




## Article

# Resource Allocation and Offloading Strategy for UAV-Assisted LEO Satellite Edge Computing

Hongxia Zhang <sup>1</sup>, Shiyu Xi <sup>1</sup>, Hongzhao Jiang <sup>2</sup>, Qi Shen <sup>3,4,\*</sup> , Bodong Shang <sup>5</sup>  and Jian Wang <sup>6</sup> 

<sup>1</sup> Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China; zhanghx@upc.edu.cn (H.Z.); xsy5059@163.com (S.X.)

<sup>2</sup> The Sixth Research Institute of China Electronics Corporation, Beijing 100083, China; lzhmjhz@163.com

<sup>3</sup> Teachers' College, Beijing Union University, Beijing 100023, China

<sup>4</sup> Institute of Science and Technology Education, Beijing Union University, Beijing 100023, China

<sup>5</sup> Eastern Institute for Advanced Study, College of Information Science and Technology, Ningbo 315200, China; bdshang@eias.ac.cn

<sup>6</sup> College of Science, China University of Petroleum (East China), Qingdao 266580, China; wangjiannl@upc.edu.cn

\* Correspondence: sftshenqi@buu.edu.cn

**Abstract:** In emergency situations, such as earthquakes, landslides and other natural disasters, the terrestrial communications infrastructure is severely disrupted and unable to provide services to terrestrial IoT devices. However, tasks in emergency scenarios often require high levels of computing power and energy supply that cannot be processed quickly enough by devices locally and require computational offloading. In addition, offloading tasks to server-equipped edge base stations may not always be feasible due to the lack of infrastructure or distance. Since Low Orbit Satellites (LEO) have abundant computing resources, and Unmanned Aerial Vehicles (UAVs) have flexible deployment, offloading tasks to LEO satellite edge servers via UAVs becomes straightforward, which provides computing services to ground-based devices. Therefore, this paper investigates the computational tasks and resource allocation in a UAV-assisted multi-layer LEO satellite network, taking into account satellite computing resources and device task volumes. In order to minimise the weighted sum of energy consumption and delay in the system, the problem is formulated as a constrained optimisation problem, which is then transformed into a Markov Decision Problem (MDP). We propose a UAV-assisted airspace integration network architecture, and a Deep Deterministic Policy Gradient and Long short-term memory (DDPG-LSTM)-based task offloading and resource allocation algorithm to solve the problem. Simulation results demonstrate that the solution outperforms the baseline approach and that our framework and algorithm have the potential to provide reliable communication services in emergency situations.

**Keywords:** deep reinforcement learning; edge computing; task offloading



**Citation:** Zhang, H.; Xi, S.; Jiang, H.; Shen, Q.; Shang, B.; Wang, J. Resource Allocation and Offloading Strategy for UAV-Assisted LEO Satellite Edge Computing. *Drones* **2023**, *7*, 383. <https://doi.org/10.3390/drones7060383>

Academic Editor: Shiva Raj Pokhrel

Received: 24 May 2023

Accepted: 6 June 2023

Published: 7 June 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

As 5G networks and IoT develop rapidly, countless promising applications and services are emerging, including a High Definition (HD) livestream, autonomous driving, industrial automation and virtual reality, which will take advantage of the benefits that 5G networks will offer, including extremely high data rates, reduced latency, enhanced reliability and large-scale connectivity [1]. The number of IoT connection types is expected to reach 25 billion by 2025. However, the computational power of some IoT devices struggle to handle the large number of tasks due to their limited resources. The emergence of mobile edge computing provides strong support for task offloading and execution.

In traditional edge computing, servers are usually deployed on terrestrial infrastructure communication facilities, which are susceptible to severe damage and loss of service capacity by natural disasters, such as earthquakes and landslides. In emergency scenarios,

where finding safe routes and field rescue is critical, these tasks are often highly latency sensitive and computationally intensive [2], requiring offloading to resource-rich locations for processing. Limited by transmission distance, bandwidth and energy, as well as the possibility of the surrounding infrastructure often being damaged in a disaster, offloading tasks to surrounding areas for execution is not always feasible. Therefore, for the reduction of energy consumption and delay, tasks are offloaded to edge servers, which have more computing power [3]. The solutions to these problems have been made possible by the fast growth of the LEO satellite network.

In the last few years, great progress has been made in the study of task offloading for LEO satellite networks. K. Jaiswal et al. investigated a task offloading scheme for LEO satellites to minimise task processing time by jointly optimising offloading decisions for IoT devices [4–6]. For the task offloading problem in SAGIN-based edge cloud computing systems, M. D. Nguyen et al. consumed energy while adhering to the task’s maximum latency constraint [7]. However, in emergency situations, none of the existing satellite task offloading efforts consider the inability of IoT devices to communicate directly with satellites for task offloading due to the disruption of the ground network infrastructure, which is limited by power.

In this paper, we wish to address these issues. First, UAVs, with their high mobility and flexible deployment, can act as aerial base stations [8–10]. We propose a satellite–UAV–IoT device network architecture where multiple LEO satellites collaborate to offload computational tasks. The UAV acts as a connecting device between LEO satellites and IoT devices, observing satellite resource information and device task information to unload ground tasks to LEO satellites. Next, the problem is modelled as a constrained optimisation non-deterministic Polynomial (NP)—hard problem considering the task offloading of IoT devices and resource allocation of LEO satellites. Then, this optimisation issue is described as a Markov decision process (MDP), and a Deep Deterministic Policy Gradient and Long short-term memory (DDPG-LSTM)-based task offloading and resource allocation algorithm is designed to tackle the issue. Finally, an experimental environment for the simulation of the algorithm was created, and the findings demonstrate that the suggested approach saves the Weighting summation of the energy and latency by an average of 64.5% compared to the benchmark algorithm. The key contributions are the following.

- A UAV-assisted air-space integrated task offload architecture is proposed in emergency scenarios, which jointly considers resource allocation and offloading schemes under the lack of ground resources of computing;
- A multi-satellite joint task offload scheme is proposed, which takes full advantage of satellite computing resources to complete the task with low delay and energy consumed;
- A Deep Reinforcement Learning (DRL) algorithm is proposed, and simulation experiments prove the functionality of the algorithm, reducing the weighted sum of the energy consumed and delay by an average of 64.5%.

The remainder of this paper is organized as listed below. Section 2 showcases related work, including LEO satellite task offloading and UAV-assisted task offloading. Section 3 presents the model, problem description, and optimization objective. The satellite selection and the DDPG-LSTM algorithm are presented in Section 4. Section 5 analyses the experimental findings. Section 6 summarises the thesis.

## 2. Related Work

Task offloading supported by LEO satellite edge computing is a promising method in dealing with energy and computationally resource-constrained IoT devices effectively. In prior works on task offloading, to reduce the consumption of energy or latency of ground-based devices, Wang et al. studied the offloading problem in multiple IoT device scenarios and proposed a strategy for the allocation of resources and offloading, which significantly reduced the average system cost [11,12]. Tan et al. introduces a multi-stage offloading scheme to obtain the most appropriate offloading strategy and reduce the average request response latency and request cost [13]. Wang et al. propose a strategy

considering the differences in terminal tasks and computing capabilities, energy and latency minimisation [14,15]. In studying hybrid task offload, [16,17] proposed hybrid cloud and edge computing LEO satellite networks with a three-tier computing architecture, jointly considering cooperation between mobile users, LEO satellites and cloud servers. Wei et al. considered cooperative offloading between LEO satellite networks, jointly optimising offloading and resource allocation strategies aiming to minimise the weighted sum of user latency and energy consumption. Tang et al. considered the LEO coverage time and computing power to minimize the energy consumption of users by optimizing offloading decisions. [18] investigated cooperative offloading strategies for satellite edge computing systems and terrestrial base stations, considering satellite orbit characteristics and optimising offloading strategies to reduce energy consumption and latency. To make full use of the computing power of cloud servers, Tang et al. proposed an LEO-assisted terrestrial satellite network architecture for collaborative computing offloading at the cloud edge to minimise system energy consumption within the constraints of latency and other Quality of Service (QoS) requirements [19].

The focus of some work has been on UAV-assisted task offloading, where mobile and flexible UAVs can enhance task offloading efficiency. To maximise computational efficiency and task queue stability, Ding et al. propose a DRL-based scheme to optimise offloading and resource allocation [20,21]. Considering the limited computational resources of UAVs, Chen et al. propose a strategy for offloading to ground-based base station servers to optimise transmit power and base station selection under the practical constraints of task completion latency and power consumption [22]. In a UAV-enabled mobile edge computing system based on device-to-device communication, the overall energy efficiency is maximised by optimising UAV and node transmit power and scheduling strategies in order to improve the balance between different types of nodes [23]. For channel uncertainty during offloading, [24] minimises the energy consumption under constraints such as the user quality of service by optimising the CPU frequency and user transmit power, etc. To effectively support the communication and computation of unmanned surface vehicles, [25] jointly considered the UAV flight speed and offloading decision to minimise the energy consumption of the UAV swarm under the condition of ensuring the time delay constraint. Wang et al. considered the energy limitation problem of UAVs and proposed the strategy of offloading the mission to the ground base station by optimising the communication scheduling and resource allocation, etc., considering the location relationship of the ground base station and energy efficiency, and aimed to minimise the total energy consumption [26].

