Received 25 February 2024, accepted 10 March 2024, date of publication 19 March 2024, date of current version 26 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3378512

RESEARCH ARTICLE

Multi-Agent Collaborative Optimization of UAV Trajectory and Latency-Aware DAG Task Offloading in UAV-Assisted MEC

CHUNXIANG ZHENG[1], KAI PAN[2], JIADONG DONG[2], LIN CHEN[2], QINGHU GUO[2], SHUNFENG WU[2], HANYUN LUO[1], AND XIAOLIN ZHANG[1]

[1]School of Computer and Information, Anqing Normal University, Anqing 246133, China
[2]School of Electronic Engineering and Intelligent Manufacturing, Anqing Normal University, Anqing 246133, China

Corresponding author: Jiadong Dong (djd@aqnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62205005.

ABSTRACT The domain of UAV-assisted Multi-Access Edge Computing (MEC) emerges as a novel frontier, characterized by the seamless integration of edge computing capabilities with UAV to facilitate advanced computational services for Terminal Devices (TDs). This research tackles two critical aspects in UAV-assisted MEC frameworks: the strategic formulation of UAV flight paths and the refinement of execution latency for Directed Acyclic Graph (DAG) tasks. We introduce an innovative solution employing Deep Reinforcement Learning (DRL), coined as the Twin Delayed Deep Deterministic Policy Gradient for UAV Trajectory Planning and Task Offloading (TD3-TT) algorithm. This algorithm harmonizes UAV flight planning, DAG task delegation, and scheduling hierarchies, thereby enabling UAV to efficiently undertake task offloading and processing concurrently along their designated optimal trajectories. Through this approach, the latency within the computational network is significantly diminished. A thorough examination of simulation outcomes reveals that the TD3-TT algorithm exhibits notable convergence and robustness, surpassing conventional benchmarks and markedly reducing the execution latency of DAG tasks.

INDEX TERMS Deep reinforcement learning, directed acyclic graph, unmanned aerial vehicle, multi-access edge computing.

I. INTRODUCTION

In the contemporary landscape, driven by the swift expansion of the Internet of Things (IoT), a plethora of intelligent devices, including sensors, cameras, and Virtual Reality/Augmented Reality (VR/AR) systems, increasingly proliferate, actively participating in data collection and processing [1], [2], [3]. Nonetheless, these devices typically harbor finite computational resources and energy provisions, possibly confronting manifold constraints when dealing with intricate computations. Concurrently, with the rapid surge of large-scale artificial intelligence models, a multitude of computationally intensive task, encompassing data prediction, object recognition, and speech discernment, among others, necessitate substantial computational resources for achieving efficient and accurate processing [4], [5]. Against this backdrop, the advent of MEC is manifest [6], [7]. MEC, by deploying computational capabilities proximate to data sources, translocates the computational paradigm from the conventional centralized cloud data centers to device edge, thereby mitigating latency and bolstering computational efficiency.

UAV-assisted MEC represents an innovative computing paradigm that amalgamates the advantages of UAV and MEC [9], [10], [11]. In traditional scenarios, computational servers are stationed at the network periphery, such as base stations (BS) or wireless access points (AP). This architecture is restricted by the static nature of MEC server placement. In contrast, UAV-assisted MEC possesses attributes that transcend these limitations. By deploying MEC servers onto

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

UAV, the inherent attributes of UAV, such as high maneuverability, extensive coverage, and low latency, are fully leveraged to provide wireless communication and computational services to remote or hard-to-reach TDs [13]. This novel UAV-assisted MEC framework is widely applicable across a diverse range of scenarios. It includes, but is not limited to, providing temporary communication and computing support to rescue teams during disaster recovery and emergency response phases, supplying temporary network and computational resources during periods of high traffic at large-scale events, offering mobile communication and computing services to researchers or residents in remote areas, and reducing transmission latency through proximate analysis in applications that necessitate real-time processing of substantial data volumes.

In this study, the flight trajectories and communication conditions of UAV are dynamic and subject to change over time. DRL is commonly regarded as well-suited for addressing policy control issues within complex dynamic environments [16]. When tackling trajectory planning and task offloading challenges in UAV-assisted MEC, nondeterministic policy algorithms like DQN encounter the dimensionality explosion problem due to rapid expansions in state and action spaces. This work proposes a multi-agent collaborative computation offloading algorithm, TD3-TT, based on the TD3 algorithm within the DRL framework, aimed at reducing the complexity of the problem-solving process. By strategically planning UAV flight trajectories, the algorithm ingeniously integrates DAG task offloading decisions with scheduling priorities, effectively minimizing task execution latency. The primary contributions of this paper are as follows:

- Unified Optimization of UAV Trajectory and Task Offloading: Our model uniquely addresses the joint optimization of UAV trajectory planning and task offloading, a comprehensive approach that distinguishes it from existing works focusing solely on one aspect.
- TD3-TT Algorithm: The introduction of the TD3-TT algorithm, based on deep reinforcement learning for multi-agent collaboration, is a innovation. This algorithm concurrently optimizes UAV trajectory and task offloading decisions, minimizing overall system latency.
- Unlike the partial offloading strategies in existing UAV-assisted MEC research, this paper models the computing task as a DAG graph, fully considers the dependencies between child node tasks, and jointly optimizes the offloading decisions and execution priorities of node tasks in the designed algorithm framework.

II. RELATED WORK

Despite endowing UAV systems with substantial potential, MEC introduces a series of challenges in practical application. The foremost challenge revolves around computation task latency [19]. UAV necessitate real-time execution of navigation, obstacle avoidance tasks, and concurrent processing of user-submitted workloads. This necessitates a delicate equilibrium between flight path planning and task offloading strategies within the constraints of limited resources [22]. Furthermore, battery constraints emerge as a significant limiting factor [23]. Since UAV rely exclusively on finite battery capacity for power, their flight duration is inevitably curtailed. To address these challenges, researchers have categorized offloading strategies into binary offloading and partial offloading schemes based on the granularity of task offloading.

A. BINARY OFFLOADING

In the context of UAV-assisted MEC, the binary offloading strategies have garnered substantial attention in recent literature [24]. Addressing concerns pertinent to user energy consumption and data queue stability within UAV-assisted MEC systems, the authors of [25] artfully employs a multi-stage stochastic optimization approach coupled with Lyapunov optimization techniques. the study successfully attains a reduction in energy consumption while concurrently enhancing data queue stability. The authors of [26] formulates the problem as a Markov Decision Process (MDP) and employs a Multi-Agent Deep Reinforcement Learning (MADRL) framework to develop models capturing the dynamics of TDs, UAV costs, and revenues. Through joint optimization of power control and resource allocation, the proposed MADRL approach effectively improves the overall performance of the UAV-assisted wireless network. Moreover, the authors of [27] addresses the joint optimization problem of computation offloading, spectrum resource allocation, computing resource allocation, and UAV deployment by formulating it as a mixed-integer nonlinear programming problem. They propose a decomposition-based approximate algorithm that divides the original problem into two subproblems solved sequentially. Simulation results demonstrate that the proposed algorithm achieves superior performance over baseline schemes. The authors of [28], the focus shifts towards enhancing energy efficiency and task fairness in UAV-assisted MEC. Through a synergistic integration of UAV flight time, trajectory, TD offloading decisions, and time allocation, a novel PDDQNLP algorithm is introduced. This algorithm adeptly combines Continuous Action Space Deep Deterministic Policy Gradient (DDPG) and Dueling Deep Q-Network (DQN) algorithms, effectively addressing the aforementioned challenges in the UAV-assisted MEC landscape. In the context of multi-UAV-assisted MEC communication scenarios, the authors of [29] delves into the intricacies of resource allocation, aiming to minimize overall system latency and energy consumption. Drawing upon the principles of DRL, a novel Unified Mobile Aircraft Power (UMAP) algorithm is introduced. This algorithm seamlessly integrates aspects of UAV mobility control, mobile user association, and mobile user power control. Demonstrating impressive results, UMAP significantly reduces system latency and energy consumption while concurrently enhancing coverage.

B. PARTIAL OFFLOADING

Regarding task partial offloading research [30], the authors of [31], a study explores the concept of divisible task offloading models and orthogonal/non-orthogonal multiple access (OMA/NOMA) access modes. By employing a block alternating descent approach, an alternating iterative algorithm is devised. This algorithm jointly optimizes the allocation of computational resources and the trajectory of UAV, effectively achieving the minimization of weighted energy consumption for both UAV and TDs. The authors of [32] introduces an optimal partial offloading scheme named POSMU. This scheme optimizes the offloading ratio, local computing frequency, transmission power, and MEC server computing frequency for each TDs. By efficiently utilizing MEC server resources, the scheme aims to maximize the number of offloaded tasks. The authors of [33], the study focuses on resource-constrained scenarios and explores the utilization of an improved metaheuristic optimization algorithm called PGL (Particle Swarm Optimization based on Genetic Learning). This approach combines the local search capability of particle swarm optimization with the genetic operations of genetic algorithms to optimize various aspects, including task offloading, user association, CPU speed of computing devices, transmission power, and resource allocation of base stations. The authors of [34], a computational offloading algorithm based on DDPG is employed to obtain optimal offloading strategies in unpredictable dynamic environments. This approach optimizes user scheduling, task offloading ratios, UAV flight angles, and flight speeds to minimize the maximum processing latency. The authors of [35], a joint optimization approach for computational offloading strategies and UAV trajectories is employed. This is achieved by utilizing the Population Diversity-based Binary Particle Swarm Optimization (PDPSO) algorithm and DDPG algorithm to obtain optimal solutions. The aim is to enable real-time communication between TDs and UAV, thereby minimizing overall energy consumption and latency while enhancing the quality of service (QoS) for users.

