along predefined flight corridors. The communication environment is divided into multiple zones (e.g. sparse, dense, and crossing zones). UAV speed is configured according to corridor type, with higher speed in sparse zones and lower speed in congested zones. The position of each UAV U_i is represented by a discrete-time model: $P_i[t] = (X_i[t], Y_i[t])$. The displacement of the UAV in each time slot is related to its speed V_i, which depends on the type of flight corridor it is in: high speed in sparse zones, medium speed in dense zones, and low speed in congested zones.

(2) The edge server layer comprises a set of $\mathcal{M}$ edge servers (ESs), denoted as $\mathcal{M} = \{1,2,\ldots,M\}$. Each edge server $m \in \mathcal{M}$ is uniformly deployed along the flight routes to provide computing and communication services for UAVs. Each edge server possesses powerful computing capabilities, and they are interconnected via high-speed wired links, forming a shared service pool. When an edge server becomes overloaded with tasks, its tasks are reassigned to other idle edge

servers. Meanwhile, each edge server maintains a task queue to process tasks offloaded by UAVs.

(3) The cloud layer contains a powerful cloud server. This server provides abundant computing resources for UAVs. The cloud server also maintains a task queue and processes computing tasks based on their priority.

3.2 Communication Model

The communication framework supports cross-layer data exchange, including U2E (UAV-to-edge server) and U2C (UAV-to-cloud server) communications. Since the energy consumption of a UAV depends on task processing duration and the corresponding power consumption, the processing time is calculated first. In this paper, a task-level full offloading mode is adopted; for each task, the UAV selects either local execution, edge offloading, or cloud offloading. Let C_k denote the data volume of task k.

Different UAV types exhibit variations in local computing capacity and energy consumption parameters. Specifically, the type-dependent computing capacity F_u^{uav} is defined as:

$$F_u^{uav} = \begin{cases} 1000, & \tau_u = light \\ 2000, & \tau_u = medium \\ 3000, & \tau_u = heavy \end{cases} \tag{1}$$

Furthermore, a type power coefficient η_u is introduced to characterize the differences in energy consumption characteristics among UAV types:

$$\eta_u = \begin{cases} 0.8, & \tau_u = light \\ 1.0, & \tau_u = medium \\ 1.2, & \tau_u = heavy \end{cases} \tag{2}$$

The power parameters for each UAV in various operating states (idle power P_u^{idle}, computing power P_u^{compute}, upload power P_u^{up}, download power P_u^{down}) are obtained by multiplying the base power values by this type coefficient

Here, the base power values are determined by hardware characteristics and are set to typical values in the simulation. These parameter values are summarized in Table 1.

3.2.1 U2E Communication

In time slot t, the signal-to-interference-plus-noise ratio *(SINR*) of the link between the UAV and the edge server is defined as:

$$SINR_{u,m}^{AG}(t) = \frac{P_u^{tx}\left|h_{u,m}^{AG}(t)\right|^2}{N_0 B^{U2E} + I_{u,m}^{U2E}(t) + I_{other}^{U2U}(t)} \tag{3}$$

where P_u^{tx} denotes the UAV's transmission power, $h_{u,m}^{AG}(t)$ represents the channel gain,

$I_{u,m}^{U2G}(t)$ indicates the total interference from concurrent transmissions, and $I_{other}^{U2U}(t)$ refers to the interference among UAVs.

During time slot t, the data transmission rate between the UAV and the edge server is:

$$R_{u,m}^{U2R}(t) = B^{U2R}\, log_2\left(1 + SINR_{u,m}^{AG}(t)\right) \tag{4}$$

the task upload time from the UAV to the edge server is:

$$T_{k,u}^{up,es}(t) = \frac{C_k}{R_{u,m}^{U2E}(t)} \tag{5}$$

The time required to download the task results from the edge server to the UAV within time slot t is

$$T_{k,u}^{down,es}(t) = \frac{\gamma C_k}{R_{u,m}^{U2E}(t)} \tag{6}$$

where γC_k denotes the calculated data volume.

3.2.2 U2C Communication (UAV to Cloud Server):

During time slot t, the communication transmission rate between the UAV and the cloud server is:

$$R_u^{U2C}(t) = B^{U2C} \tag{7}$$

Time to upload the task from the UAV to the cloud server:

$$T_{k,u}^{up,cloud}(t) = \frac{C_k}{R_u^{U2C}(t)} \tag{8}$$

Time to download the task from the cloud server to the UAV:

$$T_{k,u}^{down,cloud}(t) = \frac{\gamma C_k}{R_u^{U2C}(t)} \tag{9}$$

3.3 Energy Consumption Calculation Model

This section constructs a computational model to describe the task processing process at three offloading destinations: local execution, edge offloading, and cloud offloading.

The task completion time consists of execution time, queuing delay, and transmission delay, and the UAV's energy consumption depends on the execution time, queuing delay, transmission time, and corresponding power. To model dynamic task arrival and processing rates, the M/M/1 queue model is adopted.

3.3.1 Local Energy Consumption Calculation Model

The execution time of computing tasks performed locally by UAVs depends on its computational capabilities:

$$T_{k,u}^{comp,loc}(t) = \frac{C_k}{F_u^{uav}} \tag{10}$$

where F_u^{uav} denotes the local computing capacity.

In time slot t, the UAV locally maintains a task queue with the following queuing time:

$$T_{k,u}^{queue,loc}(t) = \frac{1}{\mu_u^{loc}(t) - \lambda_u^{loc}(t)} \tag{11}$$

where $\mu_u^{loc}(t)$ denotes the service rate of UAV tasks, while represents the arrival rate of these tasks.

Therefore, within time slot t, the energy consumption of the UAV's local computation is:

$$E_{k,u}^{loc}(t) = T_{k,u}^{comp,loc}(t) \cdot P_u^{compute} + T_{k,u}^{queue,loc}(t) \cdot P_u^{idle} \tag{12}$$

3.3.2 Energy Consumption Calculation Model for Edge Servers

When a UAV offloads computational tasks to an ES, the ES's processing time depends on its computational capacity:

$$T_{k,u}^{comp,es}(t) = \frac{C_k}{F_g^{es}} \tag{13}$$

During time slot t, the ES also maintains a task queue with the following queuing delay:

$$T_{k,u}^{queue,es}(t) = \frac{1}{\mu_u^{es}(t) - \lambda_u^{es}(t)} \tag{14}$$

where $\mu_u^{es}(t)$ denotes the task service rate of the edge server, while $\lambda_u^{es}(t)$ represents its task arrival rate.