Several efforts focused on joint UAV and LEO satellite processing task offloads. Chai et al. consider allocating computing and communication resources to build a resource allocation and task scheduling system. UAVs are responsible for collection tasks, and satellites provide edge computing services. A scheme for joint multitask offloading and resource allocation in satellite is proposed, significantly reducing the offloading cost [27]. Due to the uncertainty of the air environment, Liu et al. presents an integrated network architecture between the air and ground and designs an adaptive joint deep reinforcement learning offloading scheme to select the most suitable LEO or task offload UAVs based on energy and computational capability, which improves the energy and computational efficiency. Under the condition of satisfying energy dynamics and considering the UAV on-board computing resources and energy constraints, it is proposed in [28] that IoT could locally handle and transfer it to servers, improving the success rate of the task.

From the above analysis, few past works have focused on UAV-assisted LEO edge computing for task offloading in emergency scenarios, where tasks from ground-based IoT devices are offloaded to LEO satellites for processing with the assistance of UAVs. In addition, tasks in emergency scenarios are computationally intensive and latency sensitive. Therefore, multiple satellites are considered for collaborative task processing to reduce energy consumption and task processing delays by optimising resource allocation and offloading strategies.

### 3. System Model and Problem Description

#### 3.1. Network Scenario

Consider the emergency scenarios for 6G, where natural disasters such as earthquakes cause severe damage to ground infrastructure and prevent the provision of computing services to IoT devices. Due to the easy deployment characteristics of UAVs, we consider providing computing offload services for ground devices with the help of UAVs to meet the execution needs of computationally intensive and latency-sensitive tasks.

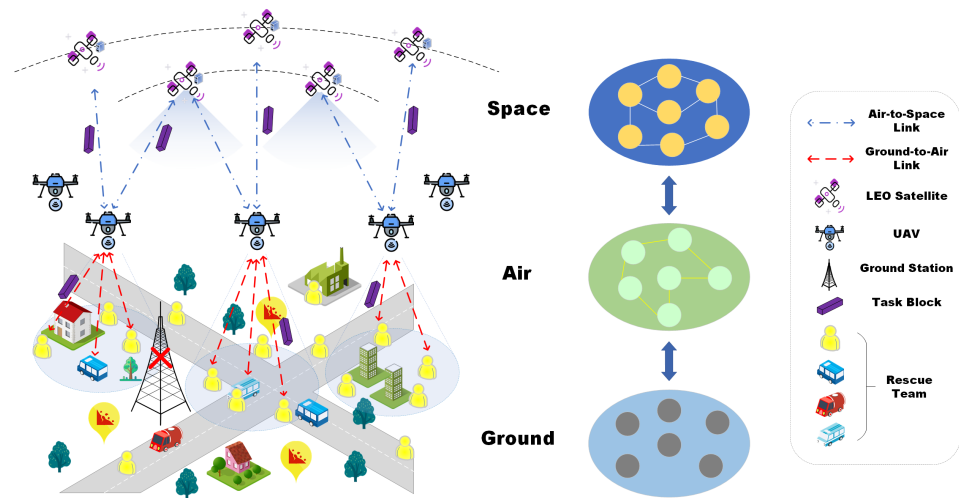
The UAVs collect tasks from ground-based IoT devices and communicate with multiple low-orbiting satellites simultaneously. In order to minimise execution latency, multiple LEOs with a relatively good channel state are selected to share the computational load. Selecting too few LEOs may result in an inability to carry the entire task load, leading to high computational latency and energy consumption. Conversely, selecting too many LEOs can take up satellite resources and other IoT devices are allocated fewer satellite resources, reducing overall system performance.

The ground-based IoT devices  $\{1, \dots, N\}$  are denoted as  $i$ , UAVs  $\{1, \dots, M\}$  are denoted as  $m$  and low-orbiting guard  $\{1, \dots, V\}$  is denoted as  $j$ . The locations of IoT devices, drones and satellites are represented using 3D coordinates. The access bandwidth of the satellite is divided into  $K$  sub-channels, each with bandwidth  $B$ , and the satellite storage capacity is  $C_j$ . Where  $S_i$  denotes the size of the device computational task, we used  $S_{ij}$  to denote the task allocation of the IoT devices and the size of the allocation;  $\beta$  denotes necessary computing cycles to complete a bit of the computational task.

#### 3.2. Architecture

Current terrestrial communication networks are prone to interruptions in the event of serious natural disasters. Instead, low earth orbiting satellites can provide communication guarantees for the emergency response to natural disasters or post-disaster relief. As communication networks continue to evolve, the integration of satellite and terrestrial networks, making full utilisation of the benefits of satellite networks to provide network support for emergency scenarios, is becoming one of the important topics.

In an emergency scenario, a multi-layered LEO satellite–UAV–ground network-integrated air–space–sky architecture is proposed in this paper, in which the ground-based communication facilities are damaged and unable to provide network services. IoT devices (search and rescue tools, rescue vehicles, etc.) are evenly distributed on the ground and UAVs are hovering over the affected area. It is assumed that the UAV has no on-board processing capability, providing relay services to ground equipment to assist in task offload. The ground equipment communicates with the UAV via a wireless channel and then offloads the task to the UAV. The ultra-dense multi-layer LEO topology ensures that seamless service coverage can be provided by multiple satellites for ground devices. The UAV selects service satellites based on satellite resources and computational tasks, forwards the tasks to be processed by the LEO satellites, and provides computational services to ground devices. The proposed task offload architecture fully exploits the resources, combines the advantages of UAVs and LEO satellites to meet the needs of ground emergency response, and provides a new solution for the ground computing task offload in emergency scenarios. The specific scenario is shown in Figure 1.



**Figure 1.** An illustration of task offloading scenario.

### 3.3. Channel Model

#### 3.3.1. IoT Device–UAV Channel

The coordinates of the UAVs are denoted as  $(X_m, Y_m, H_m)$ , and the coordinates of the IoT device are represented as  $(X_i, Y_i, 0)$ ; then, the horizontal distance is indicated as

$$d_{im} = \sqrt{(X_m - X_i)^2 + (Y_m - Y_i)^2} \quad (1)$$

It is assumed that each small affected area is covered by a drone and is within the service area of just one drone. If the IoT device  $i$  sends a task to the drone  $m$ , the IoT device has to be inside the drone's coverage [5].

$$d_{im} \leq d_{max}, \forall i \in \mathcal{N}, m \in \mathcal{M} \quad (2)$$

where  $d_{max}$  dictates the maximum coverage radius of the UAV.

The transmission from the IoT device to the drone is assumed to take place over a wireless channel [29]; to prevent significant co-channel interference, IoT devices offload their computing tasks to drones in the form of frequency division multiple access (FDMA) [30]. When an IoT device and a drone communicate, the drone flies at a low altitude, the channel is considered line-of-sight (LoS) [31] and the small fading effect of the channel is negligible [32]. The uplink data rate  $r$  can be expressed as

$$r_{im} = \omega_u \log_2 \left( 1 + \frac{p_i g_{im}}{\sigma^2} \right) \quad (3)$$

where  $\omega_u$  denotes the channel bandwidth of the IoT device to the UAV,  $g_{i,m}$  is the channel gain of the uplink,  $\sigma^2$  is the additive white Gaussian noise (AWGN) power and  $p_i$  is the transmission power of the channel [33].

$$g_{im} = \frac{g_o}{d_{im}^2 + H_m^2} \quad (4)$$

where  $g_o$  denotes the reference channel gain and  $d_{i,m}^2 + H_m^2$  denotes the Squared Euclidean Distance from the IoT device to the UAV.