In the aforementioned literature, the binary offloading mode entails completely offloading the entire task to the UAV for processing. This approach leads to higher computational latency, increased drone energy consumption, and reduced flight endurance. The partial offloading mode divides the task into two subtasks using an offloading ratio, which are then allocated to different computational nodes for parallel processing. This mode substantially reduces latency and energy consumption. Nevertheless, in real-world applications, computational tasks typically consist of multiple interdependent subtasks. For instance, tasks like image recognition involve various interconnected subtasks such as data loading, preprocessing, feature extraction, classification, and regression. These subtasks are interconnected, with the output of one subtask serving as the input for another [36]. Additionally, ensuring the interdependency and sequence of execution among these subtasks significantly impacts the overall computational efficiency of the task. It is worth noting that the task offloading problem has been proven to be a complex NP-HARD issue [37]. Hence, devising and implementing viable offloading strategies remains a substantial challenge.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this work, we consider a UAV-assisted MEC system consisting of N TDs and one UAV equipped with a MEC server. The set of TDs is denoted by $i \in \mathcal{N} = \{1, 2, 3, \ldots, N\}$. We employ time division multiple access (TDMA) to partition the continuous time into equal-length time slots, i.e., $t \in \mathcal{T} = \{1, 2, 3, \ldots, T\}$. In each time slot t, by adjusting its flying velocity $\omega(t)$ and orientation $\theta(t)$, the UAV reaches the target location within the flying time T_{fly} to provide communication and computation services to a specific TD i. The other TDs need to wait for the next time slot to get serviced.

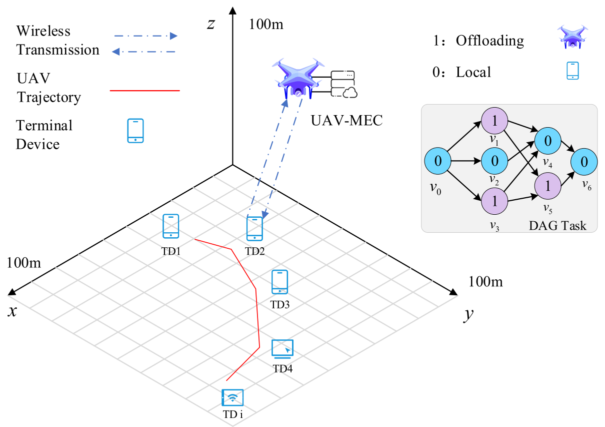


FIGURE 1. System network model.

In a three-dimensional Cartesian coordinate framework, TDs are dispersed randomly within a specified rectangular region, each side extending to a maximum distance of D_{max} meters. The location of a specific TD, referred to as TDi, is marked by L_i=$(x_i, y_i, 0)$. An UAV operates at a constant altitude H above this rectangular zone, with its dynamic position at any time t given by

$$\alpha(t) = \alpha(t-1) + \omega(t) * T_{fly} * \cos(\theta(t)) \tag{1}$$

$$\beta(t) = \beta(t-1) + \omega(t) * T_{fly} * \sin(\theta(t)) \tag{2}$$

where $\omega(t)$ is the velocity, T_{fly} the flight duration, and $\theta(t)$ the flight direction angle at time t. The conceptual framework of the system is depicted in Figure 1.

A. TASK MODEL

To enable fine-grained task offloading, we model the overall application task as a DAG, $G = (V, E)$. V denotes the set of node tasks, where each node task $v_j \in V$,

$j \in \{0, 1, 2, 3, \ldots, M\}$ constitutes the overall application task. The directed edge set $E = \{e = (v_j, v_k), k \in \{1, 2, 3, \ldots, M\}\}$ represents the dependency between adjacent node tasks. We define $in(v_j)$ and $out(v_j)$ as the set of immediate predecessor and successor tasks of node v_j, respectively. As shown in the DAG task in Fig. 1, the in-degree set of node task v_4 is $in(v_4) = \{v_1, v_2, v_3\}$, implying that v_4 can only start execution after all its predecessor tasks v_1, v_2 *and* v_3 finish execution. The data size of v_j on TDi is denoted by $d_j(t)$. In consideration of the DAG structure, characteristics of node tasks, system requirements, and optimization objectives, an appropriate offloading decision $S(t) = (s_1, s_2, s_3, \ldots, s_M)$, is determined for each node task. Here, $s_j \in \{0, 1\}$ denotes whether v_j is offloaded or not, with $s_j = 1$ indicating offloading and $s_j = 0$ indicating otherwise. $FT_j^{local}(t)$ is defined as the completion time when TDi locally executes v_j within time slot t, $FT_j^{up}(t)$ represents the completion time when TDi uploads v_j to the MEC through a wireless link, $FT_j^{calc}(t)$ corresponds to the completion time of the computation of v_j on MEC, and $FT_j^{down}(t)$ signifies the completion time when TDi downloads the output data of v_j from MEC.

B. COMMUNICATION MODEL

Channel State Information (CSI) is a crucial metric for assessing the quality of the communication link between UAV and ground TDs, encompassing key wireless channel attributes such as signal attenuation, multipath effects, and signal interference. In our UAV-assisted MEC system, to simplify problem handling and focus on the core issues, we adopted a Line-Of-Sight (LOS) communication model to estimate the channel gain, thereby calculating the CSI. The channel gain of the LOS link between the UAV and TD i can be expressed as:

$$g(t) = \frac{\mu}{d_{uu}} \tag{3}$$

where μ denotes the channel gain at the reference distance of 1 m, and $d_{uu} = \|L_i - U(t)\|$ represents the Euclidean distance between the UAV and the corresponding TD i in time slot t. As the UAV moves, d_{uu} is updated in real-time, thereby dynamically adjusting the value of CSI. Our assumption of equal uplink and downlink transmission rates is made to simplify the model and facilitate problem handling [38]. According to the Shannon formula, the uplink and downlink transmission data rates between the TD and the MEC server are given by:

$$r(t) = Blog_2\left(1 + \frac{p_i g(t)}{\sigma^2}\right) \tag{4}$$

where B represents the channel bandwidth, p_i denotes the transmission power, σ^2 is the Gaussian white noise power.

C. LOCAL COMPUTING MODEL

When the offloading decision $s_j(t) = 0$ for v_j, it indicates that v_j is computed locally. The prerequisite for v_j to start local execution is that all its predecessor tasks $in(v_j)$ have finished execution. Thus, the starting time for local computation of v_j is given by:

$$ST_j^{local}(t) = max\left\{max\left\{FT_k^{local}(t), FT_k^{down}(t)\right\}\right\}, \quad v_k \in in(v_j) \tag{5}$$

v_j local execution completion time $FT_j^{local}(t)$ is:

$$FT_j^{local}(t) = ST_j^{local}(t) + \frac{d_j(t)C_{td}}{f_i} \tag{6}$$

where C_{td} represents the number of CPU cycles required to process one byte of data at the TD, and f_i denotes the computational capability of TD i.

D. COMPUTING OFFLOADING MODEL

When the offloading decision $s_j(t) = 1$ for v_j, it indicates that v_j will be offloaded to the MEC server for computation. Thus, the starting time for computation of the offloaded v_j is:

$$ST_j^{offload}(t) = max\left\{max\left\{FT_k^{local}(t), FT_k^{calc}(t)\right\}\right\}, \quad v_k \in in(v_j) \tag{7}$$

Similar to the local computation model, the prerequisite for v_j to start execution after offloading is that all its predecessor tasks $in(v_j)$ have finished execution. However, unlike local computation, the maximum finish time of the predecessor tasks is given by $max\left\{FT_k^{local}(t), FT_k^{calc}(t)\right\}$, without considering the result downloading time. The latency of the offloading process consists of the uplink transmission latency, computation latency, and downlink transmission latency. Specifically, the uplink transmission latency of v_j is:

$$T_j^{up}(t) = \frac{d_j(t)}{r(t)} \tag{8}$$

Calculation latency of v_j:

$$T_j^{calc}(t) = \frac{d_j(t)C_{uav}}{F} \tag{9}$$

where C_{uav} represents the number of CPU cycles required to process one byte of data at the UAV, and F denotes the computational capability of MEC server. Assuming the output data size is the same as the input data size for the offloaded task, and the wireless channel is symmetric with equal achievable uplink and downlink transmission rates, the result downloading latency of v_j is:

$$T_j^{down}(t) = \frac{d_j(t)}{r(t)} \tag{10}$$

The completion instance of v_j offloaded execution, denoted as $FT_j^{offload}$, can be formulated as follows:

$$FT_j^{offload}(t) = ST_j^{offload}(t) + T_j^{up}(t) + T_j^{calc}(t) + T_j^{down}(t) \tag{11}$$

E. NODE SCHEDULING PRIORITY

Scheduling priorities of nodes determine the relative execution order of tasks during the task execution, significantly impacting the system's performance and efficiency. The modeling process for node task scheduling priorities is as follows: Firstly, for each node task v_j, belonging to the i-th layer, denoted as $Layer\left(v_j\right) = i$, we obtain the node set L_i for the i-th layer through topological sorting:

$$L_i = \left\{v_j \in V \mid Layer\left(v_j\right) = i\right\} \tag{12}$$

Secondly, to establish the scheduling priority model, we sort according to each layer node's value $P\left(v_j\right)$, defining the sorted result SortedByLayer as:

$$SortedByLayer = Sort\left(\begin{array}{l}\left\{\left(v_j, P\left(v_j\right)\right) \mid v_j \in V\right\}, \\ key = Layer\left(v_j\right)\end{array}\right) \tag{13}$$

where $P\left(v_j\right)$ represents the scheduling priority value output by the neural network for node task v_j, and the key is set to the node's layer index, ensuring sorting based on the node's layer. Finally, we define the ultimate scheduling priority set P_{node}, sorting within each layer based on the neural network's output values:

$$P_{node} = Sort\left(SortedByLayer, key = P\left(v_j\right)\right) \tag{14}$$

where the key is set to the scheduling priority of the node task, ensuring sorting based on the neural network's output values within each layer.

