drones

MDPI

Article

Intelligent Online Offloading and Resource Allocation for HAP Drones and Satellite Collaborative Networks

Cheng Gao [1], Xilin Bian [1], Bo Hu [1,*], Shanzhi Chen [2] and Heng Wang [3]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; gaoch@bupt.edu.cn (C.G.); bianxilin2018@bupt.edu.cn (X.B.)
[2] State Key Laboratory of Wireless Mobile Communication, China Academy of Telecommunication Technology, Beijing 100049, China; chensz@cict.com
[3] Chinatelecom Research Institute, Beijing 102209, China; wangh26@chinatelecom.cn
* Correspondence: hubo@bupt.edu.cn

Abstract: High-altitude platform (HAP) drones and satellites collaborate to form a network that provides edge computing services to terrestrial internet of things (IoT) devices, which is considered a promising method. In this network, IoT devices' tasks can be split into multiple parts and processed by servers at non-terrestrial nodes in different locations, thereby reducing task processing delays. However, splitting tasks and allocating communication and computing resources are important challenges. In this paper, we investigate the task offloading and resource allocation problem in multi-HAP drones and multi-satellite collaborative networks. In particular, we formulate a task splitting and communication and computing resource optimization problem to minimize the total delay of all IoT devices' tasks. To solve this problem, we first transform and decompose the original problem into two subproblems. We design a task splitting optimization algorithm based on deep reinforcement learning, which can achieve online task offloading decision-making. This algorithm structurally designs the actor network to ensure that output actions are always valid. Furthermore, we utilize convex optimization methods to optimize the resource allocation subproblem. The simulation results show that our algorithm can effectively converge and significantly reduce the total task processing delay when compared with other baseline algorithms.

Keywords: HAP drone; LEO satellite; task offloading and resource allocation; deep reinforce learning

Citation: Gao, C.; Bian, X.; Hu, B.; Chen, S.; Wang, H. Intelligent Online Offloading and Resource Allocation for HAP Drones and Satellite Collaborative Networks. *Drones* **2024**, *8*, 245. https://doi.org/10.3390/drones8060245

Academic Editors: Chinthaka Premachandra and Tomotaka Kimura

Received: 16 April 2024
Revised: 27 May 2024
Accepted: 28 May 2024
Published: 5 June 2024

1. Introduction

The internet of things (IoT) is considered an important emerging technology that can change human life and has brought great convenience to human society [1]. Internet of things applications, such as smart cities [2], smart agriculture [3], maritime detection [4], augmented reality (AR)/virtual reality (VR) [5], etc., have made great progress and raised new demands for future network development [6]. It is predicted that by 2027, more than 30 billion IoT devices will be deployed around the world [7]. Limited by geographical location, it is difficult for traditional terrestrial networks to provide reliable services for IoT devices in maritime areaa, remote areas, and other areas without terrestrial network coverage [8]. Therefore, non-terrestrial networks, unaffected by geographical limitations, can provide reliable wide-area coverage for terrestrial IoT devices, becoming a crucial direction for future network development.

According to the different deployment heights of non-terrestrial network nodes, non-terrestrial networks can be divided into satellite networks and space-based networks. Satellite network nodes are deployed at altitudes ranging from 160 km to 35,786 km above the ground [9]. In the past 20 years, satellite communication networks represented by low-orbit satellite constellations have experienced tremendous development. Typical low-earth orbit (LEO) satellite constellations include SpaceX's Starlink constellation system [10], the Iridium constellation [11], the OneWeb constellation [12], the Telesat constellation [13], etc.

By deploying edge servers, satellite network nodes can directly provide task processing services to terrestrial IoT devices. However, due to the high flight altitude of satellites and the limited energy and signal transmission power of a large number of terrestrial IoT devices, terrestrial IoT devices face challenges in sending data to satellites stably and quickly.

Different from satellite networks, space-based networks represented by high altitude platform (HAP) drones are deployed at an altitude of several hundred meters to 20 km above the ground [14]. Their flying heights are much lower than satellite networks. Therefore, ground IoT devices can transmit data to air nodes more stably and at a higher speed. According to the definition of the International Telecommunication Union (ITU), a HAP drone is unmanned aerial vehicle (UAV) deployed at an altitude of 20 km above the ground [15,16]. It can either hover to provide stable communication services for ground equipment or be deployed mobile to respond to emergency service requirements. Since the distance from the ground IoT device to the HAP drone is much smaller than the distance from the ground IoT device to the satellite, the wireless channel from the ground IoT device to the HAP drone is more stable and has a higher channel gain. Therefore, the ground IoT device can communicate with the HAP drone at a higher rate. Similarly, HAP drones can carry edge servers to provide edge computing services to IoT devices.

However, since the load capacity of HAP drones and LEO satellites is limited, a single HAP drone and satellite cannot guarantee the provision of stable and effective task processing services for the terrestrial IoT devices it serves. Therefore, sharing the computing resources of the server through the collaboration of multiple non-terrestrial nodes has become a promising method to improve the service capabilities of non-terrestrial networks. By dividing the task into multiple sub-parts and processing them on servers in different locations, the network's task processing capabilities can be effectively improved. However, for HAP drones and satellite collaborative networks, how to reasonably split tasks and optimize the allocation strategy of communication and computing resources have become important issues that need to be solved.

Inspired by these challenges, we investigate the problem of task splitting and resource allocation under the collaborative network of HAP drones and LEO satellites, as shown in Figure 1. In this paper, we propose that HAP drones can not only cooperate with satellites but also with each other to jointly provide computing services for ground IoT devices. This solution is conducive to expanding the task processing capabilities of the double-layer network and improving the resource utilization of the network. Correspondingly, we considered factors such as the task processing tolerance delay of ground IoT devices, the limited computing resources of satellites, HAP drones, and IoT devices, and the limited communication resources between devices, and we formulated the problem of minimizing the total task processing delay.

In order to minimize the total task processing delay, we propose an intelligent online offloading and resource allocation algorithm based on a combination of deep reinforcement learning and convex optimization. We first decompose the original optimization problem into a task splitting optimization sub-problem and a resource allocation optimization sub-problem. For the task splitting optimization sub-problem, we designed an intelligent solving algorithm based on the DDPG algorithm. For the resource allocation optimization sub-problem, we used a convex optimization algorithm to solve it. Through joint optimization of the two sub-problems, the efficiency of task offloading and resource allocation can be improved, and the total task processing delay can be reduced. Compared with typical reinforcement learning solutions, the algorithm proposed in this paper does not use deep reinforcement learning methods to directly output task offloading and resource-scheduling strategy, which objectively reduces the scale of the neural network, thereby reducing the training difficulty of deep reinforcement learning and improving the efficiency of neural networks, and improving the convergence speed of the network. Finally, we conduct simulation experiments to verify the convergence and performance of the proposed algorithm. The results demonstrate that, under various scenarios, such as different task data sizes and

different CPU cycles required per bit, the proposed algorithm can reduce the total task processing delay compared to baseline algorithms.

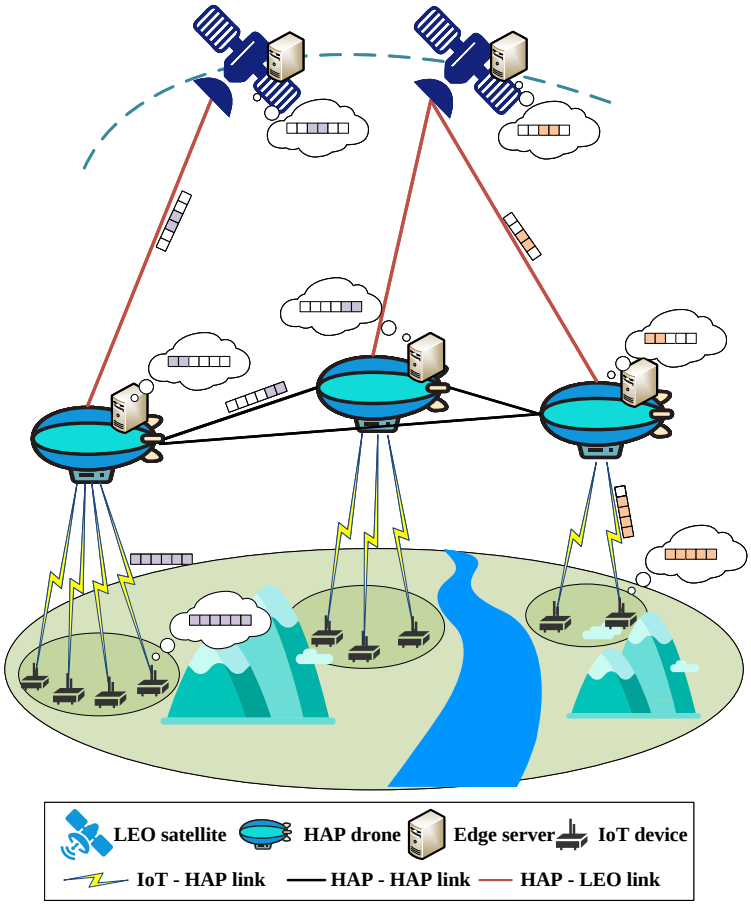

Figure 1. The scenario of LEO-HAP drone collaborative networks.

