EP-MUSTO: Entropy-Enhanced DRL-Based Task Offloading in Secure Multi-UAV-Assisted Collaborative Edge Computing

Longxin Zhang, *Member, IEEE*, Runti Tan, *Student Member, IEEE*, Buqing Cao, *Member, IEEE*, Lihua Ai, Kenli Li, *Senior Member, IEEE*, and Kenqin Li, *Fellow, IEEE*

***Abstract*—Unmanned aerial vehicles (UAVs)-assisted edge computing has emerged as an effective solution for providing contingency task offloading (TO) services when ground computing infrastructures are insufficient. However, UAVs face challenges in implementing efficient task offloading strategies due to their limited capabilities and the complexity of the privacy offloading problem. To address these challenges, this study constructs a digital twin (DT)-enabled UAV swarm-assisted secure computing model, which considers collaboration of devices, edges, and cloud resources. The model is designed to represent the three-tier computing environment as a DT virtual framework, allowing for the monitoring of network changes and the exploration of potential strategies. Furthermore, a joint optimization problem that considers time delay and energy consumption within encryption and decryption costs is formulated. To solve this problem, an entropy-enhanced proximal policy optimization-based multi-UAV assisted security-aware task offloading (EP-MUSTO) algorithm is proposed. In EP-MUSTO, the exploration capability is enhanced by utilizing an actor network with policy entropy, and the action cognition is improved through the parameterization of the hybrid action space (HAS). Experimental results demonstrate that compared with other advanced algorithms, EP-MUSTO achieves a reduction in security system costs (SSCs) and magnitude of convergence oscillations by at least 9.43% and 54.62%, respectively.**

***Index Terms*—Collaborative edge computing, digital twin (DT), security awareness, task offloading (TO), unmanned aerial vehicles (UAV) swarm assistance.**

Received 11 October 2024; revised 25 May 2025; accepted 4 June 2025. Date of publication 6 June 2025; date of current version 8 August 2025. This work was supported in part by the Postgraduate Scientific Research Innovation Project of Hunan Province under Grant CX20240916; in part by the Natural Science Foundation of Hunan Province under Grant 2023JJ50204; in part by the Scientific Research Foundation of Hunan Provincial Education Department, China, under Grant 23B0560; in part by the National Key Research and Development Program of China under Grant 2018YFB1003401; in part by the National Natural Science Foundation of China under Grant 61702178 and Grant 62072172; in part by the CCF C Sangfor "Yuanwang" Research Fund under Grant CCF-SANGFOR OF 2024002; and in part by the State Key Laboratory Program for Novel Software Technology under Grant KFKT2024B52. *(Corresponding author: Buqing Cao.)*

Longxin Zhang, Runti Tan, Buqing Cao, and Lihua Ai are with the School of Computer Science and Artificial Intelligence, Hunan University of Technology, Zhuzhou 412007, Hunan, China (e-mail: longxinzhang@hut.edu.cn; buqingcao@hut.edu.cn).

Kenli Li is with the College of Information Science and Engineering and the National Supercomputing Center, Hunan University, Changsha 410082, Hunan, China (e-mail: lkl@hnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JIOT.2025.3577487

I. Introduction

MOBILE-EDGEcomputing (MEC) has emerged as a novel computing architecture that places resources at the edge of networks. In contrast to cloud computing, MEC allows for the offloading of latency-sensitive tasks closer to terminal users (TUs), thereby alleviating the computational load on cellular networks. However, statically deployed servers face challenges in real-world, particularly in urban industrial clusters with high computational demands and in disaster-stricken or remote areas lacking sufficient infrastructure [1].

Unmanned aerial vehicles (UAVs) have been recognized as a promising technology within the MEC framework. UAV-assisted MEC (UAV-MEC) leverages the high flexibility, seamless deployment, and cost-effectiveness of UAVs to deliver emergency services. By equipping UAVs with airborne servers and utilizing wireless communication services, UAV-MEC has the potential to extend the limited network coverage of classical MEC and offer immediate services to users [2]. However, the capability of a single UAV to handle large-scale tasks is constrained, posing a challenge in meeting the increasing complexity of computation and coverage requirements.

In recent years, digital twin (DT) technology and deep reinforcement learning (DRL) have emerged as cutting-edge solutions to tackle these challenges. DT enables the creation of a virtual representation that mirrors the physical world, allowing for real-time system monitoring through this digital counterpart. It models fundamental principles of physical components, communication, and computation, collecting and analyzing action data from UAV-MEC, managing unified scheduling, and updating the DT model in response to state changes [3]. In contrast to heuristic algorithms, DRL can dynamically interact with intricate UAV environments, learning optimal solutions for sequential decision-making problems through the use of Markov decision processes (MDP) without the need for prior knowledge [4]. The collaboration between DT and DRL provides comprehensive environmental information and intelligent solutions for the computing platform.

However, the openness of wireless channels expose data sharing among devices to the risk of eavesdropping. Implementing privacy-preserving techniques in UAV-MEC and IoT is essential, yet it introduces extra encryption and decryption overheads, leading to increased offloading costs [5]. Furthermore, collaborative offloading within UAV clusters involves substantial action and state interactions. The large number of parameters and the overhead associated with privacy techniques considerably affect the learning performance and effectiveness of DRL-based optimization algorithms [6].

To tackle the challenges outlined above, this study explores the issue of security-aware task offloading (TO) in cooperative computing systems using improved DRL. We develop a DT-enabled UAV cluster-assisted TO system structured around the device-edge-cloud (D-E-C) three-layer framework. This UAV-assisted D-E-C (UAV-DEC) collaboration leverages various computational resources within the system to deliver complementary strengths for TO. Additionally, we explore an optimization algorithm based on entropy-enhanced DRL, which incorporates entropy optimization into the proximal policy optimization (PPO) method to enhance the exploration capabilities of policy generation. The primary contributions of this study are summarized below.

1) It develops a DT-empowered D-E-C collaborative privacy TO system that integrates UAV clusters. D-E-C collaboration provides computational services with complementary strengths for the task. The DT creates an effective mapping between UAV-DEC and the digital system, allowing for comprehensive monitoring of UAV-DEC parameters.
2) It devises a joint minimization problem that tackles secure system costs, encompassing delay, energy consumption, and privacy-preserving expenses, along with UAV endurance. This problem is transformed into an MDP that incorporates observation normalization and hybrid action space (HAS).
3) It proposes an entropy-enhanced PPO-based multi-UAV-assisted security-aware TO algorithm (EP-MUSTO). EP-MUSTO designs a strategy generation mechanism based on the maximum entropy (ME) mechanism and dual-action output structure to enhance the randomness of strategic actions and boost the representational capacity of mixed actions. Experimental results show the advantages of EP-MUSTO in terms of cost optimization and training stability.

The structure of the study is as follows. Section II discusses relevant work on TO schemes in MEC. Section III elaborates the multi-UAV-assisted collaborative TO model in DT and describes the optimization problem. Section IV presents the HAS-based MDP and describes the process details of EP-MUSTO. Section V evaluates the experimental performance of EP-MUSTO. Finally, Section VI summarizes the study.

II. Related Work

MEC has emerged as a key area of research for dynamic cross-area services. Generally, the existing TO schemes can be divided into two types of applications, i.e., classical MEC and UAV-MEC structures.

The TO problem has been extensively studied within the classical MEC. Zhang et al. [7] formulated the multiuser distributed computation offloading problem as a game and introduced a dynamic noncooperative game-based computational offloading algorithm focused on quality of experience. Zhang et al. [8] exploited a DT-enabled mirrored MEC system that leverages DT to create a learning population of agents from various vehicles dynamically. They proposed a distributed multiagent DRL (MADRL) scheme aimed at maximizing offloading efficiency. Xu et al. [9] discussed a secure MEC system with reconfigurable intelligent surface assistance to improve task offloading security. They introduced a joint optimization algorithm utilizing deep deterministic policy gradient (DDPG). Li et al. [10] developed a two-stage hybrid multiobjective optimization evolutionary algorithm grounded in competitive swarm optimization, aimed at minimizing delay and energy consumption in a cloud-edge collaboration system. Lin et al. [11] devised a TO scheme utilizing offline-to-online DRL, which employs a heuristic algorithm to initialize the DRL model during the offline phase.

The UAV effectively caters to the evolving demands of MEC through their adaptable computational resources. Li et al. [12] introduced a TO algorithm called maximized service efficiency PPO, which assesses the service fairness of all users using a fairness index to encourage UAVs to select service objects equitably. Zhou et al. [13] developed a secure MEC system supported by RIS-enabled UAVs and proposed low-complexity iterative algorithms to optimize RIS phase shifts and resource allocation jointly. Yan et al. [14] created a TO model for UAV-assisted vehicular networks to manage global information, along with a DRL-based algorithm for trajectory control and TO allocation. Xu et al. [15] utilized block coordinate descent and convex optimization techniques to explore the TO and UAV trajectory optimization challenges in UAV-MEC systems with task dependency constraints. Zhang et al. [16] explored the TO problem in UAV-DEC collaborative computing under task-dependent constraints and introduced a TO algorithm utilizing a saturated training rule and soft actor-critic (SAC) approach.

