UAV-Assisted Task Offloading in Edge Computing

Junna Zhang, Guoxian Zhang, Xinxin Wang, Xiaoyan Zhao, Peiyan Yuan, and Hu Jin

***Abstract*—Task offloading can meet users' demands for the latency and energy consumption by offloading tasks from resource-constrained IoT devices to relatively resource-rich edge servers. Traditional task offloading usually makes use of fixed base stations or servers as edge servers. This would lead to limited range of services and increased costs due to large-scale deployment of edge servers. Therefore, deploying unmanned aerial vehicles (UAVs) as mobile edge servers for task offloading in complex terrains (e.g., forest, desert, etc.) is a worthwhile research problem. To this end, this paper proposes a UAV-assisted task offloading mechanism. The mechanism aims to minimize the weighted sum of latency and energy consumption through jointly optimizing resource allocation, offloading decision, and UAV trajectory. We first transform the non-convex optimization problem into convex optimization subproblems to obtain the optimal resource allocation. Second, we use an improved particle swarm optimization algorithm to find the optimal offloading decision. Finally, we present the deep determination policy gradient algorithm to optimize the UAV trajectory which is a kind of deep reinforcement learning algorithm. Through simulation experiments, we show that the proposed mechanism can efficiently reduce the weighted sum of latency and energy consumption.**

***Index Terms*—Edge computing, task offloading, UAV trajectory, deep determination policy gradient algorithm, resource allocation.**

I. Introduction

The International Data Corporation predicts that by 2025, there will be 41.6 billion Internet of Things (IoT) devices worldwide, capable of generating 79.4 zettabytes of data [1]. The rapid development of 5G has driven the emergence of latency-sensitive and compute-intensive applications, such as virtual reality and augmented reality [2]–[7]. Nonetheless, executing these applications on IoT devices is challenging due to their limited computing power, storage resource and battery life. Therefore, a portion of computing tasks can be offloaded from IoT devices to edge servers (e.g., base stations or access points) to perform remote computing [8].

However, most of traditional task offloading strategies use fixed base stations or servers as edge servers [9]. These strategies have some major drawbacks, such as fixed and limited-service coverage, channel attenuation caused by long communication distances, and high costs due to the large-scale deployment of fixed edge servers [2]. How to deploy edge servers in dynamic scenarios or in complex terrain, and further shorten communication distances, is a problem that needs to be considered in the new generation of edge network architecture. To address these issues, a promising solution is to use unmanned aerial vehicles (UAVs) as mobile edge servers for task offloading. UAVs have flexible mobility so that they can be deployed and processed at any time to collect tasks offloaded from multiple IoT devices [10]–[13]. This mobility can avoid problems such as information transmission latency and can further improve task processing efficiency.

Our research motivation is to explore an efficient task offloading mechanism for UAVs and IoT devices both of which are in motion. The proposed mechanism can improve the service coverage of UAVs to reduce latency and energy consumption, while also facilitating the future development of various potential applications with different requirements. The situation where both UAVs and IoT devices are in motion is common in real-world scenarios. For example, in natural resource survey and monitoring, there is a large number of IoT devices that need to detect and monitor the state of the local natural environment and resource condition, including data related to meteorology, hydrology, geography, and environment. These IoT devices usually need to execute tasks in remote areas or in areas with limited human access (e.g., forest, desert, etc.). The tasks generated by these IoT devices may have high latency requirements due to the instability and unpredictability of the environment. To improve efficiency and accuracy, these tasks can be offloaded to UAVs for processing. By taking advantage of UAV's mobility and high speed, a large amount of high-precision natural resource data can be quickly obtained in remote or hard-to-access environments, increasing the comprehensiveness and accuracy of the data, and providing better support for geological exploration, environmental monitoring, and other fields.

In this paper, our goal is to address resource allocation, offloading decision, and trajectory optimization problem under the constraints of limited transmit power of IoT devices and computing frequency of the UAV server. Our research differs from existing studies in jointly considering the following two main aspects [14]–[21]: 1) Most of the existing works, such as [19], [21], only consider single mobility, i.e., the UAV is in a mobile state while IoT devices are in a stationary state. In fact, IoT devices in real-word applications are usually in a mobile state. Therefore, we consider a more realistic scenario where both the UAV and IoT devices are in motion. This scenario is better suited to real-world applications such as natural resource survey and monitoring and wilderness rescue. It is notable that we jointly optimize the resource allocation, offloading decision, and UAV trajectory in a dual-mobility scenario, where the mobility of IoT devices greatly increases the complexity of task offloading strategy. 2) Most researches only consider

Junna Zhang, Guoxian Zhang, Xinxin Wang, Xiaoyan Zhao and Peiyan Yuan are with the School of Computer and Information Engineering, Henan Normal University, and also with Key Laboratory of Artificial Intelligence and Personalized Learning in Education of Henan Province, Xinxiang, Henan, China. E-mail: jnzhang@htu.edu.cn; 2108283041@stu.htu.edu.cn; 2108283081@stu.htu.edu.cn; zhaoxiaoyan@htu.edu.cn; peiyan@htu.cn

Hu Jin is with the School of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China and is also with the School of Electrical Engineering, Hanyang University, Ansan, 15588, South Korea. E-mail: hjin@hanyang.ac.kr.

(*Corresponding author: Peiyan Yuan; Hu Jin.)*

energy consumption or latency as the primary performance metric [17], [20]. In fact, the simultaneous optimization of latency and energy consumption is crucial for the experience of the users and the performance of the applications. Therefore, we explore an optimization objective that integrates latency and energy consumption into a single metric using a weighted approach, as these are two critical performance metrics in practical applications. Simultaneously optimizing latency and energy consumption offers several benefits: *a*) it greatly improves user experience and effectively prolongs battery life of UAVs; *b*) it can better support a wider range of use cases, such as natural resource survey and monitoring; *c*) it enables faster and more precise completion of tasks, while consuming energy more efficiently and conservatively.

As discussed above, it is challenging but necessary to find a suitable task offloading decision under the dual-mobility scenarios and constraints of limited resources of UAV server and IoT devices. To this end, we propose a UAV-assisted task offloading mechanism (UTOM) to minimize the weighted sum of latency and energy consumption (defined as cost) spent on executing tasks. The main contributions of this paper are summarized as follows:

- We investigate cost minimization by jointly optimizing resource allocation, offloading decision, and UAV trajectory under the constraint of limited resource of a UAV server and IoT devices. We first prove that the optimization problem is non-convexity and NP-hard. In order to make the optimization problem be tractable, we decompose the problem into three subproblems: 1) Resource allocation; 2) Offloading decision; 3) UAV trajectory optimization.
- *Resource Allocation*: Despite the non-convexity of the joint optimization problem, we identify the hidden convexity of the resource allocation problem when the offloading decision and the UAV location are fixed. By exploiting this property, the optimal solution to the resource allocation problem is obtained by using the Lagrange multiplier method and Karush-Kuhn-Tucker (KKT) condition. Consequently, the optimal resource allocation can be obtained in a computationally efficient way when the offloading decision and the UAV location are given.
- *Offloading Decision*: In order to obtain the optimal offloading decision for a given UAV location, we further propose an improved particle swarm optimization (IPSO) algorithm. In searching for the optimal offloading decision with IPSO, we need to obtain the optimal offloading decision and calculate the corresponding cost at each generation of IPSO. While the computing complexity increases proportionally with the population size and the number of generations, it is still computationally implementable as the optimal resource allocation is the solution of a convex optimization problem.
- *UAV Trajectory*: We finally propose a deep determination policy gradient (DDPG) algorithm, which is a kind of deep reinforcement learning algorithm, to find the optimal UAV flight trajectory. DDPG enables us to predict the flight trajectory of the UAV for next time slot based on the optimal offloading decision made in the current time slot. Experimental results show that the proposed mechanism can significantly reduce the cost compared with the state-of-the-art methods.

The rest of paper is organized as follows. Section II reviews related work. Section III presents the system model and problem formulation. Section IV introduces the proposed mechanism. Section V shows the experiment results, and Section VI concludes the paper.

II. Related Work

In recent years, UAV-assisted task offloading has received widespread attention as one of the key technologies of edge computing. To clearly present the contributions of this paper, we summarize the related researches on task offloading from the following three related aspects: fixed-position offloading of IoT devices, single-objective optimization, and UAV trajectory optimization.

A. Fixed position offloading

With increasing user demands for lower latency and energy consumption, offloading tasks from fixed-position IoT devices to UAVs has garnered significant attention, resulting in a large volume of studies [18], [19], [21]–[27]. Xu *et al.* [22] proposed an edge computing system that maximizes computational efficiency by jointly optimizing communication and computational resources, computation requirements, and UAV flight trajectory. Xiong *et al.* [18] designed an online optimization algorithm to minimize long-term network operation costs through joint task assignment, local computing resource allocation, association control, and UAV computing resource allocation. Zeng *et al.* [23] introduced a path discretization algorithm to transform the problem into a discretized equivalent, aiming to minimize total UAV energy consumption by jointly optimizing UAV trajectory, communication time allocation among ground nodes, and total mission completion time. He *et al.* [27] proposed a 3D multi-UAV mobile edge computing system and derived the optimal offloading and UAV selection scheme to minimize energy consumption. While these studies primarily address task offloading for IoT devices in fixed locations, real-world scenarios increasingly involve mobile IoT devices, complicating task offloading compared to fixed-device scenarios.

B. Single-objective optimization

Recently, research on single-objective optimization, such as latency or energy consumption, for UAV-assisted task offloading has continuously increased [8], [14], [20], [23], [28]. Yang *et al.* [20] proposed a strategy to minimize service cost by jointly optimizing channel allocation and offloading strategy. Tun *et al.* [8] introduced a collaborative multi-UAV-assisted edge computing system that reduces latency by dividing the task offloading problem into subproblems and solving them using the Lagrangian relaxation and alternating direction method of multipliers. Ji *et al.* [29] developed an air-ground mobile edge offloading model that minimizes latency under constraints of security, energy, and coverage using a Markov decision process. Liu *et al.* [28] developed an iterative algorithm to minimize system energy consumption by jointly optimizing UAV beamforming vectors, CPU frequency, and transmit power of IoT devices. These studies focus on single-objective optimization for task offloading to UAVs. However, balancing latency and energy consumption is crucial in practical applications. Therefore, the task offloading process typically involves co-optimizing latency and energy consumption, leading to a more complex resource competition problem within single-objective optimization.

C. UAV trajectory optimization

Combining edge computing with UAVs can significantly reduce task completion latency due to the flexibility and ease of UAV deployment [8]. As a result, optimizing UAV flight trajectories has garnered significant attention [10], [13], [30]–[32]. For instance, Diao *et al.* [10] developed an alternating optimization algorithm to minimize energy consumption by optimizing offloading parameters and UAV flight trajectory. Elgendy *et al.* [13] introduced a deep reinforcement learning algorithm to minimize system cost through joint task offloading, load balancing, and resource allocation. Zhuang *et al.* [32] used a deep deterministic policy gradient algorithm to optimize task offloading, minimizing the weighted sum of latency and energy consumption under limited UAV computational and energy resources. While these studies focus on trajectory optimization, practical applications require comprehensive UAV server resource management to maximize task execution efficiency. Thus, UAV trajectory optimization must adopt flexible strategies to ensure efficient resource utilization, enhancing task execution reliability and efficiency.

While there have been plenty of previous studies that address UAV-assisted task offloading, they have not jointly investigated the mobility of IoT devices, the co-optimization of latency and energy consumption, and the limited computing resources of UAV servers. Considering these factors, we propose a novel solution that combines an IPSO algorithm with a DDPG algorithm to minimize the cost of the edge computing system. To the best of our knowledge, few studies address scenarios where both IoT devices and UAVs are moving simultaneously while solving the multi-domain resource optimization problem. It is worth mentioning that although a few recent studies [10], [13], [29], [32] have started to consider the dual mobility of IoT devices and UAVs, our proposed UTOM outperforms these approaches in scenarios where both latency and energy consumption are critically important as will be discussed in Section V.