The main contributions of this paper are summarized as follows:

(1) We construct an edge computing framework for multi-HAP drones and multi-LEO collaboration. Under this framework, ground IoT devices' tasks can be dynamically allocated to multiple HAP drones and LEO satellites for processing. In addition, the communication and computing resources of each node under this framework can also be dynamically allocated.

(2) Considering the task splitting constraints, available resource constraints, and the maximum tolerated delay constraint of the tasks, we construct a task splitting and resource allocation problem to minimize the total system delay. This is a non-convex continuous optimization problem.

(3) We propose a joint optimization algorithm of deep reinforcement learning and convex optimization. We design a task splitting optimization algorithm based on the deep deterministic policy gradient (DDPG) method and solve the optimal resource allocation strategy through a convex optimization algorithm. We design the structure of the actor network to ensure that the actions of DRL are effective.

(4) We verify the convergence and effectiveness of the algorithm proposed in this paper through experiments. By comparing the algorithm convergence under different discount factors and learning rates, we select reasonable neural network parameters. By comparing our algorithm with three other baseline schemes, we verify that the algorithm proposed in this paper can effectively reduce the total system delay.

The rest of the paper is structured as follows. The related works are presented in Section 2. The system model of HAP drones and LEO satellite collaborative networks is introduced in Section 3. In Section 4, we formulate the problem. We introduce the task offloading and resource allocation algorithm proposed in this paper in detail in Section 5. Numerical results are presented to verify the convergence and performance of the proposed algorithm in Section 6. Finally, we conclude the paper in Section 7.

2. Related Works

Recently, there have been many studies on edge computing for satellites. In [17], the authors studied the edge cloud resource scheduling problem of space–air–ground integrated networks (SAGIN). The authors proposed an improved Two_Arch2 algorithm to optimize the resource-scheduling strategy of the SAGIN network to improve the service capabilities of the internet of vehicles. In [18], the authors proposed a satellite edge computing network architecture that supports the IoT and constructed a multi-objective optimization problem that considers system delay, computing power, and energy consumption. The authors proposed a slicing-based scheduling strategy to optimize the offloading sequence and a number of offloading tasks. In [19], the authors studied a hybrid LEO satellites and MEO satellite network for IoT. In order to solve the problem of satellite load imbalance, the authors formulated a joint optimization problem of computing and communication resources and proposed an optimization algorithm based on deep reinforcement learning to solve it. In [20], the authors studied the service chain optimization problem in satellite edge computing scenarios. Aiming to minimize transmission delay, the authors designed two algorithms, an approximation algorithm and an online algorithm.

In addition, there are also studies on HAP drones carrying edge servers and providing edge computing services. Qiqi Ren et al. [21] studied HAP drone and ground network collaboration to provide computing offloading services for ground transportation. Considering the joint optimization of cache, computing, and communication resources, the authors used multi-agent reinforcement learning and the Lagrange multiplier method to solve the problem, which effectively reduced task processing delay. In [22], the authors introduced NOMA technology and deployed edge servers on HAP to provide computing offloading services for ground users. Taking into account power, transmission bandwidth, and maximum tolerated delay constraints, the authors proposed a transmission and deployment joint optimization algorithm based on successive convex approximation to minimize system energy consumption. In [23], the authors also studied the HAP network that supports NOMA. The authors proposed a power control algorithm based on DDPG to reduce energy consumption and task processing delay. In [24], the authors considered edge computing and wireless power transfer at the same time. In addition to offloading tasks to HAP drones for processing, ground IoT devices can also charge themselves through ground access points. The authors considered the problem of maximizing computing power while minimizing IoT energy consumption and designed a heuristic algorithm to solve the problem.

Research on edge computing for collaboration between HAP drones and satellites mainly focuses on resource scheduling between HAP drones and satellites. In [25], the authors studied the task offloading and resource allocation problem in the scenario where HAP carries an edge server and cooperates with the LEO satellite network. The HAP drone can directly provide computing services to ground vehicles or forward data to the ground center for processing through LEO satellites. In [26], the authors studied the application prospects of machine learning in resource scheduling problems in space–space–ground integrated networks. The authors designed an optimization algorithm based on a deep neural network to realize intelligent user scheduling in space–space–ground integrated networks. In [27], the authors studied the edge computing network where HAP and satellites collaborate. Under this framework, users' tasks can be processed collaboratively by HAP and satellites. The author proposes a task offloading and resource allocation algorithm based on block coordinate descent to improve network service capabilities.

At present, in edge computing research under HAP drones and satellite collaborative networks, HAP drones directly collaborate with satellites to provide computing services for ground users [25–28]. In this paper, HAP drones can collaborate not only with satellites but also with each other to jointly provide computing services to users on the ground. This method can expand the computing resources available to each user, and it is beneficial in reducing the delay in task processing.

In addition, many current works, such as [23,25,29,30], directly design reinforcement learning algorithms to output task offloading and resource allocation strategies. In this way, the output size of the neural network (including task offloading strategy and resource allocation strategy) is larger, resulting in a large network scale and a long training time. In this paper, we propose an optimization framework that combines deep reinforcement learning with convex optimization. Reinforcement learning only outputs task splitting strategies, and the optimization of resource allocation is realized by the convex optimization method, which can reduce the output size of neural networks and reduce the difficulty of network training.

3. System Model of HAP Drones and LEO Satellite Collaborative Networks

In this section, we consider the HAP drones and LEO satellite collaborative network, as shown in Figure 1. In this network, there are M LEO satellites flying at an altitude of 200 km [10]. They carry edge servers to provide computing services to terrestrial IoT devices. In the stratosphere, 20 km above the ground, N HAP drones are hovering. They also carry edge servers, so they can also provide computing services to terrestrial IoT devices.

We denote the collection of LEO satellites as $\mathcal{M} = \{1, 2, \ldots, M\}$. For LEO m, its computing capability can be denoted as F_m. The set of HAP drones can be labeled as $\mathcal{N} = \{1, 2, \ldots, N\}$, and the maximum computing capacity of HAP drone n can be denoted as F_n. There are J terrestrial IoT devices directly connected to different HAP drones that can offload their tasks to HAP drones. In each time slot, the input task size of IoT j can be modeled as D_j, the computing density can be labeled as c_j, and the maximum tolerable delay of this task is T_j.

3.1. Communication Model

In this system, there are two channels to consider: the IoT–HAP drone channel and the HAP drone–LEO channel.

3.1.1. IoT–HAP Drone Communication Model

The channel gain between terrestrial IoT device j and HAP drone n can be modeled as [28]

$$g_j^n = \left(\frac{c}{4\pi d_j^n f_c}\right)^2 G_j^H \left|h_j^H\right|^2 \tag{1}$$

where c is the speed of light, which is equal to 3.0×10^8 m/s. d_j^n is the distance between terrestrial IoT device j and HAP drone n. f_c is the carrier frequency of transmission signal. G_j^H is the attenuation gain, which is related to the environment. In this paper, we set the HAP drone's antenna gain to 17 dBi [31]. h_j^n is the small scale fading component that satisfies Rician distribution, and the Rician factor is 10 dB [31].

For the HAP drone n, the transmission date from the IoT $j (j \in \mathcal{J}_n)$ can be expressed as follows [32]:

$$R_j^n \leq C_j^n = b_j^n \log_2\left(1 + \frac{P_j g_j^n}{N_0 b_j^n}\right), \tag{2}$$

where N_0 is the spectral density of additional white Gaussian noise (AWGN). b_j^n is the wireless channel bandwidth between the terrestrial IoT device j and HAP drone n.

3.1.2. HAP Drone–LEO Satellite Communication Model

Because both HAP drones and LEO satellites are located at altitudes of at least 20 km above the ground, we consider the wireless channel between HAP drones and LEO satellites to be line-of-sight (LoS) channels, which comply with the free-space path loss model. According to 3GPP TR 38.811 [33], we can model this channel as follows:

$$g_n^m = 32.45 + 20\log_{10}(d_n^m) + 20\log_{10}(f_c^H), \tag{3}$$

where d_n^m is the distance between the HAP drone n and LEO satellite m. f_c^H represents the carrier frequency of the wireless signal transmitted from the HAP drone to the LEO satellite, and we set it as $f_c^H = 31$ GHz.

The corresponding data rate between the HAP drone n and LEO satellite m can be formulated as follows [34]:

$$R_n^m \leq C_n^m = b_n^m \log_2(1 + \frac{p_n^m G_{ts} G_{tr} g_n^m L_{as}}{N_0 b_n^m}), \tag{4}$$

where G_{ts} is the transmit antenna gain of the HAP drone, and G_{ts} is the receive antenna gain of the LEO satellite. L_{as} represents the path loss caused by environmental and atmospheric effects. b_n^m is the wireless channel bandwidth between the HAP drone n and LEO satellite m.

3.2. Computing Model

Each user's task can be divided into four parts: one part can be processed locally, while the remaining parts can be processed on the HAP drone's edge server or LEO satellite's edge server. Below, we introduce each part in detail.