Therefore, when the UAV offloads tasks to the edge server, its total energy consumption is

$$\begin{aligned} E_{k,u}^{es}(t) = T_{k,u}^{up,es}(t) \cdot P_u^{up} + T_{k,u}^{queue,es}(t) \cdot P_u^{idle} + \\ T_{k,u}^{comp,es}(t) \cdot P_u^{idle} + T_{k,u}^{down,es}(t) \cdot P_u^{down} \end{aligned} \tag{15}$$

3.3.3 Cloud Computing

If the UAV chooses to offload computations to the cloud server, the time required for cloud-based computation execution is

$$T_k^{queue,cloud}(t) = \frac{1}{\mu^{cloud}(t) - \lambda^{cloud}(t)} \tag{16}$$

Therefore, the total energy consumption for cloud offloading is:

$$\begin{aligned} E_{k,u}^{cloud}(t) = T_k^{up,cloud}(t) \cdot P_u^{up} + T_{k,u}^{queue,cloud}(t) \cdot P_u^{idle} \\ + T_{k,u}^{comp,cloud}(t) \cdot P_u^{idle} + T_{k,u}^{down,es}(t) \cdot P_u^{down} \end{aligned} \tag{17}$$

The energy model couples computation, communication, and queuing delays, so queue states and link quality affect energy consumption indirectly through delay terms.

3.4 Optimization Problem

The optimization problem is formulated to minimize total energy consumption subject to energy and computational resource constraints as follows:

$$min \sum_{u \in N} \sum_{j \in D} x_{u,j} \cdot E_{u,j} \tag{18}$$

where $E_{u,j}$ denotes the energy consumption when UAV u selects offloading option j.

The energy constraint is expressed as:

$$\sum_{j \in D} x_{u,j} \cdot E_{u,j} \leq E_u^{max}, \ \forall u \in U \tag{19}$$

Each UAV must select exactly one offloading option at a time.

The computational resource constraints are represented as

$$\sum_{u \in U} \sum_{k \in K(t)} x_{k,u,j}(t) \cdot \frac{C_k}{F_j} \leq F_j^{max} \tag{20}$$

Offloading option constraints are given as

$$\sum_{j \in J} x_{k,u,j}(t) = 1 \tag{21}$$
$$x_{k,u,j}(t) \in \{0,1\}, \forall k, n, j$$

The system requires the UAV to decide whether to execute the computational tasks locally, offload them to the ES, or offload them to the cloud.

3.5 Shared edge server pool and task priority queue

The proposed framework incorporates a shared edge-server pool and a task priority queue to improve resource utilization and load balancing. The edge servers are aggregated into a unified resource pool. When edge server m becomes overloaded, the task is reallocated to other underutilized edge servers in the shared pool m^*. The selection criteria for the optimal edge server are:

$$m^* = arg \min_{(j \in \mathcal{M})} \left(Q_j^{es}(t) + \frac{C_k}{F_j^{es}} \right) \tag{22}$$

where $Q_k^{es}(t)$ denotes the task queue of edge server j, and the ratio C_k/F_j^{es} is employed to approximate the processing delay of task k.

The task priority queue determines the processing order of tasks based on their specific priorities. The calculation formula is as follows:

$$S_k(t) = \frac{P_k}{1 + \alpha Q_j^{es}(t)} \tag{23}$$

The priority of task k is denoted as P_k, with α being the sensitivity coefficient. Tasks with higher priority will be processed first in the queue, thereby reducing the UAV's standby time and overall energy consumption.

The system model described above shows that task offloading decisions are strongly coupled with task size, queue congestion, communication conditions, and UAV heterogeneity. Since each UAV can only observe limited local information and the system state evolves dynamically over time, the resulting decision-making process is naturally modeled as a Dec-POMDP. Under this formulation, each UAV must infer an appropriate offloading action from partial observations while considering the impact of its decision on queue delay and energy consumption. To solve this problem, we adopt MAPPO as the core learning framework and further enhance it with a top-K sparse attention mechanism and a Gaussian-based latent policy representation to improve scalability and training stability.

4. Deep Reinforcement Learning Framework Based on the MAPPO Algorithm

The task offloading problem in the considered heterogeneous UAV edge computing system is characterized by partial observability, dynamic queue states, and frequent competition for communication and computation resources. Each UAV can only observe local information and a limited set of server states, while the global system evolves according to the joint decisions of all agents. Therefore, the offloading problem is naturally modeled as a decentralized partially observable Markov decision process (Dec-POMDP).

To solve this problem, we adopt the centralized training and decentralized execution paradigm and use MAPPO as the baseline multi-agent reinforcement learning framework. On top of MAPPO, we further propose SS-MAPPO, which introduces a Gaussian-based latent policy representation in the actor and a top-K sparse attention mechanism in the centralized critic. These two improvements are designed to enhance policy stability and scalability in large-scale multi-UAV offloading scenarios.

4.1 Dec-POMDP Formulation

The multi-UAV offloading problem is formulated as a Dec-POMDP, which can be represented by the tuple

$$\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \{\mathcal{O}_i\}_{i=1}^{N}, \{\mathcal{A}_i\}_{i=1}^{N}, P, R, \gamma \rangle. \quad (24)$$

Here, $\mathcal{N} = \{1,2,\dots,N\}$ denotes the set of UAV agents, $\mathcal{S}$ denotes the global state

space, $\mathcal{O}_i$ denotes the observation space of agent i, $\mathcal{A}_i$ denotes the action space of agent i, P denotes the state transition probability, R denotes the reward function, and $\gamma \in (0,1)$ is the discount factor.

At each time slot t, the environment is in state $s_t \in \mathcal{S}$. Agent i receives a local observation $o_t^i \in \mathcal{O}_i$, selects an action $a_t^i \in \mathcal{A}_i$, and the joint action of all agents is denoted by

$$\mathrm{a}_t = (a_t^1, a_t^2, \dots, a_t^N). \qquad (25)$$

The environment then evolves according to

$$P(s_{t+1} \mid s_t, \mathrm{a}_t), \qquad (26)$$

and each agent receives an instantaneous reward r_t^i. The objective of each agent is to maximize the expected discounted return, and the system objective is to minimize the total energy consumption of all UAVs while satisfying the task execution requirements.

4.2 Observation Space, Action Space, and Reward Design

Each UAV makes decisions using only partial local information. To reflect the factors that directly affect task execution delay and energy consumption, the local observation of agent i at time slot t is defined as

$$o_t^i = [d_t^i, q_{u,t}^i, q_{e,t}^i, q_{c,t}, h_{u,e,t}^i, h_{u,c,t}^i, e_t^i], \qquad (27)$$

where d_t^i is the size of the newly generated task, $q_{u,t}^i$ is the local queue length of UAV i, $q_{e,t}^i$ is the queue length of the accessible edge server, $q_{c,t}$ is the queue length of the cloud server, $h_{u,e,t}^i$ and $h_{u,c,t}^i$ denote the channel conditions for UAV-to-edge and UAV-to-cloud transmission, and e_t^i represents the remaining energy of UAV i.