The MEC system equipped with multiple UAVs demonstrates enhanced adaptability in handling intricate TO environments. Shi et al. [17] established a DT-enabled multi-UAV-assisted MEC system and introduced a model-free DRL method grounded in federated learning with dual-delay DDPG. Consul et al. [18] proposed a generalized federated DRL approach utilizing meta-learning techniques to address resource allocation and TO challenges in DT-enabled UAV-MEC systems. Hao et al. [19] investigated UAV-MEC with the aim of maximizing long-term system gain, introducing a DRL-based priority-aware TO algorithm featuring enhanced hybrid action processing through embedded tables with conditional variational autoencoders. Yu et al. [20] presented a QMIX-based method for delay minimization to facilitate efficient collaboration among heterogeneous UAVs in an aeronautical MEC system. Zhou et al. [21] examined a heterogeneous collaborative MADRL algorithm that employs reward sharing to maximize fair weighted throughput.

The aforementioned studies do not consider the importance of heterogeneous computation and privacy encryption in open UAV-MEC. Furthermore, they do not address the convergence challenges of the TO algorithm arising from multiserver collaboration and the demand for mixed action processing.

III. System Model and Problem Description

This section initially presents the system model of a DT-enabled UAV-DEC cooperative security-aware TO system. On this basis, it outlines a joint minimization problem involving system delay and energy consumption.

A. System Model of DT Enablement

Fig. 1 depicts the DT-enabled multi-UAV-assisted three-layer cooperative security-aware TO model, which comprises a UAV-DEC physical system and a DT network (DTN). The physical system features a formation of UAVs flying independently within a designated square area to deliver computational services via MEC onboard servers for TUs. The UAV-DEC physical model is divided into the following three components.

1) *Terminal Layer:* This layer is composed of TUs within the region, with their number defined as $i \in \mathcal{I} = \{1, 2, \ldots, I\}$. TUs generate task requests but have limited computational power. The attributes of task_n ($n \in \mathcal{N} = \{1, 2, \ldots, N\}$) can be expressed as $\text{task}_n = \{\text{TU}_i, t_n, l_n, d_n\}$, where each element includes the generator, the maximum allowable deadline, the data transfer size, and the data computational volume, respectively.
2) *Edge Layer:* This layer comprises multiple UAVs equipped with edge servers that act as processors or deliverers of tasks, with their number defined as $j \in \mathcal{J} = \{1, \ldots, J\}$.
3) *Cloud Layer:* This layer offers sufficient but costly computing resources from cloud centers.

In the proposed scenario, all TUs move randomly. On the basis of the current TO policy, the TUs can decide whether to process data locally, at the UAV, or in the cloud to maximize overall benefits. When a task necessitates data transmission from the TU locally, AES technology is used to encrypt the data, ensuring secure transmission. The costs associated with encryption and decryption for the tasks are considered.

A corresponding DTN was developed in UAV-DEC to characterize comprehensively the complex TO environment, which encompasses random environmental variations, terminal movement, and TO decision-making [22], [23]. As illustrated in Fig. 1, the DTN performs three primary functions.

1) Recording physical models and parameters in real time within the digital network space for predicting dynamic changes in the UAV-DEC real model.
2) Creating precise models of physical entity states to enhance policy accuracy in intricate scenarios.
3) Employing the DRL method to simulate task allocation schemes and refine task offloading decisions in simulations.

Through observation of the DTN, the model aims to identify the most appropriate TU for each UAV and achieve a globally optimal offloading path. The DTN is defined as

$$\text{DTN} = \{\text{DT}_i, \text{DT}_j\} \quad \forall i, j. \tag{1}$$

The DTN utilizes the 1-D vector $\text{DT}_i = \{p_i^{\text{up}}, l_i, f_i, P_{i,j}\}$ to characterize the DT model of TU_i, where each element is represented by the transmission power, computation frequency, current location, and channel obstacle condition of TU_i in turn. The DTN utilizes $\text{DT}_j = \{p_j^{\text{up}}, l_j, f_j, E_j, v_j, o_j\}$ to represent the DT model of UAV_j, where each element is represented by the transmission power, current position, computation frequency, current power, flight speed, and flight angle of UAV_j in turn.

B. Movement Model

The physical model employs a 3-D Cartesian coordinate system. During each time slot $t \in \mathcal{T} = \{1, 2, \ldots, T\}$, the position of UAV_j is denoted as $L_j(t) = [x_j(t), y_j(t), Z]$, where Z is the UAV's constant altitude. The coordinates of TU_i can be expressed as $L_i(t) = [x_i(t), y_i(t), 0]$. At the start of t, UAVs receive an action command from the DTN. At this point, the coordinates of UAV_j are updated as follows:

$$x_j(t+1) = x_j(t) + \cos\left(o_j(t)\right) \times \text{dis}_j(t) \tag{2}$$

$$y_j(t+1) = y_j(t) + \sin\left(o_j(t)\right) \times \text{dis}_j(t) \tag{3}$$

where $\text{dis}_j(t) = v_j(t) \times \Delta$ denotes the distance traveled by UAV_j, and Δ indicates the time slot allocated for the flight. In a multi-UAV system, the trajectories of any two UAVs must adhere to the collision avoidance constraint, meaning that

$$||L_j(t) - L_{j+1}(t)|| \geq \text{dis}_{\min}(t) \tag{4}$$

where $\text{dis}_{\min}(t)$ denotes the minimum distance between UAVs. Given that UAV_j and TU_i are limited to a specific area, with the constraint represented by the following:

$$L(t) \in \left\{x(t), y(t) \,|\, x(t) \in [0, L], y(t) \in [0, W]\right\} \quad \forall t \tag{5}$$

where L and W denote the length and width of the area, respectively.

C. Communication Model

This system incorporates an air-to-ground path loss model that considers both line-of-sight (LoS) and non-LoS (NLoS) links among TUs, UAVs, and the cloud center.

The channel gain between TU_i and UAV_j is represented as $g_{i,j}(t) = g_0 \cdot \text{dis}_{i,j}^{-2}(t)$, where g_0 signifies the channel power gain at a reference distance of one meter and $\text{dis}_{i,j}(t) = ((L_i(t) - L_j(t))^2 + Z^2)^{1/2}$ denotes the 3-D distance between TU_i and UAV_j. Given that obstacles frequently lead to NLoS status in dynamic UAV-DEC, the channel transmission rate between UAV_j and TU_i is defined as follows:

$$r_{i,j}(t) = B\log_2\left(1 + \frac{p_i^{\text{up}} \cdot g_{i,j}(t)}{\sigma_{\text{NLoS}}^2 + P_{i,j}(t) \cdot W_{\text{NLoS}}}\right) \tag{6}$$

where B represents the channel bandwidth. σ_{NLoS}^2 indicates the Gaussian white noise resulting from obstruction. $P_{i,j}$ denotes whether an NLoS exists between TU_i and UAV_j. W_{NLoS} denotes the transmission loss caused by NLoS.

Similarly, the channel gain $g_{j,c}(t)$ and the transmission rate $r_{j,c}(t)$ between UAV_j and the cloud can be determined. Considering the limited service distance of the UAV, the maximum communication distance $R_{\max}$ for UAV_j is expressed by

$$\text{dis}_{i,j}(t) \leq R_{\max} \quad \forall i \in I, j \in J. \tag{7}$$

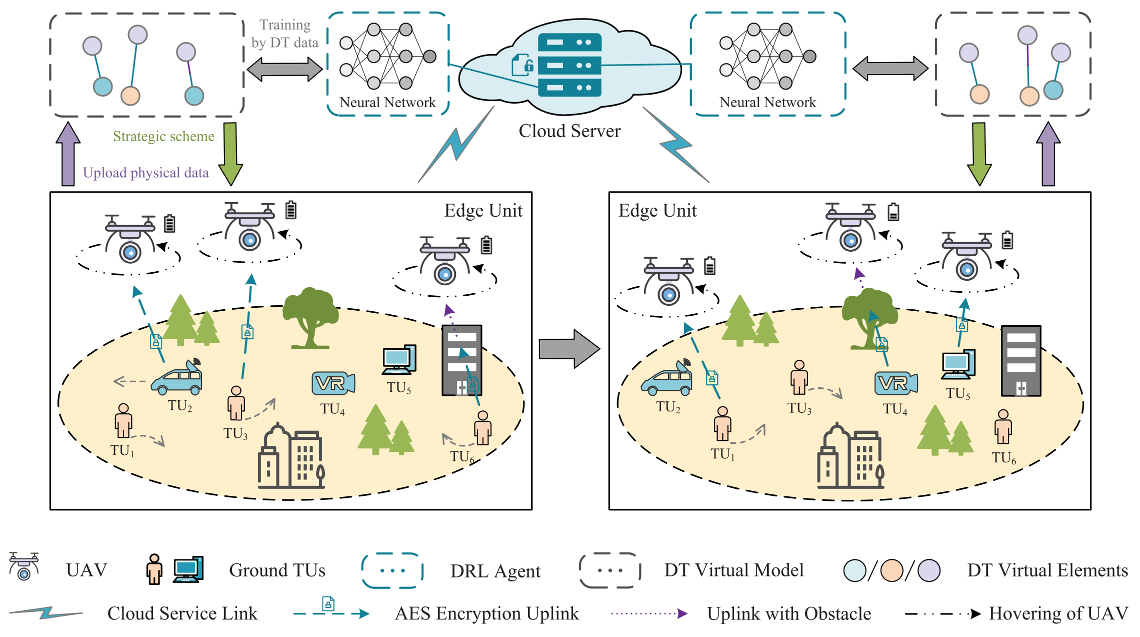

Fig. 1. DT-enabled UAV-DEC cooperative security-aware TO model.