- Local processing: Terrestrial users can put part of the task on the local CPU for processing. We denote $x_j (x_j \in [0,1], j \in \mathcal{J})$ as the proportion of task processed locally. Therefore, the data size of the task processed locally is $x_j D_j$, and the required CPU numbers can be expressed as $x_j D_j c_j$.
- Processing on directly connected HAP drones: In addition to the amount of data processed locally, the user will transmit other parts to the HAP drone directly connected to it. The amount of data in this part is $(1 - x_j)$. After the HAP drone receives this part of the data, it will directly put part of it on its own server for processing. We express the proportion of this part as $y_j (y_j \in [0,1], j \in \mathcal{J})$. Therefore, the data size and required CPU numbers can be denoted as $y_j D_j$ and $y_j D_j c_j$.
- Processing on forwarded HAP drones: The HAP drone can also offload part of the task to the HAP drone connected to it and process it on its server. The proportion of this part of the task is $z_j (z_j \in [0,1], j \in \mathcal{J})$, and the amount of data and the required CPU numbers are $z_j D_j$ and $z_j D_j c_j$.
- Processing on LEO satellites: The HAP drone is connected to the LEO satellite, so the remaining task can be forwarded to the LEO drone and processed on the LEO's server. The proportion of this part of the task is $w_j (w_j \in [0,1], j \in \mathcal{J})$.

3.3. Overall Delay Analysis

3.3.1. Local Processing

For terrestrial IoT device j, it can process part of the task on the local CPU. Assuming that the processing frequency of IoT j is f_j^l, the local processing delay can be written as follows:

$$t_j^l = \frac{x_j D_j c_j}{f_j^l}. \tag{5}$$

3.3.2. Processing on Directly Connected HAP Drones

In addition to the parts processed locally, terrestrial IoT devices will offload other parts to the HAP drone directly connected to it, and then forward them to the served using

the HAP drone, another HAP drone, or a LEO connected to it for processing. The amount of data sent by the terrestrial IoT device j is $(y_j + z_j + w_j)D_j$; therefore, the transmission delay can be expressed as follows:

$$\begin{aligned} t_j^{n,trans} &= \frac{\alpha_j^n (y_j + z_j + w_j) D_j}{R_j^n}, \\ &= \frac{\alpha_j^n (y_j + z_j + w_j) D_j}{b_j^n \log_2(1 + \frac{p_j g_j^n}{N_0 b_j^n})}, \end{aligned} \tag{6}$$

where $\alpha_j^n \in \{0, 1\}$ represents the connection relationship between the IoT j and the HAP drone n. If the IoT j is connected to the HAP drone n, $\alpha_j^n = 1$; otherwise, $\alpha_j^n = 0$.

The processing delay of the IoT j's task in the HAP drone n is as follows:

$$t_j^{n,dirc} = \frac{\alpha_j^n y_j D_j c_j}{f_j^n}. \tag{7}$$

Therefore, the total processing of the IoT j's tasks on the directly connected HAP drone can be expressed as follows:

$$\begin{aligned} t_j^{HAP,dirc} &= \sum_{n \in \mathcal{N}} (t_j^{n,trans} + t_j^{n,dirc}), \\ &= \sum_{n \in \mathcal{N}} \alpha_j^n (\frac{(y_j + z_j + w_j) D_j}{b_j^n \log_2(1 + \frac{p_j g_j^n}{N_0 b_j^n})} + \frac{y_j D_j c_j}{f_j^n}). \end{aligned} \tag{8}$$

3.3.3. Processing on Forwarded HAP Drones

After receiving the task from terrestrial IoT devices, the HAP drone can also forward it to other HAP drones connected to it. We assume that HAP drones can be connected through laser links [35], and the transmission bandwidth is very large. Therefore, we ignore the delay in data forwarding between HAP drones. The overall delay of task processing on directly connected HAP drones can be formulated as follows:

$$\begin{aligned} t_j^{HAP,fd} &= \sum_{n \in \mathcal{N}} \alpha_j^n (\frac{(y_j + z_j + w_j) D_j}{R_j^n} + \sum_{n' \in \mathcal{N}/n} \beta_{n,n'} t_j^{n,n'}), \\ &= \sum_{n \in \mathcal{N}} \alpha_j^n (\frac{(y_j + z_j + w_j) D_j}{b_j^n \log_2(1 + \frac{p_j g_j^n}{N_0 b_j^n})} + \sum_{n' \in \mathcal{N}/n} \beta_{n,n'} \frac{z_j D_j c_j}{f_j^{n',fd}}), \end{aligned} \tag{9}$$

where $\beta_{n,n'} \in \{0, 1\}$ denotes whether HAP drone n is connected to HAP drone n'. If HAP drone n is connected to HAP drone n', $\beta_{n,n'} = 1$; otherwise, $\beta_{n,n'} = 0$.

3.3.4. Processing on LEO Satellites

In addition HAP drones can also forward tasks to connected LEO satellites. After the task is forwarded to LEO, it can be processed on the server carried by LEO. The overall delay of the task processed on the LEO satellite can be formulated as follows:

$$t_j^{LEO} = \sum_{n \in \mathcal{N}} \alpha_j^n (\frac{(y_j + z_j + w_j) D_j}{R_j^n} + \sum_{m \in \mathcal{M}} \zeta_{n,m} (\frac{w_j D_j}{R_n^m} + \frac{w_j D_j c_j}{f_j^m})), \tag{10}$$

where $\zeta_{n,m} \in \{0, 1\}$ denotes whether the HAP drone n is connected to the LEO satellite m. If the HAP drone n is connected to the LEO satellite m, $\zeta_{n,m} = 1$; otherwise, $\zeta_{n,m} = 0$. f_j^m is the computing frequency assigned by the LEO satellite m to the terrestrial IoT device j.

Therefore, the total delay of task j can be formulated as follows:

$$t_j = \max\{t_j^l, t_j^{HAP,dirc}, t_j^{HAP,fd}, t_j^{LEO}\}. \tag{11}$$

For all IoT devices, the total delay can be calculated as follows:

$$t_{sum} = \sum_{j \in \mathcal{J}} t_j. \tag{12}$$

4. Problem Formulation

The original problem of this paper can be expressed as follows:

$$\begin{aligned}
\mathcal{OP}: \quad & \min_{\Theta,\Psi} t_{sum}(\Theta, \Psi), && \text{(13a)}\\
\text{s.t.} \quad & C1: \ 0 \le x_j, y_j, z_j, w_j \le 1, \forall j \in \mathcal{J}, && \text{(13b)}\\
& C2: \ x_j + y_j + z_j + w_j = 1, \forall j \in \mathcal{J}, && \text{(13c)}\\
& C3: \ 0 \le p_j \le P_j, \forall j \in \mathcal{J}, && \text{(13d)}\\
& C4: \ 0 \le p_n^m \le\le P_n, \forall n \in \mathcal{N}, m \in \mathcal{M}, && \text{(13e)}\\
& C5: \ 0 \le f_j^l \le F_j^l, \forall j \in \mathcal{J}, && \text{(13f)}\\
& C6: \ 0 \le f_j^l, f_j^n, f_j^m, \forall j \in \mathcal{J}, n \in \mathcal{N}, m \in \mathcal{M}, && \text{(13g)}\\
& C7: \ \sum_{j \in \mathcal{J}} (f_j^n + f_j^{n,fd}) \le F_n, \forall n \in \mathcal{N}, && \text{(13h)}\\
& C8: \ \sum_{j \in \mathcal{J}} f_j^m \le F_m, \forall m \in \mathcal{M}, && \text{(13i)}\\
& C9: \ t_j \le T_j, \forall j \in \mathcal{J}, && \text{(13j)}
\end{aligned}$$

where $\Theta = \{x_j, y_j, z_j, w_j \mid \forall j \in \mathcal{J}\}$ is the collection of the task splitting strategy. $\Psi = \{p_j, p_n^m, f_j^l, f_j^n, f_j^{n,fd}, f_j^m \mid \forall j \in \mathcal{J}, n \in \mathcal{N}, \text{and } m \in \mathcal{M}\}$ denote the collection of multiple resource allocation, including power control, computing resource allocation of IOT devices, HAP drones, and LEO satellites. In this problem, (C1) and (C2) are constraints on task partitioning, indicating that the task partitioning variables must be continuous values between 0 and 1, and for each task j, the sum of x_j, y_j, z_j, and w_j must be 1. (C3) and (C4) are constraints on transmit power, ensuring that the transmit power does not exceed the maximum available transmit power of the device. (C5) to (C8) are constraints on computation frequency, ensuring that the computation frequency allocated to each task by each IoT does not exceed its maximum computation capacity. (C9) indicates that the overall delay for each IoT must not exceed its maximum tolerable delay.