The action space of each UAV is discrete and defined as

$$\mathcal{A}_i = \{0,1,2\}, \qquad (28)$$

where $a_t^i = 0$ denotes local execution, $a_t^i = 1$ denotes offloading to an edge server, and $a_t^i = 2$ denotes offloading to the cloud server.

To guide the learning process toward energy-efficient offloading, the reward is designed as a negative normalized energy cost with additional penalty terms for invalid or overloaded decisions. The reward of agent i at time slot t is given by

$$r_t^i = -\frac{E_t^i}{E_{\max}} - \lambda_1 \mathbb{I}_{\text{violate}} - \lambda_2 \mathbb{I}_{\text{overflow}}, \qquad (29)$$

where E_t^i denotes the energy consumption induced by the selected action, $E_{\max}$ is a

normalization factor, $\mathbb{I}_{\text{violate}}$ is an indicator of constraint violation, $\mathbb{I}_{\text{overflow}}$ is an indicator of server overload or queue overflow, and λ_1, λ_2 are penalty coefficients.

The cumulative return of agent i is defined as

$$G_t^i = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k}^i. \qquad (30)$$

Maximizing G_t^i is equivalent to minimizing long-term energy consumption while maintaining feasible task processing behavior.

4.3 MAPPO Baseline

MAPPO follows the centralized training and decentralized execution paradigm. During training, a centralized critic uses global information to estimate the value of the joint state, while the actor of each agent is trained using only local observations. During execution, each UAV chooses actions independently according to its own policy.

Let $\pi_{\theta_i}(a_t^i \mid o_t^i)$ denote the policy of agent i parameterized by θ_i. The joint policy of the system is

$$\pi_\theta(\mathrm{a}_t \mid \mathrm{o}_t) = \prod_{i=1}^{N} \pi_{\theta_i}(a_t^i \mid o_t^i), \qquad (31)$$

where $\mathrm{o}_t = (o_t^1, o_t^2, \ldots, o_t^N)$.

The centralized critic estimates the value function $V_\phi(s_t)$, where ϕ denotes the critic parameters. To improve policy learning stability, MAPPO uses the PPO clipped objective. The probability ratio is defined as

$$\rho_t^i(\theta_i) = \frac{\pi_{\theta_i}(a_t^i \mid o_t^i)}{\pi_{\theta_i}^{\text{old}}(a_t^i \mid o_t^i)}.$$

$$L_{\text{clip}}^i(\theta_i) = \mathbb{E}_t\big[min\big(\rho_t^i(\theta_i)\hat{A}_t^i, \text{clip}\big(\rho_t^i(\theta_i), 1-\epsilon, 1+\epsilon\big)\hat{A}_t^i\big)\big], \qquad (32)$$

where $\hat{A}_t^i$ is the advantage estimate and ϵ is the clipping parameter.

The advantage is computed using generalized advantage estimation (GAE) as

$$\delta_t^i = r_t^i + \gamma V_\phi(s_{t+1}) - V_\phi(s_t),$$

$$\hat{A}_t^i = \sum_{l=0}^{T-t} (\gamma\lambda)^l \, \delta_i^{t+l}, \qquad (33)$$

where $\lambda \in [0,1]$ is the GAE parameter.

The critic is trained by minimizing the value loss. To encourage exploration, an entropy

regularization term is added. The final optimization objective of MAPPO is

$$L_{\mathrm{MAPPO}} = \sum_{i=1}^{N} \left(L_{\mathrm{clip}}^{i}(\theta_i) + c_{\mathrm{ent}} \mathbb{E}_t \left[\mathcal{H}\left(\pi_{\theta_i}(\cdot \mid o_t^i) \right) \right] \right) - c_V \mathbb{E}_t \left[\left(V_\phi(s_t) - \hat{R}_t \right)^2 \right], \tag{34}$$

where $\hat{R}_t$ is the return target, $\mathcal{H}(\cdot)$ is the entropy function, and c_{ent} and c_V are balancing coefficients.

Although MAPPO is effective for cooperative decision-making, a standard centralized critic becomes increasingly expensive in large-scale systems, and a directly parameterized discrete actor may produce unstable action transitions when queue states and channel states change rapidly. These issues motivate the proposed SS-MAPPO framework.

4.4 SS-MAPPO Algorithm Framework

SS-MAPPO improves MAPPO from two aspects. First, the actor adopts a Gaussian-based latent policy representation to smooth policy generation in the hidden space. Second, the centralized critic uses a top-K sparse attention mechanism to reduce redundant inter-agent coupling and improve scalability.

4.4.1 Gaussian-Based Latent Policy Representation

Instead of mapping observations directly to discrete action logits, the actor first encodes the local observation into a latent Gaussian distribution. Let $g_\psi(\cdot)$denote the observation encoder together with the mean and standard deviation heads. The latent policy representation is given by

$$\mu_t^i, \sigma_t^i = g_\psi\left(o_t^i\right). \tag{35}$$

A latent variable is sampled through the reparameterization form

$$z_t^i = \mu_t^i + \sigma_t^i \odot \epsilon_t^i \,,\ \epsilon_t^i \sim N(0, I),. \tag{36}$$

and the sampled latent vector is then mapped to the discrete action distribution by a projection layer:

$$p_t^i = \mathrm{softmax}\left(W_p z_t^i + b_p\right),\ a_t^i \sim \mathrm{Cat}\left(p_t^i\right), \tag{37}$$

This design does not change the discrete nature of the offloading decision. Instead, the Gaussian latent representation acts as an intermediate stochastic model that helps smooth the policy update process and reduces abrupt action changes under dynamic queue and channel conditions.

4.4.2 Top-K Sparse Attention Mechanism

To estimate the global value more efficiently, SS-MAPPO uses a top-K sparse attention mechanism in the centralized critic. Let x_t^idenote the embedding of agent i, which is obtained from

its observation and relevant state features. The query, key, and value vectors are generated as

$$q_t^i = W_q x_t^i,\ k_t^i = W_k x_t^i,\ v_t^i = W_v x_t^i \qquad (38)$$

where W_q, W_k, and W_v are learnable projection matrices. For each agent i, let $\mathcal{K}_i$ be the index set of the top-K agents with the highest attention scores. The sparse attention weight from agent i to agent j is computed as

$$\alpha_{ij} = \frac{\exp\left(q_t^{i\top} k_t^j / \sqrt{d}\right)}{\sum_{l\in\mathcal{K}_i} \exp\left(q_t^{i\top} k_t^l / \sqrt{d}\right)},\ j \in \mathcal{K}_i, \qquad (39)$$

and $\alpha_{ij} = 0$ for $j \notin \mathcal{K}_i$, where d is the feature dimension. The contextual representation of agent i is obtained by aggregating the selected value vectors:

$$c_t^i = \sum_{j\in\mathcal{K}_i} \alpha_{ij}\, v_t^j,\ V_\phi(s_t) = f_V([c_t^1, c_t^2, \dots, c_t^N]), \qquad (40)$$

where $f_V(\cdot)$ is the value head of the critic network.