#### 3.3.2. UAV-LEO Channel

The geometric distance from the UAV to the LEO satellite, neglecting other factors, is such that the satellite enters the communication window, and data can be transmitted

only when  $\alpha \geq 20$ . The coordinates of the UAVs are denoted as  $(X_j, Y_j, H_j)$ . The geocentric angle  $\theta_{mj}$  between ground-based IoT devices and satellites can be expressed as [16]

$$\theta_{mj} = \arccos\left(\frac{R + H_m}{R + H_j} \cos \alpha_j\right) - \alpha_j \quad (5)$$

where  $R$  denotes the Earth's radius,  $H_m$  represents the UAV's height,  $H_j$  denotes the satellite altitude, and  $\alpha_j$  indicates the horizontal angle between the UAV and the satellite. The maximum value of the communication window  $\theta$  is obtained when  $\alpha = 20$ . The distance is given by

$$H_{mj} = \sqrt{(R + H_m)^2 + (R + H_j)^2 - 2R(R + H_j) \cos \theta_{mj}} \quad (6)$$

According to 3GPP Release15, an additional Doppler shift due to satellite motion should be taken into account according to the following formula:

$$f_{j,m} = (v_{sat} / c) \times \left(\frac{R + H_m}{R + H_j} \cos \alpha\right) \times f_{m,j} \quad (7)$$

where  $v_{sat}$  denotes the satellite speed,  $c$  denotes the speed of light, and  $f_{m,j}$  is the carrier frequency at the transmitter. The drone antenna transmitting gain and receiving antenna gain of the satellite are given by the following formula [34]:

$$g_m = \phi \left(\frac{\pi f_{m,j} \Omega_m}{c}\right)^2 \quad (8)$$

$$g_j = \phi \left(\frac{\pi f_{j,m} \Omega_j}{c}\right)^2 \quad (9)$$

where  $\phi$  denotes the effectiveness of the antenna, and  $\Omega_m$  and  $\Omega_j$  are the antenna radii on the reflective surfaces of the UAV and satellite, respectively.  $c$  is the speed of light. According to the work that has been finished [30], the channel coefficient of the UAV-LEO channel is modelled as

$$h_{mj} = u_{mj} l_{mj} \quad (10)$$

where  $u_{mj}$  and  $l_{mj}$  represent the path loss factor and the small range fading, respectively. In particular, the path loss factor can be written as  $u_{mj} = \left(\frac{4\pi H_{m,j}}{\lambda_{m,j}}\right)^2$ ,  $\lambda_{m,j} = c / f_{m,j}$ . The small-scale fading is given by

$$l_{mj} = \sqrt{\frac{Q}{Q+1} \bar{l}_{mj} + \frac{1}{Q+1} \tilde{l}_{mj}} \quad (11)$$

where  $Q$  is the Rician fading factor,  $\bar{l}_{mj}$  denotes the LoS component satisfying  $|\bar{l}_{mj}|=1$ , and  $\tilde{l}_{mj}$  denotes the non-line-of-sight (NLA) component following  $\tilde{l}_{mj} \sim \mathcal{CN}(0, 1)$ . According to the Shannon formula, the following equation gives the data rate of the uplink

$$R_{mj} = B \log \left(1 + \frac{P_{mj} |l_{mj}|^2 u_{mj}}{N_0 B}\right) \quad (12)$$

where  $B$  refers to the bandwidth used to link the UAV and the satellite,  $P_{mj}$  denotes the UAV uplink transmission power, and  $N_0$  denotes the noise power spectral density.

### 3.3.3. Task Offloading and Computing

The ground device offloads the task to the UAV, then transfers the task block  $S_i$  via the UAV to  $leo_j$  for processing. Use  $a_{ij} = 1$  to indicate that the task can be processed by the satellite  $leo_j$  and vice versa to indicate that task  $S_i$  is not offloaded to  $leo_j$  for processing.



In the IoT device–UAV channel, the data offloading delay for the IoT device task  $S_i$  to offload the task to the UAV includes both transmission delay, propagation delay, transmission delay  $T_{im}^{tran}$  and propagation delay  $T_{im}^{prop}$ , which is given by the following equation [35].

$$T_{im}^{tran} = \frac{S_i}{r_{im}}, T_{im}^{prop} = \frac{d_{im}}{c} \quad (13)$$

where  $c$  represents the light speed. The time delay between the IoT device and the drone can be given by the following equation.

$$T_{im} = T_{im}^{tran} + T_{im}^{prop} \quad (14)$$

The energy consumed by the device to transmit to the drone is calculated as

$$E_{im} = \omega_u T_{im} \quad (15)$$

In the UAV-LEO channel, the delay in offloading task  $S_i$  from the UAV to  $leo_j$  includes a transmission delay  $T_{mj}^{tran}$ , propagation delay  $T_{mj}^{prop}$ , and computation delay  $T_j$ , which can be expressed separately as

$$T_{mj}^{tran} = \frac{S_{ij}}{R_{mj}}, T_{mj}^{prop} = \frac{H_{mj}}{c}, T_j = \frac{S_{ij}\beta}{f_{ij}} \quad (16)$$

where  $S_{ij}$  is the task volume size transferred from  $S_i$  to  $leo_j$ ,  $\beta$  represents the process cycles taken by the CPU to execute one bit of the task volume, and  $f_{ij}$  indicates the computing frequency allocated to  $S_{ij}$  by  $leo_j$ . The drone to satellite time delay is

$$T_{mj} = T_{mj}^{tran} + T_{mj}^{prop} + T_j \quad (17)$$

The energy used consists of the link transmission energy and the energy needed to calculate in the LEO satellite. The following equation gives the calculation of the transfer energy consumption [36].

$$E_{mj} = \frac{P_{mj} S_{ij}}{R_{mj}} \quad (18)$$

where  $P_{mj}$  is the uplink power of the UAV  $m$ . According to the following equation, the satellite calculates the energy consumption as

$$E_j = k S_{ij} \beta f_{ij}^2 \quad (19)$$

where  $k$  is the energy factor. The total time delay  $T$  and total energy consumption  $E$  can be expressed as

$$T = T_{prop} + T_{tran} + T_j, E = E_{im} + E_{mj} + E_j \quad (20)$$

where  $T_{prop}$  is the total propagation time delay,  $T_{prop} = T_{im}^{prop} + T_{mj}^{prop}$ .  $T_{tran}$  is the total transmission time delay,  $T_{tran} = T_{im}^{tran} + T_{mj}^{tran}$ .

### 3.4. Problem Definition

In terrestrial satellite networks, the management of available computing resources is crucial. One of the critical aspects of resource management is allocating tasks from IoT devices to satellite nodes for processing, where different offloading decisions lead to additional costs, affecting system performance and energy consumption.

Based on the system model and assumptions discussed above, the primary objective is the minimisation of the balanced totals of system latency and energy consumption by the collaborative optimisation of offloading decisions and resource allocation. The system satis-

fies the storage requirements, given the available bandwidth and computational resources for all tasks simultaneously, and the problem can be expressed mathematically, as follows.

$$\min Z = \zeta T + \eta E \quad (21)$$

$$s.t. C1 : \sum_{j \in \mathcal{V}} a_j \bar{X}_j \geq \sum S_i(t) \quad (22)$$

$$C2 : \sum_{i \in \mathcal{N}} S_{ij} \leq C_j \quad (23)$$

$$C3 : N \leq K \quad (24)$$

$$C4 : \sum_{i \in \mathcal{N}} f_{ij} \leq f_j^* \quad (25)$$

$$C5 : \sum_{j \in \mathcal{V}} S_{ij} = S_i \quad (26)$$

$$C6 : \sum_{j \in \mathcal{V}} P_{mj} \leq P_m^* \quad (27)$$

where  $\bar{X}_j$  denotes the size of the remaining storage resources of  $leo_j$ ,  $f_{ij}$  is the computing resources allocated to  $s_{ij}$  by  $leo_j$ ,  $f_j^*$  is the satellite's highest CPU frequency  $j$ ,  $P_m^*$  is the full uplink transmission power of the UAV,  $C_j$  is the total storage space,  $N$  is the number of individual satellite connections,  $K$  is the maximum number of channels, and the size of the task block is allocated by  $S_{ij}$ . Where  $\zeta, \eta \in [1, 10]$ ,  $\zeta$  and  $\eta$  are the weights of delay and energy consumption, respectively.

C1 is that the free storage capacity of the LEO satellite to which the device is connected is not less than the device's task size, C2 is that the storage space already used by the satellite is not more significant than the total storage space, C3 is that the number of individual satellite connections does not exceed the maximum number of channels, and C4 is a constraint on satellite computing resources to ensure that the CPU resources being allocated to IoT device tasks do not overwhelm the total CPU computing resources. C5 is the sum of the tasks given to different satellites by  $S_i$  and the whole task size, and C6 is the UAV to uplink the transmission power that is not more significant than the maximum UAV power.

The complexity of the problem is increased by the coupling relation among the optimizing variables. In addition, the proposed optimisation problem is a mixed integer nonlinear problem, though the function and constraints have binary variables. As IoT devices continue to rise, the complexity grows exponentially. To reduce the problem's complexity, decomposing the original problem into sub-problems provides a new solution. It decouples the optimising problem and turns it into two sub-problems: satellite selection, task volume and computational resource allocation.