III. System Model and Problem Formulation

A. Network model

Our network model shown in Fig. 1 consists of two layers: the IoT device layer and the edge layer. The edge layer includes a UAV that is equipped with an edge server in the air as an aerial base station to handle and collect the latency-sensitive tasks offloaded from the IoT device layer. The IoT device layer consists of m IoT devices, each with a computing

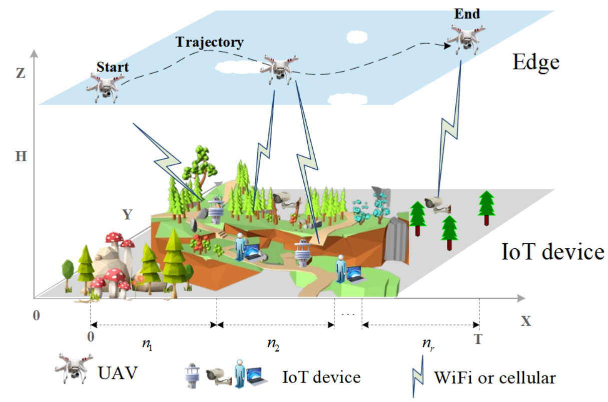


Fig. 1. Network model

processor capable of computing simple task locally. The IoT devices can also transmit partial task data wirelessly to the UAV for task offloading. The total time duration T is divided into N equal-length time slots of size δ.

Keeping the flight altitude of H meters, the UAV flies from left to right for communications with the m IoT devices whose latency-sensitive tasks may be offloaded to the UAV for processing. The flight trajectory of the UAV in time slot $n \in [1, N]$ can be determined by its flight speed and angle, represented as $v(n) \in [30 \text{ m/s}, 50 \text{ m/s}]$ and $\theta(n) \in [0, 2\pi]$, respectively. The movement trajectory of each IoT device in time slot n can be determined by its movement speed and angle, represented as $v_{iot}(n) \in [0, 5 \text{ m/s}]$ and $\theta_{iot}(n) \in [0, 2\pi]$, respectively.

Each IoT device generates a random task for calculation, and we denote the task data size of the u-th IoT device by L_u. The quality of communication link between the UAV and each IoT device depends on their distance, and to determine the distance, a 3D Cartesian coordinate system is constructed to represent the location of the UAV and each IoT device. For the u-th IoT device, the coordinate at the n-th time slot is denoted by $Q_{user,u}(n) = \{x_{u,n}, y_{u,n}, 0\}$, where $x_{u,n}$ and $y_{u,n}$ denote the x-coordinate and y-coordinate, respectively. For the UAV, the location of the UAV at the n-th time slot is represented by $Q_{uav}(n) = \{X_n, Y_n, H\}$, where X_n and Y_n respectively represent the x-coordinate and y-coordinate, and H denotes the altitude of the UAV.

B. Problem Formulation

In general, some tasks generated by IoT devices have high latency requirement, while others are more concerned with energy consumption of the IoT devices and the UAV. Consequently, our objective is to minimize the cost which balances the latency requirement and the energy consumption by introducing a weight variable $\omega \in [0, 1]$ similar to [33] while optimizing the UAV trajectory. In particular, we define the cost function as follows:

$$\mathbb{C} = (1 - \omega)\mathcal{T} + \omega E \tag{1}$$

where a larger ω means that the proposed mechanism puts more emphasis on the energy consumption requirement E

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488210

in the total flight time N of the UAV, while a smaller ω emphasizes that the proposed mechanism puts more emphasis on the latency $\mathcal{T}$.

Our objective is to minimize the cost occurred in the total duration of N time slots through optimizing the resource allocation, task offloading decision and the UAV trajectory. For the resource constraint, we consider the computing frequency of the UAV and the transmit power of the IoT devices. In particular, we set the following optimization problem:

$$\begin{aligned}
&P1: \min_{\Phi, X, \Theta} \mathbb{C} \\
&s.t.\ C_1: \sum_{u=1}^{m} X_u^{\text{uav}}(n) \leq 1, \text{ for } n \in [1, N] \\
&\quad C_2: \text{X}_u^{\text{uav}}(n) \in \{0,1\}, \forall u \in [1,m], n \in [1,N] \\
&\quad C_3: 0 \leq \lambda_u(n) \leq 1, \forall u \in [1,m], n \in [1,N] \\
&\quad C_4: \text{Q}_{user,u}, Q_{uav} \leq \{X, Y\}, \forall u \in [1,m], n \in [1,N] \\
&\quad C_5: v_{\text{min}} \leq ||v(n)|| \leq v_{\text{max}} \\
&\quad C_6: p_{\text{min}} \leq p_u^{\text{up}} \leq p_{\text{max}} \\
&\quad C_7: f_{\text{min}} \leq f_u^{\text{uav}} \leq f_{\text{max}} \\
&\quad C_8: X_u^{\text{uav}}(n)(t_u^{\text{trans}}(n) + t_u^{\text{uav}}(n)) \leq \delta\,, \text{for}\, n \in [1,, N]
\end{aligned} \tag{2}$$

where the symbols are described in Table I. In particular, $\Phi = \{p_u^{\text{up}}, f_u^{\text{uav}}, \forall u \in [1,m]\}$, where p_u^{up} and f_u^{uav} represent the transmit power of the IoT device u and the CPU frequency of the UAV. $X = \{X_u^{\text{uav}}(n), \forall n \in [1,N]\}$, where $X_u^{\text{uav}}(n)$ denotes the offloading decision: If the task of IoT device u is offloaded to the UAV in time slot n, $X_u^{\text{uav}}(n) = 1$; otherwise, $X_u^{\text{uav}}(n) = 0$. $\Theta = \{v(n), \theta_{\text{uav}}(n)\}$, where $v(n)$ and $\theta_{\text{uav}}(n) \in [0, 2\pi]$ represent the speed and flight angle of the UAV.

The following describes each constraint of the optimization problem $P1$: C_1 shows that only one IoT device can be offloaded to the UAV to execute its task in each time slot. C_2 represents that the offloading decision of each IoT device can be 0 or 1. C_3 denotes the range of the ratio of latency sensitive task offloaded from the IoT device u to the UAV. C_4 indicates the area within which both IoT devices and the UAV move. C_5 denotes the constraint of the UAV flight speed. C_6 shows the uplink transmit power constraint of each IoT device. C_7 represents the CPU frequency constraint of the UAV. C_8 indicates that the sum of the transmission time of task offloading to the UAV and the task execution time at the UAV shall be completed within one slot time duration.

The optimization problem $P1$ involves jointly optimizing resource allocation, offloading decision, and UAV trajectory. Given the resource allocation and the UAV location, the task offloading decision problem can be transformed into the 0-1 knapsack problem [34]. In addition, the UAV trajectory optimization is equivalent to the Traveling Salesman Problem [14], which is NP-hard. Thus, it follows that the problem $P1$ is NP-hard problem. Next, we detail the mathematical representation of latency $\mathcal{T}$ and energy consumption E.

1) Latency: Latency consists of communication latency and computing latency. The *communication latency* includes uplink transmission latency and downlink transmission latency.

TABLE I
MAIN SYMBOLS

Symbols	**Descriptions**
m	The number of IoT devices
N	The number of time slots
$X_u^{\text{uav}}(n)$	Whether the IoT device u is offloaded to the UAV in the n-th time slot
B_0	The channel bandwidth
σ^2	The Gaussian noise power
p_u^{up}	The transmit power of IoT device u
$\lambda_u(n)$	Ratio at which IoT device u offloads the size of the generated task data to the UAV in the n-th time slot
$L_u(n)$	The task data size generated by IoT device u
$h_{up}(n)$	The channel fading coefficient
h_0	The received power at the reference distance (i.e., d = 1m) between the transmitter and receiver
$d_{Uu}(n)$	The Euclidean distance between UAV and IoT device u
$R_u^{\text{up}}(n)$	The uplink transmission rate (bit/s) from the IoT device u to the UAV in n-th time slot
f_u^{uav}	CPU frequency of UAV
f_u^{user}	CPU frequency of IoT device u
t_u^{loc}	The time needed for executing the task generated by IoT device u locally only
t_u^{uav}	The computation time at IoT device u
t_u^{iot}	The local execution time of part of the task data size for IoT device u
e_u^{loc}	The energy consumption of the task generated by IoT device u that is executed locally only
e_u^{uav}	The energy consumption for the task partially offloaded to the UAV
e_u^{iot}	The energy consumption of the task data size portion generated by IoT device u to be executed locally
e_u^{trans}	The transmission energy consumption
k_u	The effective switched capacitance of IoT device u
k_{uav}	The effective switched capacitance of UAV
$Q_{user,u}(n)$	The coordinates of the IoT device u in the n-th time slot
$Q_{uav}(n)$	The location of UAV in the n-th time slot
$\mathbb{C}$	The weighted sum of latency and energy consumption
ω	The weight for balancing latency and energy consumption

The downlink transmission latency usually occurs due to the transmission of the execution results back to the IoT devices. It is generally ignored because of the small amount of data of the execution results [35], [36]. The *computing latency*, on the other hand, includes the time to execute the task at both the IoT devices and the UAV.

The communication latency depends on the communication link between the IoT devices and the UAV. According to [37], the spectrum efficiency from IoT device u to the UAV in the n-th time slot, Γ_u, can be represented as:

$$\Gamma_u = \log_2\left(1 + \frac{p_u^{up} h_{up}(n)}{\sigma^2}\right), \tag{3}$$

where p_u^{up} represents the uplink transmit power of IoT device u, $h_{up}(n)$ represents the channel gain between IoT device u and the UAV in the n-th time slot, and σ^2 is the Gaussian noise power.

In modeling the channel gain $h_{up}(n)$, we account for diverse channel scenarios by considering two models: 1) an ideal channel model where Line-of-Sight (LoS) is dominant, as in [38]; 2) a complex channel model that combines both LoS and Non-LoS (NLoS) components, as in [8], [39], [40].

- *Ideal LoS channel model [38]*: As the UAV flies at much higher than IoT devices, the line-of-sight (LoS) channel of the UAV communication link is more pronounced than

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488210

other channel losses (e.g., small-scale fading). The free-space path loss model [38] is utilized for the evaluation, where the channel gain $h_{up}(n)$ can be expressed as:

$$h_{up}(n) = \frac{h_0}{H^2 + \|Q_{uav}(n) - Q_{user}(n)\|^2}, \quad (4)$$

where h_0 denotes the received power at the reference distance (i.e., d = 1m).

- *Combined LoS and NLoS channel model [8], [39], [40]*: According to [39], the channel gain $h_{up}(n)$ is given by:

$$h_{up}(n) = 10^{-\mathrm{PL}_u/10}, \forall u \in [1, m], \quad (5)$$

where PL_u is the path loss between IoT device *u* and the UAV. This PL_u consists of two components: LoS path loss ($\mathrm{PL}_u^{\mathrm{LoS}}$) and non-Los path loss ($\mathrm{PL}_u^{\mathrm{NLoS}}$), which are expressed as follows [8]:

$$\mathrm{PL}_u^{\mathrm{LoS}} = 2a \log\left(\frac{4\pi d_u^U F_c}{c}\right) + L_{\mathrm{LoS}}, \quad (6)$$

$$\mathrm{PL}_u^{\mathrm{NLoS}} = 2a \log\left(\frac{4\pi d_u^U F_c}{c}\right) + L_{\mathrm{NLoS}}, \quad (7)$$

where $a \geq 2$ represents the path loss exponent, F_c is carrier frequency, c is the speed of light, and L_{LoS} and L_{NLoS} are the average additional losses for the LOS and NLOS links, respectively. Here, d_u^U is the distance which can be expressed as:

$$d_u^U = \sqrt{(X_n - x_{u,n})^2 + (Y_n - y_{u,n})^2 + H}. \quad (8)$$

According to [40], the probability of a LOS connection between IoT device *u* and the UAV is given by:

$$\mathcal{P}_u^{\mathrm{LoS}} = \frac{1}{1 + C \exp[D(\frac{180}{\pi}\tan^{-1}\frac{H}{d_u^U} - C)]}, \quad (9)$$

where H denotes the flying altitude of the UAV, and C and D are constants. Consequently, the probability of a NLOS connection can be given by:

$$\mathcal{P}_u^{\mathrm{NLoS}} = 1 - \mathcal{P}_u^{\mathrm{LoS}}. \quad (10)$$

Consequently, the total path loss between IoT device *u* and the UAV is given by:

$$\mathrm{PL}_u = \mathrm{PL}_u^{\mathrm{LoS}} \cdot \mathcal{P}_u^{\mathrm{LoS}} + \mathrm{PL}_u^{\mathrm{NLoS}} \cdot \mathcal{P}_u^{\mathrm{NLoS}}. \quad (11)$$

Finally, the uplink transmission data rate (bit/s) from IoT device *u* to the UAV in n-th time slot, $R_u^{up}(n)$, can be obtained as:

$$R_{up}(n) = B_0 \Gamma_u, \forall u \in [1, m], \quad (12)$$

where B_0 denotes the channel bandwidth.