5. Algorithm Design for $\mathcal{OP}$

5.1. Problem Conversion

Firstly, to simplify the problem-solving process, we introduce auxiliary variables $\boldsymbol{t} = \{t_1, t_2, \ldots, t_J\}$, representing the overall delay for each terrestrial IoT device. Therefore, the original problem $\mathcal{OP}$ can be rewritten as follows:

$$\mathcal{P}1: \quad \min_{\Theta,\Psi,\boldsymbol{t}} \sum_{j\in\mathcal{J}} t_j, \tag{14a}$$

$$\text{s.t.} \quad \frac{x_j D_j c_j}{f_j^l} \le t_j, j \in \mathcal{J}, \tag{14b}$$

$$\sum_{n\in\mathcal{N}} \alpha_j^n \left(\frac{(y_j+z_j+w_j)D_j}{b_j^n \log_2\left(1+\frac{p_j g_j^n}{N_0 b_j^n}\right)} + \frac{y_j D_j c_j}{f_j^n}\right) \le t_j, j \in \mathcal{J}, \tag{14c}$$

$$\sum_{n\in\mathcal{N}} \alpha_j^n \left(\frac{(y_j+z_j+w_j)D_j}{b_j^n \log_2\left(1+\frac{p_j g_j^n}{N_0 b_j^n}\right)} + \sum_{n'\in\mathcal{N}/n} \beta_{n,n'} \frac{z_j D_j c_j}{f_j^{n',fd}}\right) \le t_j, j \in \mathcal{J}, \tag{14d}$$

$$\sum_{n\in\mathcal{N}} \alpha_j^n \left(\frac{(y_j+z_j+w_j)D_j}{R_j^n} + \sum_{m\in\mathcal{M}} \zeta_{n,m}\left(\frac{w_j D_j}{R_n^m} + \frac{w_j D_j c_j}{f_j^m}\right)\right) \le t_j, j \in \mathcal{J}, \tag{14e}$$

$$(13b) - (13i). \tag{14f}$$

Since the variables Θ and Ψ are coupled, this makes problem $\mathcal{P}1$ non-convex and difficult to be solved directly. Therefore, to reduce the complexity of the problem, we decompose problem $\mathcal{P}1$ into two subproblems: Subproblem 1 involves determining task splitting decisions, while Subproblem 2 involves determining the resource scheduling decisions. Below, we explain the two subproblems, respectively.

Subproblem 1, used to solve the task splitting strategy, can be represented as follows:

$$\mathcal{SP}1: \quad \min_{\Theta} V(\Theta), \tag{15a}$$

$$\text{s.t.} \quad (13b),(13c). \tag{15b}$$

Subproblem 2 is used to determine the optimal resource scheduling strategy. It is important to note that the solution to Subproblem 2 is carried out under the assumption that the task splitting strategy is fixed, i.e., Subproblem 2 can be formulated as follows:

$$\mathcal{SP}2: \quad V(\Theta) = \min_{\Psi,\boldsymbol{t}} \sum_{j\in\mathcal{J}} t_j, \tag{16a}$$

$$\text{s.t.} \quad (13d) - (13i),(14b) - (14e). \tag{16b}$$

Below, we prove that jointly solving $\mathcal{SP}1$ and $\mathcal{SP}2$ can obtain the optimal solution of the original problem.

Proof. Assume that two task splitting strategies, Θ_1 and Θ_2, are given. Their corresponding resource allocation strategies can be obtained by solving $\mathcal{SP}2$. If $V(\Theta_1) < V(\Theta_2)$, then Θ_1 is better than Θ_2. That is, (Θ_1, Ψ_1) is better than (Θ_2, Ψ_2). By obtaining different Θ, we can obtain (Θ, Ψ) that can make the total task processing workload smaller. Furthermore, by solving $\mathcal{SP}1$ to obtain Θ, and the Ψ corresponding to Θ by solving $\mathcal{SP}2$, we can obtain the optimal solution to the original problem. This proves that, by jointly solving $\mathcal{SP}1$ and $\mathcal{SP}2$, we can obtain the optimal solution to the original problem. □

In this paper, the quality of the solution obtained for Subproblem 1 (i.e., the task splitting strategy) depends on solving Subproblem 2. For each solution derived from Subproblem 1, Subproblem 2 must be tackled to determine the optimal resource allocation decision given the task splitting strategy. Then, by iteratively solving Subproblem 1, optimized task offloading and resource allocation decisions are obtained. Therefore, we will first introduce the solution method for Subproblem 2, followed by an explanation of how to address Subproblem 1.

5.2. Algorithm Design for the Optimization of $\mathcal{SP}2$

Under the fixed task splitting strategy (Θ), the total delay of all tasks depends on the allocation of communication and computation resources. Since we are aiming to minimize the total delay of all IoT devices' tasks, we can infer that when the resource scheduling strategy is optimal, the transmission power is set to the maximum available transmission power of the devices, and the optimal allocation strategy for local computation frequency is determined by the maximum available computation frequency for each user. Therefore, problem $\mathcal{SP}2$ can be rewritten as follows:

$$\mathcal{SP}2a: \quad \min_{\boldsymbol{F},\boldsymbol{t}} \sum_{j\in\mathcal{J}} t_j, \tag{17a}$$

$$\text{s.t.} \quad \frac{x_j D_j c_j}{F_j^l} \le t_j, j \in \mathcal{J}, \tag{17b}$$

$$\sum_{n\in\mathcal{N}} \alpha_j^n \left(\frac{(y_j+z_j+w_j)D_j}{b_j^n \log_2\left(1+\frac{P_j g_j^n}{N_0 b_j^n}\right)} + \frac{y_j D_j c_j}{f_j^n}\right) \le t_j, j \in \mathcal{J}, \tag{17c}$$

$$\sum_{n\in\mathcal{N}} \alpha_j^n \left(\frac{(y_j+z_j+w_j)D_j}{b_j^n \log_2\left(1+\frac{P_j g_j^n}{N_0 b_j^n}\right)} + \sum_{n'\in\mathcal{N}/n} \beta_{n,n'} \frac{z_j D_j c_j}{f_j^{n',fd}}\right) \le t_j, j \in \mathcal{J}, \tag{17d}$$

$$\sum_{n\in\mathcal{N}} \alpha_j^n \left(\frac{(y_j+z_j+w_j)D_j}{R_j^{n,max}} + \sum_{m\in\mathcal{M}} \zeta_{n,m}\left(\frac{w_j D_j}{R_n^{m,max}} + \frac{w_j D_j c_j}{f_j^m}\right)\right) \le t_j, j \in \mathcal{J}, \tag{17e}$$

$$(13g) - (13i), \tag{17f}$$

where $R_j^{n,max}$ and $R_n^{m,max}$ denote the maximum data transmission rates achieved when transmitting power is maximized for IoT devices and HAP drones, respectively.

Clearly, problem $\mathcal{SP}2a$ is a convex optimization problem, and we can directly determine its optimal solution using existing convex optimization solvers. In this paper, since our simulations are implemented in Python, we use CVXPY to solve this problem [36,37].

5.3. Algorithm Design for the Optimization of $\mathcal{SP}1$

To address the task splitting subproblem, we devised a solution framework based on the DDPG method. Initially, we transformed $\mathcal{SP}1$ into a Markov decision process (MDP) model. Subsequently, we employed the DDPG-based method to solve the transformed problem.

5.3.1. MDP

In general, an MDP typically consists of states, actions, a reward function, a state transition function, and a discount factor. In this paper, we designate a specific HAP drone as the agent interacting with the environment, which includes satellites, terrestrial IoT devices, and other HAP drones. The HAP drone collects information such as user data and wireless channel gain to form state representation and then devises a task splitting strategy (i.e., actions) based on the states. Finally, the decision information is transmitted to satellites, HAP drones, and terrestrial IoT devices for execution. Below, we provide complete definitions for each concept.

(1) State Space: In this paper, state is defined as the environment variables of the system, expressed as $s_t = \{\boldsymbol{D}_t, \boldsymbol{C}_t, \boldsymbol{G}_t, \boldsymbol{P}_t, \boldsymbol{F}_t, \boldsymbol{T}_t\}$, specifically including the following:

- $\boldsymbol{D}_t = \{D_j(t) | \forall j \in \mathcal{J}\}$, which represents the data sizes of all terrestrial IoTs' tasks.
- $\boldsymbol{C}_t = \{C_j(t) | \forall j \in \mathcal{J}\}$, which represents computing density of all terrestrial IoTs' tasks.
- $\boldsymbol{G}_t = \{g_j^n, g_n^m | j \in \mathcal{J}, n \in \mathcal{N}, m \in \mathcal{M}\}$, which represents the gains of wireless channels between nodes in the system.
- $\boldsymbol{P}_t = \{P_j(t) | \forall j \in \mathcal{J}\}$, which represents the maximum transmitter power of all terrestrial IoT devices.

- $\boldsymbol{F}_t = \{F_j^l(t), F_n(t), F_m(t) | \forall j \in \mathcal{J}, n \in \mathcal{N}, and m \in \mathcal{M}\}$, which represent maximum computing frequency of terrestrial IoT devices, HAP drones, and LEO satellites.
- $\boldsymbol{T}_t = \{T_j(t) | \forall j \in \mathcal{J}$, which represents the maximum tolerated delay of all terrestrial IoT devices' tasks.

(2) Action Space: In this MDP, the agent needs to output the task splitting strategy for each task. Because each task can be divided into four parts for processing, processing locally at the user, processing at the connected HAP drone, processing at the adjacent HAP drone, and processing at the LEO satellite, action a_t can be expressed as $\{x_j(t), y_j(t), z_j(t), w_j(t) | \forall j \in \mathcal{J}\}$ (i.e., $\Psi(t)$).