D. TO Model

During time slot t, tasks in the environment can opt for one of three offloading methods: 1) local processing; 2) offloading to UAV processing; or 3) transferring to cloud center processing.

When task$_n$ chooses to execute locally at TU$_i$, the time delay and energy consumption of task$_n$ depend solely on TU$_i$. In contrast to [24], the local time delay $T_i(n)$ is denoted as

$$T_i(n) = \frac{d_n s}{f_i} \tag{8}$$

where s represents the CPU performance factor. Similar to [25], the nonoffloading energy consumption $E_i(n)$ is expressed as

$$E_i(n) = \kappa f_i^2 d_n s \tag{9}$$

where κ denotes the effective capacitance coefficient.

When task$_n$ chooses to offload to UAV$_j$ for execution, the offloading cost for task$_n$ includes the transmission delay $T_{i,j}^{\text{tra}}(n)$ and energy consumption $E_{i,j}^{\text{tra}}(n)$ from TU$_i$ to UAV$_j$. Meanwhile, the computation cost includes the delay $T_j^{\text{cal}}(n)$ and the energy consumption $E_j^{\text{cal}}(n)$ from UAV$_j$. The edge delay $T_{i,j}(n)$ and energy consumption $E_{i,j}(n)$ are defined by

$$T_{i,j}(n) = T_{i,j}^{\text{tra}}(n) + T_j^{\text{cal}}(n) = \frac{l_n}{r_{i,j}(t)} + \frac{d_n s}{f_j} \tag{10}$$

$$E_{i,j}(n) = E_{i,j}^{\text{tra}}(n) + E_j^{\text{cal}}(n) = \frac{p_i^{\text{up}} l_n}{r_{i,j}(t)} + \kappa f_j^2 d_n s. \tag{11}$$

When task$_n$ is designated for execution at the cloud center, the total completion delay of task$_n$ consists of three parts, which in addition to the transmission delay from TU$_i$ to UAV$_j$, also includes the transmission delay $T_{j,c}^{\text{tra}}(n)$ from UAV$_j$ to the cloud and the computation delay $T_c^{\text{cal}}(n)$ of the cloud. The time delay of the cloud $T_{i,c}(n)$ is represented by

$$\begin{aligned} T_{i,c}(n) &= T_{i,j}^{\text{tra}}(n) + T_{j,c}^{\text{tra}}(n) + T_c^{\text{cal}}(n) \\ &= \frac{l_n}{r_{i,j}(t)} + \frac{l_n}{r_{j,c}(t)} + \frac{d_n s}{f_c}. \end{aligned} \tag{12}$$

Similarly, the cloud energy consumption $E_{i,c}(n)$ is defined as

$$\begin{aligned} E_{i,c}(n) &= E_{i,j}^{\text{tra}}(n) + E_{j,c}^{\text{tra}}(n) + E_c^{\text{cal}}(n) \\ &= \frac{p_i^{\text{up}} l_n}{r_{i,j}(t)} + \frac{p_j^{\text{up}} l_n}{r_{j,c}(t)} + \kappa f_c^2 d_n s. \end{aligned} \tag{13}$$

E. Security Cost Model

Security cost model employs 128-bit AES for encrypting and decrypting data during task transmission, thereby ensuring data security. Once the TO target for task$_n$ is established, UAV$_j$ provides the key used for data encryption to TU$_i$ and employs the same key with the cloud to decrypt the received data. Similar to [26], the energy consumption and delay related to the AES of task$_n$ are expressed as

$$t_n^{\text{pvc}} = \frac{d_n^{\text{enc}}}{f_i} + \frac{d_n^{\text{dec}} \times \gamma_n^1}{f_u} + \frac{d_n^{\text{dec}} \times \gamma_n^2}{f_c} \tag{14}$$

$$\begin{aligned} e_n^{\text{pvc}} &= p_i \times \frac{d_n^{\text{enc}}}{f_i} + p_j \times \frac{d_n^{\text{dec}} \gamma_n^1}{f_u} + p_i \times \frac{d_n^{\text{dec}} \gamma_n^2}{f_c} \\ &= \kappa \times \left(f_i^2 d_n^{\text{enc}} + \left(\gamma_n^1 f_j^2 + \gamma_n^2 f_c^2 \right) \times d_n^{\text{dec}} \right) \end{aligned} \tag{15}$$

where d_n^{enc} and d_n^{dec} represent the CPU cycles of data encryption and decryption. ω_n^m is a binary variable that indicates whether task$_n$ can choose between local computing or offloading computing. $m \in \{0, 1, 2\}$ represents three types of facilities: 1) the TU; 2) the UAV; and 3) the cloud server. When $\omega_n^1 = 1$, task$_n$ is categorized as an edge offloading state. Therefore, the TO delay $T(n)$ and the energy consumption $E(n)$ for task$_n$ are improved as

$$\begin{aligned} T(n) &= \omega_n^0 \times T_i(n) + \omega_n^1 \times \left(T_{i,j}(n) + t_n^{\text{pvc}} \right) + \omega_n^2 \\ &\quad \times \left(T_{i,c}(n) + t_n^{\text{pvc}} \right) \end{aligned} \tag{16}$$

$$E(n) = \omega_n^0 \times E_i(n) + \omega_n^1 \times \left(E_{i,j}(n) + e_n^{\text{pvc}} \right) + \omega_n^2$$

$$\times \left(E_{i,c}(n) + e_n^{\text{pvc}}\right). \tag{17}$$

According to (16) and (17), the total cost of task$_n$ under security considerations is defined as $C_n = \alpha \times T(n) + \beta \times E(n)$, where α and β are the weights used to denote the varying levels of importance assigned to delay and energy consumption, respectively. Consequently, the primary evaluation metric is defined as security system cost (SSC).

Definition 1 (SSC): The SSC is defined as the total cost incurred by each task$_n$ during computation and transmission across the entire time sequence. It accounts for the costs of encryption and decryption and the energy consumed for hovering and flying while the UAV cluster is in operation. The SSC is defined as follows:

$$\text{SSC} = \sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{n=1}^{N}\left(C_n + E_j^{\text{fly}}(t) + E_j^{\text{hov}}(t)\right). \tag{18}$$

F. Problem Description

The aim of this study is to optimize the SSC for all tasks and UAVs while taking into account the costs associated with AES technology and the constraints of computational resources. The minimization problem can be defined as follows:

$$\begin{aligned}
(P_1):\ & \underset{\{\delta,\theta,\gamma,o,v\}}{\text{minimize}} \quad \text{SSC}\\
\text{s.t.}\ & C_1: \alpha + \beta = 1\\
& C_2: \omega_n^m \in \{0, 1\} \quad \forall m, n\\
& C_3: \sum_{m=0}^{2} \omega_n^m = 1 \quad \forall n\\
& C_4: \sum_{t=1}^{T}\left(E_j^{\text{fly}}(t) + E_j^{\text{hov}}(t) + E_j^{\text{sev}}(t)\right) \le E_j \quad \forall t, j\\
& C_5: T(n) \le D_n^{\max} \quad \forall n\\
& C_6: 0 \le o_j(t) \le 2\pi,\ 0 \le v_j(t) \le v_{\max} \quad \forall t\\
& C_7: (4), (5), (7)
\end{aligned} \tag{19}$$

where C_1 represents the weight constraint. C_2 and C_3 specify the task attribution constraints, with ω_n^m ensuring that each task is processed individually. C_4 represents the UAV power constraint, where $E_j^{\text{fly}}(t)$ and $E_j^{\text{hov}}(t)$ indicate the energy consumed during flying and hovering by UAV$_j$ [16]. The service energy consumption $E_j^{\text{sev}}(t)$ encompasses $E_{j,c}^{\text{tra}}$ and E_j^{cal}. C_5 denotes a tolerance constraint, stipulating that the completion time of task$_n$ must be less than its maximum deadline. C_6 establishes the UAV motion constraint. C_7 defines the lawful action constraint, specifying the legality of the actions of individuals in the environment.