F. PROBLEM FORMULATION

Building upon the aforementioned modeling framework, this study formulates an optimization problem that encompasses the joint optimization of TDs scheduling priority P_{td}, UAV flight velocity ω, flight angle θ, DAG task offloading decision S, and node scheduling priority P_{node}. The primary objective is to minimize the maximum processing latency of each TD within every time slot. This optimization problem is defined as follows:

$$\begin{aligned}
\min_{P_{td}, \omega, \theta, S, P_{node}} \; & \sum_{t=1}^{T}\left(T_{fly} + \max\left(FT_M^{local}(t), FT_M^{offload}(t)\right)\right) \\
s.t.\; & C1: 0 < \theta(t) \leq 2\pi, \quad \forall t \in \mathcal{T} \\
& C2: s_j(t) = \{0, 1\}, \quad \forall t \in \mathcal{T} \\
& C3: 0 \leq \alpha(t) \leq D_{max}, \quad \forall t \in \mathcal{T} \\
& C4: 0 \leq \beta(t) \leq D_{max}, \quad \forall t \in \mathcal{T} \\
& C5: 0 \leq x_i \leq D_{max}, \quad \forall i \in \mathcal{N} \\
& C6: 0 \leq y_i \leq D_{max}, \quad \forall i \in \mathcal{N}
\end{aligned} \tag{15}$$

In the above formulation, the maximum processing latency of a TD in each time slot consists of the fixed flying latency of the UAV and the finish time of the last node task v_M in the DAG. Constraint $C1$ represents the horizontal orientation angle limits of the UAV. Constraint $C2$ enforces the offloading decision variables. Constraints $C3$ and $C4$ restrict the UAV to fly within the predefined area. Constraints $C5$ and $C6$ confine the randomly generated locations of the TDs.

IV. DRL BASED OPTIMIZATION ALGORITHM

In this section, we formulate the optimization problem as a MDP to minimize the overall system latency. First, we introduce the TD3 algorithm that supports continuous action spaces. Then, we propose a TD3-TT algorithm based on TD3 by defining the states, actions, and reward function in the MDP to realize multi-agent collaborative optimization of UAV trajectory planning and task offloading.

A. TD3 ALGORITHM FRAMEWORK

DRL is an amalgamation of deep learning and reinforcement learning techniques designed to tackle intricate tasks characterized by high-dimensional state and action spaces. Within this realm, the TD3 emerges as a cutting-edge algorithm within the Actor-Critic framework of DRL. It comprises six neural networks (NNs): an Actor policy network denoted as μ_ϕ, two Critic value networks represented by Q_{θ_1} and Q_{θ_2}, along with their corresponding Target Actor target policy network, $\mu_{\phi'}$, and Target Critic target value networks, $Q_{\theta_1'}$ and $Q_{\theta_2'}$. Refer to Fig. 2 for an illustration.

1) EXPERIENCE GENERATION

The agent computes the output action from the policy network μ_ϕ based on the current state $s(t)$ at each timeslot t.

$$a(t) = \mu_\phi(s(t)) + \varepsilon, \varepsilon \sim N(0, \sigma) \tag{16}$$

where ϕ represents the parameters of the actor network, and ε is Gaussian noise. The addition of noise can help avoid getting stuck in local optima and improve exploration efficiency. The agent interacts with the environment using the noisy action $a(t)$ and obtains the reward $r(t)$ and next state $s'(t)$, constituting a state transition sample $(s(t), a(t), r(t), s'(t))$ stored in the experience replay buffer as the training dataset for the online networks.

2) TRAINING AND UPDATING THE NETWORK

When the number of state transition samples in the experience replay buffer exceeds the predetermined capacity, a batch of state transition samples (bs, bs, br, bs') is randomly drawn from it for training the online network. Differing from the DDPG algorithm, the TD3 algorithm introduces a regularization strategy by incorporating truncated Gaussian noise onto the output actions $\mu_{\theta_2}(bs')$ of the Target Actor network. This is formulated as follows:

$$bs' = \mu_{\phi_2}\left(bs'\right) + \varepsilon', \varepsilon' \sim clip\left(N\left(0,\sigma^2\right), -c, c\right) \tag{17}$$

The parameter ϕ_2 corresponds to the Target Actor network. The introduction of truncated Gaussian noise serves to ren der the output Q-values of the two Target Critic networks smoother, thus mitigating the occurrence of overfitting. Additionally, recognizing the need to alleviate excessive bias in Q-function estimates, the TD3 algorithm incorporates dual

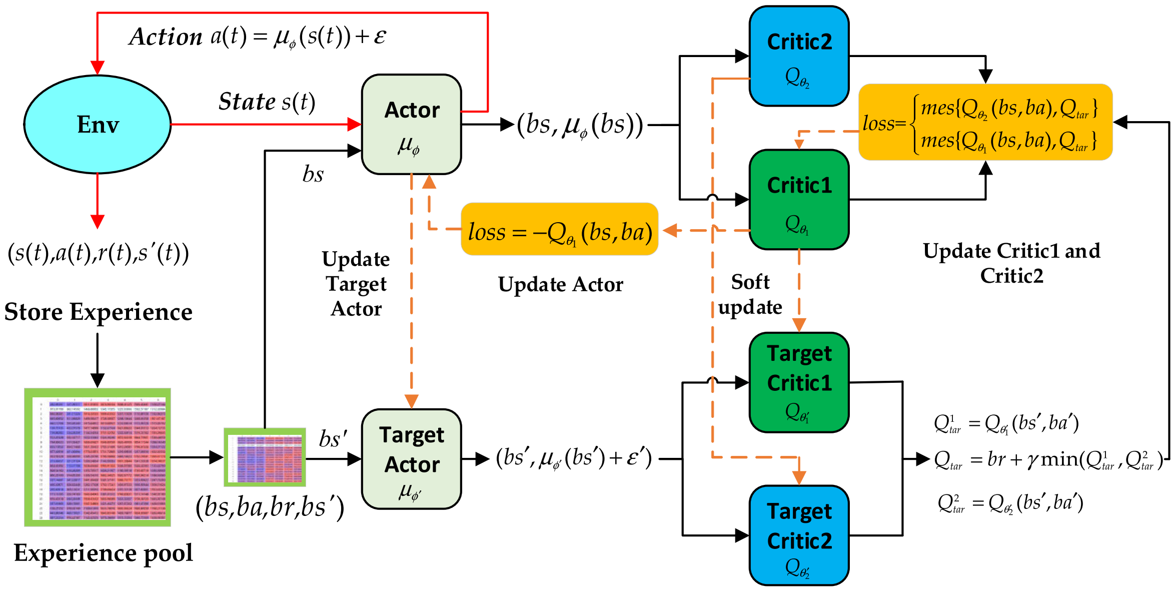

Action $a(t)=\mu_\phi(s(t))+\varepsilon$
Env
State $s(t)$
Actor μ_ϕ
$(bs,\mu_\phi(bs))$
Critic2 Q_{θ_2}
Critic1 Q_{θ_1}
$loss=\begin{cases} mes\{Q_{\theta_2}(bs,ba),Q_{tar}\} \\ mes\{Q_{\theta_1}(bs,ba),Q_{tar}\}\end{cases}$
bs
$(s(t),a(t),r(t),s'(t))$
Store Experience
Update Target Actor
$loss=-Q_{\theta_1}(bs,ba)$
Update Actor
Soft update
Update Critic1 and Critic2
Target Critic1 $Q_{\theta_1'}$
bs'
Target Actor $\mu_{\phi'}$
$(bs',\mu_{\phi'}(bs')+\varepsilon')$
$Q_{tar}^1=Q_{\theta_1'}(bs',ba')$
$Q_{tar}=br+\gamma\min(Q_{tar}^1,Q_{tar}^2)$
$Q_{tar}^2=Q_{\theta_2'}(bs',ba')$
Target Critic2 $Q_{\theta_2'}$
(bs,ba,br,bs')
Experience pool

FIGURE 2. TD3 Network Architecture and Algorithm Flow.