Suppose that IoT device *u* has $L_u(n)$ bits of latency-sensitive task in the *n*-th time slot, which can be computed locally or offloaded to the UAV for computation. Let $\lambda_u(n)$ denote the ratio of the latency-sensitive task offloaded from IoT device *u* to the UAV in the *n*-th time slot. Then, the uplink transmission latency for IoT device *u* to be offloaded to the UAV can be expressed as:

$$t_u^{\mathrm{trans}} = \frac{\lambda_u(n) L_u(n)}{R_{up}(n)}. \quad (13)$$

The computation latency can be categorized into two scenarios: one is that the whole task is computed locally at the IoT device, and another one is that partial latency-sensitive task is offloaded to the UAV for computation, while the remaining task is computed locally. In the following, we explain the computational latency for the local computation only and the computational latency for partial offloading, respectively.

a) Latency of the whole task computed locally: The time when the task generated by IoT device *u* is computed locally only, denoted as t_u^{loc}, can be expressed as:

$$t_u^{\mathrm{loc}} = \frac{(1 - X_u^{\mathrm{uav}}(n)) L_u(n)}{f_u^{\mathrm{user}}}, \quad (14)$$

where $X_u^{\mathrm{uav}}(n)$ is the offloading decision for the IoT device *u* in the *n*-th time slot and f_u^{user} represents the CPU frequency of IoT device *u*.

b) Computation latency with partial offloading: In the case of partial offloading, we need to consider the size of the task data $\lambda_u(n)L_u(n)$ that is offloaded from the IoT device *u* to the UAV, and the size of the task data $(1 - \lambda_u(n))L_u(n)$ that is partially used for local computing at the IoT device.

For the task offloaded to the UAV from the IoT device *u*, the computation time at the UAV can be obtained as:

$$t_u^{\mathrm{uav}} = \frac{\lambda_u(n) X_u^{\mathrm{uav}}(n) L_u(n)}{f_u^{\mathrm{uav}}}, \quad (15)$$

where f_u^{uav} represents the computing capacity of the UAV. Meanwhile, when allocating the computing resource of the UAV to its associated IoT devices, the following constraint must be satisfied:

$$\sum_{u=1}^{m} X_u^{\mathrm{uav}}(n) f_u^{\mathrm{uav}} \leq f_{\max}, \quad (16)$$

where $f_{\max}$ is the maximum computing capacity available at the UAV. The local computation time of the remaining partial latency-sensitive task at the IoT device *u*, denoted as t_u^{iot}, can be expressed as:

$$t_u^{\mathrm{iot}} = \frac{(1 - \lambda_u(n)) X_u^{\mathrm{uav}}(n) L_u(n)}{f_u^{\mathrm{user}}}. \quad (17)$$

For the IoT device *u*, the computation latency with partial offloading can be expressed as follows:

$$t_u^{\mathrm{off}}(n) = \max\left(t_u^{\mathrm{iot}}, t_u^{\mathrm{trans}} + t_u^{\mathrm{uav}}\right). \quad (18)$$

Finally, the total latency of the IoT device *u* in the *n*-th time slot can be expressed as:

$$\begin{aligned} \mathcal{T}_u(n) &= (1 - X_u^{\mathrm{uav}}(n)) t_u^{\mathrm{loc}} + X_u^{\mathrm{uav}}(n) t_u^{\mathrm{off}}(n) \\ &= (1 - X_u^{\mathrm{uav}}(n)) t_u^{\mathrm{loc}} \\ &\quad + X_u^{\mathrm{uav}}(n) \max(t_u^{\mathrm{iot}}, t_u^{\mathrm{trans}} + t_u^{\mathrm{uav}}), \end{aligned} \quad (19)$$

where $X_u^{\mathrm{uav}}(n)$ can be 0 or 1.

Based on the above analysis, the total latency of the tasks generated by all the IoT devices is formulated as:

$$\mathcal{T} = \sum_{u=1}^{m} \sum_{n=1}^{N} \mathcal{T}_u(n). \quad (20)$$

2) Energy: The energy consumption of the whole edge computing system consists of four parts: a) uplink transmission energy consumption; b) the energy consumption for the task with local computing only; c) the energy consumption by the UAV performing partial offloading; d) the flight energy consumption of the UAV.

a) Energy consumption for transmitting offloaded task: When offloading the latency-sensitive task from the IoT device *u* to the UAV, the corresponding energy consumption can be expressed as follows:

$$e_u^{\text{trans}} = p_u^{\text{up}} t_u^{\text{trans}} = p_u^{\text{up}} \frac{\lambda_u(n) L_u(n)}{R_u^{\text{up}}(n)}, \tag{21}$$

where p_u^{up} denotes the uplink transmit power.

b) Energy consumption for the task with local computing only: If the task generated by the IoT device *u* is computed locally only, the energy consumption can be expressed as:

$$e_u^{\text{loc}} = k_u (f_u^{\text{user}})^3 t_u^{\text{loc}}, \tag{22}$$

where k_u denotes the effective switched capacitance of the IoT device *u* [2].

c) Energy consumption for the task with partial offloading: The energy consumption generated by offloading part of the latency-sensitive task to the UAV can be expressed as:

$$e_u^{\text{uav}} = k_{uav} (f_u^{\text{uav}})^3 t_u^{\text{uav}}, \tag{23}$$

where k_{uav} represents the effective switched capacitance coefficient of the UAV [2].

In addition, the energy consumption required by IoT device *u* to compute the remaining task, e_u^{iot} , can be expressed as:

$$e_u^{\text{iot}} = k_u (f_u^{\text{user}})^3 t_u^{\text{iot}}. \tag{24}$$

Consequently, the energy consumption for the task with partial offloading, $e_u^{off}(n)$, can be expressed as follows:

$$e_u^{off}(n) = e_u^{\text{trans}} + e_u^{\text{uav}} + e_u^{\text{iot}}, \tag{25}$$

where the energy consumption spent on transmitting the partial latency-sensitive task is also considered.

d) Flight energy consumption of the UAV: We adopt the refined UAV propulsion energy consumption model for quadrotor UAV following [38]. The propulsion energy consumption of the UAV is related to its instantaneous acceleration and velocity, then the propulsion energy consumption can be expressed using the following formula:

$$E_{\text{uav}}^{\text{fly}} = \delta \sum_{n=1}^{N} (w_1 \|\mathbf{v}_n\|^3 + \frac{\alpha_2}{\|\mathbf{v}_n\|} + \frac{w_2 \|\mathbf{a}_n\|^2}{g^2 \|\mathbf{v}_n\|}), \tag{26}$$

where w_1 and w_2 are two parameters related to UAV's weight, wing area, air density, etc. δ represents the size of each time slot. g and $\mathbf{v}_n$ denote the gravitational acceleration and the UAV's velocity in the *n*-th time slot, respectively. $\mathbf{a}_n = \mathbf{v}_n/\delta$ represents the UAV's acceleration [27].

The total energy consumption of the IoT device *u* for transmission and computation at the *n*-th time slot is expressed as follows:

$$E_u(n) = (1 - X_u^{\text{uav}}(n)) e_u^{\text{loc}} + X_u^{\text{uav}}(n) e_u^{off}(n). \tag{27}$$

In summary, the energy consumption by all IoT devices can be expressed as:

$$E = \sum_{u=1}^{m} \sum_{n=1}^{N} E_u(n) + E_{\text{uav}}^{\text{fly}}. \tag{28}$$

IV. Algorithm Design

In this section, we propose a UAV-assisted task offloading mechanism (UTOM) to solve $P1$ which consists of the following three steps: 1) When the offloading decision and the location of the UAV are given, the optimal transmit power for IoT device u offloading to the UAV and CPU frequency of the UAV are obtained based on the Lagrange multiplier method and KKT condition; 2) Given the optimal resource allocation and the UAV location, an improved particle swarm optimization (IPSO) algorithm is introduced to obtain the optimal offloading decision of all the IoT devices; 3) A deep determination policy gradient (DDPG) based algorithm is introduced to obtain the optimal trajectory of the UAV.

A. Optimal allocation of the transmit power and the CPU frequency

Assume that offloading decision X and the UAV location Θ are given. It can be shown that the constraints C_6, C_7 and C_8 of $P1$ are relevant for determining the transmitted power and the CPU frequency. Therefore, the subproblem can be formulated as:

$$\begin{aligned} P2: \quad & \min_{p_u^{\text{up}}, f_u^{\text{uav}}} \mathbb{C} = (1-\omega)\mathcal{T} + \omega E \\ s.t. \quad & C_6, C_7, \\ & C_9 : t_u^{\text{trans}}(n) + t_u^{\text{uav}}(n) \le \delta \\ & C_{10} : f_u^{\text{uav}} \le f_{\max} \end{aligned} \tag{29}$$

As $P2$ does not satisfy the property of the concavity or convexity of the function, it is a non-convex optimization problem. To address the $P2$ problem, we can utilize the variable substitution technique [33] to transform $P2$ into the subsequent optimization problem.

$$\begin{aligned} P3: \min_{p_u^{\text{up}}, f_u^{\text{uav}}} &= (1-\omega)[t_u^{\text{loc}} + \max\{t_u^{\text{iot}}, \tfrac{\lambda_u(n) L_u(n)}{B_0 x} + t_u^{\text{uav}}\}] \\ &+ \omega(e_u^{\text{loc}} + \tfrac{\sigma^2(2^x-1)}{h_{up}(n)} \tfrac{\lambda_u(n) L_u(n)}{B_o x} + e_u^{\text{uav}} + e_u^{\text{iot}} + E_{\text{uav}}^{\text{fly}}) \\ s.t. C_7&, C_9, C_{10} \\ C_{11} : \log_2&(1 + \tfrac{h_{up}(n) p_{\min}}{\sigma^2}) \le x \le \log_2(1 + \tfrac{h_{up}(n) p_{\max}}{\sigma^2}), \end{aligned} \tag{30}$$

where $x = \log_2(1 + h_{up}(n) p_u^{up}/\sigma^2)$. The Lagrange multiplier function of problem $P3$ is:

$$\begin{aligned} L = L(f_u^{\text{uav}}, x, \varphi) &= (1-\omega)\mathcal{T} + \omega E \\ &+ \zeta(t_u^{\text{trans}} + t_u^{\text{uav}} - \delta) + \varphi(f_u^{\text{uav}} - f_{\max}). \end{aligned} \tag{31}$$

where $(\zeta, \varphi) \ge 0$ represents the Lagrange multipliers corresponding to the constraints C_9 and C_{10}, respectively. Our next step is to show that the optimal solution can be found if the problem $P3$ is a convex optimization problem.

Lemma 1: $P3$ is a convex optimization problem.