IV. UAV-Assisted TO Algorithm Based on Entropy-Enhanced PPO

This section transforms the previous problem into an MDP for processing with DRL. It proceeds to presents the principle of the entropy-enhanced PPO algorithm, followed by a detailed exploration of the EP-MUSTO.

A. MDP Formulation

Evidently, P_1 is classified as mixed-integer nonlinear programming, which is known to be NP-hard and nonconvex. Traditional algorithms struggle to handle it efficiently, which impels this study to design other schemes. Given that P_1 is a sequential decision-making problem within a time-varying environment, it is well-suited for an MDP-based DRL approach.

The MDP offers a mathematical framework for DRL, allowing the agent to engage with a sequential decision-making environment and learn optimal schemes through neural networks and actions in accordance with Markovian characteristics. This section reformulates P_1 into an MDP consisting of three components: 1) state; 2) action; and 3) reward function.

1) State: Within the UAV-DEC environment, the state space is constructed from the UAV data received by the DTN and the current environmental information, which is modeled as

$$S = \left\{s_t | s_t = \{E_r(t),\ L_U(t),\ L_{TU}(t),\ d_i(t),\ p_{i,j}(t)\}\right\} \tag{20}$$

where $E_r(t)$ represents the remaining power of each UAV at time slot t. $L_U(t)$ and $L_{TU}(t)$ indicate the location of each UAV and TU. $d_i(t)$ denotes the number of tasks waiting offloading by TU$_i$. Normalizing the state space can enhance the processing efficiency of the neural network.

Definition 2 [State Normalization (SN)]: SN employs the difference between the maximum value state$_{\max}$ and the minimum value state$_{\min}$ of each state as a normalization factor, subsequently carrying out the normalization operation. The SN for each element in the state space is defined as

$$\text{SN}(\text{state}_t) = \text{state}_t/(\text{state}_{\max} - \text{state}_{\min}) \quad \forall\ \text{item}_t \in s_t \tag{21}$$

where state$_t$ denotes the five state elements within S.

2) Action: The UAV makes decisions based on the current state of the environment. The action space is modeled as

$$A = \{a_t | a_t = \{U(t), O_u(t), V_u(t), \omega(t)\}\} \tag{22}$$

where $U(t)$ denotes the service target set of the UAV cluster during the time slot t. $O_u(t)$ and $V_u(t)$ indicate the flight angle set and flight speed set.

A encompasses discrete actions [such as $U(t)$ and $\omega(t)$] and continuous actions [such as $O_u(t)$ and $V_u(t)$], constituting an HAS. To effectively handle it, a parameterized action (PA) space is implemented in the form of a hierarchical structure. The PA is represented as $PA = \{(a^d,\ a^c)|a^c \in C_d$ for all $a^d \in D\}$. In PA, the agent selects a discrete action a^d from the discrete action set D to determine which task should be offloaded to a specific computational facility. Each a^d is linked to a continuous parameter set C_d, which includes $o_j(t)$ and $v_j(t)$ [27]. On this basis, continuous parameters a^c are assigned from C_d to specify the flight actions of the UAV.

3) Reward: In the context of P_1, which focuses on the joint optimization of UAV energy consumption and SSC, the reward is minimized according to the level of optimization achieved for this goal. The reward is formulated by

$$R = r(s_t, a_t) = -\sum_{j=1}^{J}\sum_{n=1}^{N}\left(C_n + E_j^{\text{fly}}(t) + E_j^{\text{hov}}(t)\right). \tag{23}$$

B. Policy Optimizer Considering Exploration Entropy and Mixed Action Outputs

HAS effectively represents action design in complex environments. However, discretizing continuous actions in value-based algorithms can affect training accuracy. By contrast, policy-based approaches utilize stochastic gradient ascent to optimize a policy function that directly learns a specific policy through interaction of the environment. Given that θ represents the policy parameter, the policy π_θ is defined as

$$J(\theta) = \mathbb{E}_{s_0}\left[V^{\pi_\theta}(s_0)\right] = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right] \tag{24}$$

where V^{π_θ} represents the total state value of π_θ. γ^t indicates the reward discount factor for the time slot t. This implies that the strategy is not optimized discretely for individual action in each time slot; instead, it focuses on the cumulative reward of the entire strategy from the initial state s_0 to the end while updating θ in the direction of the gradient $\nabla_\theta J(\theta)$. The difference between the old and new strategies is represented as follows:

$$J(\theta^*) - J(\theta) = \sum_{t=0}^{T} \gamma^t \mathbb{E}_{s_t \sim P_t^{\pi_{\theta^*}}} \mathbb{E}_{a_t \sim \pi_{\theta^*}(\cdot|s_t)}\left[A^{\pi_\theta}(t)\right] \tag{25}$$

where $P_t^{\pi_{\theta^*}}$ represents the current state transfer distribution. $A^{\pi_\theta}(t) = r(s_t, a_t) + \gamma V_\delta^{\pi_\theta}(s_{t+1}) - V_\delta^{\pi_\theta}(s_t)$ denotes the temporal difference residual (TDR) at single step. $V_\delta^{\pi_\theta}(s)$ is the state value function with parameter δ. The raw TDR can introduce considerable variance. Stability is updated by utilizing TDRs from multiple future time steps to compute the advantage function estimate while weighting current and future rewards with the GAE coefficient $\varphi \in [0, 1]$. Similar to [28], the dominance function estimation is expressed as

$$\widehat{A}(t) = \sum_{i=0}^{\infty}\left[(\gamma\varphi)^i A^{\pi_\theta}(t+i)\right]. \tag{26}$$

However, directly tracing the policy gradient without constraints may result in excessive policy changes. This issue is mitigated by using the trust region (TR) mechanism to limit the updated step size within a defined region. However, using the conjugate gradient method to solve Kullback-Leibler divergence constrained optimization can affect the flexibility and efficiency of TR Policy Optimization (TRPO) [29]. By contrast, the PPO algorithm manages the disparity by incorporating a regularization term constraint. It directly imposes upper and lower bounds on the objective function, simplifying TRPO. The loss function of the policy network is defined as

$$L^C(\theta) = \widehat{\mathbb{E}}_t\Big[\min\Big\{I(t, \theta)\widehat{A}(t),\ \text{clip}\,(I(t, \theta),\ 1-\tau, 1+\tau)\widehat{A}(t)\Big\}\Big] \tag{27}$$

where $I(t, \theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ represents the importance weight sampling (IS) ratio. The IS technique corrects the bias between the past and current gradients. $\text{clip}(x, y, z) := \max(\min(x, z), y)$ denotes the trimming function, which restricts x to $[y, z]$. Values outside the interval will be clipped to the boundary, with τ serving as the cropping boundary constant. As a result, PPO is chosen as the policy generation kernel for EP-MUSTO.

PPO gradient direction update method provides a certain exploration ability through a randomly distributed strategy function. Entropy serves as a measure of uncertainty for a random variable. In a DRL framework grounded in ME theory, maximizing the strategy gain and the entropy value encourages the agent to behave as randomly as possible. At this point, the strategy is represented as

$$\pi^{\text{MERL}} = \arg\max_{\pi^{\text{MERL}}} \mathbb{E}_{(s_t,a_t)\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t, a_t) + \psi E(\pi_\theta(\cdot|s_t))\right] \tag{28}$$

where ψ represents the temperature coefficient, which is adjusted to control the randomness weight of the strategy. $E(\pi_\theta(\cdot|s_t)) = \mathbb{E}_{a\sim\pi(\cdot|s_t)}[-\log \pi_\theta(a|s_t)]$ is the information entropy, serving as a specific measure of the degree of randomness in the strategy π_θ at state s_t. By maximizing entropy, the strategy network can utilize its chaos and randomness to capture multiple extreme points of the policy, avoiding local optima. Therefore, this study applies the concept of MERL to PPO strategy networks, with the optimization objective based on (27) and (28), which is rewritten as

$$L^{CE}(\theta) = \widehat{\mathbb{E}}_t\Big[\min\Big\{I(t, \theta)\widehat{A}(t)\text{clip}(I(t, \theta)\ 1-\tau, 1+\tau) \times \widehat{A}(t)\Big\} - \psi \log \pi_\theta(a_t|s_t)\Big]. \tag{29}$$