(3) Reward Function: The goal of the MDP problem is to maximize rewards, while the goal of this paper is to minimize the total task processing delay, so we take the negative of all task processing delays and then accumulate them as rewards.

$$r_t = -\sum_{j \in \mathcal{J}} t_j. \tag{18}$$

Therefore, the long-term discounted reward can be formulated as follows:

$$U_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{i=0}^{\infty} \gamma^i r_{t+i}, \tag{19}$$

where $\gamma \in [0, 1]$ is the discount factor, which is used to discount the value of future rewards when calculating cumulative rewards. If the discount factor is too low, then the value of future rewards will be severely underestimated, which may lead to the agent making less informed decisions. If the discount factor is too high, the value of future rewards will be overestimated, which may lead to the agent adopting an overly conservative strategy. Therefore, choosing the right discount factor is very important. The total task processing delay is obtained by solving $\mathcal{SP}2a$ based on state and action.

Based on the above discussion, the problem of maximizing the long-term discount reward can be expressed as follows:

$$\max_{\Theta} \sum_{i=0}^{\infty} -\gamma^i \sum_{j \in \mathcal{J}} t_j(i), \tag{20a}$$

$$\text{s.t.} \quad (13b), (13c). \tag{20b}$$

5.3.2. Proposed Task Splitting Algorithm Based on DDPG

The algorithm for the optimization of task splitting in this paper is based on the DDPG method, as shown in Figure 2. In this algorithm, four neural networks are included, namely an actor network, a critic network, a target actor network, and a target critic network. Below, we first introduce the actor and critic networks.

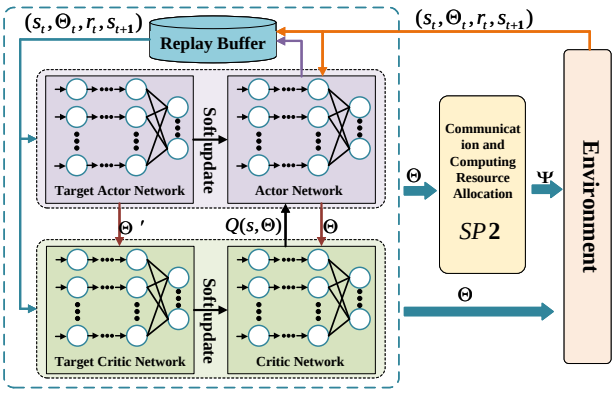

Figure 2. The schematics of the proposed intelligent algorithm.

(1) Actor and Critic Networks:

The output of the actor network is a continuous action value, expressed as $a = \phi(s|\eta)$, where η represents the parameters of the actor network. The output of the critic network is the evaluation of the action a output by the actor network, which can be expressed as $Q(s, a|\theta)$, where θ represents the parameters of the critic network. During each round of training, the actor network will output an action decision a based on the current environment state s, and then the critic network will evaluate the decision; that is, obtain $Q(s, a)$.

Our constructed actor network consists of two parts: a deep neural network (DNN) module and an action validity assurance module based on a normalization function, as shown in Figure 3. The DNN module is responsible for making optimized task partitioning decisions based on the input state. However, the output of the DNN cannot guarantee compliance with constraints (13b) and (13c). Therefore, we designed the action validity assurance module to ensure that the actor network's output meets these constraints.

First, we restructure the DNN's output into a two-dimensional tensor (denoted as δ), with dimensions of J rows and four columns. We represent the j-th row of δ as $\delta_j = \{\delta_{1,j}, \delta_{2,j}, \delta_{3,j}, \delta_{4,j}\}$. Then, we normalize each $\delta_j, j \in \mathcal{J}$, and the processed result is expressed as ϵ_j and ϵ_j satisfies the following:

$$\epsilon_{i,j} = \frac{\delta_{i,j}}{\sum_{k=1}^{4} \delta_{k,j}}, i \in \{1, 2, 3, 4\}, j \in \mathcal{J}. \tag{21}$$

Obviously, each ϵ_j satisfies $\epsilon_{1,j} + \epsilon_{2,j} + \epsilon_{3,j} + \epsilon_{4,j} = 1$; that is, ϵ_j satisfies constraints (13b) and (13c). In this way, we can ensure that the output of the actor network (i.e., the task splitting decision) is valid.

DDPG introduces an experience buffer mechanism to store states, actions, and rewards information, represented as $\{s_t, a_t, r_t, s_{t+1}\}$. During training, to update the critic and actor networks, a batch of Ξ samples is first drawn from the replay buffer. For the sample s_t, a_t, r_t, s_{t+1}, the Critic network first computes the target value, satisfying the following:

$$y_t = r_t + \gamma Q'(s_{t+1}, \phi(s_{t+1}|\eta')|\theta'). \tag{22}$$

Then, by minimizing the loss function $L(\theta) = \sum_{t \in \Xi}(y_t - Q(s_t, a_t|\theta))^2$, we can update the critic network by one step of gradient descent using $\frac{1}{|\Xi|}\nabla_\theta \sum L(\theta)$.

In the same way, we can update the actor network parameters by one step of gradient descent using $\frac{1}{|\Xi|}\sum(\nabla_\eta \phi(s_t) \nabla_{a_t} Q(s_t, a_t|\theta))$.

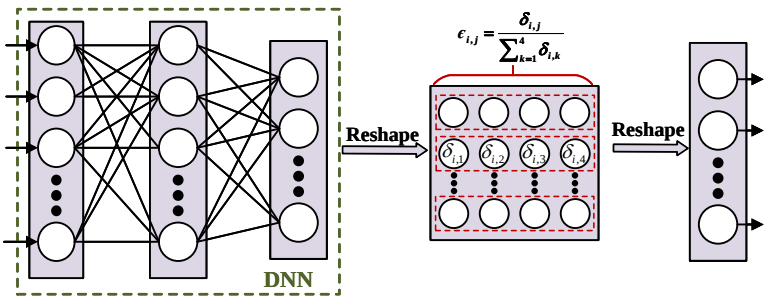

Figure 3. The framework of the actor and target actor networks.

(2) Target Actor and Target Critic Networks:

In order to improve the stability of the training process, reduce the fluctuation of the training process, and reduce the variance of the training process, the DDPG method introduces the target actor network and the target critic network. The target actor network is used to output the best action of the next state, and the target critic network is used to evaluate the Q value of the next state. The structures of the target actor network and the target critic network are the same as the actor network and critic network. Moreover, the

parameters of the target actor network and target critic network (corresponding parameters are η' and θ') are slowly updated from the actor network and critic network. The most commonly used parameter update is soft update, and the update method is as follows [7]:

$$\eta' = \tau\eta + (1-\tau)\eta', \tag{23}$$
$$\theta' = \tau\theta + (1-\tau)\theta', \tag{24}$$

where τ is the soft update factor, and τ satisfies $\tau \ll 1$, which is beneficial to improving the stability of training.

Based on the above analysis, we can summarize the intelligent task splitting and resource allocation algorithm proposed in this paper as Algorithm 1.

Algorithm 1 Task splitting and resource allocation algorithm

1: Randomly initialize the parameters of the actor network and critic network η, θ.
2: Initialize the parameters of the target actor network and target critic network $\eta' = \eta$, $\theta' = \theta$.
3: Initialize the replay buffer Λ.
4: **for** $i = 1, 2, \ldots,$ **do**
5: The actor network generates action decisions a_t based on the current state s_t.
6: Execute action a_t and obtain the corresponding reward r_t based on solving $\mathcal{SP}2a$, an obtain new state s_{t+1}.
7: Store $\{s_t, a_t, r_t, s_{t+1}\}$ into the replay buffer Λ.
8: Randomly sample small samples Ξ from the replay buffer Λ.
9: The critic network calculates the target value $y_t = r_t + \gamma Q'(s_{t+1}, \phi(s_{t+1}|\eta')|\theta')$
10: Update the critic network parameters by calculating the loss function $L(\theta) = \sum_{t\in\Xi}(y_t - Q(s_t, a_t|\theta))^2$, and update critic network by one step of gradient descent using $\frac{1}{|\Xi|}\nabla_\theta \sum L(\theta)$.
11: Update the actor network parameters by one step of gradient descent using $\frac{1}{|\Xi|}\sum(\nabla_\eta \phi(s)\nabla_{a_t} Q(s_t, a_t|\theta))$.
12: Update the parameters of the target actor network and target critic network through soft update (23) and (24).
13: **end for**

5.4. Complexity Analysis

The complexity of the algorithm proposed in this article mainly consists of two parts: reinforcement learning and convex optimization. First, we analyze the complexity of reinforcement learning. The complexity of the reinforcement learning method based on DDPG designed in this paper mainly depends on the scale of the actor network and critic network. We assume that the actor network and critic network are composed of X layer and Y layer fully connected networks. If the number of neurons in each layer of the network is m_x and n_y, the training complexity of DDPG is $O(|A|m_1 + |S|m_{X-1} + \sum_{x=1}^{X-2} m_x m_{x+1} + (|A| + |S|)n_0 + \sum_{y=1}^{Y-2} n_y n_{y+1})$ [38]. Among them, $|A|m_1$, $|S|m_{X-1}$, and $(|A| + |S|)n_0$ represent the computational complexity of the actor network input layer, actor network output layer, and critic network input layer, respectively; $|A|$ and $|S|$ are the number of elements in action and state. The complexity of DDPG online decision-making is $O(|A|m_1 + |S|m_{X-1} + \sum_{x=1}^{X-2} m_x m_{x+1})$.