Q-functions, denoted as $Q_{\theta_1'}(\cdot)$ and $Q_{\theta_1'}(\cdot)$. During the gradient descent process, the smaller Q-value from the outputs is selected, and the TD target is computed based on the Bellman expectation equation of the state-action value function:

$$Q_{tar}=br+\gamma min(Q_{\theta_1'}(bs',ba'),Q_{\theta_2'}(bs',ba')) \tag{18}$$

γ represents the discount factor, while θ_1' and $\theta_2{}'$ stand as the parameters of the two Target Critic networks. With the intention to enhance the estimated Q-values towards the target Q-values, the calculation of the TD error is undertaken. This process establishes two Mean Squared Error (MSE) loss functions for the purpose of gradient descent, facilitating the update of the Critic networks' parameters. The update equations are as follows:

$$L(\theta_i)=\frac{1}{b}\sum\left(Q_{\theta_i}(bs,ba)-Q_{tar}\right)^2;\quad i=1,2 \tag{19}$$

$$\theta_i\leftarrow\theta_i-l_{Cri}\nabla_{\theta_i}L(\theta_i);\quad i=1,2 \tag{20}$$

$Q_{\theta_{1,2}}(bs,ba)$ represent the estimated Q-values output by the two Critic value networks, with θ_1 *and* θ_2 denoting their respective parameters, and l_{Cri} signifying the learning rate. In the context of RL, the agent's objective is to maximize cumulative rewards, where Q-values can gauge the long-term cumulative rewards of taking specific actions in the current state. Throughout the training process, the aspiration is for the Actor network to yield actions $\mu_\phi(\cdot)$ that maximize Q-values, thereby maximizing cumulative rewards. Consequently, the loss function and parameter update equation for the Actor network are formulated as follows:

$$J(\phi)=-\frac{1}{b}\sum Q_{\theta_1}(bs,\mu_\phi(bs)) \tag{21}$$

$$\phi\leftarrow\phi-l_{Act}\nabla_\phi J(\phi) \tag{22}$$

Finally, the parameters ϕ of the Actor network and the parameters θ_1 *and* θ_2 of the two Critic networks are synchronized with the corresponding target networks using soft updates as follows:

$$\theta_i'\leftarrow\tau\theta_i+(1-\tau)\theta_i',\ i=1,2 \tag{23}$$

$$\phi'\leftarrow\phi+(1-\tau)\phi' \tag{24}$$

where τ is the soft update coefficient, ϕ' represents the parameters of the Target Actor policy network, and θ_1', θ_2' denote the parameters of the two Target Critic networks.

B. TD3-TT ALGORITHM DESIGN AND IMPLEMENTATION

In this paper, we innovatively propose the TD3-TT algorithm based on TD3 that achieves efficient task offloading through the collaboration of three intelligent agents. First, Agent

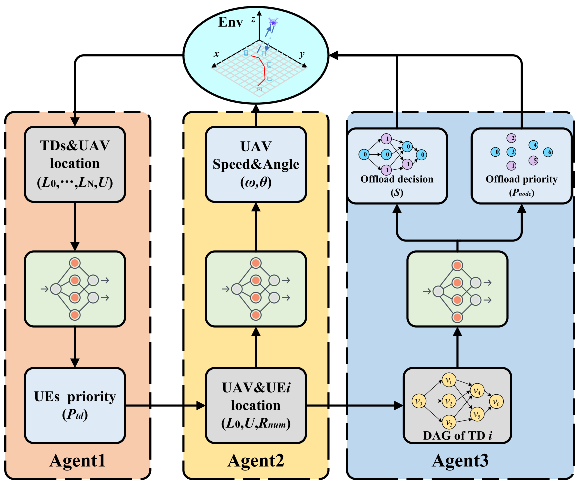

Env
TDs&UAV location $(L_0,\cdots,L_N,U)$
UAV Speed&Angle (ω,θ)
Offload decision (S)
Offload priority (P_{node})
UEs priority (P_{td})
UAV&UEi location (L_0,U,R_{num})
DAG of TD i
Agent1
Agent2
Agent3

FIGURE 3. Cooperative Optimization for Multi-Agent.

1 accurately determines the scheduling priority of TDs based on their location distribution and task requirements. Second, following the scheduling priority of TDs, Agent 2 controls the velocity and orientation of the UAV to plan its flying trajectory. Finally, Agent 3 formulates the offloading decisions and scheduling priorities for the DAG tasks, as depicted in Fig. 3.

The entire task offloading process can be modeled as a MDP. Below we define the state space, action space, and reward function for each of the three agents.

1) STATE SPACE

The location distribution of TDs is crucial for determining the offloading sequence. By observing the global information, Agent 1 comprehends the location details between the UAV and TDs, including coordinates, relative positions, and distances, to identify the TDs at critical locations that require prioritized processing. Thus, the state input for Agent 1 is defined as:

$$ob^1(t) = \{L_1, \ldots, L_N, U(t)\} \tag{25}$$

In contrast, Agent 2 focuses on local information. The state input for Agent 2 consists of the location L_i of TD i, the location $U(t)$ of the UAV in the current time slot t, as well as the number R_{num} of remaining TDs to be processed:

$$ob^2(t) = \{L_i, U(t), R_{num}\} \tag{26}$$

In order to facilitate a comprehensive understanding by Agent 3 of the interdependencies and structure existing between the v_j and v_k within the DAG, the state input of Agent 3 is composed of a binary $M \times M$ adjacency matrix A, along with the node count N_{layer} per layer within the DAG and the node task size $d_j(t)$:

$$ob^3(t) = \left\{ A = \begin{bmatrix} A_{00} & \cdots & A_{0M} \\ \vdots & \ddots & \vdots \\ A_{M0} & \cdots & A_{MM} \end{bmatrix}, N_{layer}, d_{ij}(t) \right\}$$

$$A_{jk} = \begin{cases} 1, & (v_j, v_k) \in E \\ 0, & (v_j, v_k) \notin E \end{cases} \tag{27}$$

where A_{jk} denotes the connectivity between the current $v_j and v_k$, when a connection exists, the matrix element is set to 1, otherwise it remains 0.

2) ACTION SPACE

The action space of Agent 1 is the scheduling priority P_{td} for all TDs. A reasonable scheduling priority order can enable the UAV to more intelligently plan its flight path and avoid unnecessary flight distance and repetitive routes:

$$ac^1(t) = \{P_{td}\} \tag{28}$$

The action space of the Agent 2 consists of the UAV's flight speed $\omega(t)$ and Angle $\theta(t)$:

$$ac^2(t) = \{\omega(t), \theta(t)\} \tag{29}$$

The action space of Agent 3 is the offloading decision $S(t)$ and scheduling priority $P_{node}(t)$ of node tasks:

$$ac^3(t) = \{S(t), P_{node}(t)\} \tag{30}$$

By incorporating the TANH activation function into the output layer of Agent 3's policy network, the output values are constrained within the range of -1 and 1, with a dimensionality of $2M$. The first M dimensions are binarized to 0 and 1, serving as the offloading decisions for each node task. The latter M dimensions are employed to establish the offloading priority sequence of node tasks based on the DAG layer node count N_{layer} and their dependency relationships, as depicted in Fig. 4. This design ensures that the priority of any v_i is always lower than any of its predecessor tasks $in(v_i)$. It comprehensively considers the relationship between $S(t)$ and $P_{node}(t)$ to enable efficient parallel computation, thereby minimizing execution latency.

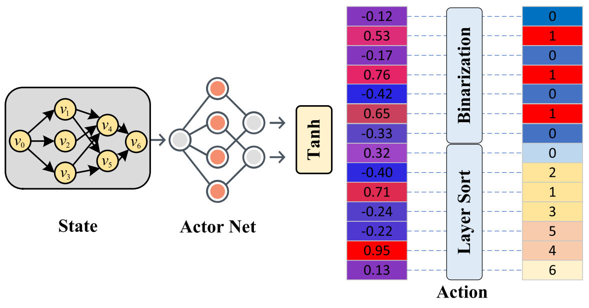


FIGURE 4. Action mapping.

3) REWARD

The optimization objective studied in this paper is the system latency throughout the offloading process. In RL, the reward function should be negatively correlated with the objective function to ensure minimizing the optimization goal while maximizing the cumulative discounted reward. The UAV's flight distance affects its energy consumption, while the LOS distance $d_{uu}(t)$ between the UAV and TD impacts the data transmission rate, thereby influencing the execution latency. Under the premise of a relatively short flight distance, by maximizing the reduction of the LOS distance between the UAV and TD and considering both factors synthetically, we define the reward functions for Agent 1 and Agent 2 as:

$$re^{1,2}(t) = -\left(\varphi_1\left(\varepsilon(t)T_{fly}\right) + \varphi_2\left(d_{uu}(t) - H\right)\right) \tag{31}$$

where $\omega(t)T_{fly}$ denotes the flight distance in time slot t, and $d_{uu}(t) - H$ represents the difference between the distance from the UAV to TD and the flight altitude of the UAV. φ_1 and φ_2 are respective weight coefficients. By precisely tuning the weight coefficients, the goal is to enable the UAV to select the optimal flight trajectory to maximize reduction of the LOS distance between the UAV and TD, thereby effectively minimizing the flight distance and flight energy consumption. During offloading, the finish time of the tail node task v_M determines the execution latency of the DAG task. Thus, according to Equations (4) and (9), the reward function for Agent 3 is set as:

$$re^3(t) = -max\left(FT_M^{local}(t), FT_M^{offload}(t)\right) \tag{32}$$

The detailed process of TD3-TT is shown in **Algorithm 1.**

Algorithm 1 TD3-TT

```
Input: $L_1, \ldots, L_N, U(t), L_i, R_{num}, A, N_{layer}, d_j(t)$
Output: $P_{td}, \omega(t), \theta(t), S(t), P_{node}(t)$
1: Initialize the network parameters for agents 1,2,3.
2: Set the capacity of experience buffer, and specify the batch
   size for training.
3: for $episode = 1, 2, \ldots,$ Max_Episode do
4:     Resetting the global environment to obtain the initial
       $ob^1(t) : L_1, \ldots, L_N, U(t)$.
5:     Input $ob^1(t)$ to the actor network $\mu_\phi^1$ of agent 1 to
       obtain the $ac^1(t) : P_{td}$.
6:     for $t = 1, 2, \ldots, T$ do
7:         Obtain states $ob^2(t), ob^3(t)$ for agents2,3 based on $P_{td}$
8:         Use $\mu_\phi^2$ for Agent 2 with input $\omega(t), \theta(t)$ to get $ac^2(t)$
9:         Use $\mu_\phi^3$ for Agent 3 with input $S(t), P_{node}(t)$ to get $ac^3(t)$
10:        Calculate reward $re^{1,2}(t), re^3(t)$ using formulas (29,30)
11:        Store the $(ob^2(t), ac^2(t), re^{1,2}(t), ob^2(t+1))$ in the
           replay buffer of agent 2.
12:        Store the $(ob^3(t), ac^3(t), re^3(t), ob^3(t+1))$ in
           the replay buffer of agent 3.
13:        if batch size < the current capacity of buffer Then
14:            for agent $i = 2, 3$ do
15:                Sample a batch of experiences randomly.
16:                Calculate critic loss $Q_{\theta_{1,2}}^i$ using equation (19)
17:                Update critic parameters $\theta_1, \theta_2$ using equation (20)
18:                Calculate actor loss of $\mu_\phi^i$ using equation (21).
19:                Update actor parameter $\phi$ using the equation (22).
20:                Update target parameters according to (23,24)
21:      Store the $(ob^1(t), ac^1(t), re^{1,2}(t), ob^1(t+1))$ in the replay
         buffer of agent 1.
22:        if batch size < the current capacity of buffer Then
23:                Train and update the actor parameters for agent 1.
```

V. SIMULATION EXPERIMENT

In this section, we commence by providing an elaborate exposition of the simulated environment for experimentation and the hyperparameter configuration of the TD3-TT algorithm. Subsequently, we conduct an in-depth exploration of the UAV flight trajectory optimization strategy based on the TD3-TT algorithm. Building upon this foundation, we devise various benchmark schemes and comprehensively compare them with the TD3-TT unloading strategy.

A. EXPERIMENTAL SETTINGS

We utilized the Python 3.8 development environment and the PyTorch deep learning framework as the simulation software platform for our experiments, with the hardware platform consisting of an Intel Xeon Gold 6148 processor and a 3090 graphics card. Experiment parameters and the hyperparameter settings for the TD3-TT algorithm were established based on experimental requirements, as illustrated in **Table 1** and **Table 2**.

TABLE 1. Experiment parameter settings.

Definition	Value
Number of TD (N)	5
Number of node task (M)	10
UAV flight area (D_{max})	100m
Fixed UAV flight altitude (H)	100m
Fixed flight time per slot (T_{fly})	1s
CPU frequency of UAV-MEC (F)	4GHz
CPU frequency of TD (f_i)	2Ghz
UAV CPU Cycles for Task (C_{uav})	1500 cycles/bit
TD CPU Cycles for Task (C_{td})	1000 cycles/bit
Bandwidth of channel (B)	8 Mhz
Node task data size (d_i)	Unif(31457, 419430) Mbits
The transmission power of TD (p_i)	0.1w
The noise power (σ^2)	−100 dBm
Total time slots (T)	5
Weight coefficient 1 (φ_1)	0.9
Weight coefficient 2 (φ_2)	0.1

TABLE 2. Hyperparameter settings for TD3-TT.

Definition	hyperparameter
Max_episode	1000
The learning rate of actor lr_a	0.0001
The learning rate of critic lr_c	0.001
memory size	10000
batch size	256
discount factor γ	0.99
soft update coefficient τ	0.05
action noise ε	$N(0,0.1)$

In the TD3-TT algorithm, three Agents are employed to accomplish UAV trajectory optimization, node task offloading decisions, and scheduling priority optimization. The training process of each Agent consists of two Actor networks and four Critic networks. The structural parameters of these Actor and Critic networks are depicted in **Table 3.**

B. TRAJECTORY OPTIMIZATION FOR UAV FLIGHT

The UAV departs from its initial position at (10, 20, 100) and gradually approaches the vicinity of the TDs to provide computational services. Fig. 5 shows the process of training Agents 1 and 2 under three different learning rates to optimize the UAV flight trajectory. Typically, the learning rate for Critic networks is set higher than that for Actor networks, ensuring a more accurate estimation of the value of state-action pairs and providing reliable feedback

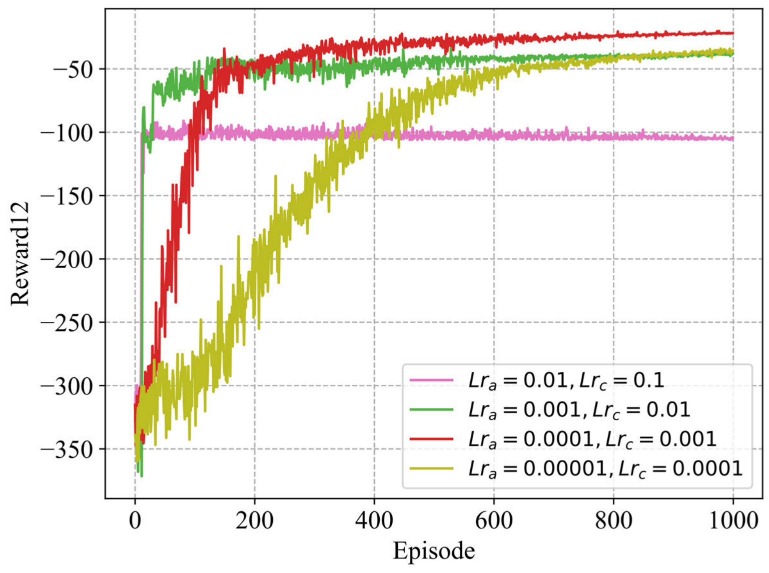


FIGURE 5. Convergence Curves of Agents 1 and 2 training under different learning rates.

TABLE 3. Network Structure for DA-TD3.

Network	Num	Network Structure
Actor of Agent 1	2	fc1(state_dim,256),relu fc2(256,128),relu fc3(128,action_dim), tanh
Critic of Agent 1	4	fc1(state_action_dim,256),relu fc2(256,128),relu fc3(128,1)
Actor of Agent 2	2	fc1(state_dim,256),relu fc2(256,128),relu fc3(128,action_dim), tanh
Critic of Agent 2	4	fc1(state_action_dim,256),relu fc2(256,128),relu fc3(128,1)
Actor of Agent 3	2	fc1(state_dim,256),relu fc2(256,128),relu fc3(128,action_dim),tanh
Critic of Agent 3	4	fc1(state_action_dim,256),relu fc2(256,128),relu fc3(128,1)

to guide policy updates for Actor networks. We designed three sets of learning rates with varying magnitudes, consistently maintaining the Critic network's learning rate one order of magnitude higher than that of the Actor network. The findings illustrate that with learning rates set to $lr_a = 0.01$ and $lr_c = 0.1$, the algorithm converges rapidly to a local optimum within around 30 episodes, attributed to the excessive learning rate. On the other hand, the use of learning rates $lr_a = 0.00001$ and $lr_c = 0.0001$ proves to be too conservative. The model updates at a slow pace during the learning process, hindering its ability to swiftly adapt to dynamic environmental changes, consequently affecting overall performance enhancement.

When the learning rate is set as $lr_a = 0.0001$ *and* $lr_c = 0.001$, the convergence curve achieves the optimal result with the highest reward value. As can be seen from Fig. 6a, during the initial 100 episodes of training, Agents 1 and 2 mainly conduct random exploration to search for the optimal actions, thus the optimization effect is limited. As the training episodes increase to around 400 (Fig. 6b), the direct distance between the UAV and TDs reduces compared to the initial stages, gradually bringing the UAV closer to the center of the dense TD region. Around 800 training rounds (Fig. 6c), as evident from the planar view in Fig. 6d, the P_{td} for UAV to provide computation services to TDs are [2,3,0,1,4]. Due to the influence of the distance optimization term in the reward functions $re^{1,2}(t)$, the UAV flies to the vicinity above each TD in an optimal trajectory every time slot, rather than directly above the TDs, thereby reducing the flight distance. Additionally, to ensure thorough exploration of the action space by the Agents, we introduce progressively diminishing normal distribution noise into the action outputs of the Actor networks during the training process. This introduces significant fluctuations in the initial stages of training, but as training experience accumulates, the agents increasingly rely on existing experiences to make decisions, leading to a smoother curve.