Compared with dense attention, top-K sparse attention reduces redundant pairwise interaction modeling and lowers computational overhead. At the same time, it preserves the most influential inter-agent dependencies, which is important in offloading systems where queue contention and shared resource competition dominate the system dynamics.

4.4.3 Overall Optimization Objective

Combining the latent actor and sparse critic, the final SS-MAPPO objective is written as

$$L_{\text{SS-MAPPO}} = \sum_{i=1}^{N} \left(L_{clip}^i(\theta_i) + c_{ent} H\left(\pi_{\theta_i}(\cdot \mid o_t^i)\right)\right) - c_V E_t\left[\left(V_\phi(s_t) - \hat{R}_t\right)^2\right], \qquad (41)$$

where the actor loss and critic loss are computed using the latent-policy-based action distribution and the sparse-attention-based value estimate, respectively.

4.5 Theoretical Analysis

This section provides an intuitive analysis of the advantages of SS-MAPPO in terms of computational complexity, policy smoothness, and training stability.

4.5.1 Computational Complexity

In a dense attention-based critic, each agent must consider its relationship with all other agents. As a result, the interaction cost grows quadratically with the number of UAVs, which becomes expensive in large-scale systems.

With top-K sparse attention, each agent retains only the K most relevant neighbors. Therefore,

the per-agent interaction cost is reduced from dependence on all N agents to dependence on only K selected agents. When $K \ll N$, the computational and memory overhead is substantially reduced. This makes the critic more scalable and more suitable for large UAV swarms.

4.5.2 Policy Smoothness

In the actor, the latent Gaussian variable introduces stochasticity in a structured way. Since the final discrete policy is generated from a latent representation rather than directly from the raw observation, the policy is less sensitive to small perturbations in the input features. This is useful in dynamic offloading environments where queue lengths, link quality, and server load vary over time.

Therefore, the proposed policy parameterization may help the actor avoid abrupt changes in action preference and improve exploration behavior during training.

4.5.3 Training Stability

PPO already limits drastic policy updates through the clipped objective. When combined with the Gaussian-based latent policy representation, the update process becomes smoother in practice. Meanwhile, sparse attention reduces the complexity of the critic and avoids excessive coupling between weakly related agents.

As a result, SS-MAPPO can improve the stability of multi-agent policy learning, especially in environments with partial observability, queue congestion, and frequent resource contention.

Algorithm 1 Training Process of SS-MAPPO

Algorithm 1 Training Procedure of SS-MAPPO

Input: number of agents N, environment Env, actor learning rate α_a, critic learning rate α_c, discount factor γ, PPO clipping parameter ϵ, GAE parameter λ, entropy coefficient β, PPO update epochs K, minibatch size B, training epochs E, and time slots per epoch T

Output: trained actor networks and critic network

1. Initialize N actor networks $\{\pi_{\theta_i}\}_{i=1}^{N}$, one centralized critic network V_ϕ, and their optimizers.
2. Initialize the rollout buffer.
3. For epoch $= 1$ to E do
4. Initialize the UAV swarm, edge servers, cloud server, and all queue states in the environment.
5. For time slot $t = 1$ to T do
6. Collect the local observation $o_{i,t}$ for each agent i.

7. For each agent i, compute the action distribution using the Gaussian-based latent policy representation actor and sample an action $a_{i,t}$.
8. Execute the joint action $a_t = \{a_{1,t}, a_{2,t}, \dots, a_{N,t}\}$ in the environment.
9. Obtain the immediate reward $R_{i,t}$, next observation $o_{i,t+1}$, and termination signal $done_t$ from the environment.
10. Store the transition $(o_t, a_t, R_t, o_{t+1}, done_t, \log \pi_t)$ into the rollout buffer.
11. Update the task queues of UAVs, edge servers, and the cloud server.
12. End for
13. Compute state values $V_\phi(s_t)$ and $V_\phi(s_{t+1})$ using the centralized critic.
14. Estimate advantages with GAE:

$$\delta_t = R_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

$$\hat{A}_t = \delta_t + \gamma\lambda\hat{A}_{t+1}$$

15. Normalize the advantages.
16. For update step $k = 1$ to K do
17. Sample a minibatch of size B from the rollout buffer.
18. Compute the probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_{i,t} \mid o_{i,t})}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid o_{i,t})}$$

19. Update the Actor networks by maximizing the PPO clipped objective with entropy regularization:

$$L_t^{\text{CLIP}}(\theta) = \min(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$$

20. Update the critic network by minimizing the value regression loss:

$$L^V(\phi) = (V_\phi(s_t) - \hat{G}_t)^2$$

21. End for
22. Clear the rollout buffer.
23. End for

5. Simulation Results

In this paper, the algorithm performance is evaluated through extensive simulation experiments. The simulation scenarios include 120 UAVs and tasks of four different scales. The simulation parameters are given as follows in Table 1.

To comprehensively evaluate the effectiveness of the proposed method, we select baselines from three representative categories: learning-based multi-agent methods, classical heuristic algorithms, and rule-based scheduling policies. MAPPO is included as the most direct baseline because it shares the same CTDE framework with SS-MAPPO, and the performance gap between them can clearly reflect the contribution of the proposed top-K sparse attention mechanism and policy parameterization. MADDPG and MATD3 are selected as representative multi-agent actor-critic methods to assess the advantage of SS-MAPPO over widely used deterministic MARL approaches under partial observability and dynamic queue states. Genetic algorithm, simulated annealing, and round-robin scheduling are used as heuristic and rule-based references to provide non-learning benchmarks and to demonstrate the superiority of the proposed method in terms of adaptability, scalability, and decision quality. This baseline set covers the major solution paradigms for UAV MEC offloading and therefore enables a fair and balanced comparison.

Table 1 Simulation Parameters

Parameter	Value	Unit
Number of UAVs	120	
Number of edge servers	24	
Number of cloud servers	1	
Wireless transmission power	10	W
Task size	300-3300	KB
UAV computing capacity	1000/2000/3000	KB
Edge server computing capacity	20000	KB
Cloud computing capacity	200000	KB
Bandwidth of UAV to edge server	10	Mbps
Bandwidth of UAV to cloud server	20	Mbps
Base idle power	8	W
Base computing power	40	W
Base upload/download power	10	W
Edge server transmission power	8	W
Training epochs	200	
Time slots	100	
Rollout buffer size	64	
Batch size	64	
Discount factor (γ)	0.9	
PPO clip (ε)	0.2	
Entropy	0.2	

The environment parameters are configured for a UAV edge computing scenario, aiming

to preserve simulation feasibility while capturing the key characteristics of dynamic, resource-constrained, and delay-sensitive task offloading. The number of UAVs is set to 120 to emulate a dense aerial workload, thereby ensuring sufficient competition for task requests and computing resources. A multi-region edge server deployment is adopted to form a typical air-ground cooperative computing architecture, where the service capability and workload of each node vary dynamically with UAV mobility, reflecting the heterogeneous nature of practical UAV networks. Meanwhile, the task arrival rate, task size, computation density, bandwidth, and mobility-related parameters are selected within commonly used ranges in UAV networking studies to capture wireless link fluctuations, limited edge computing capacity, and scheduling pressure in realistic scenarios. A workload that is too light would fail to reveal performance differences among policies, whereas an excessively heavy workload may keep the system persistently congested and thus hinder fair comparison. Therefore, the adopted parameter settings strike a balance between challenge and feasibility, and are representative of practical UAV edge computing environments.