The computational complexity of the convex optimization algorithm depends on the computational complexity of solving $\mathcal{SP}2a$. The number of variables in $\mathcal{SP}2a$ is $I_1 = (2 + M + 2N)J$, and the number of constraints is $I_2 = 4J + M + N$. Therefore the computation complexity of solving $\mathcal{SP}2a$ is $O((I_1^2 I_2 + I_1^3)I_2^{0.5})$ [39].

In summary, the computational complexity of the algorithm proposed in this paper in the training phase is $O((|A|m_1 + |S|m_{X-1} + \sum_{x=1}^{X-2} m_x m_{x+1} + (|A| + |S|)n_0 + \sum_{y=1}^{Y-2} n_y n_{y+1})$

$(I_1^2 I_2 + I_1^3) I_2^{0.5})$, and the computational complexity of the online decision-making stage is $O((|A|m_1 + |S|m_{X-1} + \sum_{x=1}^{X-2} m_x m_{x+1})(I_1^2 I_2 + I_1^3) I_2^{0.5})$.

6. Performance Evaluation

In this section, we adopt a series of experiments to verify the convergence and performance of our proposed algorithm for the optimization of task offloading and resource allocation.

6.1. Simulation Setup

Our experiments were conducted on a laptop with 13th Gen Intel(R) Core(TM) i7-13650HX 2.60 GHz, 16.0 GB RAM, and NVIDIA GeForce RTX4060 Laptop GPU. All simulations were developed based on Python 3.11, the neural network method was designed based on PyTorch, and the solution of convex optimization problems was performed through CVXPY. We considered the following scenario: two LEO satellites are located in high-altitude orbits 200 km above the ground [27], while three HAP drones are positioned 20 km above the ground, randomly distributed within a square area with a side length of 2 km. On the ground, there are J randomly distributed terrestrial IoT devices. Both the IoT-HAP drone and HAP drone-LEO satellite links adopt the access strategy with maximum channel gain. In each time slot, we assume each terrestrial IoT device generates a task for processing. For each task, the default generated data size is in the range of $[1, 1.2]$ Mbits. The number of CPU cycles required per bit is 1000 [40], and the maximum tolerable latency is $T = 3$ s [27]. In this paper, we assume that the available bandwidth for each wireless channel is 20 MHz, allocated evenly to each device. The maximum local computing capacity for each ground user is the same, set at 1 GHz. Both HAP drones and LEO satellites have a maximum computing capacity of 5 GHz. Other important simulation environment parameters are shown in Table 1.

Choosing appropriate hyper parameters is crucial for the performance of DDPG. In this paper, we mainly focused on the following key hyper parameters: batch size, replay buffer size, learning rates of the actor network and critic network, the soft update factor for the actor network and critic network, and the discount factor.

Our goal in selecting the batch size and replay buffer size was to ensure the stability and convergence speed of DDPG training while minimizing computational and memory resources. We chose a batch size of 64 and a replay buffer size of 100,000 as the hyperparameters for subsequent simulations. The soft update factor was chosen to control the update speed of the weights of the target network (including the actor target network and critic network). We chose $\tau = 0.0001$. A smaller value was chosen to increase the stability of training. Finally, we chose appropriate discount factors and learning rates to ensure the convergence, convergence speed, and performance of the DDPG algorithm in this scenario. In this regard, we designed comparative experiments with different discount factors ([0.95, 0.75, 0.55, 0.35]) and different learning rates ($[1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}]$). The experimental results are shown in Section 6.2. The specific parameters are shown in Table 2.

Table 1. Simulation environment settings.

Parameter	Value
AWGN spectral density (N_0)	-174 dBm/Hz [34]
Maximum transmit power of terrestrial IoT device (P_j)	23 dBm
Maximum transmit power of HAP drone (P_n)	43 dBm
Transmit antenna gain of HAP drone (G_{ts})	0 dB [34]
Receive antenna gain of LEO satellite (G_{tr})	53 dB [34]
Path loss (L_{as})	5.2 dB [34]

Table 2. Algorithm parameter settings.

Parameter	Value
Batch size	64
Replay buffer size	100,000
Learning rate of actor network	0.001
Learning rate of critic network	0.001
Soft update factor of actor network	0.0001
Soft update factor of critic network	0.0001
Discount factor	0.75

6.2. Convergence and Parameter Analysis

In this subsection, we demonstrate the impact of the discount factor and learning rate on the performance of the algorithm proposed in this paper to help us choose appropriate parameter values.

(1) Impact of the Discount Factor: Figure 4 illustrates the impact of the discount factor on the performance of the algorithm proposed in this paper. We selected four typical discount factors: 0.95, 0.75, 0.55, and 0.35. From this figure, it can be observed that all curves converge after approximately 1500 iterations. Among all the curves, the curve corresponding to a discount factor of 0.75 achieved the highest reward, and its convergence speed was only slightly slower than the curve with a discount factor of 0.35. Therefore, in all subsequent simulations, we set the discount factor to 0.75.

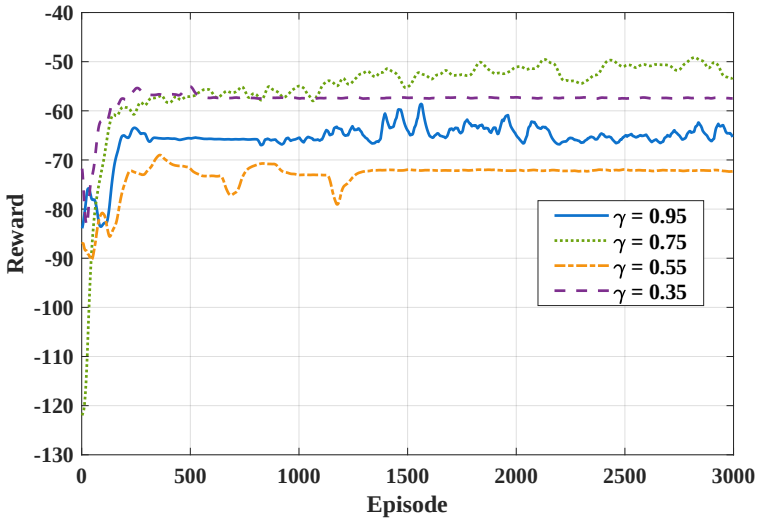

Figure 4. Impact of discount factor.

(2) Impact of the Learning Rate: Figure 5 illustrates the impact of different learning rates on the performance of the algorithm proposed in this paper. In the simulation, we compared four different learning rates: 0.0001, 0.001, 0.01, and 0.1. The learning rates for both the actor and critic networks are the same. From each curve, it can be observed that the curves corresponding to learning rates of 0.0001, 0.001, and 0.01 converge at around 1000 iterations. The curve corresponding to a learning rate of 0.1 does not converge because when the learning rate is too large, the algorithm skips over local or global optimal solutions, resulting in poor performance of the model. Among the four curves, the curve corresponding to a learning rate of 0.01 converges the fastest and exhibits the best performance. Therefore, we chose a learning rate of 0.01 as the parameter for subsequent simulations.

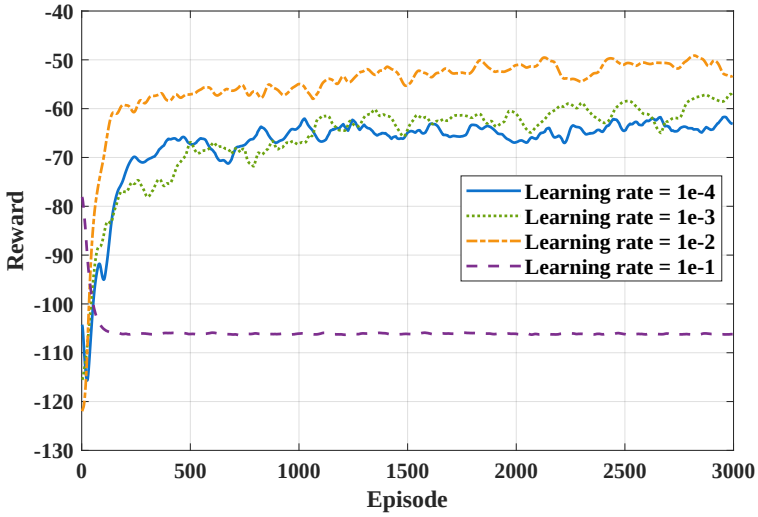

Figure 5. Impact of learning rate.

6.3. Performance Analysis

In order to verify the performance of the algorithm proposed in this paper, we compared the performance of multiple algorithms under different environmental parameters. We set up four baseline algorithms as follows:

- Local processing (LP): All tasks are processed locally on the IoT device's CPU, and the processing computational frequency is set to the IoT's maximum available computational frequency.
- Pure HAP processing (PHAP): All tasks are offloaded to HAP drones for processing, and the HAP drones do not offload tasks to LEO satellites. The resource allocation strategy can be obtained by solving problem $\mathcal{SP}2$.
- Random offloading (RAND): The tasks are randomly allocated (i.e., the sizes of x_j, y_j, z_j and w_j are random and satisfy $x_j + y_j + z_j + w_j = 1$). The resource allocation strategy can be obtained by solving problem $\mathcal{SP}2$.
- Block coordinate descent (BCD-based): Taking the task splitting problem and the resource allocation problem as two sub-problems and alternately solving the two sub-problems iteratively to obtain the optimal solution of the original problem. The original solution of the algorithm is obtained by the RAND algorithm.