Simultaneously, the structure and outputs of the policy network must be adjusted in accordance with the parameterized HAS. The policy network is required to optimize the mutually independent discrete policy π_{θ_d} and continuous policy π_{θ_c} separately. The optimization objectives for both is defined as

$$L_d^{CE}(\theta_d) = \widehat{\mathbb{E}}_t\Big[\min\Big\{I_d(t, \theta_d)\widehat{A}(t),\ \text{clip}\Big(I_d(t, \theta_d)\ 1-\tau,\ 1+\tau\Big)\widehat{A}(t)\Big\} - \psi \log \pi_{\theta_d}\Big(a_t^d|s_t\Big)\Big] \tag{30}$$

$$L_c^{CE}(\theta_c) = \widehat{\mathbb{E}}_t\Big[\min\Big(I_c(t, \theta_c)\widehat{A}(t),\ \text{clip}(I_c(t, \theta_c), 1-\tau,\ 11-\tau,\ 1+\tau)\widehat{A}(t)\Big) - \psi \log \pi_{\theta_c}\big(a_t^c|s_t\big)\Big] \tag{31}$$

where $I_d(t, \theta_d) = \pi_{\theta_d}(a_t^d|s_t)/\pi_{\theta_d^{\text{old}}}(a_t^d|s_t)$ and $I_c(t, \theta_c) = \pi_{\theta_c}(a_t^c|s_t)/\pi_{\theta_c^{\text{old}}}(a_t^c|s_t)$ represent the IS ratios of π_{θ_d} and π_{θ_c}, respectively [29]. Thus, the total loss function is defined as

$$L^{\text{total}} = -\Big(\mu L_d^{CE}\big(\theta_{u,\omega}\big) + L_c^{CE}(\theta_o) + L_c^{CE}(\theta_v)\Big) \tag{32}$$

where the discrete actions are updated in accordance with $L_d^{CE}(\theta_d)$, while the continuous actions also follow the update of $L_c^{CE}(\theta_c)$, with μ serving as the weighting factor.

Algorithm 1 outlines the parameter update rules for the PPO algorithm. The PPO algorithm utilizes an actor-critic architecture, comprising two types of networks: 1) the actor and 2) the critic. The actor interacts with the environment to generate strategies that follow a importance sampling. Meanwhile, the critic learns a value function from the existing experience and computes $\widehat{A}(t)$ for the actor using V_δ. The

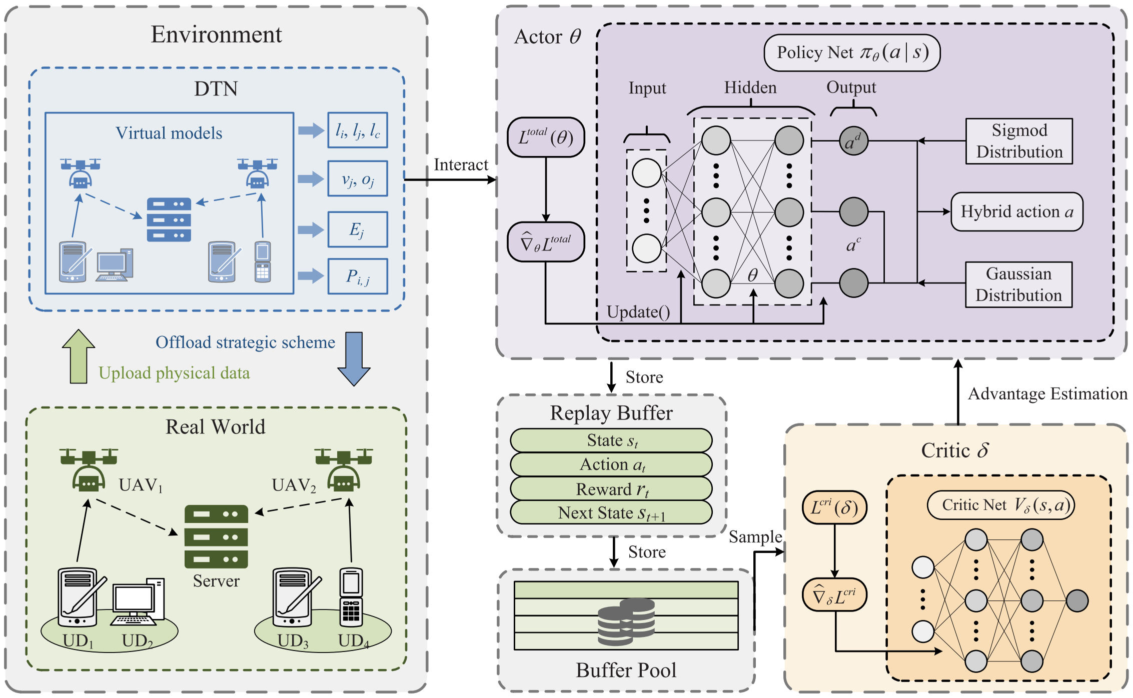

Fig. 2. Overall structure of the EP-MUSTO algorithm.

Algorithm 1 PPO Algorithm Based on Policy Entropy and HAS

```
Require: Policy parameters $\theta_d$ and $\theta_c$, critic parameter $\delta$,
    number of updates $U$, batch size $M$.
Ensure: The updated parameters $\theta_d$, $\theta_c$, and $\delta$.
1: for $u = 1, 2, \ldots, U$ do
2:     Calculate the total reward $V^{\pi_\theta}$ for each trajectory in $M$;
3:     Calculate $\widehat{A}(t)$ for each trajectory by using (26);
4:     Calculate the loss of actor $L^{total}$ by using (32);
5:     Calculate the loss of critic $L^{cri}$ by using (33);
6:     Update $\theta_d$ and $\theta_c$ by using $L^{total}$ and $\theta = \theta - \lambda_\theta \widehat{\nabla}_\theta L^{total}$;
7:     Update $\delta$ by using $L^{cri}$ and $\delta = \delta - \lambda_\delta \widehat{\nabla}_\delta L^{cri}(\delta)$;
8: end for
```

update for the critic network is represented through mean-square error as

$$L^{\text{cri}}(\delta) = \left(V_\delta(s_{t+1}) - V^{\pi_\theta}(s_t)\right)^2. \tag{33}$$

C. EP-MUSTO Algorithm

To address problem P1 efficiently, we propose an entropy-enhanced PPO-based multi-UAV-assisted security-aware task optimization algorithm, named EP-MUSTO. The framework of EP-MUSTO is illustrated in Fig. 2. It comprises three primary components: 1) the actor; 2) the critic; and 3) the DT-enabled UAV-DEC environment. With the inclusion of a parameterized HAS, the policy network output is divided into three distinct branches. These branches comprise a discrete actor network π_{θ_d} that identifies the service goal and endpoint, along with two continuous actor networks π_{θ_c} that determine the flight actions. π_{θ_d} acquires discrete actions through random sampling, while π_{θ_c} derives continuous actions from the mean and variance of continuous parameters in a Gaussian distribution. The pseudocode for EP-MUSTO is presented in Algorithm 2. The overall flow of EP-MUSTO is organized into the following four phases.

Initialization phase (lines 1–3). Once the environment is initialized, the system enters a standby state. At the beginning of each exploration, the DTN replicates the state parameters of each device in the environment as a virtual device.

Interaction phase (lines 4–17). The DTN sends the current state s_t to the actor. The policy network creates a group of HAS a_t using the current policies π_{θ_d} and π_{θ_c}. After the UAV cluster executes a_t, the SSC of a_t is calculated. The reward r_t and the status for the next time slot s_{t+1} are obtained from the DTN. Finally, the tuple (s_t, a_t, r_t, s_{t+1}) is stored in the buffer.

Parameter update phase (lines 18–23). During a maximum number of updates, the agent randomly samples a strategy trajectory from the experience pool. Next, a dominance estimate $\widehat{A}(t)$ is generated from the critic. Finally, the two strategy networks are updated independently of the critic network.

Prepreparation phase (line 24). Given that EP-MUSTO is an online learning algorithm, the buffer pool is cleared, making way for the training of new experiences.

V. Performance Evaluation and Analysis

This section outlines the experimental setup and assesses EP-MUSTO's performance in various environments.

A. Experiment Setting

The UAV-DEC operates within a square area measuring 600 m on each side, with UAVs evenly distributed throughout this area. The TUs are generated at random locations on the

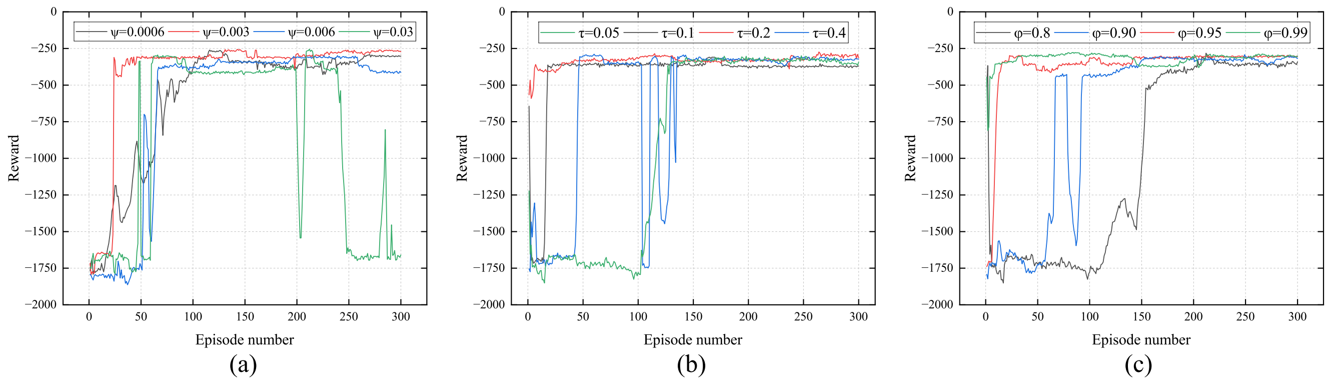

Fig. 3. Effect of different hyperparameters of EP-MUSTO on episode rewards. (a) Learning rate. (b) Clip factor. (c) GAE coefficient.