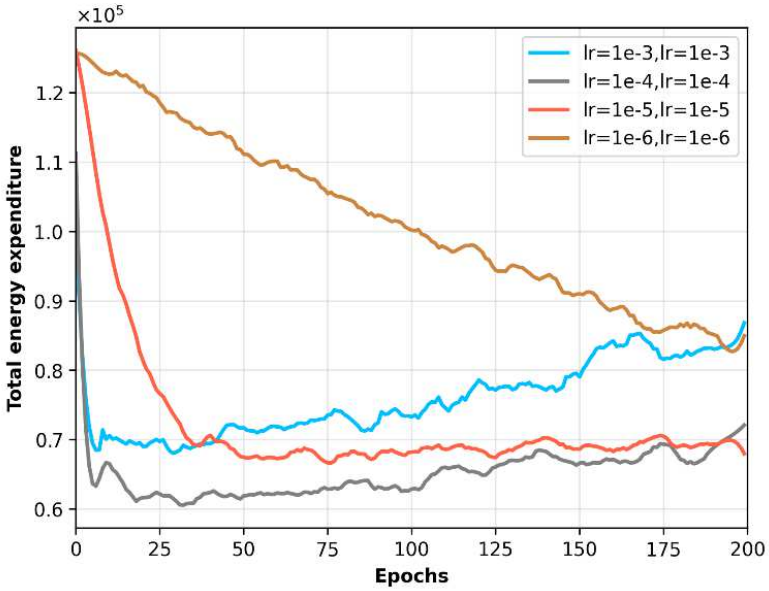
5.1 Experiment 1: Effect of Learning Rate on Algorithm Convergence

The learning rate is an important hyperparameter in deep reinforcement learning because it directly affects the update step size of the policy network and value network, and therefore influences both convergence behavior and training stability. To examine the effect of different learning rate settings on the training performance of SS-MAPPO, four groups of comparative experiments were conducted. The learning rate combinations for the Actor and Critic networks were set to 1×10^{-4}, 1×10^{-5}, 1×10^{-6}, and a mixed setting (Actor: 1×10^{-4}, Critic: 1×10^{-5}). All experiments were performed under the same simulation conditions, and the evaluation metric was the average energy consumption over 200 training episodes.

As shown in Figure 2, when the learning rate was set to 1×10^{-4}, the average energy consumption decreased rapidly during the early training stage, indicating faster initial convergence. However, after approximately 30 episodes, the curve became noticeably more volatile, suggesting that a relatively large learning rate may lead to unstable policy updates. When the learning rate was set to 1×10^{-6}, the training curve decreased more slowly, and the model did not reach a stable state within 200 episodes, which indicates lower training efficiency. In contrast, with a learning rate of 1×10^{-5}, the model achieved relatively stable convergence after

about 50 episodes, and the final energy consumption was lower than that of the other tested settings. The mixed learning rate setting converged quickly in the early stage, but its later-stage stability was slightly weaker than that of the uniform setting. Based on these results, this paper adopts 1×10^{-5} as the learning rate for both the Actor and Critic networks in order to balance convergence speed and training stability.

Figure 2 Energy consumption curves during training under different learning rates



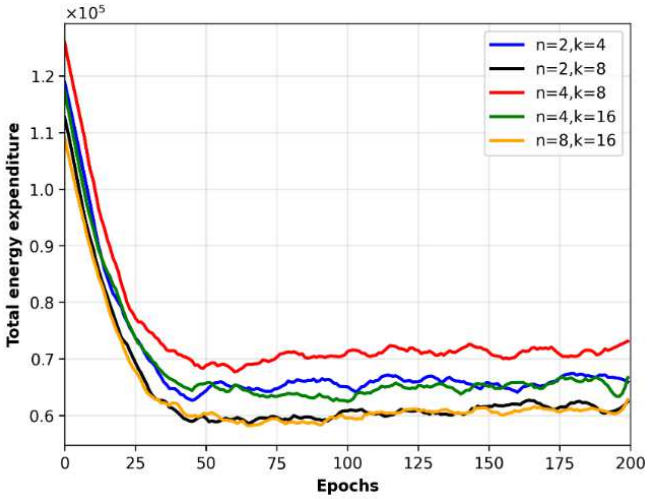
5.2 Experiment 2: Impact of top-K Sparse Attention Mechanism on Algorithm Convergence

To evaluate the effect of the top-K sparse attention mechanism on the convergence behavior of SS-MAPPO, a set of comparative experiments was carried out by varying the number of attention heads (H) and the sparsification parameter (K), which determines how many neighboring agents each agent focuses on. With other hyperparameters fixed, combinations including $n = 2, k = 4$, $n = 2, k = 8$, $n = 4, k = 8$, and $n = 8, k = 16$ were tested, and the average energy consumption over 200 training episodes was recorded.

As shown in Figure 3, the experimental results indicate that the combinations $n = 2, k = 8$ and $n = 8, k = 16$ both achieved low energy consumption levels, with $n = 2, k = 8$ showing a slight advantage in convergence speed. When $k = 4$, each agent focused on too little neighbor information, resulting in insufficient collaborative information, poorer policy convergence, and higher energy consumption. The combination $n = 4, k = 8$ performed slightly better but still fell short of the $n = 2, k = 8$ combination. Considering that training time complexity is positively correlated with the product $n \times k$, the $n = 2, k = 8$ combination ensures good convergence

performance while significantly reducing computational overhead. Therefore, this paper selects $n = 2, k = 8$ as the default parameter configuration for the top-K sparse attention mechanism to achieve a balance between performance and efficiency.