In the UAV trajectory optimization experiment, we examined the impact of different batch sizes and memory sizes on the model's convergence performance. As depicted in Figure 7, we observed poorer convergence when the batch size and memory size were set to smaller values (64 and 100). This could be attributed to an insufficient number of samples, making it challenging for the network to capture the complex features of the environment. Conversely, setting excessively large batch sizes and memory sizes (512 and 100000) resulted in better final convergence but came at a significantly increased training cost with limited

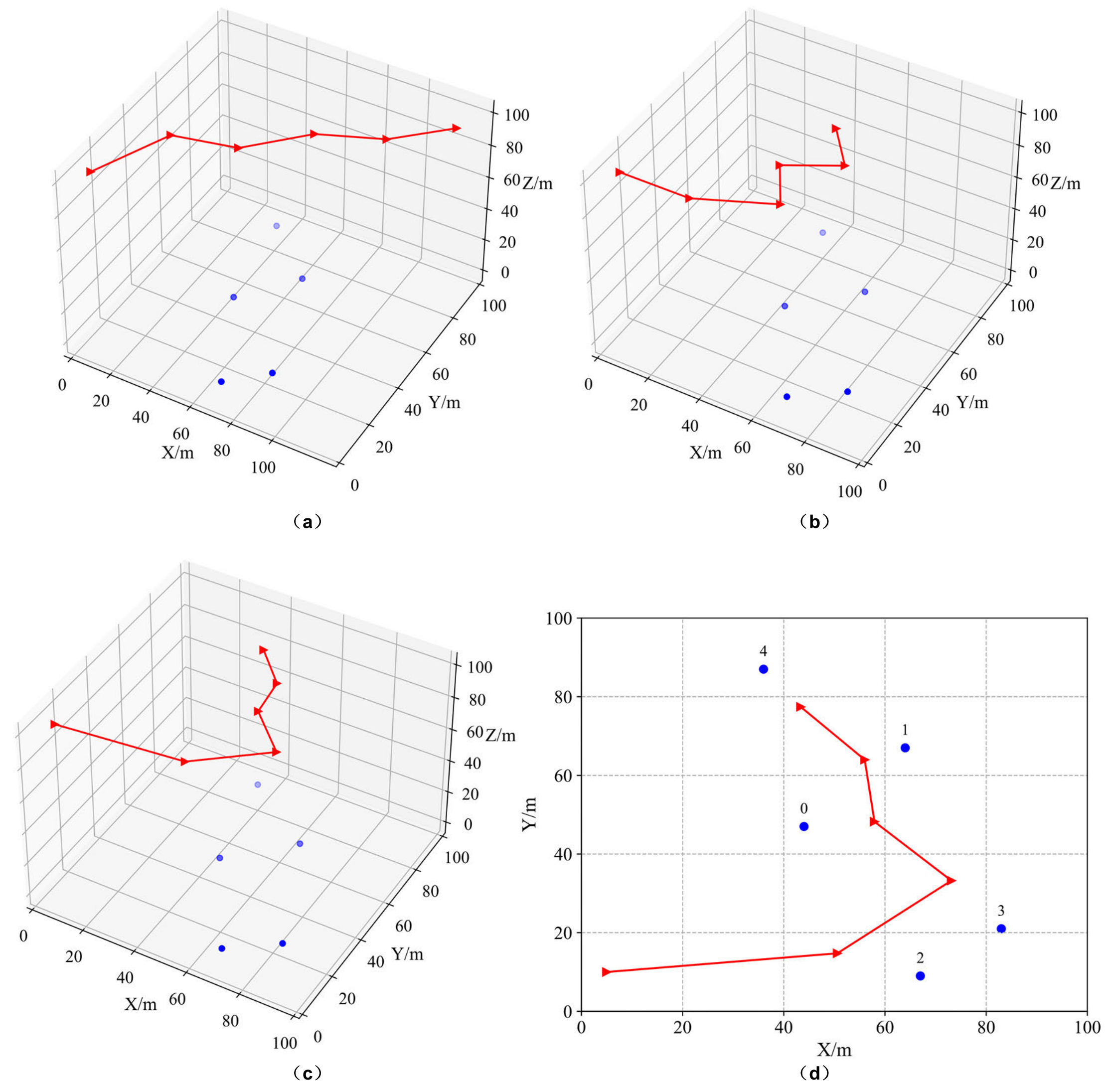


FIGURE 6. UAV Trajectory Optimization Process.

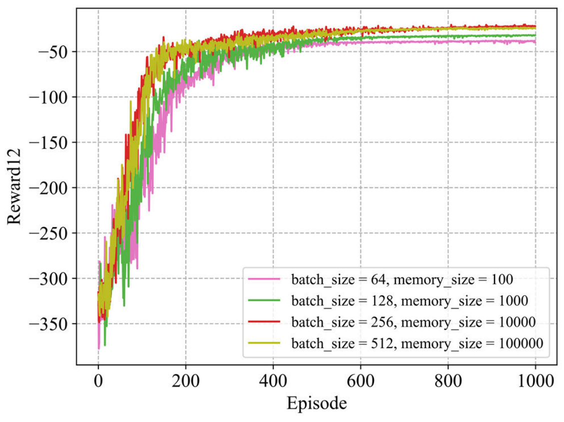


FIGURE 7. Convergence Curves of Agent 1,2 with Varying Memory and Batch Sizes.

performance improvement. Considering factors such as convergence speed, stability, and computational costs, a balanced choice was made with a batch size of 256 and a memory size of 10000.

C. COMPARISON OF DAG TASK OFFLOADING SCHEMES

Fig. 8 shows the training process of Agent 3 under different learning rate settings. It can be observed from the results that the curve converges optimally when the learning rates are $lr_a = 0.0001$ and $lr_c = 0.001$. Unlike the continuous variables of ω and θ in UAV trajectory optimization, the state input of Agent 3 consists of discrete variables including S and P_{node}. This leads to a discrete optimal value for the corresponding reward function. Therefore, there is considerable noise and instability in the convergence curve during early training. However, after approximately 800 episodes of training, the curve eventually becomes smooth and stable.

In the study of DAG task offloading optimization, we also analyzed the impact of different batch sizes and memory sizes on the model's convergence performance. As depicted in Figure 9, the hyperparameter combination of batch size 256

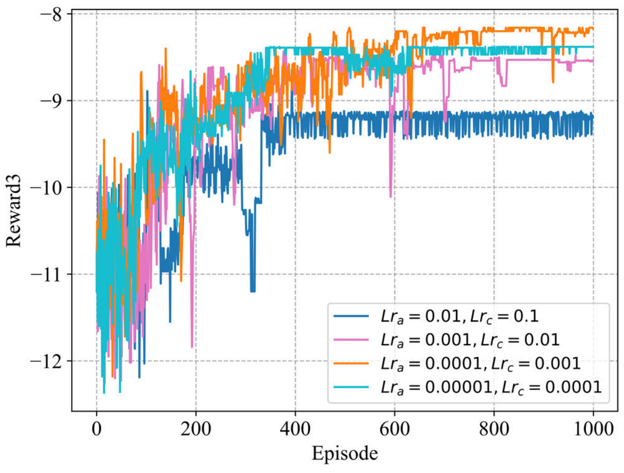


FIGURE 8. Convergence Curves of Agents 3 training under different learning rates.

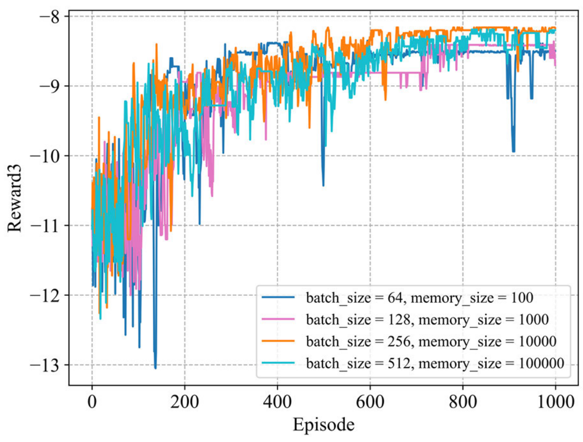


FIGURE 9. Convergence Curves of Agent 3 with Varying Memory and Batch Sizes.

and memory size 10,000 was chosen, and it exhibited the best convergence performance.

To further investigate the significant advantages of TD3-TT in reducing task execution latency, based on the optimal trajectory strategies for the UAV generated by Agent 1 and 2, we designed the following four baseline schemes for comprehensive comparison and analysis of the execution latency performance:

- **Random Offloading Recursion Priority (RORP):** In this scheme, the S for v_j is randomly generated, while the head and tail node tasks must be executed locally to ensure data transmission integrity. The P_{node} is recursively generated based on the offloading latency cost using the following equation:

$$P\left(v_j\right) = \max\left(P\left(v_k\right)\right) + T_j^{up} + T_j^{calc} + T_j^{down}, \quad v_k \in out\left(v_j\right) \tag{33}$$

The termination condition of recursion is:

$$P\left(v_j\right) = T_j^{up} + T_j^{calc} + T_j^{down}, j = M \tag{34}$$

$P\left(v_j\right)$ can be regarded as the critical path cost from node task v_j to the exit task. By sorting $P\left(v_j\right)$, the P_{node} can be obtained. This priority definition first ensures that the priority of any node task v_j is always higher than any of its successor tasks v_k. It also reinforces that among peer tasks, the priority of the task with higher execution cost is strengthened.

- **ALL Offloading Recursion Priority (AORP):** In this scheme, all node tasks except the head and tail tasks are fully offloaded to the MEC server for execution. The P_{node} is the same as the RORP scheme.
- **ALL Local Recursion Priority (ALRP):**In this scheme, all node tasks are executed locally, and the P_{node} is the same as the RORP scheme.
- **DDPG-TT:** In this scheme, the TD3 network architecture in TD3-TT is replaced with a DDPG network, where the actor network generates the S and P_{node}.