Algorithm 2 EP-MUSTO

```
Require: Environment information, iteration rounds $e_{\max}$, maximum update length $z_{\max}$, time sequence length $t_{\max}$.
Ensure: Optimum TO strategy $\pi_{\max}$.
1: Randomly initialize the parameters $\theta_d$, $\theta_c$, and $\delta$;
2: for $e \leftarrow 1$ to $e_{\max}$ do
3:     Randomly initialize the DTN;
4:     for $t \leftarrow 1$ to $t_{\max}$ do
5:         Get $s_t$ from the DTN virtual environment;
6:         Perform $SN$ on $s_t$ by using (21);
7:         The discrete action policy receives $s_t$ and outputs $a_t^d = \{U(t), \omega(t)\} \sim \pi_{\theta_d}(\cdot|s_t)$;
8:         The continuous action policy receives $s_t$ and outputs $a_t^c = \{O_u(t), V_u(t)\} \sim \pi_{\theta_c}(\cdot|s_t)$;
9:         The UAV swarm executes the actions $a_t^d$ and $a_t^c$ and updates the position by using Eqs. (2) and (3);
10:        if $L_U(t)! = L_U(t-1)$ then
11:            Calculate $E_j^{fly}(t)$ for each UAV;
12:        end if
13:        Calculate $E_j^{hov}(t)$ of each UAV hovering;
14:        Calculate SCC by using (18);
15:        Obtain $r(s_t, a_t)$ and $s_{t+1}$ by using (23);
16:        Store the tuple $(s_t, a_t, r_t, s_{t+1})$ into the buffer $D$;
17:    end for
18:    for $z \leftarrow 1$ to $z_{\max}$ do
19:        Randomly sample a strategy trace from $D$;
20:        Update $\theta_d$, $\theta_c$, and $\delta$ by using Algorithm 1;
21:        $\theta_d(old) \leftarrow \theta_d$;
22:        $\theta_c(old) \leftarrow \theta_c$
23:    end for
24:    Empty the buffer;
25: end for
```

ground. A cloud server is positioned outside the region. Table I provides the main experimental parameters.

To validate the effectiveness of EP-MUSTO, this study compares it with four other advanced algorithms:

1) *Raw PPO-based UAV-assisted task offloading algorithm (PPOTO):* This algorithm is a version of EP-MUSTO that does not include any additional enhancements.

TABLE I
LIST OF EXPERIMENTAL PARAMETERS

Parameters	Values
Bandwidth (B)	1 MH
Gaussian white noise (σ^2)	–100 dBm
Noise with obstacles (σ^2_{NLoS})	–80 dBm
Computing frequency of TU$_i$ (f_i)	1 GHz
Computing frequency of UAV (f_j)	5 GHz
Computing frequency of cloud (f_c)	10 GHz
Channel gain per unit distance (g_0)	1×10^{-5}
Computational density (s)	1×10^3 cycles/s
Uplink transmit power of TU$_i$ (p_i^{up})	1 W
Uplink transmit power of UAV (p_j^{up})	2 W
Power coefficient of CPU (κ)	1×10^{-28}
Battery capacity of UAV (E_b)	500 KJ

2) *SAC-Enc [30]:* SAC-Enc utilizes a multitask agent-based SAC algorithm to solve the energy minimization problem.
3) *TD3-based UAV-assisted computational offloading algorithm (TD3CO) [31]:* TD3CO employs the TD3 algorithm to minimize processing delay and energy consumption for IoT devices.
4) *DDPG-based UAV-assisted computation offloading algorithm (DDPGCO) [32]:* DDPGCO employs a DDPG-based scheme to solve a delay minimization problem with discrete variables and energy consumption constraints.
5) *LocalOnly algorithm:* This algorithm focuses on keeping all tasks localized to the TU.
6) *Round-robin algorithm (RR):* RR sequentially offloads tasks to multiple UAVs following a predetermined flight path.

B. Performance Analysis

1) Analysis of Hyperparameters: Fig. 3(a) demonstrates the effect of the learning rate ψ on the neural network. Notably, when ψ is set to 0.003 in EP-MUSTO, the reward value quickly and stably converges to the optimal solution. By contrast, when ψ is 0.03, learning stability is compromised, leading to rapid convergence with considerable fluctuations; when ψ is 0.0006, learning capability is hindered, resulting in slow convergence. Thus, ψ is set to 0.003. Fig. 3(b)

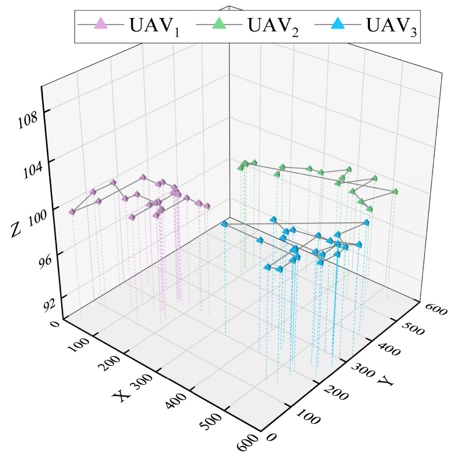

Fig. 4. Flight trajectory diagram of UAV cluster.

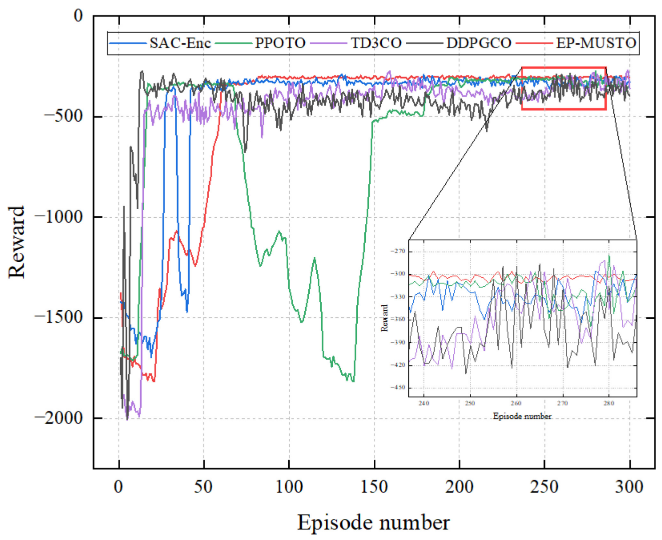

Fig. 5. Overall performance comparison of different algorithms.

illustrates the effect of the clip factor τ in EP-MUSTO. Particularly, when τ equals 0.03, the reward curve has the lowest convergence amplitude and eventually converges to the optimal solution. Conversely, when τ is 0.05, the excessive update of the old and new strategies results in oscillations in convergence; when τ is 0.01 or 0.005 restricts the extent of strategy updates, resulting in slow convergence progress. Therefore, τ is set to 0.03. Fig. 3(c) demonstrates the effect of the GAE coefficient φ. Specifically, when φ equals 0.95, the reward function converges rapidly and stably to the optimal solution. However, at $\varphi = 0.99$, the large decay factor favors future rewards, leading to optimistic convergence near the solution interval; when φ is 0.90 or 0.80, the smaller decay factor bias the focus toward immediate rewards, resulting in instability during the convergence phase. Therefore, φ is set to 0.95.

2) Analysis of UAV Trajectory: Fig. 4 depicts a 3-D scatter diagram that illustrates the flight trajectories of the UAV swarm under an optimal policy. The hovering point of each UAV indicating the provision of TO service during specific time slots. As shown in Fig. 4, the service trajectories of UAV swarm cover nearly the entire area without any overlap, signifying that in the optimal offloading strategy, UAVs can avoid energy waste caused by overlapping trajectories. Moreover, the closed-loop service paths effectively minimize flight distance and conserve UAV power. These findings demonstrate that EP-MUSTO adeptly manages UAV energy-efficient flight paths within the environment through constraints, DTNs, and MDPs.