In Figure 6, the changes in total system delay under five different algorithms are depicted for various task data sizes. It can be observed from the figure that with the increase in task data size, the overall system delay shows an upward trend for all schemes. This is attributed to the fact that as the data volume increases, devices require more time to transmit and process user tasks. Among them, the curve of the RAND algorithm shows occasional decreases at some points due to its random nature, leading to unstable solution quality. The performance of the BCD-based algorithm depends largely on the quality of the original solution. Therefore, in this figure, the overall system delay obtained by it is better than RAND but inferior to LP and the algorithm proposed in this paper. Among the five algorithms, the proposed algorithm consistently achieves the minimum total delay in all scenarios, confirming the effectiveness of the proposed approach.

Figure 7 shows the changes in total delay of the four algorithms under different computing densities. It can be seen from the figure that, as the computing density $c_j (j \in \mathcal{J})$ increases, the total delay obtained by the four algorithms continues to increase. This is because, as the computing density increases, the number of CPU cycles required to process the task continues to increase, and the processing delay of the task increases, resulting in an increase in the total delay. The RAND algorithm has no solution when $c = 1100$ and 1200. This is because the task splitting strategy obtained by the RAND algorithm cannot

allow the resource allocation problem to meet all constraints, resulting in task timeout for some IoT devices. The initial solution of the BCD-based algorithm is obtained through the RAND algorithm; therefore, when $c = 1100$ and $c = 1200$, the BCD-based algorithm has no solution. Among the five algorithms, the algorithm proposed in this paper can always obtain the smallest total delay, which proves that the algorithm proposed in this article can effectively reduce the total system delay.

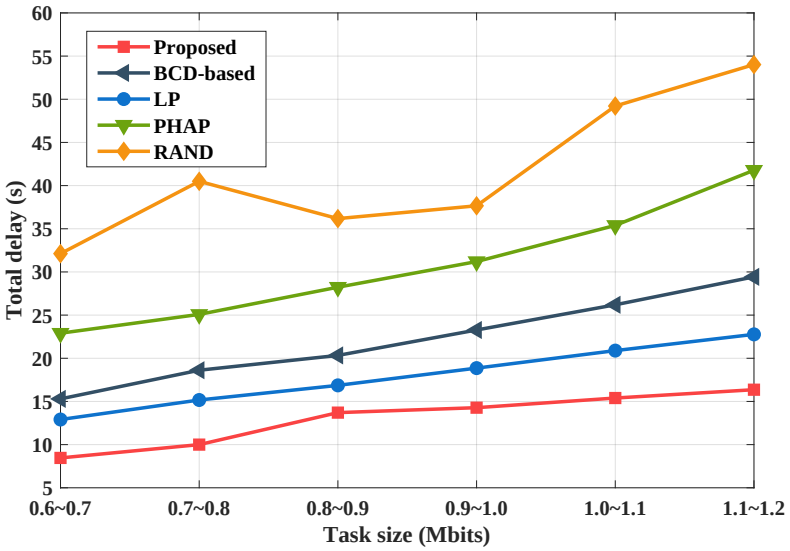

Figure 6. Total delay of the system versus the task size of tasks.

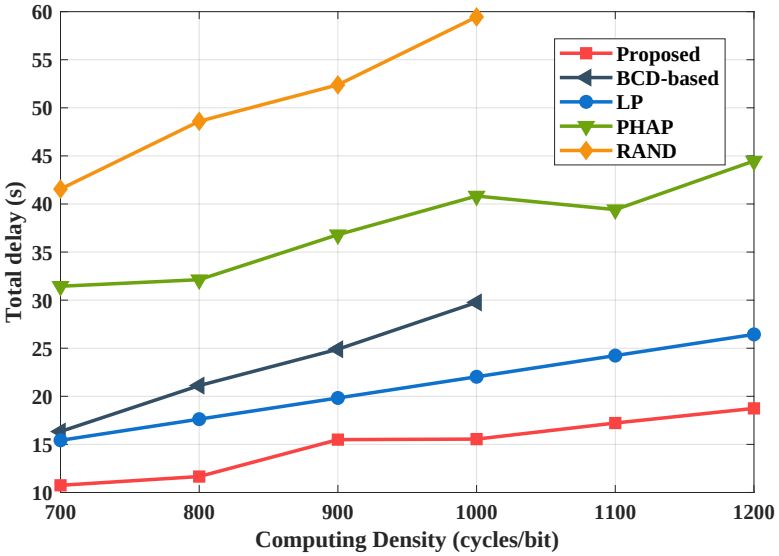

Figure 7. Total delay of the system versus the computing density of all tasks.

Figure 8 shows the total system delay compared with the computing capacity of terrestrial IoTs. With the increase in local computing capacity, except for the PHAP solution, the total system delay of other solutions shows a downward trend. This is because, as the local computing capacity increases, terrestrial IoT devices can process more tasks on the local CPU, thereby reducing overall system delay. In the PHAP solution, no task is processed locally, so the total system delay of this solution does not change with changes in local computing capabilities. Compared with the other four schemes, the proposed algorithm in this paper achieves the smallest total system delay.

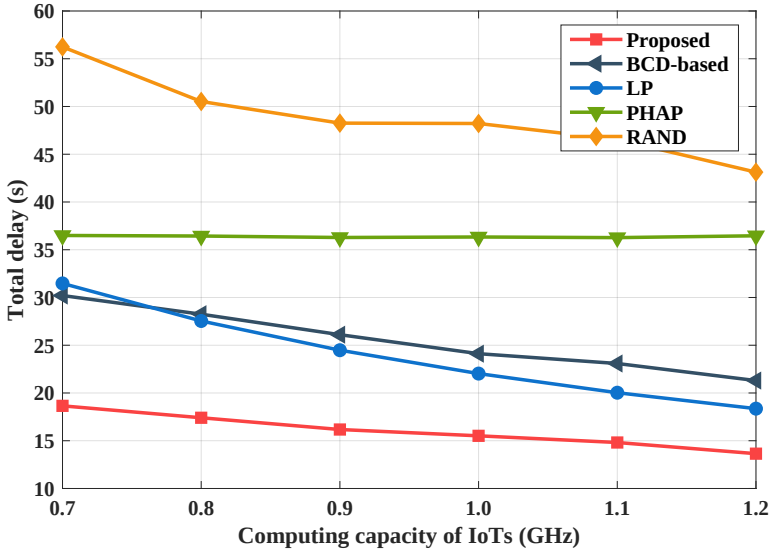

Figure 8. Total delay of the system versus the computing capacity of IoTs.

7. Conclusions

In this paper, we investigated the task offloading and resource allocation problem for multiple-HAP drones and multiple-satellite collaborative networks. The problem was formulated to minimize the total delay of all IoTs' tasks. We transformed and decomposed it into two subproblems: the task splitting optimization subproblem and the resource allocation optimization subproblem. We designed a DDPG-based algorithm to obtain an optimal task splitting strategy, and we optimized resource allocation strategies through convex optimization. Simulation results show that the proposed algorithm can achieve lower total delay compared with baseline algorithms. As the number of IoT devices, HAP drones, and satellites increases, the scale of the neural network in the proposed algorithm will also expand. This can lead to difficulties in training the algorithm and challenges in achieving convergence. Therefore, we believe that future research should focus on designing optimization algorithms that combine multi-agent reinforcement learning with convex optimization techniques.

Author Contributions: Conceptualization, B.H. and C.G.; methodology, C.G., B.H. and X.B.; validation, C.G. and H.W.; formal analysis, C.G. and B.H.; investigation, C.G. and X.B.; writing—original draft preparation, C.G.; writing—review and editing, C.G., X.B. and B.H.; supervision, S.C. and H.W.; project administration, B.H. and S.C.; funding acquisition, B.H. and S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 61931005.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Laghari, A.A.; Wu, K.; Laghari, R.A.; Ali, M.; Khan, A.A. A review and state of art of Internet of Things (IoT). *Arch. Comput. Method Eng.* **2022**, *29*, 1395–1413. [CrossRef]
2. Salih, H.S.; Jaber, M.M.; Ali, M.H.; Abd, S.K.; Alkhayyat, A.; Malik, R.Q. Application of edge computing-based information-centric networking in smart cities. *Comput. Commun.* **2023**, *211*, 46–58. [CrossRef]
3. Luo, Y.; Pu, L. UAV Remotely-Powered Underground IoT for Soil Monitoring. *IEEE Trans. Ind. Inform.* **2024**, *20*, 972–983. [CrossRef]