Transformation, according to the optimization objective, yields

$$\min Z = \zeta T + \eta E = \zeta T_{prop} + [\zeta (T_{tran} + T_j) + \eta E] \quad (28)$$

The satellite selection needs to satisfy the constraints C1, C2 and C3, and the objective of optimisation is phrased as

$$\min Z_1 = \zeta T_{prop} \quad (29)$$

The task and computational resource allocation policy must satisfy constraints C4, C5 and C6 with the following optimization objectives.

$$\min Z_2 = \zeta (T_{tran} + T_j) + \eta E \quad (30)$$



#### 4. Algorithm Design

In this section, the above sub-problems are analyzed. Two algorithms are proposed to solve problems based on task separability and resource separability, respectively. To better understand our proposed solutions, we will briefly introduce the algorithmic process, explaining the concepts related to Monte Carlo methods, Markov decision processes, reinforcement learning, and the mathematical definitions of value and reward functions.

##### 4.1. Satellite Selection

The satellite selection problem is a mixed integer programming problem, as observed from the objective function and constraints; thus, a Monte Carlo random sampling method is considered for the solution. A Monte Carlo method-based satellite selection algorithm is proposed for satellite and task matching, where satellites move continuously according to a predetermined orbit. IoT devices generate computational tasks and obtain the optimal satellite combination for IoT device task offloading.

##### Monte Carlo-Based Satellite Selection Algorithm

Monte Carlo methods are also known as random sampling or statistical test methods. The Monte Carlo method is a computational method but is different from the general numerical computational methods. It is a method based on probabilistic statistical theory. It solves problems that are difficult to solve by numerical methods, which is why it is increasingly used in many applications.

The UAV collects the current satellite storage resource information  $(C_1, \dots, C_j)$  and the IoT device task  $(S_1, \dots, S_i)$ . According to the managed satellite resource and task information, the satellite storage resource information is randomly sampled by the Monte Carlo method to approximate the task size  $S_i$  and obtain the approximate satellite subsequence  $X_t = (a_1, \dots, a_j)$ .

$$|\sum Random(C_1, \dots, C_j) - \sum S_i| \geq 0 \quad (31)$$

Select the satellite subsequence  $A_i = (X_1, \dots, X_t)$  that satisfies the task amount  $\sum S_i$ .

$$\sum Random(C_1, \dots, C_j) \geq \sum S_i \quad (32)$$

UAV  $m$  to Satellite propagation time delay

$$T_{m,j} = (D_{m1}, \dots, D_{mj}) \quad (33)$$

Maximum propagation time delay is

$$T_{max} = \max(T_{mj} \times X_i) = \max(D_{m1} \times a_1, \dots, D_{mj} \times a_j) \quad (34)$$

The optimal combination of offloading satellites is obtained by minimizing the maximum propagation delay of IoT devices. As shown in Algorithm 1.

The algorithm inputs the task size processed and the satellite resources. Minimizing the propagation delay obtains the optimal offloading decision for the IoT device tasks, which are transmitted via a link between the UAV and the LEO satellite.

**Algorithm 1** Satellite selection algorithm**Input:** Task  $S = \{S_1, S_2, \dots, S_i\}$ , Storage Resource  $C = \{C_1, C_2, \dots, C_j\}$ ;**Output:** Selection Vector  $a$ ;

```

1: Initialize  $cost = 0$ ;
2: Monte Carlo random sampling of a satellite combination  $X = \{a_1, \dots, a_t\}$ , according to (31);
3: for  $n = 1, j$  do
4:   Compute distance between Satellite and UAV  $H_{m,n}$ , according to (6);
5:   Compute Propagation Delay between Satellite and UAV  $D = \{D_{m,1}, \dots, D_{m,j}\}$ , according to  $D_{m,n} = \frac{H_{m,n}}{c}$ ;
6: end for
7: for  $u = 1, t$  do
8:   if  $a_u C^T \geq \sum S$  then
9:     Compute Propagation Delay  $T_u$ , according to  $T_u = \max(a_u \times D)$ ;
10:    if  $cost > T_u$  then
11:       $cost = T_u$ ;
12:       $a = a_u$ ;
13:    end if
14:  end if
15: end for

```

**4.2. Task and Computing Resource Allocation Strategy**

Due to the presence of correlations in several restrictions, to reduce the problem's difficulty, several deterministic factors have to be taken into account when offloading satellite edge tasks. These include the satellite edge server's state and the environment of the communication. Therefore, the optimizing problem is converted to an offloading scheme based on DRL-making methods and solving it by Markov decision. The DRL architecture consists of interactions through the agent to solve the above problem by training the best policy to maximise the cumulative reward [37].

**DDPG-Based Task Offloading and Resource Allocation Algorithm**

Through iterative trials, reinforcement learning optimizes the action selection in multiple situations based on a given reward function. The intelligent body perceives the state and performs actions to change it. During each iteration, the competent body observes the state as input and selects the action to be completed. The execution of the action produces a reward, and the intellectual body judges the quality of the action by observing the prize. The selection of activities by intelligence tends to increase the long-term total compensation and maximize the reward function.

The DDPG algorithm is one of the most popular methods for dealing with problems in RL, described as  $(S, a, r_t, \gamma)$ , where  $S$  is the state,  $a$  denotes the action,  $r_t$  is the immediacy reward of the time slot, and  $\gamma \in (0, 1)$ . The anticipated long-term discounting compensation has the following definition:

$$R_t = \sum_{t=1}^T \gamma^t r(s_t, a_t) \quad (35)$$

Here, at time  $t$ , the state and action are  $s_t$  and  $a_t$ , respectively;  $r(s_t, a_t)$  is the straight reward. Taking into account that we have to deal with continuous actions, we decide to follow a determined policy and write the value function, as follows

$$Q^\mu(s_t, a_t) = r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, a_{t+1}) \quad (36)$$

In an effort to achieve maximum expected discounted benefits over the long term, at each slots, we use the time-series difference method learned from the previous period's experience to update the action function.

The DDPG algorithm makes the approximation  $Q(s, a|\theta)$ . The actor uses the strategy function  $\mu(s|\theta)$  to decide on an action, the critic uses the value function to judge the policy functional, and the network of values and strategic network are renewed in accordance with the critic's output. The loss function has the form

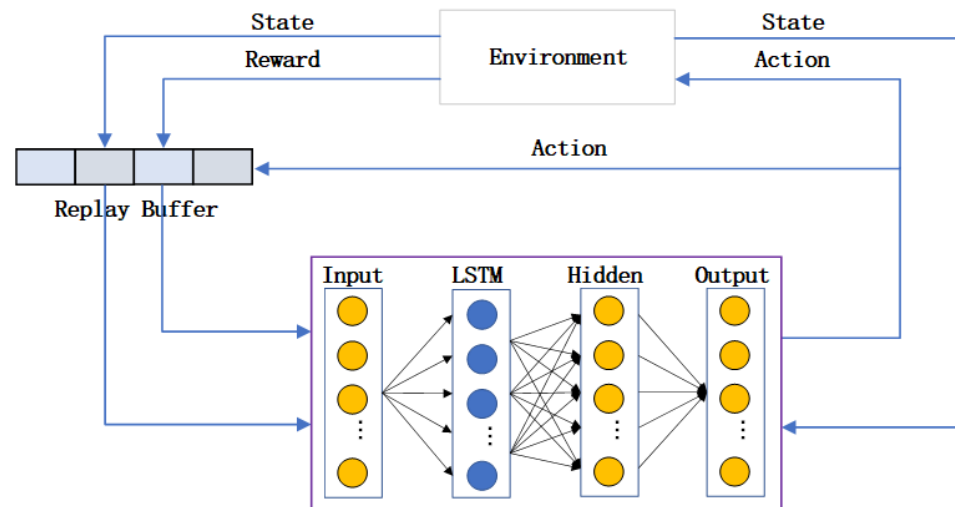
$$L(\theta^Q) = (Q(\theta^Q) - y_t)^2 \quad (37)$$

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \theta^Q) \quad (38)$$

Refine the critic network by minimising  $L(\theta^Q)$ , and  $Q(\theta^Q)$  is the maximum future payoff that will be earned by proceeding with the policy  $\mu(s_{t+1}|\theta^Q)$  to the state following the implementation of action  $a_t$ . The actor network must change the action parameters in the general direction where the maximum Q is more probable.

Instead of updating all the parameters, the fixed goal Q-network can stabilise the learning by keeping a part of the parameters updated. Let  $\theta^\mu$  and  $\theta^Q$  be the parameters before the update, respectively, and  $\theta^{\mu'}$  and  $\theta^{Q'}$  after the update.