3) Analysis of Overall Convergence: As illustrated in Fig. 5, among all algorithms, EP-MUSTO achieves the optimal convergence position, demonstrating remarkable accuracy and stability as it steadily converges to 300 in approximately 80 generations. TD3CO and DDPGCO are the fastest converging algorithms; however, they exhibit great oscillations upon reaching convergence. By contrast, EP-MUSTO improves the accuracy of the optimal solution by 14.29% and 22.06% compared with these algorithms, respectively. This enhancement is attributed to EP-MUSTO with online updates, which boosts update stability, along with the incorporation of strategy entropy into the actor network to enhance exploration at extreme points. The SAC-Enc algorithm achieves the second-best convergence accuracy after EP-MUSTO, also leveraging the strategy entropy mechanism and utilizing a multicritic network for value assessment. However, EP-MUSTO tackles the training complexity of the HAS through a parametric design approach, thereby improving the network to understand actions. Experimental results indicate that EP-MUSTO effectively enhances convergence accuracy and stability compared with PPOTO.

4) Analysis of Comparative Experiments: The LocalOnly and Round-robin algorithm are considered to represent the scenarios of fully local computing and polling offloading. Fig. 6(a) depicts the effect of varying TU numbers on the SSC of each algorithm. Among the six algorithms, EP-MUSTO consistently achieves the lowest SSC, followed by the four DRL-based algorithms. This performance advantage is attributed to the ability of DRL to learn effective strategies iteratively through interaction with the DT model. By contrast, the RR and LocalOnly algorithms demonstrate relatively lower performance. The RR algorithm performs better due to its efficient use of resources, distributing tasks among servers in a rotational manner. Fig. 6(b) examines the effect of varying task sizes on the SSC of each algorithm. This experiment lists three task size scenarios: 1) small tasks (1.5–3 MB); 2) medium tasks (3–4.5 MB); and 3) large tasks (4.5–6 MB). Among all algorithms, EP-MUSTO achieves the best SSC values across all task sizes. Compared to SAC-Enc, which also incorporates the MERL, EP-MUSTO's parameterized HAS offers enhanced cognitive ability for exploration. Fig. 6(c) illustrates the effect of varying the number of UAVs on the time delay of each algorithm. Given that energy consumption by UAVs is proportional to their quantity, this experiment prioritizes delay as the performance metric. Among all algorithms, EP-MUSTO delivers optimal results across different UAV cluster sizes. The delay of LocalOnly remains constant because it operates independently of the UAVs. However, the RR algorithm is heavily reliant on the computational power of the UAVs, leading to great performance fluctuations as the number of

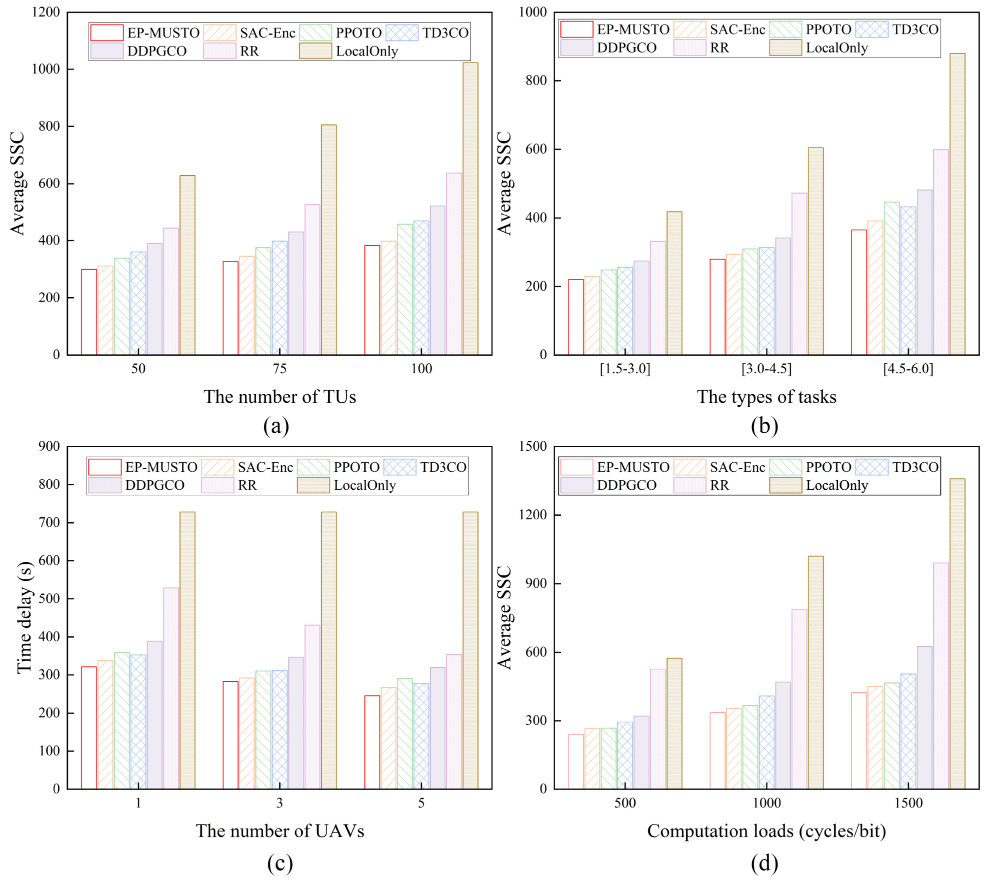

Fig. 6. Result of (a) number of TUs, (b) task size, (c) number of UAVs quantity, and (d) computation loads on evaluation metrics across various algorithms.

UAVs changes. Fig. 6(d) depicts the overall SSC performance of various algorithms under different computational loads. This experiment lists three computational load scenarios: 1) low load (500 times/bit); 2) medium load (1000 times/bit); and 3) high load (1500 times/bit). Task computation efficiency is measured by the number of processor cycles needed to process 1 bit of data. The load conditions of the facilities can be influenced by controlling their computational efficiency. Among all algorithms, EP-MUSTO displayed optimal optimization across all computational load scenarios. Furthermore, during the experiment, both the PPOTO and DDPG algorithms failed to converge normally within the designated duration under high computational loads due to their inefficiency in handling extensive volumes of action data under such circumstances. To tackle these issues, this experiment suggests enhancements to them. Results show that even in high-load environments, EP-MUSTO improves the metrics by 5.56%, 8.42%, 5.03%, and 6.02% compared with the suboptimal algorithm.

VI. Conclusion

This study explores the TO problem while incorporating privacy techniques in a UAV swarm-assisted D-E-C cooperative computing environment. We first construct a model of a cooperative edge computing system supported by multiple UAVs, utilizing a DT virtual network to gather and train realistic device parameters. To tackle effectively the joint minimization problem of SSC and UAV endurance under security and resource constraints, we develop an MDP with an HAS. Global states are collected and virtually trained using DT technology, leading to the proposal of an algorithm called EP-MUSTO. By integrating policy entropy and parameterized action outputs, EP-MUSTO enhances the PPO algorithm to identify optimal solutions, addressing the hybrid action training issue. Experimental results across various environments validate the effectiveness of EP-MUSTO, with improvements of at least 58.6% and 9.43% in SSC compared to other advanced algorithms. Additionally, it achieves an average delay improvement of at least 8.62% across different UAV swarms. Future work will explore the multi-UAV-assisted TO and trajectory optimization problem using MADRL.

Acknowledgment

The authors would like to thank two anonymous reviewers for their suggestions to improve the manuscript.

References

[1] H. Zhou, Z. Wang, H. Zheng, S. He, and M. Dong, “Cost minimization-oriented computation offloading and service caching in mobile cloud-edge computing: An A3C-based approach,” *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1326–1338, May/Jun. 2023.

[2] C. Chen, S. Gong, W. Zhang, Y. Zheng, and Y. C. Kiat, “DRL-based contract incentive for wireless-powered and UAV-assisted backscattering MEC system,” *IEEE Trans. Cloud Comput.*, vol. 12, no. 1, pp. 264–276, Mar. 2024.

[3] Z. Zhou et al., “Secure and latency-aware digital twin assisted resource scheduling for 5G edge computing-empowered distribution grids,” *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4933–4943, Jul. 2022.

[4] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, “Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing,” *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, Jan. 2022.

[5] T. Zhao, F. Li, and L. He, “Secure video offloading in MEC-enabled IIoT networks: A multicell federated deep reinforcement learning approach,” *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 1618–1629, Feb. 2024.

[6] Z. Tong, J. Wang, J. Mei, K. Li, and K. Li, “FedTO: Mobile-aware task offloading in multi-base station collaborative MEC,” *IEEE Trans. Veh. Technol.*, vol. 73, no. 3, pp. 4352–4365, Mar. 2024.

[7] J. Zhang, Y. Wu, G. Min, and K. Li, “Neural network-based game theory for scalable offloading in vehicular edge computing: A transfer learning approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 7, pp. 7431–7444, Jul. 2024.

[8] K. Zhang, J. Cao, and Y. Zhang, “Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks,” *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 1405–1413, Feb. 2022.