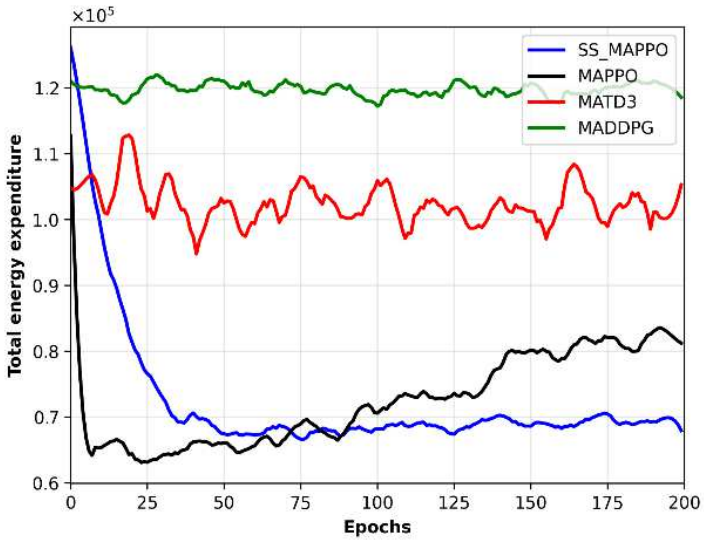
Figure 3 Energy consumption curves during training under different attention parameters



5.3 Experiment 3: Convergence Performance Comparison of Different MARL Algorithms

To compare the convergence behavior of SS-MAPPO with other multi-agent reinforcement learning methods in the considered task offloading scenario, SS-MAPPO was evaluated against MAPPO, MATD3, and MADDPG. All methods were trained under the same simulation settings for 200 episodes, and the average system energy consumption during training was recorded.

Figure 4 Energy consumption convergence curves of different multi-agent reinforcement learning algorithms



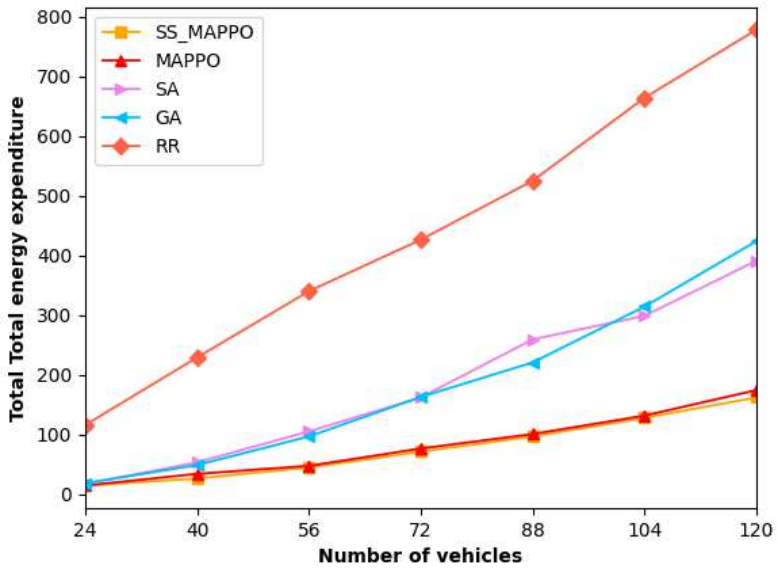
As shown in Figure 4, MADDPG and MATD3 converged relatively slowly in this scenario and did not stabilize at a low energy consumption level within 200 episodes. Their training curves also

exhibited noticeable fluctuations, which suggests that policy updates may be less stable under partial observability and dynamic multi-agent conditions. The original MAPPO algorithm reached a relatively low energy consumption level after about 60 episodes, but its curve still showed some oscillations during later training. In comparison, SS-MAPPO reached stable convergence after about 50 episodes and maintained a lower energy consumption level than the other tested methods in the training process. These results indicate that the introduced top-K sparse attention mechanism and Gaussian-based latent policy representation may help improve convergence behavior and reduce training fluctuations in this scenario.

5.4 Experiment 4: Robustness to Variations in the Number of UAVs

To evaluate the robustness of the algorithms under different swarm sizes, the number of UAVs was increased from 24 to 120 in steps of 16. The average system energy consumption over 200 time slots was measured for SS-MAPPO, MAPPO, the genetic algorithm (GA), simulated annealing (SA), and round-robin (RR). The purpose of this experiment was to observe how each method behaves as the system scale increases and resource competition becomes more intense.

Figure 5 average energy consumption under varying numbers of UAVs



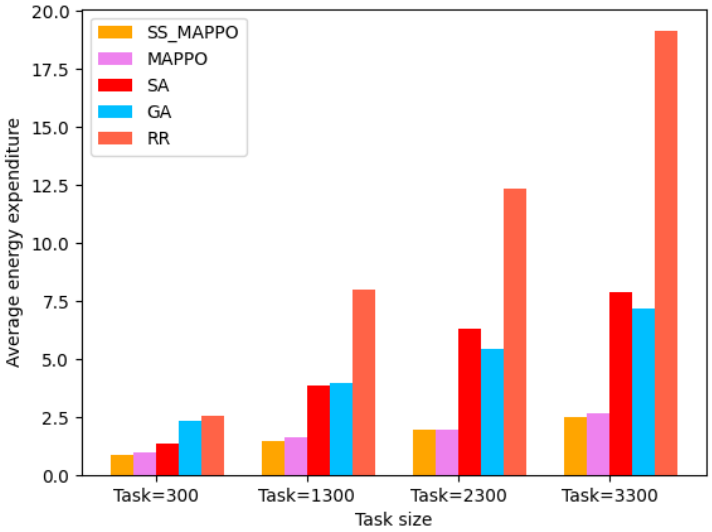
As shown in Figure 5, SS-MAPPO maintained a lower average energy consumption than the compared methods across all tested swarm sizes. In addition, as the number of UAVs increased, the energy consumption of SS-MAPPO rose more gradually than that of the baseline methods, indicating relatively better scalability in the tested scenarios. MAPPO performed slightly worse than SS-MAPPO, while SA and GA showed acceptable performance at smaller scales but degraded more

noticeably as the swarm size increased. RR produced the highest energy consumption and also exhibited larger fluctuations. Overall, these results suggest that the top-K sparse attention mechanism may help reduce the impact of redundant inter-agent interactions in larger swarms, thereby supporting more stable collaborative offloading decisions.

5.5 Experiment 5: Energy Consumption Distribution for Task Completion

To examine the effect of task size on algorithm performance, four task data sizes were considered: 300 KB, 1300 KB, 2300 KB, and 3300 KB. The average energy consumption per completed task over 200 time slots was measured for SS-MAPPO, MAPPO, GA, SA, and RR. This experiment was designed to evaluate how sensitive each method is to task scale changes.