As changes in task offloading are time-continuous and the offloading decision taken in the previous time slot has an impact on the current observation, we use LSTM to capture the correlation between the previous observation and the current observation and more potential information by learning a series of past experiences. The algorithm overcomes the inability of the Deep Deterministic Policy Gradient (DDPG) to handle partial observability and history-dependent decisions by adding a recursive mechanism. The DDPG-LSTM is therefore proposed, and the algorithm architecture is shown in Figure 2.



**Figure 2.** Architecture for the DDPG-LSTM-based computation offloading scheme.

The organisation of the DDPG-LSTM is mainly built on the actor–critic model. The actor network of the agent is responsible for generating actions and contains two components: the actor network  $\mu^\theta$  and  $\mu^{\theta'}$ , where  $\theta$  and  $\theta'$  are the network parameters. The critic network of agent is responsible for evaluating actions and contains two components: the critic network  $Q^\phi$  and  $Q^{\phi'}$ , where  $\phi$  and  $\phi'$  are the network parameters.

The DDPG-LSTM algorithm has three main elements.

- State

The state consists of the IoT device task information and satellite resource information.

$$S(t) = \{S_i, f_j\} \quad (39)$$

where  $S_i$  denotes the computational task size of the ground-based IoT device and  $f_j$  denotes the computational resources of the LEO satellite.

- Action.

The action is composed of the task allocation vector  $S_{ij}$  and the computational resource allocation vector  $F_{ij}$  to obtain the action space  $A$  of the system.

$$A(t) = \{S_{i1}, S_{i2}, \dots, S_{iv}, f_{j1}, f_{j2}, \dots, f_{jn}\} \quad (40)$$

- Reward.

In this work, a function is introduced to explain the amount of change in the system cost when action  $A_t$  is taken in system state  $S_t$ . It is expressed as

$$R(t) = U^t - U^{t+1} \quad (41)$$

Here,  $U^t$  and  $U^{t+1}$  means the latency and energy consumption at time point  $t$  and the next time point. The magnitude of the difference represents the cost reduction achieved by  $A_t$ ; the system benefit of taking action  $A_t$  is  $U^t$ .

The MDP is aimed at maximising the Reward Sum expected to be received, and can therefore be formulated as follows

$$\max Y = \sum_{t=0}^{T-1} Y^t \quad (42)$$

The process of DDPG-LSTM is shown in the Algorithm 2, where the weight parameters and the replay buffer are initialised. In each training round, the agent is given a state  $Z_t$ , decides on an action  $a_t$ , performs the action and receives a reward  $R_t$ . Then, the replay buffer stores the experience transformation and the chosen batch size of  $M$ . Finally, the networks are refreshed.

---

**Algorithm 2** DDPG-LSTM-based task and resource allocation algorithm

---

**Input:** task  $S_i$ , computing resource  $f_j$

**Output:** task allocation  $S_{ij}$ , computing resource  $f_{ji}$

```

1: Actor_Critic weight parameters randomised initialisation :  $\omega, \theta, \omega', \theta'$ 
2: Initialising Replay Buffer  $M$ , mini-batch size  $B$ , train episode threshold  $C$ ;
3: for  $m = 1, Ep$  do
4:   Initialize environment;
5:   Initialize LSTM states in the network;
6:   Reception of primary state  $Z_t$ ;
7:   for  $t = 1, T$  do
8:     Choose  $a_t$  by online network,  $a_t = H(Z_t|w) + n_0$ ;
9:     Implementation  $a_t$ , obtain  $R_t$  and switch to the state  $Z_{t+1}$ ;
10:    Store  $(Z_t, a_t, R_t, Z_{t+1})$  in  $M$ ;
11:    if  $m \geq C$  then
12:      Sample random mini-batch transitions  $(Z_i, a_i, R_i, Z_{i+1})$  from  $R$ ;
13:      Using the loss function and policy gradient to refresh online critic and actor
parameters;
14:      Refresh parameters of the target network;
15:    end if
16:  end for
17: end for
```

---

## 5. Performance Evaluation

Within this chapter, we examine the behaviour of the proposed DDPG-based scheme. First, we present the simulation scenario. Then, many simulation experiments are performed, and the results are compared and analyzed. Lastly, we verify the training efficiency

under different parameters and find the optimal parameter settings, comparing the performance of various task offloading schemes and further demonstrating the method's effectiveness.

### 5.1. Simulation Environment and Parameters

A UAV-assisted air-space integration network scenario is considered; we used Python to simulate and evaluate the proposed algorithm. In our simulations, we consider a geographical area of  $1 \text{ km} \times 1 \text{ km}$ , in which IoT devices are then randomly deployed within the area [38]. With LEO satellites flying at  $[700, 1000] \text{ km}$  and UAVs flying at  $100 \text{ m}$  [39], the LEO satellites provide a seamless coverage of the geographical area. We assume that the efficiency of the satellite, IoT device and UAV antenna are 0.6, 0.55 and 0.6, respectively [34]. The maximum available computing resources of the satellite are evenly distributed in the  $[3, 8] \text{ GHz}$  interval. The detailed simulation parameters are given in Table 1 [40–42].

**Table 1.** Simulation parameters.

Parameter	Value
Computing resources of LEO satellite, $f_j$	$[3, 8] \text{ GHz}$
CPU cycles for processing one bit, $\beta$	100 cycles/bit
Effective switched capacitance, $k$	$10^{-27}$
IoT maximum transmit power, $p_i$	23 dBm
UAV maximum transmit power, $p_m$	40 dBm
Coverage radius of UAV, $d_{max}$	1 km
Height of LEO satellite, $H_j$	$[700, 1000] \text{ km}$
Number of bandwidth chunks, $K$	6
Noise spectral density, $N_0$	$-174 \text{ dBm/Hz}$
Bandwidth of UAVs, $\omega_m$	1 MHz
Bandwidth of LEO, $B$	10 MHz
UAV altitude, $H_m$	100 m
Light speed, $c$	$3 \times 10^8 \text{ m/s}$
Task size, $S_i$	$[2.5, 3.5] \text{ MB}$

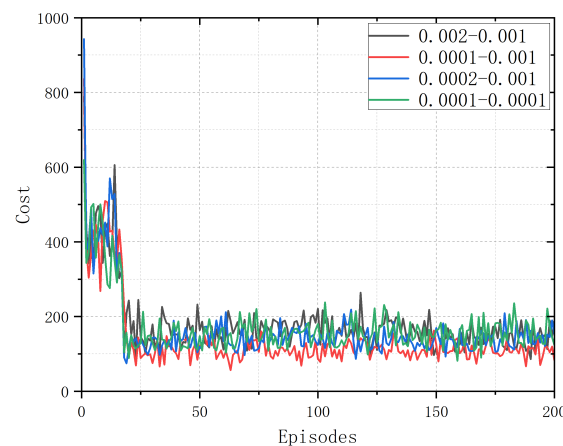
The DDPG-LSTM algorithm with a LSTM layer and the neural network [43] uses the Relu, Tanh and sigmoid functions as the activation function, while the end actor network results use softmax to restrict actions. Some critical parameters are analyzed to explore the impact of the algorithm parameters. For each parameter studied, we provide some possible reference values. Energy consumption and the delay weighted sum is used as an evaluation criterion to explore the effect of parameters. The detailed algorithm parameters are given in Table 2.

**Table 2.** Algorithm parameters.

Parameter	Value
Buffer capacity, $M$	1,000,000
Batch size, $B$	256
Learning rate, $\delta_a, \delta_c$	0.0001, 0.001
Soft update rate, $\tau$	0.005
Discount factor, $\gamma$	0.99
Exploration rate, $\epsilon$	0.01

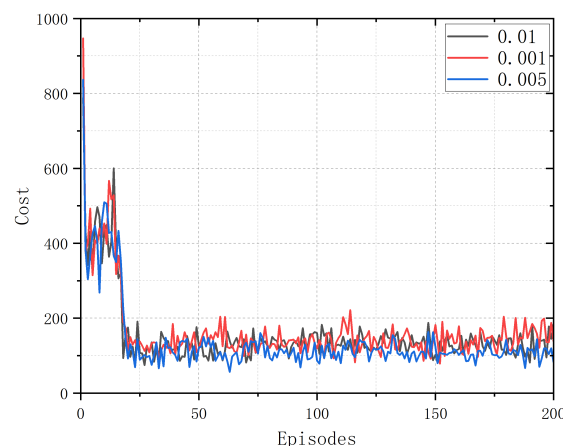
Figure 3 represents the performance of the algorithm at different learning rates. The update step's size affects the convergence speed, and when the learning rate is too low, the algorithm converges slowly. When the learning rate is too high, the maximum value may be missed due to the excessive size of the update step. The graph shows that the algorithm performs optimally if  $\delta_a = 0.0001$  and  $\delta_c = 0.001$ . In addition, it can be found that the network performs better when  $\delta_c$  is greater than  $\delta_a$  because the actor network needs guid-

ance from the critic network to learn. When the critic network learns faster than the actor network, it can better guide the update direction of the actor network.