Proof: We will show that $P3$ satisfies the three conditions involved in the theory of convex optimization: 1) The objective function is a convex function; 2) The equation constraint

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488210

functions are linear functions; 3) The functions are convex functions. As can be seen from the following (32), we can divide L into two distinct cases:

$$L=\begin{cases}(1-\omega)(t_u^{\rm loc}+t_u^{\rm iot})\\ +\omega(e_u^{\rm loc}+\frac{\sigma^2(2^x-1)}{h_{up}(n)}\frac{\lambda_u(n)L_u(n)}{B_0x}+e_u^{\rm uav}+e_u^{\rm iot}+E_{\rm uav}^{\rm fly})\\ +\zeta(\frac{\lambda_u(n)L_u(n)}{B_0x}+t_u^{\rm uav}-\delta)+\varphi(f_u^{\rm uav}-f_{\max}),\\ \text{ if } t_u^{\rm iot}>\max(t_u^{\rm trans}+t_u^{\rm uav}),\\ \\ (1-\omega)(t_u^{\rm loc}+\frac{\lambda_u(n)L_u(n)}{B_0x}+t_u^{\rm uav})\\ +\omega(e_u^{\rm loc}+\frac{\sigma^2(2^x-1)}{h_{up}(n)}\frac{\lambda_u(n)L_u(n)}{B_0x}+e_u^{\rm uav}+e_u^{\rm iot}+E_{\rm uav}^{\rm fly})\\ +\zeta(\frac{\lambda_u(n)L_u(n)}{B_0x}+t_u^{\rm uav}-\delta)+\varphi(f_u^{\rm uav}-f_{\max}),\\ \text{ if } t_u^{\rm iot}\le\max(t_u^{\rm trans}+t_u^{\rm uav}).\end{cases} \tag{32}$$

We first consider the first case when $t_u^{\rm iot}>\max(t_u^{\rm trans}+t_u^{\rm uav})$.

1) By calculating the Hessian matrix of the objective function in $P3$, we can obtain the following results:

$$\begin{aligned}H_{\rm cost}&=\begin{bmatrix}\frac{\partial^2L}{\partial(f_u^{\rm uav})^2} & \frac{\partial^2L}{\partial f_u^{\rm uav}\partial x}\\ \frac{\partial^2L}{\partial x\partial f_u^{\rm uav}} & \frac{\partial^2L}{\partial x^2}\end{bmatrix}\\ &=\begin{bmatrix}\frac{2A_1}{(f_u^{\rm uav})^3}+B_1 & 0\\ 0 & \frac{\partial^2L}{\partial x^2}\end{bmatrix}>0\end{aligned} \tag{33}$$

where $\partial^2L/\partial^2x^2=\omega\sigma^2\lambda_u(n)L_u(n)/h_{up}(n)B_02^x x\ln^2(2)+2\lambda_u(n)L_u(n)\zeta/x^3B_0>0$, $B_1=2\omega\lambda_u(n)L_u(n)k_{\rm uav}$ and $A_1=\lambda_u(n)L_u(n)\zeta$. As $H_{\rm cost}$ is a semi-positive definite matrix, it follows that the objective function in $P3$ is a convex function.

2) There are multiple inequality constraints in problem $P3$, and we consider the inequality constraint C_9 as an example for the proof. For $t_u^{\rm trans}+t_u^{\rm uav}-\delta\le 0$, let $l_1(x)=\delta-t_u^{\rm trans}-t_u^{\rm uav}$ and $-l_1(x)=t_u^{\rm trans}+t_u^{\rm uav}-\delta$. Then, the Hessian matrix of function $-l_1(x)$ is:

$$\begin{aligned}H_{-l_1(x)}&=\begin{bmatrix}\frac{\partial^2L}{\partial(f_u^{uav})^2} & \frac{\partial^2L}{\partial f_u^{uav}\partial x}\\ \frac{\partial^2L}{\partial x\partial f_u^{uav}} & \frac{\partial^2L}{\partial x^2}\end{bmatrix}\\ &=\begin{bmatrix}\frac{2\lambda_u(n)L_u(n)}{(f_u^{uav})^3} & 0\\ 0 & \frac{2L_u(n)\lambda_u(n)}{x^3B_0}\end{bmatrix}>0\end{aligned} \tag{34}$$

3) As any equation constraint is not involved in problem $P3$, additional proof is not needed.

In summary, it can be concluded that $L\left(f_u^{uav},x\right)$ in the first case is a convex function. With similar logic, we can prove that the second case of $t_u^{\rm iot}\le\max\left(t_u^{\rm trans}+t_u^{\rm uav}\right)$ is also a convex function. Therefore, $P3$ is a convex optimization problem, which can be resolved by Lagrange multiplier method.

Hence, the statements in this lemma are proved.

According to **Lemma 1**, we can obtain the optimal CPU frequency $f_u^{\rm uav}$ of the UAV and the transmit power p_u^{up} through the Lagrange multiplier method. In the following, we shall derive them one by one.

Lemma 2: The optimal CPU frequency of the UAV can be derived as:

$$f_u^{uav^*}=\begin{cases}f_{\max}, \text{if } \omega=0 \text{ or } f_u^{uav^{opt}}>f_{\max}\\ f_{\min}, \text{ if } \omega=1\\ f_u^{uav^{opt}}, \text{otherwise}\end{cases} \tag{35}$$

where $f_u^{{\rm uav}^{opt}}=\sqrt[3]{\zeta\lambda_u(n)L_u(n)/2\omega k_{\rm uav}\lambda_u(n)L_u(n)+\varphi}$, φ and ζ are the nonnegative Lagrange multipliers related to C_{10} and C_9.

Proof: Through the KKT condition of $P3$, $f_u^{\rm uav}$ can be obtained. As shown in (1), if $\omega=0$, the original problem can be converted to minimize latency, and therefore the optimal CPU frequency is $f_u^{\rm uav*}=f_{\max}$. If $\omega=1$, the original problem can be transformed into the problem of minimizing energy consumption. Hence, the optimal CPU frequency is $f_u^{\rm uav*}=f_{\min}$. Consequently, the optimal strategy of the CPU frequency can be obtained by (35).

Hence, the statements in this lemma are proved.

Depending on whether the task is partially offloaded to the UAV, the problem of transmit power allocation can be discussed by two cases.

a) First case of $t_u^{\rm iot}>\max\left(t_u^{\rm trans}+t_u^{\rm uav}\right)$: In this case, the corresponding Lagrange multiplier function is the first case of (32).

Lemma 3: For the IoT device *u* offloading part of its task to the UAV, the optimal transmit power can be obtained as:

$$p_u^{up*}=\begin{cases}p_{\max}, \text{ if } \omega=0 \text{ or } \mathrm{h}\left(\mathrm{p_{max}}<0\right)\\ p_{\min}, \text{ if } \omega=1 \text{ or } \mathrm{h}\left(\mathrm{p_{min}}>0\right)\\ p_u^{{\rm up}^{opt}}, \text{otherwise}\end{cases} \tag{36}$$

where $h\left(p_u^{\rm up}\right)=\omega(\sigma^2\lambda_u(n)L_u(n)/h_{up}(n)B_0[(1+p_u^{\rm up}h_{up}(n)/\sigma^2)(\log_2(1+p_u^{\rm up}h_{up}(n)/\sigma^2)-1)+1])-\lambda_u(n)L_u(n)\zeta/x^2B_0$ and $x=\log_2(1+h_{up}(n)p_u^{up}/\sigma^2)$.

Proof: Here, $p_u^{{\rm up}^{opt}}$ is the solution of $h\left(p_u^{\rm up}\right)=0$ and can be obtained through binary search method [33]. According to the KKT condition of P3, the optimal transmission power should meet $h\left(p_u^{\rm up}\right)=0$. By taking the first order derivative of $h\left(p_u^{\rm up}\right)$, we are able to find that $h\left(p_u^{\rm up}\right)>0$. Therefore, $h\left(p_u^{\rm up}\right)$ is monotonically increasing. When $h\left(p_{\max}\right)<0$ and $h\left(p_{\min}\right)<0$, there is no solution for $h\left(p_u^{\rm up}\right)=0$. At this time, we take the maximum value at the boundary point.

Hence, the statements in this lemma are proved.

b) Second case of $t_u^{\rm iot}\le\max\left(t_u^{\rm trans}+t_u^{\rm uav}\right)$: In this case, the Lagrange multiplier function is shown in the second case of (32), $h\left(p_u^{\rm up}\right)=\omega(\sigma^2\lambda_u(n)L_u(n)/h_{up}(n)B_0[(1+p_u^{\rm up}h_{up}(n)/\sigma^2)(\log_2(1+p_u^{\rm up}h_{up}(n)/\sigma^2)-1)+1])-\lambda_u(n)L_u(n)/x^2B_0(\zeta+1-\omega)$, and $x=\log_2(1+h_{up}(n)p_u^{up}/\sigma^2)$. It is easy to show that $h\left(p_u^{\rm up}\right)$ is monotonously increasing and $h\left(p_{\max}\right)<0$. Therefore, $p_u^{up*}=p_{\max}$.

B. Optimal task offloading decision with the proposed IPSO

If the resource allocation and the UAV location are given, according to Section III, we know that the task offloading decision problem is NP-hard. To find the optimal task offloading decision, we propose an IPSO algorithm, which is an efficient evolutionary computation algorithm.

Firstly, particle swarm optimization (PSO) is a population-based evolutionary algorithm which aims to find the optimal solution by promoting cooperation and information sharing among individuals in a group. In PSO, we initialize a population $\mathcal{P}$ of particles. Each particle represents a possible task offloading decision that indicates which tasks are processed

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488210

by the IoT devices and which tasks are offloaded to the UAV. In each iteration, the position and velocity of each particle are updated by utilizing the current local and global optimal solutions. The position of the k-th particle denoted as x_k and the velocity of particle k denoted as v_k. The position and velocity of particles are updated as follows:

$$v_k(t+1) = \varpi v_k(t) + c_1 \cdot \mathrm{rand}() \cdot [p_{\mathrm{best}} - x_k(t)] + c_2 \cdot \mathrm{rand}() \cdot [g_{\mathrm{best}} - x_k(t)], \tag{37}$$

$$x_k(t+1) = x_k(t) + v_k(t+1), \tag{38}$$

where ϖ is the inertia weight, c_1 and c_2 are the learning factors, $\mathrm{rand}()$ denotes a random number in [0,1], p_{best} represents the optimal solution of each particle in iteration t, g_{best} indicates the optimal solution of the whole population, and t denotes the number of iterations.

1) Improved Particle Swarm Optimization (IPSO): While the classical PSO algorithm converges very quickly, it suffers from disadvantages such as the tendency to fall into the local optima. The reason is that the classical PSO algorithm uses fixed inertia weights. The larger inertia weights facilitate jumping out of the local optima for global search, while smaller inertia weights facilitate accurate local search within the current search region. To balance these global and local search capabilities, we introduce population diversity G to optimize the inertia weight ϖ for each iteration:

$$G(t+1) = \sqrt{\frac{1}{m-1}\sum_{k=1}^{m}\left(\overline{d_k(t+1)} - d_k(t+1)\right)^2} \tag{39}$$

where m represents the number of particles, $\overline{d_k(t)}$ denotes the average Euclidean distance between the k-th particle and other particles, and $d_k(t)$ indicates the minimum Euclidean distance between the k-th particle and other particles. We calculate the inertia weight function $\varpi_k(t)$ according to the population diversity G(t).

$$\varpi_k(t) = \begin{cases} \varpi_k(t)(e^{\frac{1}{G(t)+1}-1}+1), G(t) \geq G(t-1) \\ \varpi_k(t)(e^{\frac{1}{G(t)+1}-1}) \quad\quad, G(t) < G(t-1) \end{cases} \tag{40}$$

As the IPSO algorithm converges, the position of particle x_k is concentrated near the optimal value, with the particle speed v_k approaching zero. Therefore, we introduce sigmoid function to determine the offloading decision:

$$sigmoid\left(v_k(t)\right) = \frac{1}{1+e^{-v_k(t)}}. \tag{41}$$

Then, the task offloading decision $X_k^{\mathrm{uav}}(t)$ for particle k is determined as:

$$X_k^{\mathrm{uav}}(t) = \begin{cases} 1, \mathrm{rand}() < sigmoid\left(v_k(t)\right), \\ 0, \mathrm{otherwise}. \end{cases} \tag{42}$$

In general, IPSO uses fitness to evaluate the optimization objective. In this paper, the weighted sum of latency and energy consumption is used as the optimization objective. Therefore, we define the fitness function as follows:

$$\mathrm{Fitness} = (1-\omega)\mathcal{T} + \omega E. \tag{43}$$

The optimal task offloading decision can be determined based on the minimum objective function value obtained with IPSO. Specifically, the solution with the smallest objective function value can be found from the set of solutions, and the corresponding task offloading decision can be identified as the optimal decision.