Figure 6 Average energy consumption under different task sizes



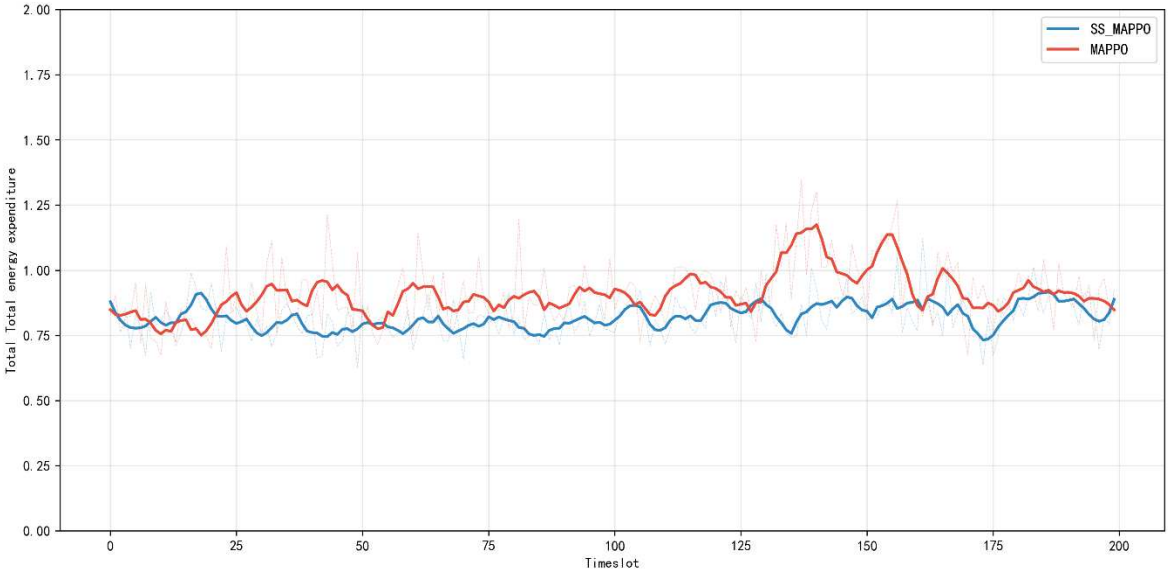
As shown in Figure 6, for the smallest task size of 300 KB, SS-MAPPO and SA achieved similar energy consumption levels. However, as the task size increased to 1300 KB and beyond, the energy consumption of SA and GA increased more rapidly, indicating that these methods may be less effective when handling computation-intensive tasks under resource constraints. SS-MAPPO showed relatively smaller variations across different task sizes and maintained a lower energy consumption level in most cases. Compared with the original MAPPO algorithm, SS-MAPPO did not always produce a large absolute improvement, but it exhibited smaller fluctuations as task size changed, which suggests that the proposed policy representation may contribute to smoother decision behavior and better robustness under the tested conditions.

5.6 Experiment 6: Algorithm Stability

To quantitatively evaluate decision stability, the standard deviation of energy consumption for completing tasks by 120 UAVs over 200 consecutive time slots was calculated and compared between SS-MAPPO and the original MAPPO algorithm. A smaller standard deviation indicates less variation in decisions across time slots and thus relatively higher stability.

As shown in Figure 7, the standard deviation of energy consumption for SS-MAPPO was lower than that of MAPPO throughout the tested period, and the corresponding curve showed smaller fluctuations. This suggests that SS-MAPPO produces more consistent decisions under the tested conditions. The improvement may be related to the top-K sparse attention mechanism, which reduces the influence of less relevant neighbor information, and the Gaussian-based latent policy representation, which may help smooth the action selection process. Overall, these results indicate that SS-MAPPO exhibits relatively more stable decision making in the considered multi-UAV offloading scenario.

Figure 7 standard deviation of energy consumption over 200 time slots



6 Conclusion

This paper investigates the task offloading problem in multi-UAV edge computing environments and develops an improved Multi-Agent Proximal Policy Optimization framework, SS-MAPPO, by incorporating a top-K sparse attention mechanism and a Gaussian-based latent policy representation. A three-layer heterogeneous computing architecture consisting of a UAV layer, an edge server layer, and a cloud layer is constructed, together with a shared edge server pool and a task priority queue, to support resource scheduling and load balancing in the considered scenario.

The proposed SS-MAPPO framework is designed to reduce redundant inter-agent interactions and to provide a smoother policy parameterization for discrete offloading decisions. Under the tested simulation settings, the results indicate that SS-MAPPO achieves lower energy consumption, faster convergence, and smaller decision fluctuations than the selected baseline methods across different swarm sizes and task scales. These observations suggest that the proposed framework may be a useful approach for energy-aware task offloading in dynamic multi-UAV edge computing environments.

However, this study still has several limitations. The current model considers UAVs moving along fixed routes and adopts a full offloading mode, which may not cover all practical deployment conditions. Future work will further investigate more flexible mobility patterns, partial offloading strategies, and more complex task dependencies.

Acknowledgement

This work was supported by the Youth Talent Project of the Scientific Research Program of the Yunnan Key Research and Development Program under Grant 202503AP140020, the Major Science and Technology Special Project of Yunnan Province under Grant202502AD080003, the Hubei Provincial Department of Education under Grant Q20241809, Hubei Provincial Natural Science Foundation under Grand 2026AFC0361, and the Doctoral Scientific Research Foundation of Hubei University of Automotive Technology under Grant BK202404.

Funding Declaration

The Youth Talent Project of the Scientific Research Program of the Yunnan Key Research and Development Program under Grant 202503AP140020
The Major Science and Technology Special Project of Yunnan Province under Grant202502AD080003
The Hubei Provincial Department of Education under Grant Q20241809
Hubei Provincial Natural Science Foundation under Grand 2026AFC0361
The Doctoral Scientific Research Foundation of Hubei University of Automotive Technology under Grant BK202404

Ethics, Consent to Participate, and Consent to Publish declarations: not applicable

Datasets Generated： The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Author contributions

[Changhao Duan and Zheng Yao]: Writing–original draft;[Changhao Duan]: Data curation;

[Changhao Duan]:Formal analysis; [Zheng Yao]: Conceptualization; [Zheng Yao]: Funding acquisition; [Zheng Yao]: Supervision; [Changhao Duan, Zheng Yao, ChangJun Deng and Rui Li]: Writing–review & editing. All authors have read and agreed to the published version of the manuscript.

References

[1] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 596–630, 2020. https://doi.org/10.1109/COMST.2020.3037674

[2] Y. Liu, K. Xiong, Q. Ni, et al., "UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2777–2790, 2020. https://doi.org/10.1109/JIOT.2019.2958975

[3] W. Pan, Y. Liu, and C. Yang, "Adaptive task offloading of rechargeable UAV edge computing network based on double decision value iteration," *Comput. Commun*, vol. 193, pp. 136–145, 2022. https://doi.org/10.1016/j.comcom.2022.06.026

[4] M. Y. Akhlaqi and Z. B. Mohd Hanapi, "Task offloading paradigm in mobile edge computing current issues, adopted approaches, and future directions," *J. Netw. Comput. Appl.*, vol. 212, p. 103568, 2023. https://doi.org/10.1016/j.jnca.2022.103568

[5] Z. Wei, X. B. Yu, and L. Zou, "Multi-resource computing offload strategy for energy consumption optimization in mobile edge computing," *Processes*, vol. 10, no. 9, p. 1762, 2022. https://doi.org/10.3390/pr10091762