**Figure 3.** Training process with different Learning Rate settings.

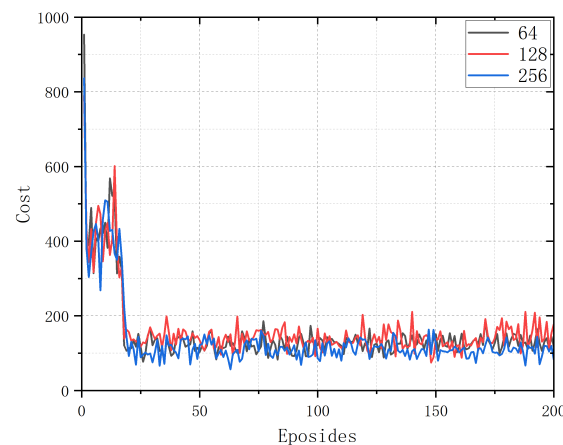
Figure 4 shows the algorithm's performance under different soft update rates. Compared to the complex update strategy, the upgrade interval of the objective network is reduced by the soft update strategy. It ensures that the target mesh is updated in every iteration, increasing the frequency of updating the target mesh and helping to decrease the time taken for the algorithm to converge. The smaller the soft update coefficient, the more stable the algorithm will be and the less the parameters of the target mesh will change, resulting in a too-slow convergence of the algorithm. If the soft update coefficient is too large, the algorithm will be unstable. Therefore, an appropriate soft update factor can make the algorithm stable and fast. The figure illustrates that the best performance of the algorithm is achieved when the soft update factor is  $\tau = 0.005$ .



**Figure 4.** Training process with different Soft Update Rate settings.

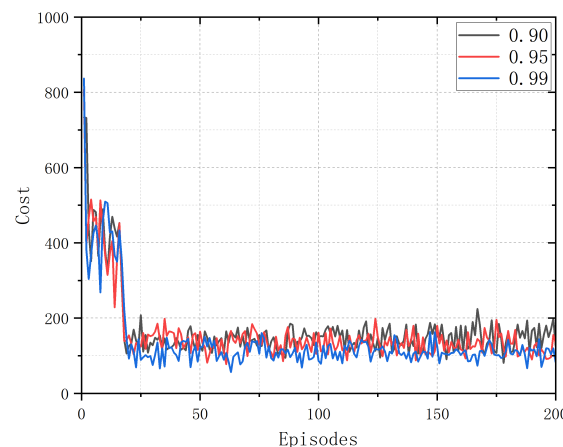
Figure 5 shows the algorithm's performance under a varying batch size. It uses small batch learning to increase the speed of model training to reduce the cost per iteration. Small batches converge faster for training compared to extensive data collection. Still, they can lead to poor performance as the data stored in the buffer is initially over-utilized [44], reducing the importance of the data at a later stage. Large batches of data cause the network to update too slowly and may also perform poorly. The behaviour of the algorithm on a variety of sets is depicted in the graph, from which it is clear that the algorithm achieves better results when the batch size = 256. In addition, it can be found that when the batch size is 64 and 128, the system loss is higher, and the curve fluctuates more, making it difficult for the algorithm to converge quickly and reducing its performance.





**Figure 5.** Training process with different batch size settings.

Figure 6 represents the performance of the algorithm under different discount factors. The actor network will make actions that give the critic network a high rating, and the rating calculation will use the discount factor. To reflect the continuity of the decision, the actor-network is expected to consider the reward and the next prize when choosing action  $a_t$ . Too small a discount factor prevents the critic network from anticipating the future in time and affects the algorithm's performance. Conversely, when the discount factor is too significant, it may reduce the critic network's prediction accuracy. In the figure, the method performs better when the discount factor = 0.99.



**Figure 6.** Training process with different discount factor settings.

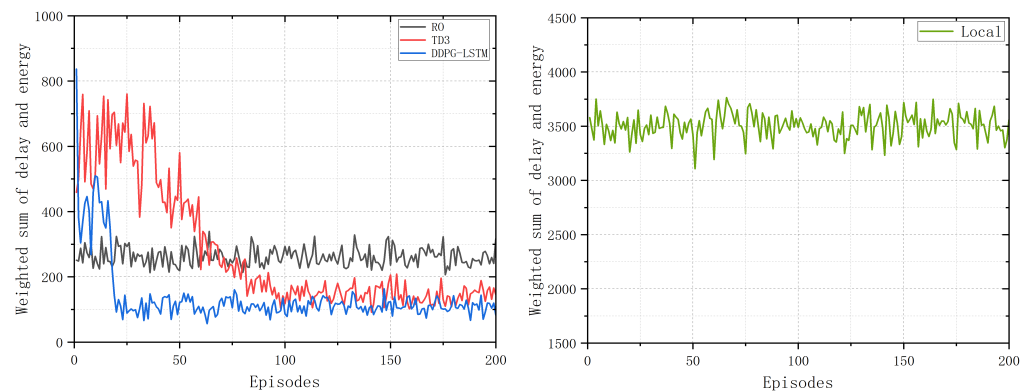
## 5.2. Performance Comparison

In simulation experiments, the DDPG-LSTM algorithm proposed in this paper is compared with random offloading (RO) [45], Twin Delayed DDPG (TD3) [46–48] and local computing (Local), using the weighted sum of latency and energy consumption as evaluation criteria [36,49]. Then, the performance of the algorithm is compared under different computational resources and task volumes, validating the performance advantages of the algorithm.

Assume that a stack of task orders arrive at the MEC server at each slot, and the device generates a task at a slot. The task execution cost is used to compare the performance of different policies. The offload expense is the summation of the time taken by the devices to fulfil their individual tasks throughout the time slot.

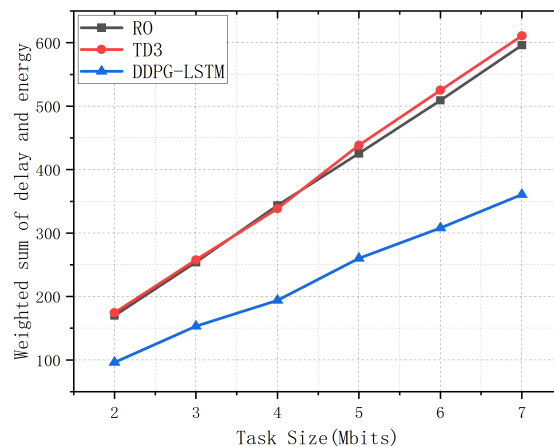
Figure 7 reveals that energy consumption and the delay weighted sum of executing the task using the DDPG-LSTM method is lower than the other three strategies. The performance of the algorithm continues to improve as the training progresses. DDPG-LSTM takes into account the satellite status information to ensure that resources are fully utilised and

continuously optimises the resource allocation strategy to ensure that the task is completed with the lowest possible latency. In addition, the figure shows that the cost of the task execution is much higher than the other strategies due to the lack of IoT device computing resources. Compared to the TD3 algorithm, DDPG has a faster convergence speed, and LSTM is easier to capture temporal information. Therefore, the DDPG-based scheme has a memory function to store valuable historical data, thus achieving better performance and validating the effectiveness of LSTM for task offloading strategies.



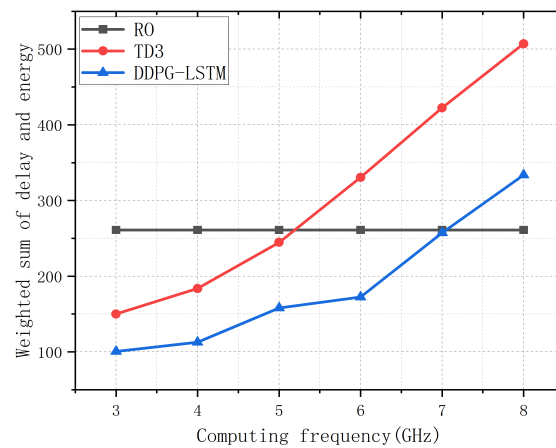
**Figure 7.** Comparison of different algorithms.

Figure 8 displays the cumulative expense of different data sizes. As the task data becomes more significant, the server has to use additional time and energy to handle the tasks, and the average full system expense for handling tasks is trending up. In contrast, the DDPG-LSTM algorithm has the smaller rising trend and the better performance compared to other algorithms. As the volume of the task size becomes larger, the expense of the locally computed increases faster.