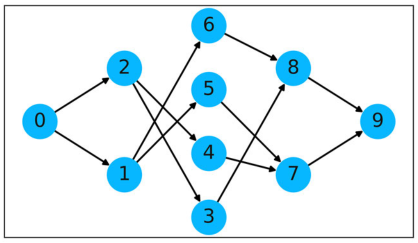


FIGURE 10. DAG task.

Fig. 10 illustrates a DAG task comprising 10 nodes and validates the effectiveness of the TD3-TT algorithm using the proposed offloading scheme. In Fig. 11a, the RORP is applied, resulting in S = [0,1,0,1,1,1,0,0,0,0] and P_{node} = [0,1,2,6,5,3,4,8,7,9]. Below is an analysis of the entire offloading process: When the v_0 completes local computation in 0.23s, its outdegree nodes v_1 and v_2 transition from blocked to ready states. Following S, they simultaneously compute locally and on the MEC server. As $out\left(v_1\right) = \{v_5, v_6\}$ and $out\left(v_2\right) = \{v_3, v_4\}$ enter the ready state, only one node task can be involved in uploading, computation, and result transmission at a given time. Accordingly, based on P_{node}, v_6 completes its upload task at 0.8s, after which v_5 commences uploading. Similar principles are followed for computation and result transmission. v_7, v_8 and v_9 are scheduled for local execution, with v_8's execution starting at 1.41s based on the maximum completion time of its $in(v_8)$. Finally, v_7 and v_9 are computed locally in sequence. The entire task duration is 2.08s.

In Fig. 11b, the AORP is employed. Here, after the initial local computation of the v_0, v_1 to v_8 are offloaded to the MEC server for computation based on S and P_{node}. The total completion time in this case is 2.22s. In contrast to the RORP, the ALRP entirely disregards the MEC server computational

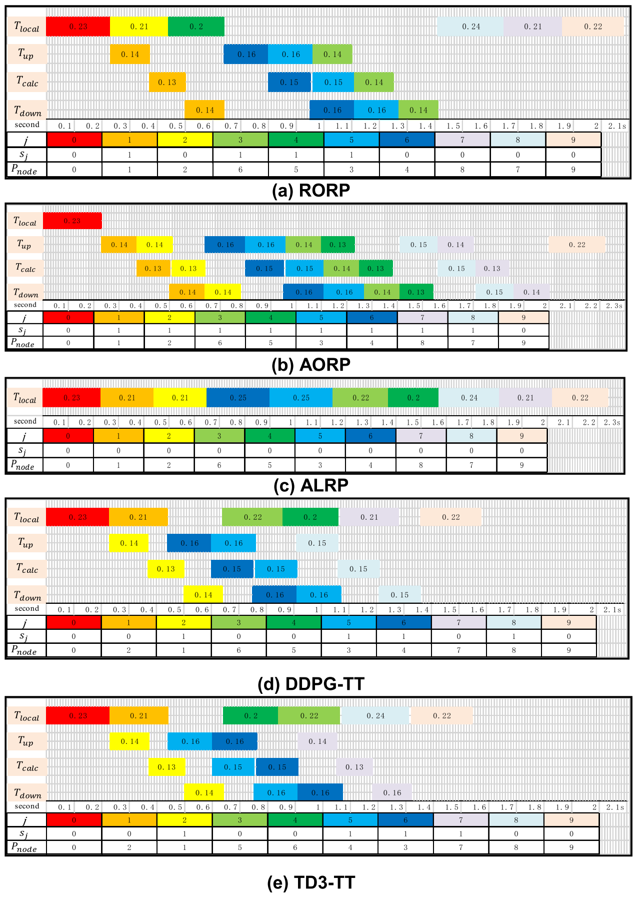


FIGURE 11. System latency of Different Offloading Schemes.

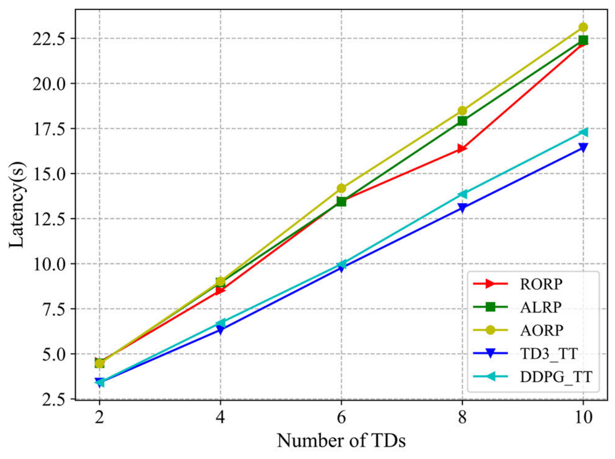


FIGURE 12. System Latency under Different N.

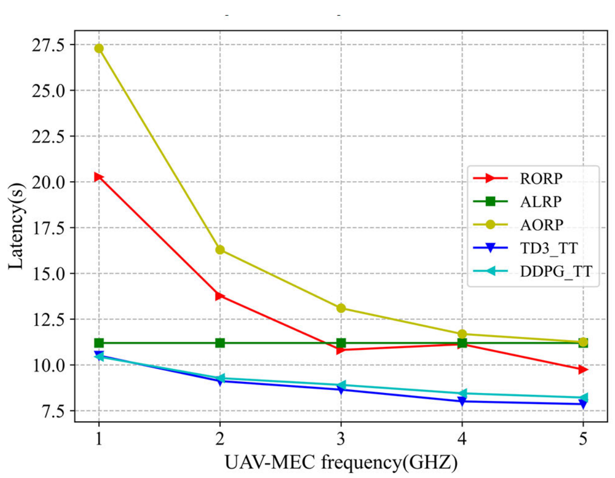


FIGURE 13. System Latency under Different F.

resources, resulting in a execution time of 2.24s. Fig. 11c and 9d depict the offloading scheduling schemes utilizing the DDPG-TT and TD3-TT algorithms, respectively. It is evident from these figures that their computation task latency is 1.54s and 1.58s, both of which are lower than the other schemes. This reduction in latency is due to the collaborative optimization of S and P_{node} by Agent 3, which efficiently utilizes both local and MEC server computational resources for parallel computation.

To evaluate the performance of the TD3-TT under varying TD counts, we conducted tests with various offloading schemes using a consistent random seed. The results from Fig. 12 clearly indicate that as the number of TD increases, the performance advantage of the TD3-TT becomes more pronounced compared to other benchmark schemes. Considering that the MEC server's computing capability directly affects the task execution latency, we adjust the MEC server's computing resource frequency while keeping the number of TDs unchanged at 5, to observe its impact on the execution latency, as shown in Fig. 13. When the computational resource frequency is set at 1GHz, it can be observed that the execution latency of both TD3-TT and DDPG-TT are close to the ALRP. This behavior arises from the inclination of Agent 1 to offload tasks locally when the offloading cost exceeds the local computation cost. As the F increases gradually, TD3-TT reasonably offloads tasks to the local device or MEC server based on their size, achieving the maximum degree of parallel computing to reduce task execution latency.

Considering that all the experimental task node quantities M=10 remain fixed, to investigate the impact of task node quantities on execution latency, Fig. 14 demonstrates the adaptability of the TD3-TT scheme under varying types of DAG tasks. By altering the DAG's node quantity M within the simulation environment and accordingly adjusting the dimensions of the Actor network's outputs, we compare the execution latency of different offloading schemes for varying node quantities in the DAG tasks. From the observation of the graph, it is evident that with the increase in the quantity of node tasks within a single time slot, the execution latency for all schemes also increase. TD3-TT demonstrates its capacity to adapt to diverse types of DAG tasks by analyzing the dependency relationships and scales of node tasks, thereby optimizing the outputs of the Actor network to achieve the optimal S and P_{node}. In comparison to RORP, TD3-TT showcases an average reduction in execution latency

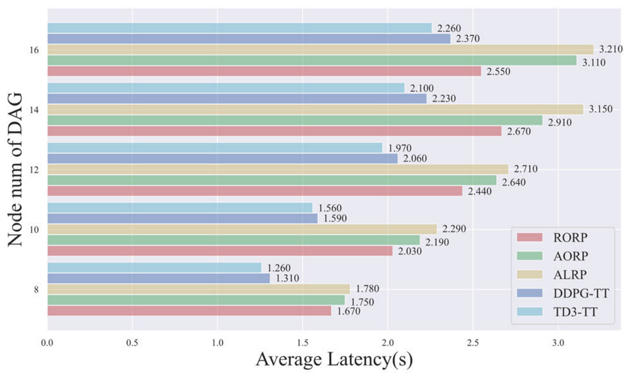


FIGURE 14. System Latency under Different Number of DAG Node Tasks.

of around 19%. In comparison to AORP, this reduction is approximately 27%, around 30% when compared to ALRP, and also a 4% decrease compared to DDPG-TT.