C. DDPG-based UAV trajectory optimization

In this section, we propose a DDPG algorithm to address the UAV trajectory problem, provided that the offloading decision X, CPU frequency f_u^{uav}, and transmit power p_u^{up} are ready to be obtained. As a kind of deep reinforcement learning, the state, action, and reward of the DDPG are defined as follows:

1) State: The states mainly includes the location of UAV Q_{uav}, the locations of IoT devices $Q_{\mathrm{user}}(n)$, the movement speed of IoT devices $v_{\mathrm{iot}}(n)$, the angle of IoT devices $\theta_{\mathrm{iot}}(n)$, and the amount of task data generated by IoT devices $L_u(n)$. In summary, the state of agent can be expressed as:

$$s_n = \{Q_{\mathrm{user}}(n), Q_{\mathrm{uav}}(n), v_{\mathrm{iot}}(n), \theta_{\mathrm{iot}}(n), L_u(n)\}, \quad n \in [1,N], u \in [1,m] \tag{44}$$

where $v_{iot}(n) \in [0, 5 \text{ m/s}]$ and $\theta_{\mathrm{iot}}(n) \in [0, 2\pi]$. Note that as the time goes over, the five elements of the state may vary over time, which means that the IoT devices are also moving.

2) Action: The agent maps the state space to the action space. That is, the agent needs to map the corresponding action according to the current state. We define the offloading decision X, the velocity of the UAV $v(n)$, horizontal deflection angle $\theta_{\mathrm{uav}}(n)$ as the actions of agent, denoted as:

$$a_n = \{X_u^{\mathrm{uav}}(n), v(n), \theta_{\mathrm{uav}}(n)\}, n \in [1,N], u \in [1,m] \tag{45}$$

where $v(n) \in [30 \text{ m/s}, 50 \text{ m/s}]$ and $\theta_{\mathrm{uav}}(n) \in [0, 2\pi]$.

3) Rewards: The behavior of agent is reward-based, and the choice of an appropriate reward function plays a vital role in the performance of the DDPG. Each step of the UAV obtains an action and flights to a new location for data reception from the IoT devices. Therefore, the reward function is the sum of the total steps over the UAV trajectory, denoted as:

$$r(s,a) = (1-\omega)\mathcal{T} + \omega E. \tag{46}$$

As shown in Fig. 2, the framework of the DDPG-based algorithm consists of the environment, the main network, the target network, and the replay buffer. Specially, the agent is to sense the environment state s_n in n-th round trajectory update. The main network and target network are composed of the actor and critic network, respectively. The actor network aims to obtain the optimal policy μ, which generates the action a_n based on the state s_n. The critic network is responsible for obtaining the action value function, which updates the weights of the critic network. The replay buffer is used to store the tuples (s_n, a_n, r_n, s_{n+1}). Next, three key elements of the DDPG that emerge with each update are discussed.

a) Action selection: The main actor network selects an action a_n based on the current state s_n and the current deterministic policy μ.

$$a_n = \mu\left(s_n \mid \theta^{\mu}\right), \tag{47}$$

where θ^{μ} denotes the weights of the main actor network.

b) Update action value: After the action a_n is determined and executed, the network state transitions to s_{n+1}, and an immediate reward r_n is received. Based on the Bellman equation [41], the action value function can be given as follows:

$$Q^{\mu}(s_n, a_n) = \mathbb{E}_{s_{n+1} \sim \rho^{\mu}} \times [r_n(s_n, a_n) + \gamma Q^{\mu}(s_{n+1}, \mu(s_{n+1}|\theta^Q))], \tag{48}$$

where $\mathbb{E}[\cdot]$ denotes the expectation, and ρ^{μ} is the distributing function of state s_{n+1} under the current policy μ. $\gamma \in [0, 1)$ denotes the discounting factor in reinforcement learning that represents the uncertainty of future revenue. θ^Q represents the weights of the main critic network.

c) Weight and gradient: The weights θ^{μ} of the main actor network are updated based on the gradient method using the gradients of the action value function $Q(s, a)$ and action policy $\mu(s|\theta^{\mu})$. Then the policy gradient of θ^{μ} can be expressed by the chain rule as:

$$\nabla_{\theta^{\mu}} J = \frac{1}{N} \times \sum_{n=1}^{N} (\nabla_a Q(s, a|\theta^Q)|_{s=s_n, a=\mu(s_n)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_n}), \tag{49}$$

where N denotes the number of transitions selected from the replay buffer. Next, the weights θ^{μ} of the main actor network can be expressed as:

$$\theta^{\mu} = \theta^{\mu} + \beta^{\mu} \nabla_{\theta^{\mu}} J, \tag{50}$$

where β^{μ} means the learning rate of the main actor network. We train the main critic network θ^Q by using the gradient method, and we adopt the following loss function:

$$L(\theta^Q) = \mathbb{E}_{s_n \sim \kappa^{\mu}} \left[\left(Q(s_n, a_n|\theta^Q) - y_n \right)^2 \right], \tag{51}$$

where κ^{μ} denotes the distribution function of state s_n under the current policy μ. The target Q-value y_n of target critic network is calculated as follows:

$$y_n = r(s_n, a_n) + \gamma Q(s_{n+1}, \mu(s_{n+1}|\theta^{\mu})\theta^Q), \tag{52}$$

where $r(s_n, a_n)$ represents the reward after UAV trajectory update, and the symbols θ^{μ} and θ^Q represent the weights of the actor network and the critic network, respectively. $Q(\cdot|\theta^Q)$ and $\mu(\cdot|\theta^{\mu})$ indicate the action value function and the actor policy, respectively. Next, the weights θ^Q of the main critic network are expressed as:

$$\theta^Q = \theta^Q + \beta^Q \nabla_{\theta^Q} L(\theta^Q), \tag{53}$$

where β^Q is the learning rate of the main critic network. Lastly, the weights of the target actor network and the target critic network are updated by using a soft updating method, which proceeds as follows:

$$\theta^{\mu'} \leftarrow \tau \theta^{\mu} + (1 - \tau) \theta^{\mu'}, \tag{54}$$

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}, \tag{55}$$

where τ is the soft update coefficient.

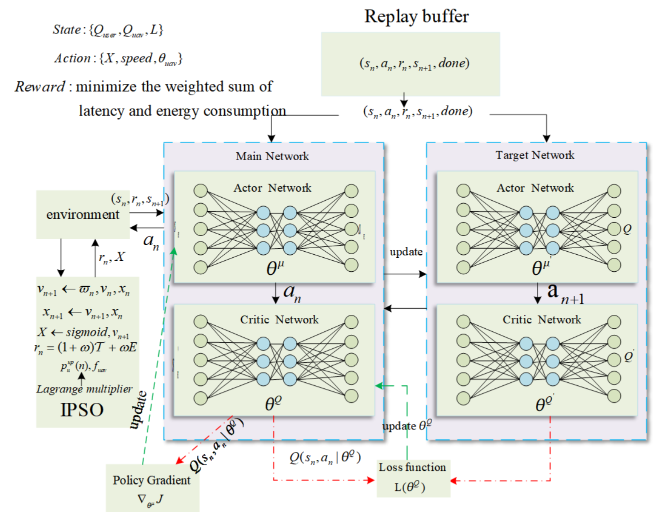


Fig. 2. Optimization process of UTOM mechanism based on offloading decision and resource allocation.

D. The Proposed Mechanism with Joint IPSO and DDPG

As shown in Fig. 2, we combine IPSO and DDPG to solve the resource allocation, offloading decision and trajectory optimization problem for the UAV-assisted task offloading environment. The idea to solve this problem mainly include: 1) the environment interacts with IPSO network to obtain the best reward r_n and offloading strategy X; 2) the environment interacts with the four networks in the DDPG to obtain the best action for each step of the UAV trajectory. The complete algorithm is shown in Algorithm 1.

The process of Algorithm 1 is as follows: We first randomly initialize the state s_1 (lines 1-2). Second, enter the action a_t and state s_t into the environment and obtain the reward r_{t+1} and new state s_{t+1} by updating the particle position and particle velocity (lines 3-14). Third, we add conversion tuple to replay buffer *rp* and take random samples from *rp* (lines 15-16). Fourth, we update the weights of main network and target network (lines 17-20). Finally, the above four steps are repeated until the loop ends.

E. Time complexity analysis

Here, we discuss the time complexity of Algorithm 1. Note that Algorithm 1 is based on the DDPG algorithm, and the actor network and critic network need to be respectively updated in each episode. The most time-consuming operation is in lines 7-13, and the time complexity is $O(I \times K)$, where I and K represent the number of iterations and particles, respectively. Hence, the complexity of Algorithm 1 is $O(N(Z_a F_a + Z_a) + N(Z_c F_c + Z_c)) + (I \times K))$, where N denotes the number of episodes. Z_a and Z_c denote the number of units in the hidden layers of the actor and critic networks, respectively. F_a and F_c represent the number of hidden layers in the actor and critic networks, respectively.

V. Experiments

In this section, we evaluate the performance of our proposed UTOM.

Algorithm 1 UAV-assisted task offloading mechanism (UTOM)

Input: Main actor network $a_t = \mu(s_t|\theta^{\mu})$; main critic network $Q(s_t, a_t|\theta^Q)$; target actor network $\mu^{'}$; target critic network $Q^{'}$ with weights θ^{μ}, $\theta^{\mu^{'}}$, θ^Q, $\theta^{Q^{'}}$; replay memory as *rp*

Output: Optimal offloading X, CPU frequency f_u^{uav}, transmit power p_{user}^{up} and UAV trajectory.

```
for each episode n = 1, ··· , N do
    Randomly initialize the state s_1.
    for step t = 1, ··· , T do
        Select the action a_t = μ(s_t|θ^μ) based on current state.
        Enter action a_t and current state s_t into the environment.
        Initialize particle position x_u, particle velocity v_u, inertia weights ϖ_u, local optimal solution p_best and global optimal solution g_best.
        for step i = 1, ··· , I do
            for each particle k = 1, ··· , K do
                The velocity v_{i+1} and position x_{i+1} of the next particle are updated and the offloading strategy X is calculated according to (42).
                The optimal transmit power and CPU frequency are calculated according to (35) and (36).
            end for
            Obtain instant rewards r_t for each step of the UAV trajectory and update p_best, g_best and inertia weights ϖ.
        end for
        Update feedback instant reward r_{t+1} and new state s_{t+1}.
        add (s_t, a_t, r_{t+1}, s_{t+1}) to replay buffer rp.
        Sample a batch size of (s_t, a_t, r_{t+1}, s_{t+1}) from rp.
        Calculate the target Q-value y_t according to (52) and update critic by minimizing the loss.
        Update the actor network according to the policy gradient ∇_{θ^μ} J.
        Update the weights of the main actor network and the main critic network according to (50) and (53).
        Update the weights of the target actor network and the target critic network according to (54) and (55).
    end for
    reward = Σ_{t=1}^{T} r_t
end for
```

A. Experiment Settings

We first set the experiment parameters to evaluate the UTOM referring to the literature [2] and [14]. The movement range of the UAV and IoT devices is set to $X_n, Y_n = [700\,\text{m}, 700\,\text{m}]$ and the altitude of the UAV flight is set to H = 100 m. The initial positions of the IoT devices are generated randomly in a uniform manner. Then, we update the flight trajectory of the UAV based on the best offloading decision of the IoT devices at each time slot. The trajectory of the UAV is from point (0, 0, 100 m) to end point (700 m, 700 m, 100 m). The speed and angle of flight of the UAV are limited to [30 m/s, 50 m/s] and [0, 2 π], respectively. The speed and angle of IoT devices are limited to [0, 5 m/s] and [0, 2 π], respectively. The minimum and maximum CPU frequencies of the UAV are $f_{\min} = 0.1\,\text{GHz}$ and $f_{\max} = 1.2\,\text{GHz}$, respectively. The minimum and maximum transmit power values of the IoT devices are $p_{\min} = 1\,\text{W}$ and $p_{\max} = 3\,\text{W}$, respectively. The effective switched capacitances of the IoT device and the UAV are $k_u = 10^{-26}$ and $k_{uav} = 10^{-27}$, respectively. The computing resource of the UAV is $C(m) = 10^8$ cycles. As the UAV flies to each IoT device to perform task offloading, it is highly likely to maintain an LoS channel. Consequently, in the experimental results, we primarily consider the ideal LoS channel [38], except in Fig. 8, where we also demonstrate the effectiveness of our proposed UTOM in combined LoS and NLoS channels. The complete parameters utilized for the experiment are summarized in Table II.