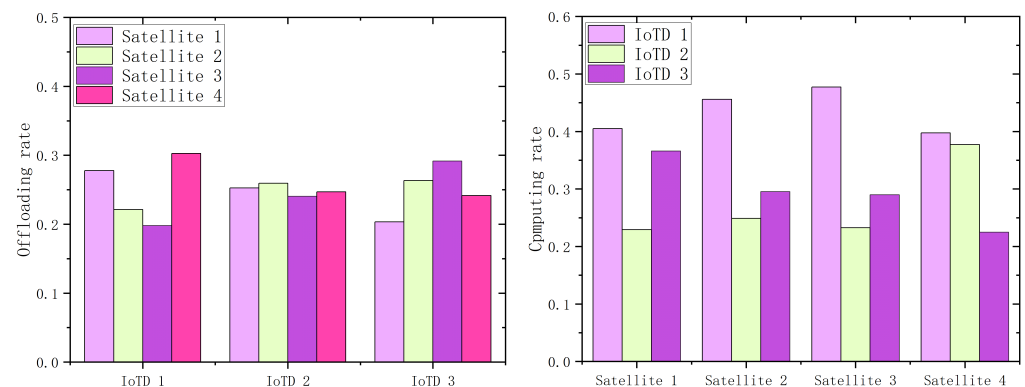
**Figure 8.** Relationship between cost and  $S_i$ .

Figure 9 indicates the changing of the cost as the LEO satellite server compute frequency rises between [3, 8] GHz. It is apparent from the figure that the combined expense of the three strategies tends to be greater as the processing frequency of the server becomes higher. As the frequency of LEO satellite server processing rises, the expense of the DDPG-LSTM suggested in this paper is the lowest.



**Figure 9.** Relationship between cost and  $f_j$ .

Demonstration of the distribution of offload tasks across several LEO satellites; Figure 10 shows four IoT devices, with three of the satellites providing edge computing capability. The altitude and computing resources of the satellites affect the mission offload, and similarly, the distribution of computing resources is affected by the mission offload. For satellite communication, transmission costs increase with altitude, and higher altitudes prolong the latency and energy consumption spent in the space segment during the computed offload. In satellite edge computing, lower computational resources and larger task offloads increase the processing latency and energy consumption of the task.



**Figure 10.** Distributions of offloading task and computing resource.

## 6. Conclusions

In this paper, we put forward an integrated air–space–sky network architecture for UAV-assisted task offloading to provide more available computational resources for ground devices and to ensure computational requirements in emergency scenarios. To minimise the delay and energy consumed by offloading tasks, described the problem as MDP. We further develop an algorithm to solve it. The solution enables the UAV controller to determine the best unloading decision based on dynamic channel conditions and the satellite position, including task offloading scenarios and resource allocation strategies. Finally, a series of trials are conducted to validate the validity and superiority of our proposed unloading scheme.

In the future, we need to consider more of drones' auxiliary access LEO mobile Internet of things in the edge of the network equipment. In some real-world situations, IoT devices are mobile at high speed; the approach we have proposed may not be appropriate for such scenarios. In the case of mobile IoT devices, by using satellite switching to overcome this problem, in this paper, IoT devices can only perform task offloading. When the number

of IoT devices or tasks increases, this offloading strategy puts a lot of bandwidth pressure on the satellite network and increases the energy consumption for task transmission. Therefore, a partial offload strategy can be explored for future work associated with air-based edge computing.

**Author Contributions:** Conceptualization, H.Z. and B.S.; methodology, S.X.; software, S.X.; validation, H.J., Q.S. and J.W.; formal analysis, B.S.; investigation, H.J.; resources, H.Z.; data curation, S.X.; writing—original draft preparation, S.X.; writing—review and editing, H.Z.; visualization, B.S.; supervision, H.Z.; project administration, S.X. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is partially supported by the Natural Science Foundation of Shandong Province under Grant ZR2020MF006 and ZR2022LZH015, R&D Program of Beijing Municipal Education Commission under Grant KM202211417014, Academic Research Projects of Beijing Union University under Grant ZK20202215 and Open Foundation of State Key Laboratory of Integrated Services Networks (Xidian University) under Grant ISN23-09.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Zhang, H.; Yang, Y.; Shang, B.; Zhang, P. Joint Resource Allocation and Multi-Part Collaborative Task Offloading in MEC Systems. *IEEE Trans. Veh. Technol.* **2022**, *71*, 8877–8890. [\[CrossRef\]](#)
2. Luo, Q.; Luan, T.H.; Shi, W.; Fan, P. Deep Reinforcement Learning Based Computation Offloading and Trajectory Planning for Multi-UAV Cooperative Target Search. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 504–520. [\[CrossRef\]](#)
3. Shang, B.; Yi, Y.; Liu, L. Computing over Space-Air-Ground Integrated Networks: Challenges and Opportunities. *IEEE Netw.* **2021**, *35*, 302–309. [\[CrossRef\]](#)
4. Jaiswal, K.; Dahiya, A.; Saxena, S.; Singh, V.; Singh, A.; Kushwaha, A. A Novel Computation Offloading Under 6G LEO Satellite-UAV-based IoT. In Proceedings of the 3rd International Conference on Computation, Automation and Knowledge Management (ICCAKM), Dubai, United Arab Emirates, 15–17 November 2022; pp. 1–6. [\[CrossRef\]](#)
5. Yu, K.; Cui, Q.; Zhang, Z.; Huang, X.; Zhang, X.; Tao, X. Efficient UAV/Satellite-assisted IoT Task Offloading: A Multi-agent Reinforcement Learning Solution. In Proceedings of the 27th Asia Pacific Conference on Communications (APCC), Jeju Island, Republic of Korea, 19–21 October 2022; pp. 83–88.
6. Liu, Y.; Jiang, L.; Qi, Q.; Xie, S. Energy-Efficient Space-Air-Ground Integrated Edge Computing for Internet of Remote Things: A Federated DRL Approach. *IEEE Internet Things J.* **2023**, *10*, 4845–4856. [\[CrossRef\]](#)
7. Nguyen, M.D.; Le, L.B.; Girard, A. Computation Offloading, UAV Placement, and Resource Allocation in SAGIN. In Proceedings of the 2022 IEEE Globecom Workshops (GC Wkshps), Rio de Janeiro, Brazil, 4–8 December 2022; pp. 1413–1418. [\[CrossRef\]](#)
8. Shang, B.; Shafin, R.; Liu, L. UAV Swarm-Enabled Aerial Reconfigurable Intelligent Surface (SARIS). *IEEE Wirel. Commun.* **2021**, *28*, 156–163. [\[CrossRef\]](#)
9. Shang, B.; Liu, L. Mobile-Edge Computing in the Sky: Energy Optimization for Air-Ground Integrated Networks. *IEEE Internet Things J.* **2020**, *7*, 7443–7456. [\[CrossRef\]](#)
10. Akter, S.; Kim, D.Y.; Yoon, S. Task Offloading in Multi-Access Edge Computing Enabled UAV-Aided Emergency Response Operations. *IEEE Access* **2023**, *11*, 23167–23188. [\[CrossRef\]](#)
11. Wang, B.; Feng, T.; Huang, D. A Joint Computation Offloading and Resource Allocation Strategy for LEO Satellite Edge Computing System. In Proceedings of the IEEE 20th International Conference on Communication Technology (ICCT), Nanning, China, 28–31 October 2020; pp. 649–655. [\[CrossRef\]](#)
12. Wang, Z.; Yu, H.; Zhu, S.; Yang, B. Curriculum Reinforcement Learning-Based Computation Offloading Approach in Space-Air-Ground Integrated Network. In Proceedings of the 13th International Conference on Wireless Communications and Signal Processing (WCSP), Virtual, 20–21 October 2021; pp. 1–6. [\[CrossRef\]](#)
13. Tan, H.; He, M.; Xia, T.; Zheng, X.; Lai, J. A Novel Multi-level Computation Offloading Scheme at LEO Constellation Broadband Network Edge. In Proceedings of the IEEE World Congress on Services (SERVICES), Beijing, China, 18–24 October 2020; pp. 281–286. [\[CrossRef\]](#)
14. Wang, B.; Xie, J.; Huang, D.; Xie, X. A Computation Offloading Strategy for LEO Satellite Mobile Edge Computing System. In Proceedings of the 14th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 10–12 June 2022; pp. 75–80. [\[CrossRef\]](#)