[6] A. Bozorgchenani, F. Mashhadi, D. Tarchi, et al., "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2992–3005, 2021. https://doi.org/10.1109/TMC.2020.2994232

[7] H. Zhou, Z. Y. Zhang, D. W. Li, et al., "Joint optimization of computing offloading and service caching in edge computing-based smart grid," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1122–1132, 2023. https://doi.org/10.1109/TCC.2022.3163750

[8] Y. Wang, W. Chen, T. H. Luan, et al., "Task offloading for post-disaster rescue in unmanned aerial vehicles networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1525–1539, 2022. https://doi.org/10.1109/TNET.2022.3140796

[9] J. X. Chen, Q. H. Wu, Y. H. Xu, et al., "A multi-leader multi-follower stackelberg game for coalition-based UAV MEC networks," *IEEE Wireless Commun. Lett.*, vol. 10, no. 11, pp. 2350–2354, 2021. https://doi.org/10.1109/LWC.2021.3100113

[10] Z. Ning, Y. Yang, X. Wang, et al., "Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2628–2644, 2023. https://doi.org/10.1109/TMC.2021.3129785

[11] Z. Han, T. Zhou, T. Xu, et al., "Joint user association and deployment optimization for delay-minimized UAV-aided MEC networks," *IEEE Wireless Commun. Lett.*, vol. 12, no. 10, pp. 1791–1795, 2023. https://doi.org/10.1109/LWC.2023.3294749

[12] S. Pang, X. He, C. H. Hsu, et al., "Joint trajectory and energy consumption optimization based on UAV wireless charging in cloud computing system," *IEEE Trans. Cloud Comput.*, vol. 11, no. 4, pp. 3426–3438, 2023. https://doi.org/10.1109/TCC.2023.3288527

[13] S. B. Chen, L. Shi, X. Ding, et al., "Energy efficient resource allocation and trajectory optimization in UAV-assisted mobile edge computing system," in *Proc. 7th Int. Conf. Big Data Comput. Commun. (BigCom)*, 2021, pp. 7–13. https://doi.org/10.1109/BigCom53800.2021.00004

[14] Z. Y. Zhu, L. P. Qian, J. F. Shen, et al., "Joint optimisation of UAV grouping and energy consumption in MEC-enabled UAV communication networks," *IET Commun.*, vol. 14, no. 16, pp. 2723–2730, 2020. https://doi.org/10.1109/JSEN.2024.3370924

[15] Q. Liu, H. Zhang, X. Zhang, et al., "Improved DDPG based two timescale multi-dimensional resource allocation for multi-

access edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 73, no. 6, pp. 9153–9158, 2024. https://doi.org/10.1109/TVT.2024.3360943

[16] Y. Wang, Z. Gao, J. Zhang, et al., "Trajectory design for UAV-based Internet of Things data collection: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3899–3912, 2021. https://doi.org/10.1109/JIOT.2021.3102185

[17] M. H. Adnan, Z. A. Zukarnain, and S. K. Subramaniam, "Path planning for multi-UAV-assisted mobile edge computing framework using reinforcement learning in urban environment disaster scenarios," *Ad Hoc Netw.*, vol. 178, Art. no. 103928, 2025. https://doi.org/

[18] N. Lin, H. Tang, L. Zhao, et al., "A PDDQNLP algorithm for energy efficient computation offloading in UAV-assisted MEC," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 8876–8890, 2023. https://doi.org/ 10.1109/TWC.2023.3266497

[19] X. Liu, Z. Y. Chai, Y. L. Li, et al., "Multi-objective deep reinforcement learning for computation offloading in UAV assisted multi-access edge computing," *Inf. Sci.*, vol. 642, p. 119154, 2023. https://doi.org/10.1016/j.ins.2023.119154

[20] C. Wang, K. Liu, Y. Yuan, S. Peng, and G. Li, "Joint trajectory and offloading optimization in UAV-assisted MEC via federated multi-agent reinforcement learning and potential fields," *Comput. Netw.*, vol. 272, Art. no. 111681, 2025. https://doi.org/10.1016/j.comnet.2025.111681

[21] Y. Huo, Y. Liu, A. Jiang, and Y. Yang, "Research on UAV–MEC cooperative scheduling algorithms based on multi-agent deep reinforcement learning," *Comput. Mater. Continua*, vol. 86, no. 3, 2026. https://doi.org/10.32604/cmc.2025.072681

[22] L. Zhang, Z. Y. Zhang, L. Min, et al., "Task offloading and trajectory control for UAV-assisted mobile edge computing using deep reinforcement learning," *IEEE Access*, vol. 9, pp. 53708–53719, 2021. https://doi.org/10.1109/ACCESS.2021.3070908

[23] Q. Liu, L. Shi, L. Sun, et al., "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5723–5728, 2020. https://doi.org/10.1109/TVT.2020.2982508

[24] Z. Q. Wang, H. G. Rong, H. B. Jiang, et al., "A load-balanced and energy-efficient navigation scheme for UAV mounted mobile edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3659–3674, 2022. https://doi.org/10.1109/TNSE.2022.3188670

[25] L. Wang, K. Wang, C. Pan, et al., "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, 2021. https://doi.org/10.1109/TCCN.2020.3027695

[26] Y. W. Nie, J. H. Zhao, F. F. Gao, et al., "Semi-distributed resource management in UAV-aided MEC systems: A multi-agent federated reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13162–13173, 2021. https://doi.org/10.1109/TVT.2021.3118446

[27] T. Ju, L. Li, S. Liu, and Y. Zhang, "A multi-UAV assisted task offloading and path optimization for mobile edge computing via multi-agent deep reinforcement learning," *J. Netw. Comput. Appl.*, vol. 229, Art. no. 103919, 2024. https://doi.org/10.1016/j.jnca.2024.103919

[28] X. Li, X. Du, N. Zhao, et al., "Computing over the sky: Joint UAV trajectory and task offloading scheme based on optimization embedding multi-agent deep reinforcement learning," *IEEE Trans. Commun.*, vol. 72, no. 3, pp. 1355–1369, 2024. https://doi.org/

[29] B. T. Hu, W. Z. Zhang, Y. Gao, et al., "Multiagent deep deterministic policy gradient-based computation offloading and resource allocation for ISAC-aided 6G V2X networks," *IEEE Internet Things J.*, vol. 11, no. 20, pp. 33890–33902, 2024. https://doi.org/10.1109/TCOMM.2023.3331029

[30] P. Qin, Y. Fu, Y. Xie, et al., "Multi-agent learning-based optimal task offloading and UAV trajectory planning for AGIN-power IoT," *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 4005–4017, 2023. https://doi.org/10.1109/TCOMM.2023.3274165

[31] A. M. Seid, G. O. Boateng, B. Mareri, et al., "Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4531–4547, 2021. https://doi.org/10.1109/TNSM.2021.3096673