TABLE II
EXPERIMENT PARAMETERS

Parameters	Value
The amount of task data size, $L(n)$	[1,10] Mbits
The height of UAV, H	100 m
Moving range of UAV and IoT devices, X_n, Y_n	[700 m,700 m]
The reference channel gain, h_0	-50 dB
Time slot size, δ	1 s
The bandwidth allocated to each IoT device, B_0	10^7 Hz
The noise power, σ^2	-70 dBm/Hz
The IoT devices computing resources, f_u^{user}	10^8Hz
The gravity acceleration vector, $\boldsymbol{g}$	9.8 m/s^2 [27]
Parameters related to UAV, w_1	9.26×10^{-4} [22]
Parameters related to UAV, w_2	2250 [22]
Discount coefficient, γ	0.9
Soft update coefficient, τ	0.01
Learning rate of main actor network, β^{μ}	0.001
Learning rate of main critic network, β^{Q}	0.001
Iterative rounds	1000
The carrier frequency, F_c	2 GHz
The constant values, C	9.61
The constant values, D	0.16
The path loss exponent, a	2
The average additional loss of LoS, L_{LoS}	0-6 dB
The average additional loss of NLoS, L_{NLoS}	10-30 dB

[1] Simulation data references [27] and [22].

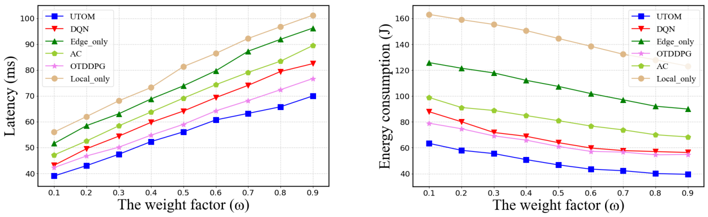
We implement the experiment using Python 3.9.0. All our experiments are run on at Lenovo, AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx@2.10 GHz, Windows 11.

B. Comparison Algorithms

The following algorithms are considered as benchmarks to check the effectiveness of our proposed UTOM.

a) Task offloading algorithm based on a Deep Deterministic Policy Gradient algorithm (OTDDPG) [32]: It aims to minimize maximum processing latency by jointly optimizing the task offloading decision, task offloading ratio, and flight speed. However, OTDDPG addresses single-objective optimization and does not consider resource constraints. To ensure a fair comparison, we modified OTDDPG's objective to minimize the cost for all the IoT devices while satisfying resource constraints. Additionally, the action space of OTDDPG is

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488210



(a) Latency (b) Energy

Fig. 3. Latency and Energy of different weight.

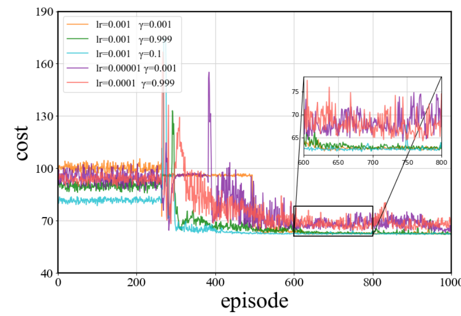


Fig. 4. The effect of different network parameters on the training rewards

continuous, enabling it to solve the same problem proposed in this paper.

b) Deep Q-network algorithm (DQN) [42]: Its aim is to minimize the energy consumption and latency by jointly optimizing users' offloaded task ratio scheduling and UAV trajectory. Since DQN does not consider resource allocation optimization and limited storage and computational resources. To be fair, we implement the resource allocation optimization on the original basis of this algorithm while satisfying the resource constraints of the UAV server.

c) Actor-Critic-based computation offloading algorithm (AC) [29]: It aims to minimize the weighted sum of latency and energy consumption by optimizing offloading decisions under resource constraints. For a fair comparison with the mechanism proposed in this paper, we also optimize the transmission power and computational power for AC.

d) Edge_only: It offloads the whole latency-sensitive task from the IoT device to be executed on the UAV.

e) Local_only: It makes all the latency-sensitive tasks on IoT devices be executed locally.

C. Experiment Results

This paper investigates the concept of dual mobility, where IoT devices move randomly over time slots. The UAV can predict flight trajectory for the next time slot based on the offloading decisions made by by IoT devices in the current time slot. To demonstrate the feasibility and effectiveness of our proposed UTOM, we analyze six different sets of experimental results.

1) The effect of different weights on latency and energy: Here, the experiment contains 10 IoT devices. Figs. 3(a) and 3(b) show the effects of different weights on latency and energy consumption, respectively. It can be found that as the weight increases, the latency of all IoT devices increases while the energy consumption decreases. When $\omega < 0.5$, the system has a higher demand for latency. Therefore, we can increase the transmit power of IoT devices to reduce transmission latency. When $\omega > 0.5$, the system has a higher demand for energy consumption. For this, we can increase CPU frequency of the UAV, and reduce the execution energy consumption of the UAV. To verify that the proposed UTOM can balance both the latency and energy consumption, in the following experiments we set $\omega = 0.5$ as a representative example.

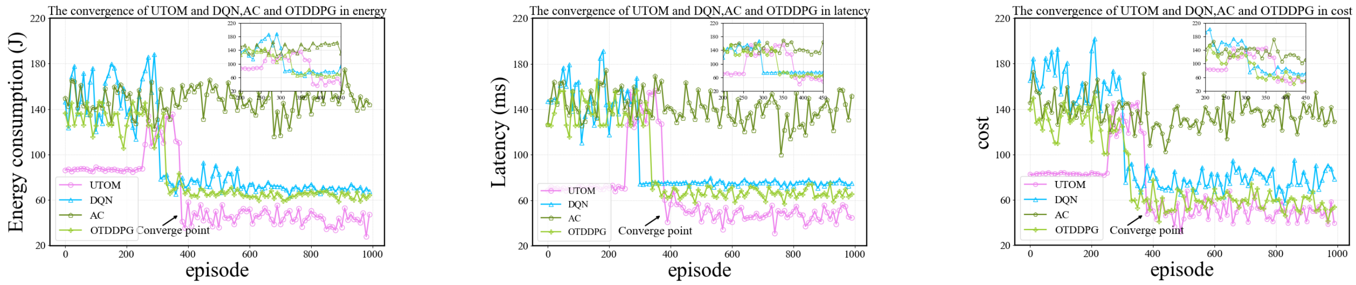
2) The comparison of different network parameters: In this subsection, we explore the effect of different network parameters on the training rewards. Our experiments involve 10 IoT devices, with IPSO iterating 100 times and the weight factor of 0.5.

Fig. 4 shows the convergence performance of the proposed UTOM with different learning rates and discount factor γ. We assume that the learning rates of the actor and critic network are the same. On the one hand, when the learning rate $lr = 0.001$ and the discount factor $\gamma = 0.1,\ 0.999,\ 0.001$, the proposed UTOM can converge. we can clearly observe that when $lr = 0.001$, the change of discount factor γ only slightly effects the convergence speed of the reward function but does not affect its convergence result. On the other hand, when ($lr = 0.00001$, $\gamma = 0.001$) and ($lr = 0.0001$, $\gamma = 0.999$), we can find that the UTOM fails in converging.

Based on the above observation, we found that an appropriate combination of learning rate and discount factor can lead to optimal performance in the reward function. Specifically, when the learning rate is set to 0.001 and the discount factor is set to either 0.999 or 0.1, the model shows the best performance, achieving an average reward function of 62.12 on the test set. In contrast, when the learning rate is set to 0.00001 and the discount factor is set to 0.001, the reward function may not converge.

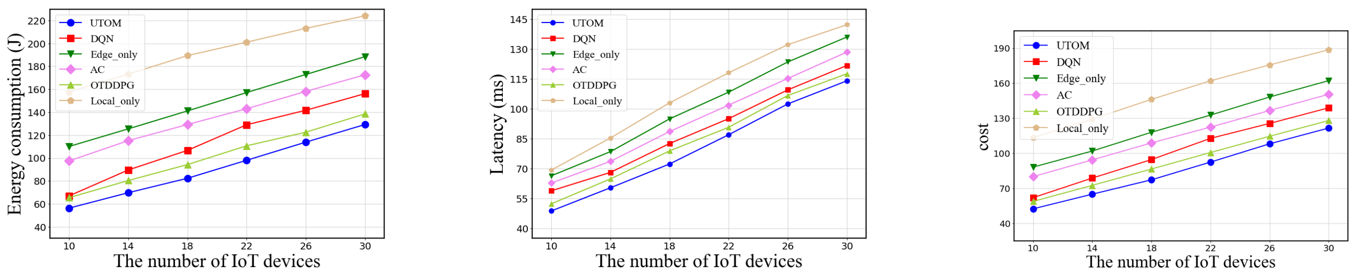
3) Convergence performance: Fig. 5 shows the convergence performance of the proposed UTOM in comparison with the DQN, AC and OTDDPG algorithms in terms of energy consumption, latency, and cost. As can be seen, the UTOM converges when the number of iterations reaches 380. In addition, we can observe that the UTOM achieves better performance in terms of energy consumption, latency, and cost compared to the DQN, AC and OTDDPG algorithms. The AC algorithm struggles with updating both the actor and critic networks. The action selection by the actor network depends on the value function of the critic network, which is challenging to converge. The DQN algorithm needs to explore the non-negligible space between the discrete action space and the available actions, making it hard to accurately find the optimal offloading decision. In contrast, the UTOM allows for more effective learning on sequential actions, and it combines the DQN structure to improve the stability and convergence of the Actor-Critic. Compared to the OTDDPG algorithm, the UTOM mechanism incorporates an IPSO algorithm, allowing it to explore a larger search space more efficiently and find a better offloading strategy.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488210



(a) Energy (b) Latency (c) Cost

Fig. 5. The convergence of our proposed UTOM in comparison with the DQN, AC and OTDDPG algorithms in cost, latency, and energy consumption.



(a) Energy (b) Latency (c) Cost

Fig. 6. Energy consumption, Latency, cost over different IoT device numbers.