[9] J. Xu, A. Xu, L. Chen, Y. Chen, X. Liang, and B. Ai, “Deep reinforcement learning for RIS-aided secure mobile edge computing in Industrial Internet of Things,” *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 2455–2464, Feb. 2024.

[10] L. Li, Q. Qiu, Z. Xiao, Q. Lin, J. Gu, and Z. Ming, “A two-stage hybrid multi-objective optimization evolutionary algorithm for computing offloading in sustainable edge computing,” *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 735–746, Feb. 2024.

[11] H. Lin, L. Yang, H. Guo, and J. Cao, “Decentralized task offloading in edge computing: An offline-to-online reinforcement learning approach,” *IEEE Trans. Comput.*, vol. 73, no. 6, pp. 1603–1615, Jun. 2024.

[12] W. Li, S. Li, H. Shi, W. Yan, and Y. Zhou, “UAV-enabled fair offloading for MEC networks: A DRL approach based on actor-critic parallel architecture,” *Appl. Intell.*, vol. 54, pp. 3529–3546, Feb. 2024.

[13] Y. Zhou et al., “Secure multi-layer MEC systems with UAV-enabled reconfigurable intelligent surface against full-duplex eavesdropper,” *IEEE Trans. Commun.*, vol. 72, no. 3, pp. 1565–1577, Mar. 2024.

[14] J. Yan, X. Zhao, and Z. Li, “Deep-reinforcement-learning-based computation offloading in UAV-assisted vehicular edge computing networks,” *IEEE Internet Things J.*, vol. 11, no. 11, pp. 19882–19897, Jun. 2024.

[15] B. Xu, Z. Kuang, J. Gao, L. Zhao, and C. Wu, “Joint offloading decision and trajectory design for UAV-enabled edge computing with task dependency,” *IEEE Trans. Wireless Commun.*, vol. 22, no. 8, pp. 5043–5055, Aug. 2023.

[16] L. Zhang, R. Tan, Y. Zhang, J. Peng, J. Liu, and K. Li, “UAV-assisted dependency-aware computation offloading in device-edge-cloud collaborative computing based on improved actor-critic DRL,” *J. Syst. Archit.*, vol. 154, Sep. 2024, Art. no. 103215.

[17] J. Shi, C. Li, Y. Guan, P. Cong, and J. Li, “Multi-UAV-assisted computation offloading in DT-based networks: A distributed deep reinforcement learning approach,” *Comput. Commun.*, vol. 210, pp. 217–228, Oct. 2023.

[18] P. Consul, I. Budhiraja, D. Garg, N. Kumar, R. Singh, and A. S. Almogren, “A hybrid task offloading and resource allocation approach for digital twin-empowered UAV-assisted MEC network using federated reinforcement learning for future wireless network,” *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 3120–3130, Feb. 2024.

[19] H. Hao, C. Xu, W. Zhang, S. Yang, and G.-M. Muntean, “Joint task offloading, resource allocation, and trajectory design for multi-UAV cooperative edge computing with task priority,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 9, pp. 8649–8663, Sep. 2024.

[20] H. Yu, S. Leng, and F. Wu, “Joint cooperative computation offloading and trajectory optimization in heterogeneous UAV-swarm-enabled aerial edge computing networks,” *IEEE Internet Things J.*, vol. 11, no. 10, pp. 17700–17711, May 2024.

[21] X. Zhou et al., “Joint UAV trajectory and communication design with heterogeneous multi-agent reinforcement learning,” *Sci. China Inf. Sci.*, vol. 67, Feb. 2024, Art. no. 132302.

[22] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, “Deep reinforcement learning for stochastic computation offloading in digital twin networks,” *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4968–4977, Jul. 2021.

[23] L. Zhang, R. Tan, M. Ai, H. Xiang, C. Peng, and Z. Zeng, “DSUTO: Differential rate SAC-based UAV-assisted task offloading algorithm in collaborative edge computing,” in *Proc. IEEE 29th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2023, pp. 2328–2335.

[24] G. Wu, X. Chen, Z. Gao, H. Zhang, S. Yu, and S. Shen, “Privacy-preserving offloading scheme in multi-access mobile edge computing based on MADRL,” *J. Parallel Distrib. Comput.*, vol. 183, Jan. 2024, Art. no. 104775.

[25] X. Li, Y. Qin, J. Huo, and W. Huangfu, “Computation offloading and trajectory planning of multi-UAV-enabled MEC: A knowledge-assisted Multiagent reinforcement learning approach,” *IEEE Trans. Veh. Technol.*, vol. 73, no. 5, pp. 7077–7088, May 2024.

[26] Z. Tong, B. Liu, J. Mei, J. Wang, X. Peng, and K. Li, “Data security aware and effective task offloading strategy in mobile edge computing,” *J. Grid Comput.*, vol. 21, p. 41, Jul. 2023.

[27] W. Masson, P. Ranchod, and G. Konidaris, “Reinforcement learning with parameterized actions,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, 2016, pp. 1934–1940.

[28] G. Wu, Z. Xu, H. Zhang, S. Shen, and S. Yu, “Multi-agent DRL for joint completion delay and energy consumption with queuing theory in MEC-based IIoT,” *J. Parallel Distrib. Comput.*, vol. 176, pp. 80–94, Jun. 2023.

[29] T. Wang, Y. Deng, Z. Yang, Y. Wang, and H. Cai, “Parameterized deep reinforcement learning with hybrid action space for edge task offloading,” *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10754–10767, Mar. 2024.

[30] S. Zhang, H. Jin, and P. Guo, “IRS-assisted energy efficient communication for UAV mobile edge computing,” *Comput. Netw.*, vol. 246, Jun. 2024, Art. no. 110387.

[31] N. Fatima, P. Saxena, and G. Giambene, “Deep reinforcement learning based computation offloading for xURLLC services with UAV-assisted IoT-based multi-access edge computing system,” *Wireless Netw.*, vol. 30, pp. 7275–7291, Dec. 2024.

[32] F. Xu, S. Wang, W. Su, and L. Zhang, “Joint optimization task offloading and trajectory control for unmanned-aerial-vehicle-assisted mobile edge computing,” *Comput. Electr. Eng.*, vol. 111, Oct. 2023, Art. no. 108916.

Longxin Zhang (Member, IEEE) received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2015.

He is currently an Associate Professor of computer science with the Hunan University of Technology, Zhuzhou. He was also a Visiting Scholar with the University of Florida from 2019 to 2020. He has published more than 50 technique papers in international journals and conferences, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, *Information Sciences*, *Journal of Systems Architecture*, IEEE SENSORS JOURNAL, *Journal of Grid Computing*, and ICA3PP. His major research interests include distributed system reliability, parallel algorithms, edge computing, cloud computing, and deep learning.

Dr. Zhang is the reviewer of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *ACM Transactions on Intelligent Systems and Technology*, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE INTERNET OF THINGS, INS, and ASC. He is a Senior Member of the China Computer Federation.

Runti Tan (Student Member, IEEE) received the B.S. degree from Hunan University of Technology, Zhuzhou, China, in 2022, where he is currently pursuing the M.S. degree in computer system architecture.

His research interests include UAV control, edge computing, and reinforcement learning for task offloading.

Buqing Cao (Member, IEEE) received the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 2011.

He is a Postdoctoral Fellow with the Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, NC, USA, in 2016, and currently a Professor with the School of Computer Science, Hunan University of Technology, Zhuzhou, China. His current interests include service computing, software engineering, and machine learning.

Dr. Cao has published more than 100 papers in conferences and journals, such as IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, ICWS, and SEKE. He is on the editorial boards of several journals and has served as a program committee member for multiple international conferences. He is an Outstanding Member of CCF.

Lihua Ai received the Ph.D. degree in computer science and technology from Tongji University, Shanghai, China, in 2023.

She is currently a Lecturer with the School of Computer Science, Hunan University of Technology, Zhuzhou, China. Her current research interests include mobile edge computing, artificial intelligence, and wireless networking.

Keli Li (Senior Member, IEEE) received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2003.

From 2004 to 2005, he was a Visiting Scholar with the University of Illinois Urbana–Champaign, Champaign, IL, USA. He is currently a Full Professor of computer science and technology with Hunan University, Changsha, China, and the deputy director of National Supercomputing Center, Changsha. He has authored or co-authored more than 300 papers in international conferences and journals, including the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE TRANSACTIONS ON SIGNAL PROCESSING. His research interests include parallel computing, cloud computing, and Big Data computing.

Prof. Li is currently on the editorial boards of the IEEE TRANSACTIONS ON COMPUTERS and *International Journal of Pattern Recognition and Artificial Intelligence*. He is a CCF Fellow.

Keqin Li (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of computer science. He has published over 320 journal articles, book chapters, and refereed conference papers. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things, and cyber–physical systems.

Dr. Li has received several best paper awards. He is currently or has served on the editorial boards of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and *Journal of Parallel and Distributed Computing*.