VI. CONCLUSION

This paper concentrates on optimizing UAV trajectories and the execution latency of DAG tasks in UAV-assisted MEC systems. It innovatively introduces the TD3-TT algorithm, which leverages multiple agents to collaboratively optimize the scheduling priorities for TDs, UAV flight speed, direction angles, offloading decisions, and scheduling priorities for DAG tasks. Within this algorithm, Agents 1 and 2 utilize heterogeneous reward functions, enabling the UAV to select optimal flight paths that not only minimize the line-of-sight distance to TDs but also effectively reduce the flight distance. Moreover, the algorithm ingeniously maps the network output of Agent 3 to the offloading decisions and scheduling priorities of DAG tasks. Extensive numerical simulation results demonstrate that, compared to other benchmark schemes, the TD3-TT algorithm exhibits remarkable adaptability across different types of DAG tasks, significantly reducing task execution latency. However, this paper focuses solely on latency optimization. Future work will need to extend to joint optimization of latency and energy consumption and collaborative planning in complex flight environments with obstacles, as well as cooperation among multiple UAVs.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[2] L. Chettri and R. Bera, "A comprehensive survey on Internet of Things (IoT) toward 5G wireless systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 16–32, Jan. 2020.

[3] M. A. Jamshed, K. Ali, Q. H. Abbasi, M. A. Imran, and M. Ur-Rehman, "Challenges, applications, and future of wireless sensors in Internet of Things: A review," *IEEE Sensors J.*, vol. 22, no. 6, pp. 5482–5494, Mar. 2022.

[4] C. Zhang and Y. Lu, "Study on artificial intelligence: The state of the art and future prospects," *J. Ind. Inf. Integr.*, vol. 23, Sep. 2021, Art. no. 100224.

[5] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and AI for UAVs: Opportunities and challenges," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15435–15459, Sep. 2022.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[7] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020.

[8] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 11, pp. 3836–3851, Nov. 2022.

[9] M. Abrar, U. Ajmal, Z. M. Almohaimeed, X. Gui, R. Akram, and R. Masroor, "Energy efficient UAV-enabled mobile edge computing for IoT devices: A review," *IEEE Access*, vol. 9, pp. 127779–127798, 2021.

[10] Y. Nie, J. Zhao, F. Gao, and F. R. Yu, "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, Dec. 2021.

[11] W. Lu, Y. Ding, Y. Gao, Y. Chen, N. Zhao, Z. Ding, and A. Nallanathan, "Secure NOMA-based UAV-MEC network towards a flying eavesdropper," *IEEE Trans. Commun.*, vol. 70, no. 5, pp. 3364–3376, May 2022.

[12] A. Mohammed, H. Nahom, A. Tewodros, Y. Habtamu, and G. Hayelom, "Deep reinforcement learning for computation offloading and resource allocation in blockchain-based multi-UAV-enabled mobile edge computing," in *Proc. 17th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. (ICCWAMTIP)*, Chengdu, China, Dec. 2020, pp. 295–299.

[13] S. M. A. Huda and S. Moh, "Survey on computation offloading in UAV-enabled mobile edge computing," *J. Netw. Comput. Appl.*, vol. 201, May 2022, Art. no. 103341.

[14] A. M. Seid, J. Lu, H. N. Abishu, and T. A. Ayall, "Blockchain-enabled task offloading with energy harvesting in multi-UAV-assisted IoT networks: A multi-agent DRL approach," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3517–3532, Dec. 2022.

[15] A. M. Seid, G. O. Boateng, B. Mareri, G. Sun, and W. Jiang, "Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4531–4547, Dec. 2021.

[16] J. Chen, H. Xing, Z. Xiao, L. Xu, and T. Tao, "A DRL agent for jointly optimizing computation offloading and resource allocation in MEC," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17508–17524, Dec. 2021.

[17] J. Wu, L. Wang, Q. Jin, and F. Liu, "Graft: Efficient inference serving for hybrid deep learning with SLO guarantees via DNN re-alignment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 2, pp. 280–296, Feb. 2024.

[18] J. Wu, L. Wang, Q. Pei, X. Cui, F. Liu, and T. Yang, "HiTDL: High-throughput deep learning inference at the hybrid mobile edge," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4499–4514, Dec. 2022.

[19] Q. Wu and R. Zhang, "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6614–6627, Dec. 2018.

[20] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "On-edge multi-task transfer learning: Model and practice with data-driven task allocation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1357–1371, Jun. 2020.

[21] S. Chen, L. Wang, and F. Liu, "Optimal admission control mechanism design for time-sensitive services in edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, London, U.K., May 2022, pp. 1169–1178.

[22] X. Zhou, L. Huang, T. Ye, and W. Sun, "Computation bits maximization in UAV-assisted MEC networks with fairness constraint," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 20997–21009, Nov. 2022.

[23] Z. Xie, X. Song, J. Cao, and W. Qiu, "Providing aerial MEC service in areas without infrastructure: A tethered-UAV-based energy-efficient task scheduling framework," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25223–25236, Dec. 2022.

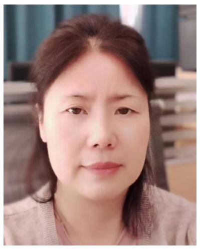
[24] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, Feb. 2019.

[25] Z. Yang, S. Bi, and Y. A. Zhang, "Online trajectory and resource optimization for stochastic UAV-enabled MEC systems," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5629–5643, Jul. 2022.

[26] J. Xue, Q. Wu, and H. Zhang, "Cost optimization of UAV-MEC network calculation offloading: A multi-agent reinforcement learning method," *Ad Hoc Netw.*, vol. 136, Nov. 2022, Art. no. 102981.

[27] L. Zhang and N. Ansari, "Optimizing the operation cost for UAV-aided mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6085–6093, Jun. 2021.

[28] N. Lin, H. Tang, L. Zhao, S. Wan, A. Hawbani, and M. Guizani, "A PDDQNLP algorithm for energy efficient computation offloading in UAV-assisted MEC," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 8876–8890, Dec. 2023.
[29] J. Chen, X. Cao, P. Yang, M. Xiao, S. Ren, Z. Zhao, and D. O. Wu, "Deep reinforcement learning based resource allocation in multi-UAV-aided MEC networks," *IEEE Trans. Commun.*, vol. 71, no. 1, pp. 296–309, Jan. 2023.
[30] V. D. Tuong, T. P. Truong, T.-V. Nguyen, W. Noh, and S. Cho, "Partial computation offloading in NOMA-assisted mobile-edge computing systems using deep reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13196–13208, Sep. 2021.
[31] J. Ji, K. Zhu, C. Yi, and D. Niyato, "Energy consumption minimization in UAV-assisted mobile-edge computing systems: Joint resource allocation and trajectory design," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8570–8584, May 2021.
[32] Q. Tang, L. Chang, K. Yang, K. Wang, J. Wang, and P. K. Sharma, "Task number maximization offloading strategy seamlessly adapted to UAV scenario," *Comput. Commun.*, vol. 151, pp. 19–30, Feb. 2020.
[33] J. Bi, H. Yuan, K. Zhang, and M. Zhou, "Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 4, pp. 1941–1954, Oct. 2022.
[34] Y. Wang, W. Fang, Y. Ding, and N. Xiong, "Computation offloading optimization for UAV-assisted mobile edge computing: A deep deterministic policy gradient approach," *Wireless Netw.*, vol. 27, no. 4, pp. 2991–3006, May 2021.
[35] Y. Gan and Y. He, "Trajectory optimization and computing offloading strategy in UAV-assisted MEC system," in *Proc. Comput., Commun. IoT Appl. (ComComAp)*, Shenzhen, China, Nov. 2021, pp. 132–137.
[36] J. Liang, K. Li, C. Liu, and K. Li, "Joint offloading and scheduling decisions for DAG applications in mobile edge computing," *Neurocomputing*, vol. 424, pp. 160–171, Feb. 2021.
[37] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.
[38] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.



CHUNXIANG ZHENG received the master's degree in computer application.

She is a Professor with Anqing Normal University, with a strong background in the specialized field of computer science. Her research interests include edge computing, deep learning, and reinforcement learning. Her current research is dedicated to seamlessly integrating deep learning and reinforcement learning methodologies, aiming to provide innovative solutions for the optimization of communication resources.

KAI PAN is currently a Graduate Student majoring in information and communication engineering with Anqing Normal University. He has contributed to internationally recognized journals with published academic papers and actively engaged in numerous well-known academic conferences and professional practices. His research interests include artificial intelligence, deep reinforcement learning, edge computing, and machine vision.

JIADONG DONG received the Ph.D. degree in mechanical design and theory. He is a Distinguished Professor with Anqing Normal University, possesses a robust professional background in the field of mechanical engineering. His primary research interests include industrial internet, industrial edge computing, and machine learning. He serves as a Valuable Member for the Industrial Internet Expert Advisory Committee of Anhui Province.

LIN CHEN is currently a Graduate Student majoring in information and communication engineering with Anqing Normal University. She has published academic articles in well-known national journals and has participated in many academic conferences during her graduate studies. Her main research interests include industrial internet and edge computing.

QINGHU GUO is a Graduate Student with Anqing Normal University. His research interests include industrial internet, intelligent manufacturing, and machine vision.

SHUNFENG WU received the master's degree in information and communication engineering. He has a strong professional background in wireless communications. He is currently an Assistant Professor with the School of Electronic Engineering and Intelligent Manufacturing, Anqing Normal University. His research interests include artificial intelligence, machine learning, and wireless communications.

HANYUN LUO received the master's degree in computer application. He is a Professor with Anqing Normal University. His main research interests include edge computing, data mining, and network security.

XIAOLIN ZHANG received the master's degree in computer application. He is an Associate Professor with Anqing Normal University. His main research interests include edge computing, data mining, and network security.