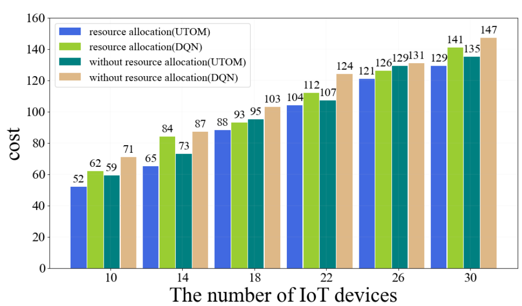


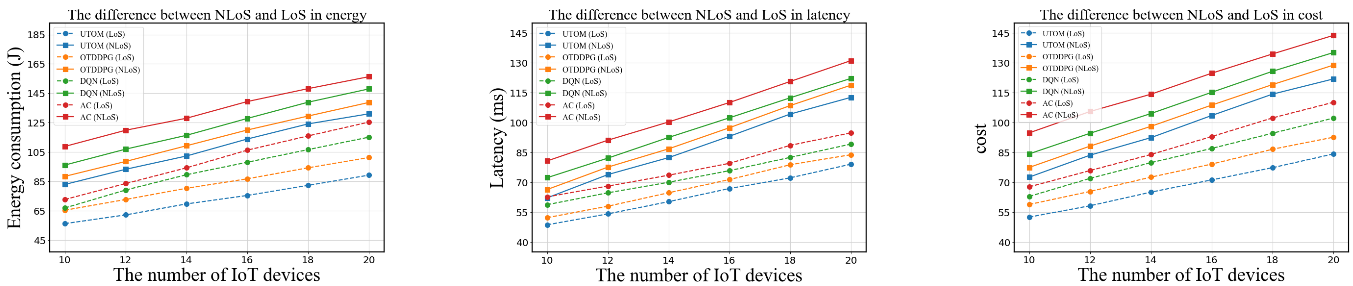
Fig. 7. The effect of resource allocation on cost.

The experimental results show the proposed mechanism performs better than the DQN, AC and OTDDPG algorithm. In the experimental process, the UTOM can achieve the optimal policy in a short time through sufficient training and remain stable after reaching the optimal policy. Therefore, compared to the DQN, AC and OTDDPG algorithm, the UTOM has better convergence performance and a wider range of applicability.

4) Effect of IoT device numbers: Fig. 6 shows the variation in energy consumption, latency, and cost over different number of IoT devices. It can be observed that as the number of IoT devices increases, the energy consumption, latency, and cost of all the six algorithms increase. For instance, when there are 10 IoT devices, the UTOM exhibits energy consumption, latency, and cost of 56.32, 48.72, and 52.52, respectively. The latency of DQN, Edge_only, AC, OTDDPG and Local_only is 58.82, 66.35, 62.72, 52.31, and 69.24, respectively, and energy consumption is 67.00, 110.09, 97.52, 65.43, and 157.72, respectively. The results clearly indicate that the UTOM outperforms the other algorithms.

For Edge_only, the whole latency-sensitive task is offloaded to the UAV, thus incurring significant offload latency and energy consumption, which increases the cost of offloading task. For Local_only, all tasks are performed locally, and the limited computing resources of IoT devices cannot handle that many latency-sensitive tasks, thus leading to excessive cost. In the AC algorithm, the critic network struggles to accurately estimate the value function, affecting the decision quality of the actor network and resulting in poor convergence speed and stability, particularly in complex environments. Therefore, the cost of Edge_only, Local_only, and AC are higher than the other three algorithms. Although DQN and OTDDPG perform better than the previous two algorithms, the UTOM still shows lower energy consumption, latency, and cost than DQN and OTDDPG. This is because the UTOM can output multiple continuous actions and take resource allocation into account, whereas DQN is used to deal with discrete actions and OTDDPG does not consider the effect of resource allocation on cost. Therefore, the proposed UTOM can accurately find a factor that has a large effect on the cost, latency and energy consumption of a continuous action control system. In addition, the proposed UTOM comprehensively considers the resource allocation problem, i.e., obtain the optimal transmit power and CPU frequency of the UAV, so it outperforms DQN and OTDDPG. Compared to DQN and OTDDPG, our proposed UTOM reduces the cost by approximately 14.23% and 6%, respectively.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488210



(a) Energy (b) Latency (c) Cost

Fig. 8. The difference between NLoS and LoS in terms of energy consumption, latency and cost.

TABLE III
RESULTS OF ABLATION EXPERIMENTS WITH DIFFERENT METHODS. (A SMALLER COST INDICATES A BETTER RESULT.)

Cost (*with* resource allocation) \ IoT number / Method	10	14	18	22	26	30
IPSO+DDPG	52.52	65.04	88.01	104.49	121.14	129.62
PSO+DDPG	62.85	76.73	94.96	112.28	129.34	137.32
Cost (*without* resource allocation) \ IoT number / Method	10	14	18	22	26	30
IPSO+DDPG	59.45	73.87	95.88	107.49	129.83	135.42
PSO+DDPG	68.55	83.86	103.78	123.83	135.66	146.72

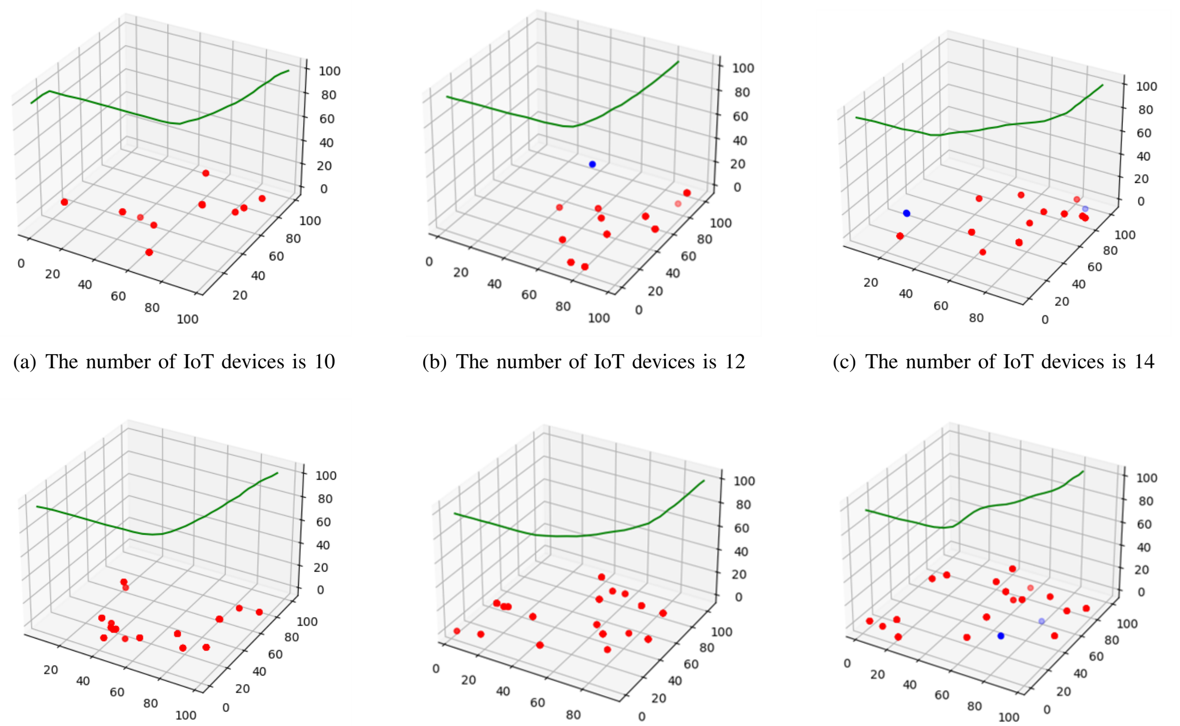
5) Effect of resource allocation: Fig. 7 provides a comparison of the cost with and without resource allocation for different numbers of IoT devices. For example, when the number of IoT devices is 10, we can observe that the cost of the UTOM with resource allocation is 52.52 and the cost of one without resource allocation is 59.45. As the number of IoT devices continues to increase, it can be observed that the cost is also increasing. For UTOM and DQN, the cost with resource allocation considered are overall smaller than the cost without resource allocation considered. This is mainly due to the optimal transmit power and CPU frequency solved by using the Lagrange multiplier method and the KKT condition.

Compared to the situation without resource allocation, using the Lagrange multiplier method and KKT condition to solve the resource allocation problem can achieve better performance. Specifically, the optimal transmit power and CPU frequency can make the most of the computing resources of the UAV, reducing the conflicts and interference among IoT devices, and thus reducing the cost of collaborative task completion.

6) Ablation experiment: To investigate the impact of each component of the UTOM, we conducted ablation experiments for different methods. For comparison, we consider PSO, a classical optimization algorithm, to serve as a benchmark for evaluating IPSO. By testing under different conditions, we can systematically analyze whether IPSO significantly improves performance in terms of cost reduction. Table III summarizes the experimental results under the conditions we considered.

For this ablation experiment, the discussion is divided into two parts: the impact of the two methods on cost after optimizing resource allocation, and their impact on cost without optimizing resource allocation. *First*, with optimized resource allocation, the IPSO+DDPG method rapidly finds the optimal offloading decision by introducing population diversity and optimizing inertia weights in each iteration to balance global and local search capabilities, thereby significantly reducing task offloading costs. In contrast, while the PSO+DDPG method also reduces costs to a certain extent, it is less efficient due to PSO's tendency to converge on local optima during the search process. Experimental results demonstrate that the IPSO+DDPG method can reduce the cost by approximately 14% under the same environmental settings. *Second*, without considering resource allocation, the IPSO+DDPG method still outperforms PSO+DDPG, indicating that even in scenarios lacking resource allocation optimization, IPSO+DDPG effectively reduces task offloading costs through more efficient policy learning and task scheduling mechanisms. Specifically, the cost of the IPSO+DDPG method is approximately 13% lower than that of the PSO+DDPG method without resource allocation optimization. In summary, IPSO+DDPG demonstrates superiority over PSO+DDPG, with or without resource allocation optimization. For example, when the number of IoT devices is 18, the costs of the IPSO+DDPG and PSO+DDPG methods with resource allocation are 88.01 and 94.96, respectively, while the costs without resource allocation are 95.88 and 103.78, respectively.

7) Effect of NLoS channels: To examine the effectiveness of our proposed UTOM across diverse channels, we further evaluate its performance specifically in terms of energy consumption, latency, and cost under combined LoS and NLoS channel conditions, based on (11), as shown in Fig. 8. For comparison, the performance of the OTDDPG, DQN, and AC algorithms is also presented. All algorithms exhibit worse performance under the combined LoS and NLoS channel conditions (denoted as 'NLoS' in the figure) compared to the ideal LoS channel conditions, primarily due to the increased



(a) The number of IoT devices is 10 (b) The number of IoT devices is 12 (c) The number of IoT devices is 14

(d) The number of IoT devices is 16 (e) The number of IoT devices is 18 (f) The number of IoT devices is 20

Fig. 9. The optimal trajectory of UAV for different IoT devices.

path loss caused by obstacles. Nonetheless, our proposed UTOM outperforms the OTDDPG, DQN, and AC algorithms, further confirming its effectiveness.

8) The optimal trajectory of UAV: In this experiment, we set the number of IoT device increases from 10 to 20. As shown in Fig. 9, we can observe that the optimal trajectory of the UAV for different IoT devices distribution can converge eventually. In the 3D space, the green line represents the flight path of the UAV from the initial position to the end position, while the both red and blue dots represent IoT devices. The red dot represents the scenario where the IoT device partially offloads the task to the UAV and partially computs it locally, while the blue dots represent the scenario where the IoT device computes the whole task locally. The two colors help visualize how the task is being offloaded for each IoT device.

In Fig. 9, IoT devices are randomly distributed and the UAV flies from the initial coordinates (0,0,100) to the end coordinates (700,700,100). Specifically, as shown in Figs. 9(a), 9(c), 9(e) and 9(f), the UAV chooses the shortest path to fly when the locations of the IoT devices are scattered or centrally deployed near the diagonal of the 3D space. This is done to minimize the weighted sum of latency and energy consumption. As shown in Figs. 9(b) and 9(d), the UAV makes a prediction based on the IoT devices current decision. In other words, the UAV follows the position of the IoT device to fly, which can minimize the cost.

Based on the above discussion, we can see that regardless of the location distribution of the IoT devices and the number of IoT devices, the flight trajectory of the UAV can be predicted for the next moment based on the minimum cost of IoT devices.