4. Xu, F.; Yang, F.; Zhao, C.; Wu, S. Deep reinforcement learning based joint edge resource management in maritime network. *China Commun.* **2020**, *17*, 211–222. [CrossRef]
5. Feng, J.; Liu, L.; Hou, X.; Pei, Q.; Wu, C. QoE fairness resource allocation in digital twin-enabled wireless virtual reality systems. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 3355–3368. [CrossRef]
6. Ruby, R.; Yang, H.; de Figueiredo, F.A.P.; Huynh-The, T.; Wu, K. Energy-Efficient Multiprocessor-Based Computation and Communication Resource Allocation in Two-Tier Federated Learning Networks. *IEEE Internet Things J.* **2023**, *10*, 5689–5703. [CrossRef]
7. Zhang, Y.; Hu, J.; Min, G. Digital Twin-Driven Intelligent Task Offloading for Collaborative Mobile Edge Computing. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 3034–3045. [CrossRef]
8. Chen, S.; Sun, S.; Kang, S. System integration of terrestrial mobile communication and satellite communication—The trends, challenges and key technologies in B5G and 6G. *China Commun.* **2020**, *17*, 156–171. [CrossRef]
9. Communications Satellite. Avariable online: https://en.wikipedia.org/wiki/Communications_satellite#Low_Earth_orbit_(LEO) (accessed on 29 May 2024).
10. Ding, C.; Wang, J.-B.; Cheng, M.; Lin, M.; Cheng, J. Dynamic Transmission and Computation Resource Optimization for Dense LEO Satellite Assisted Mobile-Edge Computing. *IEEE Trans. Commun.* **2023**, *71*, 3087–3102. [CrossRef]
11. Huang, Y.; Zhang, X. Microservice Scheduling for Satellite-Terrestrial Hybrid Network with Edge Computing. In Proceedings of the 2022 IEEE/CIC International Conference on Communications in China (ICCC Workshops), Sanshui, Foshan, China, 11–13 August 2022; pp. 24–29.
12. Cassará, P.; Gotta, A.; Marchese, M.; Patrone, F. Orbital Edge Offloading on Mega-LEO Satellite Constellations for Equal Access to Computing. *IEEE Commun. Mag.* **2022**, *60*, 32–36. [CrossRef]
13. Cao, X.; Yang, B.; Shen, Y.; Yuen, C.; Zhang, Y.; Han, Z.; Poor, H.V.; Hanzo, L. Edge-Assisted Multi-Layer Offloading Optimization of LEO Satellite-Terrestrial Integrated Networks. *IEEE J. Sel. Areas Commun.* **2022**, *41*, 381–398. [CrossRef]
14. Renga, D.; Meo, M. Can High Altitude Platform Stations Make 6G Sustainable? *IEEE Commun. Mag.* **2022**, *60*, 75–80. [CrossRef]
15. Euler, S.; Lin, X.; Tejedor, E.; Obregon, E. High-Altitude Platform Stations as International Mobile Telecommunications Base Stations: A Primer on HIBS. *IEEE Veh. Technol. Mag.* **2022**, *17*, 92–100. [CrossRef]
16. Cumalı, İ.; Özbek, B.; Kurt, G.K.; Yanikomeroglu, H. User Selection and Codebook Design for NOMA-Based High Altitude Platform Station (HAPS) Communications. *IEEE Trans. Veh. Technol.* **2022**, *72*, 3636–3646. [CrossRef]
17. Cao, B.; Zhang, J.; Liu, X.; Sun, Z.; Cao, W.; Nowak, R.M.; Lv, Z. Edge–Cloud Resource Scheduling in Space–Air–Ground-Integrated Networks for Internet of Vehicles. *IEEE Internet Things J.* **2022**, *9*, 5765–5772. [CrossRef]
18. Kim, T.; Kwak, J.; Choi, J.P. Satellite Edge Computing Architecture and Network Slice Scheduling for IoT Support. *IEEE Internet Things J.* **2022**, *9*, 14938–14951. [CrossRef]
19. Cui, G.; Duan, P.; Xu, L.; Wang, W. Latency Optimization for Hybrid GEO–LEO Satellite-Assisted IoT Networks. *IEEE Internet Things J.* **2023**, *10*, 6286–6297. [CrossRef]
20. Xia, Q.; Wang, G.; Xu, Z.; Liang, W.; Xu, Z. Efficient Algorithms for Service Chaining in NFV-Enabled Satellite Edge Networks. *IEEE. Trans. Mob. Comput.* **2023**, *early access*. [CrossRef]
21. Ren, Q.; Abbasi, O.; Kurt, G.K.; Yanikomeroglu, H.; Chen, J. Caching and Computation Offloading in High Altitude Platform Station (HAPS) Assisted Intelligent Transportation Systems. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 9010–9024. [CrossRef]
22. Zhang, Y.; Na, Z.; Wang, Y.; Ji, C. Joint power allocation and deployment optimization for HAP-assisted NOMA–MEC system. *Wirel. Netw.* **2022**, 1–13. [CrossRef]
23. Nguyen, T.-H.; Truong, T.P.; Dao, N.-N.; Na, W.; Park, H.; Park, L. Deep Reinforcement Learning-based Partial Task Offloading in High Altitude Platform-aided Vehicular Networks. In Proceedings of the 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 19–21 Ocober 2022; pp. 1341–1346.
24. Nauman, A.; Alruwais, N.; Alabdulkreem, E.; Nemri, N.; Aljehane, N.O.; Dutta, A.K.; Assiri, M.; Khan, W.U. Empowering smart cities: High-altitude platforms based Mobile Edge Computing and Wireless Power Transfer for efficient IoT data processing. *Internet Things* **2023**, *24*, 2542–6605. [CrossRef]
25. Waqar, N.; Hassan, S.A.; Mahmood, A.; Dev, K.; Do, D.-T.; Gidlund, M. Computation Offloading and Resource Allocation in MEC-Enabled Integrated Aerial-Terrestrial Vehicular Networks: A Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 21478–21491. [CrossRef]
26. Dahrouj, H.; Liu, S.; Alouini, M.-S. Machine Learning-Based User Scheduling in Integrated Satellite-HAPS-Ground Networks. *IEEE Netw.* **2023**, *37*, 102–109. [CrossRef]
27. Ding, C.; Wang, J.-B.; Zhang, H.; Lin, M.; Li, G.Y. Joint Optimization of Transmission and Computation Resources for Satellite and High Altitude Platform Assisted Edge Computing. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 1362–1377. [CrossRef]
28. Alsharoa, A.; Alouini, M.-S. Improvement of the Global Connectivity Using Integrated Satellite-Airborne-Terrestrial Networks With Resource Optimization. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 5088–5100. [CrossRef]
29. Xu, C.; Tang, Z.; Yu, H.; Zeng, P.; Kong, L. Digital Twin-Driven Collaborative Scheduling for Heterogeneous Task and Edge-End Resource via Multi-Agent Deep Reinforcement Learning. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 3056–3069. [CrossRef]
30. Jiang, Y.; Liu, J.; Humar, I.; Chen, M.; AlQahtani, S.A.; Hossain, M.S. Age-of-Information-Based Computation Offloading and Transmission Scheduling in Mobile-Edge-Computing-Enabled IoT Networks. *IEEE Internet Things J.* **2023**, *10*, 19782–19794. [CrossRef]

31. Ren, Q.; Abbasi, O.; Kurt, G.K.; Yanikomeroglu, H.; Chen, J. Handoff-Aware Distributed Computing in High Altitude Platform Station (HAPS)—Assisted Vehicular Networks. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 8814–8827. [CrossRef]
32. Fang, F.; Wang, K.; Ding, Z.; Leung, V.C.M. Energy-Efficient Resource Allocation for NOMA-MEC Networks with Imperfect CSI. *IEEE Trans. Commun.* **2021**, *69*, 3436–3449. [CrossRef]
33. *Standard 3GPP TR 38.811 (V15.4.0)*; Study on New Radio (NR) to Support Non-Terrestrial Networks (Release 15). 2020. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3234 (accessed on 29 May 2024).
34. Chen, Q.; Meng, W.; Quek, T.Q.S.; Chen, S. Multi-tier hybrid offloading for computation-aware IoT applications in civil aircraft-augmented SAGIN. *IEEE J. Sel. Areas Commun.* **2022**, *41*, 399–417. [CrossRef]
35. Fidler, F.; Knapek, M.; Horwath, J.; Leeb, W.R. Optical Communications for High-Altitude Platforms. *IEEE J. Sel. Top. Quantum Electron.* **2010**, *5*, 1058–1070. [CrossRef]
36. Diamond, S.; Boyd, S. CVXPY: A python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.* **2016**, *17*, 1–5.
37. Zhang, X.J.; Zhang, X.L.; Yang, W.T. Joint Offloading and Resource Allocation Using Deep Reinforcement Learning in Mobile Edge Computing. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 3454–3466. [CrossRef]
38. Cheng, Z.; Min, M.; Liwang, M.; Huang, L.; Gao, Z. Multiagent DDPG-Based Joint Task Partitioning and Power Control in Fog Computing Networks. *IEEE Internet Things J.* **2022**, *9*, 104–116. [CrossRef]
39. Dai, Y.; Xu, D.; Maharjan, S.; Zhang, Y. Joint computation offloading and user association in multi-task mobile edge computing. *IEEE Trans. Veh. Technol.* **2018**, *67*, 12313–12325. [CrossRef]
40. Ren, Q.; Abbasi, O.; Kurt, G.K.; Yanikomeroglu, H.; Chen, J. High Altitude Platform Station (HAPS) Assisted Computing for Intelligent Transportation Systems. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021; pp. 1–6.