VI. Conclusion and Future Work

In this paper, we address the problem of UAV-assisted task offloading with the limited resources of UAVs and IoT devices. First, we construct a cost function associated with latency and energy consumption in the studied scenario and formulate the optimization problem. We then propose a UAV-assisted task offloading mechanism (UTOM) to minimize the cost by jointly optimizing resource allocation, task offloading decisions, and UAV trajectory. To efficiently solve the formulated problem, we decompose it into three easily solvable subproblems: 1) The optimal solution for resource allocation is obtained using the Lagrange multiplier method and the KKT conditions; 2) The optimal offloading decision is determined using an improved particle swarm optimization (IPSO) algorithm; 3) The UAV flight trajectory is derived using a deep deterministic policy gradient (DDPG). Finally, extensive experimental results demonstrate the high efficiency of our proposed UTOM.

While our research has achieved positive results, there are still some limitations. First, we plan to incorporate multi-UAV systems for scenarios with many IoT devices, as a single UAV may not meet task demands. Second, considering the joint use of UAVs and edge servers is important, as it improves the overall performance and efficiency of task offloading. We shall investigate these research issues in future work.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Nos. 62472147 and 62072159).

REFERENCES

[1] T. Tan, M. Zhao, and Z. Zeng, "Joint offloading and resource allocation based on uav-assisted mobile edge computing," *ACM Transactions on Sensor Networks*, vol. 18, no. 3, pp. 1–21, 2022.

[2] A. M. Seid, G. O. Boateng, S. Anokye, T. Kwantwi, G. Sun, and G. Liu, "Collaborative computation offloading and resource allocation in multi-uav-assisted iot networks: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 203–12 218, 2021.

[3] R. Bi, T. Peng, J. Ren, X. Fang, and G. Tan, "Joint service placement and computation scheduling in edge clouds," in *2022 IEEE International Conference on Web Services (ICWS)*, 2022, pp. 47–56.

[4] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, and Y. Yang, "Efficient dependent task offloading for multiple applications in mec-cloud system," *IEEE Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2147–2162, 2021.

[5] C. Ding, Y. Li, and S. Wang, "Edge/cloud-assisted feature extraction in iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 594–21 606, 2022.

[6] J. Zhang, G. Zhang, X. Bao, C. Ding, P. Yuan, X. Zhang, and S. Wang, "Dependent application offloading in edge computing," *IEEE Transactions on Cloud Computing*, vol. 11, no. 4, pp. 3439–3451, 2023.

[7] J. Zhang, J. Chen, X. Bao, C. Liu, P. Yuan, X. Zhang, and S. Wang, "Dependent task offloading mechanism for cloud–edge-device collaboration," *Journal of Network and Computer Applications*, vol. 216, pp. 103 656–103 668, 2023.

[8] Y. K. Tun, N. D. Tri, K. Kim, M. Alsenwi, W. Saad, and C. S. Hong, "Collaboration in the sky: A distributed framework for task offloading and resource allocation in multi-access edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 221–24 235, 2022.

[9] H. Wang, "Collaborative task offloading strategy of uav cluster using improved genetic algorithm in mobile edge computing," *Journal of Robotics*, vol. 2021, pp. 1–9, 2021.

[10] X. Diao, X. Guan, and Y. Cai, "Joint offloading and trajectory optimization for complex status updates in uav-assisted internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 881–23 896, 2022.

[11] J. Zhou and X. Zhang, "Fairness-aware task offloading and resource allocation in cooperative mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3812–3824, 2022.

[12] 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 Transactions on Wireless Communications*, vol. 22, no. 12, pp. 8876 – 8890, 2023.

[13] I. AElgendy, S. Meshoul, and M. Hammad, "Joint task offloading, resource allocation, and load-balancing optimization in multi-uav-aided mec systems," *Applied Sciences*, vol. 13, no. 4, pp. 2625–2647, 2023.

[14] L. P. Qian, H. Zhang, Q. Wang, Y. Wu, and B. Lin, "Joint multi-domain resource allocation and trajectory optimization in uav-assisted maritime iot networks," *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 539–552, 2023.

[15] F. Li, J. Luo, Y. Qiao, and Y. Li, "Joint uav deployment and task offloading scheme for multi-uav-assisted edge computing," *Drones*, vol. 7, no. 5, pp. 284–302, 2023.

[16] G. Zheng, C. Xu, M. Wen, and X. Zhao, "Service caching based aerial cooperative computing and resource allocation in multi-uav enabled mec systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 10, pp. 10 934–10 947, 2022.

[17] F. Li, J. Luo, Y. Qiao, and Y. Li, "Joint uav deployment and task offloading scheme for multi-uav-assisted edge computing," *Drones*, vol. 7, no. 5, pp. 284–302, 2023.

[18] J. Xiong, H. Guo, and J. Liu, "Task offloading in uav-aided edge computing: Bit allocation and trajectory optimization," *IEEE Communications Letters*, vol. 23, no. 3, pp. 538–541, 2019.

[19] F. Song, H. Xing, X. Wang, S. Luo, P. Dai, Z. Xiao, and B. Zhao, "Evolutionary multi-objective reinforcement learning based trajectory control and task offloading in uav-assisted mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 12, pp. 7387–7405, 2022.

[20] C. Yang, B. Liu, H. Li, B. Li, K. Xie, and S. Xie, "Learning based channel allocation and task offloading in temporary uav-assisted vehicular edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9884–9895, 2022.

[21] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in uav-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[22] Y. Xu, T. Zhang, Y. Liu, D. Yang, L. Xiao, and M. Tao, "Uav-assisted mec networks with aerial and ground cooperation," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7712–7727, 2021.

[23] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing uav," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[24] Z. Xiang, S. Deng, Y. Zheng, D. Wang, J. Tehari, and Z. Zheng, "Energy-effective iot services in balanced edge-cloud collaboration systems," in *2022 IEEE International Conference on Web Services (ICWS)*, 2021, pp. 219–229.

[25] H. Kang, X. Chang, J. Mišić, V. B. Mišić, J. Fan, and Y. Liu, "Cooperative uav resource allocation and task offloading in hierarchical aerial computing systems: A mappo based approach," *IEEE Internet of Things Journal*, vol. 10, no. 12, pp. 10 497 – 10 509, 2023.

[26] J. Du, Z. Kong, A. Sun, J. Sun, D. Niyato, X. Chu, and F. R. Yu, "Maddpg-based joint service placement and task offloading in mec empowered air-ground integrated networks," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 10 600–10 615, 2023.

[27] Y. He, Y. Gan, H. Cui, and M. Guizani, "Fairness-based 3d multi-uav trajectory optimization in multi-uav-assisted mec system," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11 383–11 395, 2023.

[28] B. Liu, Y. Wan, F. Zhou, Q. Wu, and R. Q. Hu, "Resource allocation and trajectory design for miso uav-assisted mec networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4933–4948, 2022.

[29] B. Ji, M. Zhang, J. Huang, Y. Huang, L. Xing, T. Li, C. Han, and S. Mumtaz, "Research on offloading strategy of twin uavs edge computing tasks for emergency communication," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 684 – 696, 2023.

[30] R. Zhong, X. Liu, Y. Liu, and Y. Chen, "Multi-agent reinforcement learning in noma-aided uav networks for cellular offloading," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1498–1512, 2022.

[31] L. Chu, X. Li, J. Xu, A. G. Neiat, and X. Liu, "A holistic service provision strategy for drone-as-a-service in mec-based uav delivery," in *2021 IEEE International Conference on Web Services (ICWS)*. IEEE, 2021, pp. 669–674.

[32] W. Zhuang, F. Xing, and Y. Lu, "Task offloading strategy for unmanned aerial vehicle power inspection based on deep reinforcement learning," *Sensors*, vol. 24, no. 7, pp. 2070–2089, 2024.

[33] Y. Ding, C. Liu, X. Zhou, Z. Liu, and Z. Tang, "A code-oriented partitioning computation offloading strategy for multiple users and multiple mobile edge computing servers," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4800–4810, 2020.

[34] M. R. Garey and D. S. Johnson, ""'strong"np-completeness results: Motivation, examples, and implications," *Journal of the ACM (JACM)*, vol. 25, no. 3, pp. 499–508, 1978.

[35] C. Swain, M. N. Sahoo, and A. Satpathy, "Leto: An efficient load balanced strategy for task offloading in iot-fog systems," in *2021 IEEE International Conference on Web Services (ICWS)*, 2021, pp. 459–464.

[36] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading tasks with dependency and service caching in mobile edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 11, pp. 2777–2792, 2021.

[37] X. He, J. Zheng, Q. He, H. Dai, B. Liu, W. Dou, and G. Chen, "Confect: Computation offloading for tasks with hard/soft deadlines in edge computing," in *2021 IEEE International Conference on Web Services (ICWS)*, 2021, pp. 262–271.

[38] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3424–3438, 2020.

[39] H. Guo, Y. Wang, J. Liu, and C. Liu, "Multi-uav cooperative task offloading and resource allocation in 5g advanced and beyond," *IEEE Transactions on Wireless Communications*, vol. 23, no. 1, pp. 347–359, 2023.

[40] A. Manzoor, K. Kim, S. R. Pandey, S. A. Kazmi, N. H. Tran, W. Saad, and C. S. Hong, "Ruin theory for energy-efficient resource allocation in uav-assisted cellular networks," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3943–3956, 2021.

[41] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *2016 International Conference on Learning Representations (ICLR)*, 2016.

[42] L. Zhang, Z. Zhang, L. Min, C. Tang, H. Zhang, Y. Wang, and P. Cai, "Task offloading and trajectory control for uav-assisted mobile edge computing using deep reinforcement learning," *IEEE Access*, vol. 9, pp. 53 708–53 719, 2021.

Junna Zhang received the Ph.D. degree in Institute of Network Technology from Beijing University of Posts and Telecommunications in 2019. She is currently a professor in the College of Computer and Information Engineering, Henan Normal University, China. Her research interests include Edge Computing and Service Computing, and she authored or coauthored more than thirty papers in these fields. She received the Best Paper Award of CCF NCSC in 2022. She is a member of the IEEE.

Guoxian Zhang is currently working toward the master degree in the College of Computer and Information Engineering, Henan Normal University, China. His research interests include Edge Computing and Service Computing.

Xinxin Wang received the bachelor's degree in Computer and Information Technology from Xinyang Normal University in 2021. She is currently working toward the master's degree with the College of Computer and Information Engineering, Henan Normal University, China. Her research interests include Edge Computing and Service Computing.

Xiaoyan Zhao received a Ph.D., degree in School of Software from Sun Yat-sen University in 2010 and a Ph.D. degree in the Department of Computer Science and Engineering from Hong Kong University of Science and Technology in 2014. He is currently an associate professor at South China University of Technology. His research interests include mobile crowdsensing, edge computing, and mobile computing. He is a member of the IEEE and the ACM.

Peiyan Yuan received the B.S., the M.S. and Ph.D. degrees from Henan Normal University, Xinxiang, China, Wuhan University of Technology, Wuhan, China, and Beijing University of Posts and Telecommunications, Beijing, China, respectively, all in computer science. He is a professor of Computer Science, Henan Normal University. He also worked as a post doctoral researcher at the University of Texas at Dallas, USA. His research interests include future networks and distributed systems, and he authored or coauthored more than fifty papers and one book in these fields. He won the national scholarship for Ph. D students from Ministry of Education of the P. R. China in 2012, and he received the Best Paper Award of IEEE CSE in 2014. He is a senior member of CCF and a member of ACM and IEEE.

Hu Jin (Senior Member, IEEE) received the B.E. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2004, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2006 and 2011, respectively. From 2011 to 2013, he was a Postdoctoral Fellow with the University of British Columbia, Vancouver, BC, Canada. From 2013 to 2014, he was a Research Professor with Gyeongsang National University, Tongyeong, Korea. Since 2014, he has been with the School of Electrical Engineering, Hanyang University, Ansan, Korea, where he is currently a Professor. His research interests include medium-access control and radio resource management for random access networks and scheduling systems considering advanced signal processing, and queueing performance.