15. Song, Z.; Hao, Y.; Liu, Y.; Sun, X. Energy-Efficient Multiaccess Edge Computing for Terrestrial-Satellite Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 14202–14218. [\[CrossRef\]](#)
16. Wei, K.; Tang, Q.; Guo, J.; Zeng, M.; Fei, Z.; Cui, Q. Resource Scheduling and Offloading Strategy Based on LEO Satellite Edge Computing. In Proceedings of the IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Virtual, 27 September–28 October 2021; pp. 1–6. [\[CrossRef\]](#)
17. Tang, Q.; Fei, Z.; Li, B.; Han, Z. Computation Offloading in LEO Satellite Networks with Hybrid Cloud and Edge Computing. *IEEE Internet Things J.* **2021**, *8*, 9164–9176. [\[CrossRef\]](#)
18. Yang, X.; Hong, B. Cost-Efficient Task Offloading for Satellite Edge Computing Systems. In Proceedings of the International Symposium on Networks, Computers and Communications (ISNCC), Shenzhen, China, 19–22 July 2022; pp. 1–5. [\[CrossRef\]](#)
19. Tang, Z.; Zhou, H.; Ma, T.; Yu, K.; Shen, X.S. Leveraging LEO Assisted Cloud-Edge Collaboration for Energy Efficient Computation Offloading. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 September 2021; pp. 1–6. [\[CrossRef\]](#)
20. Ding, Y.; Feng, Y.; Lu, W.; Zheng, S.; Zhao, N.; Meng, L.; Nallanathan, A.; Yang, X. Online Edge Learning Offloading and Resource Management for UAV-Assisted MEC Secure Communications. *IEEE J. Sel. Top. Signal Process.* **2023**, *17*, 54–65. [\[CrossRef\]](#)
21. Lu, W.; Ding, Y.; Feng, Y.; Huang, G.; Zhao, N.; Nallanathan, A.; Yang, X. Dinkelbach-Guided Deep Reinforcement Learning for Secure Communication in UAV-Aided MEC Networks. In Proceedings of the GLOBECOM 2022—2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; pp. 1740–1745. [\[CrossRef\]](#)
22. Chen, P.; Luo, X.; Guo, D.; Sun, Y.; Xie, J.; Zhao, Y.; Zhou, R. Secure Task Offloading for MEC-aided-UAV system. *IEEE Trans. Intell. Veh.* **2022**, *8*, 3444–3457. [\[CrossRef\]](#)
23. Qi, X.; Yuan, M.; Zhang, Q.; Yang, Z. Joint power-trajectory-scheduling optimization in a mobile UAV-enabled network via alternating iteration. *China Commun.* **2022**, *19*, 136–152. [\[CrossRef\]](#)
24. Mao, W.; Xiong, K.; Lu, Y.; Fan, P.; Ding, Z. Energy Consumption Minimization in Secure Multi-antenna UAV-assisted MEC Networks with Channel Uncertainty. *IEEE Transactions Wirel. Commun.* **2023**, *1*. [\[CrossRef\]](#)
25. Liao, Y.; Chen, X.; Xia, S.; Ai, Q.; Liu, Q. Energy Minimization for UAV Swarm-Enabled Wireless Inland Ship MEC Network with Time Windows. *IEEE Trans. Green Commun. Netw.* **2022**, *7*, 594–608. [\[CrossRef\]](#)
26. Wang, J.; Yang, D.; Wu, F.; Xu, Y.; Xiao, L.; Zhang, T. Energy Minimization for Cellular-Connected UAV-MEC Patrol Inspection Systems. In Proceedings of the 2022 IEEE/CIC International Conference on Communications in China (ICCC), Foshan, China, 11–13 August 2022; pp. 389–394. [\[CrossRef\]](#)
27. Zhang, H.; Yang, Y.; Huang, X.; Fang, C.; Zhang, P. Ultra-Low Latency Multi-Task Offloading in Mobile Edge Computing. *IEEE Access* **2021**, *9*, 32569–32581. [\[CrossRef\]](#)
28. Chai, F.; Zhang, Q.; Yao, H.; Xin, X.; Gao, R.; Guizani, M. Joint Multi-task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT. *IEEE Trans. Veh. Technol.* **2023**, 1–15. [\[CrossRef\]](#)
29. Ebrahim, M.A.; Ebrahim, G.A.; Mohamed, H.K.; Abdellatif, S.O. A Deep Learning Approach for Task Offloading in Multi-UAV Aided Mobile Edge Computing. *IEEE Access* **2022**, *10*, 101716–101731. [\[CrossRef\]](#)
30. Mao, S.; He, S.; Wu, J. Joint UAV Position Optimization and Resource Scheduling in Space-Air-Ground Integrated Networks With Mixed Cloud-Edge Computing. *IEEE Syst. J.* **2021**, *15*, 3992–4002. [\[CrossRef\]](#)
31. El Hammouti, H.; Benjillali, M.; Shihada, B.; Alouini, M.S. Learn-As-You-Fly: A Distributed Algorithm for Joint 3D Placement and User Association in Multi-UAVs Networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 5831–5844. [\[CrossRef\]](#)
32. Du, S.; Chen, X.; Jiao, L.; Lu, Y. Energy Efficient Task Offloading for UAV-assisted Mobile Edge Computing. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 6567–6571. [\[CrossRef\]](#)
33. Chen, B.; Li, N.; Li, Y.; Tao, X.; Sun, G. Energy Efficient Hybrid Offloading in Space-Air-Ground Integrated Networks. In Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC), Austin, TX, USA, 10–13 April 2022; pp. 1319–1324. [\[CrossRef\]](#)
34. Chen, T.; Liu, J.; Tang, Q.; Huang, T.; Liu, Y. Deep Reinforcement Learning Based Data Offloading in Multi-Layer Ka/Q Band LEO Satellite-Terrestrial Networks. In Proceedings of the IEEE 21st International Conference on Communication Technology (ICCT), Tianjin, China, 13–16 October 2021; pp. 1417–1422. [\[CrossRef\]](#)
35. Zhao, J.; Hu, X.; Du, X. Algorithm of task offloading and resource allocation based on reinforcement learning in edge computing. In Proceedings of the 2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Xi'an, China, 15–17 October 2021; Volume 5, pp. 1266–1269. [\[CrossRef\]](#)
36. Chen, X.; Ge, H.; Liu, L.; Li, S.; Han, J.; Gong, H. Computing Offloading Decision Based on DDPG Algorithm in Mobile Edge Computing. In Proceedings of the IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), Chengdu, China, 24–26 April 2021; pp. 391–399. [\[CrossRef\]](#)
37. Zhang, P.; Wang, C.; Kumar, N.; Liu, L. Space-air-ground integrated multi-domain network resource orchestration based on virtual network architecture: A DRL method. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 2798–2808. [\[CrossRef\]](#)
38. Liu, J.; Zhao, X.; Qin, P.; Geng, S.; Meng, S. Joint Dynamic Task Offloading and Resource Scheduling for WPT Enabled Space-Air-Ground Power Internet of Things. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 660–677. [\[CrossRef\]](#)
39. Wu, H.; Yang, X.; Bu, Z. Deep Reinforcement Learning for Computation Offloading and Resource Allocation in Satellite-Terrestrial Integrated Networks. In Proceedings of the 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), Helsinki, Finland, 19–22 June 2022; pp. 1–5. [\[CrossRef\]](#)

40. Nguyen, M.D.; Le, L.B.; Girard, A. Joint Computation Offloading, UAV Trajectory, User Scheduling, and Resource Allocation in SAGIN. In Proceedings of the GLOBECOM 2022—2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; pp. 5099–5104. [\[CrossRef\]](#)
41. Gao, Z.; Liu, A.; Han, C.; Liang, X. Max Completion Time Optimization for Internet of Things in LEO Satellite-Terrestrial Integrated Networks. *IEEE Internet Things J.* **2021**, *8*, 9981–9994. [\[CrossRef\]](#)
42. Cao, H.; Yu, G.; Chen, Z. Cooperative Task offloading and Dispatching Optimization for Large-scale Users via UAVs and HAP. In Proceedings of the 2023 IEEE Wireless Communications and Networking Conference (WCNC), Glasgow, UK, 26–29 March 2023; pp. 1–6. [\[CrossRef\]](#)
43. Xie, X.; Zhang, H.; Wang, J.; Chang, Q.; Wang, J.; Pal, N.R. Learning optimized structure of neural networks by hidden node pruning with  $L_{\{1\}}$  regularization. *IEEE Trans. Cybern.* **2019**, *50*, 1333–1346. [\[CrossRef\]](#) [\[PubMed\]](#)
44. Zhang, H.; Wang, J.; Sun, Z.; Zurada, J.M.; Pal, N.R. Feature selection for neural networks using group lasso regularization. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 659–673. [\[CrossRef\]](#)
45. Wang, B.; Li, X.; Huang, D.; Xie, J. A Profit Maximization Strategy of MEC Resource Provider in the Satellite-Terrestrial Double Edge Computing System. In Proceedings of the 2021 IEEE 21st International Conference on Communication Technology (ICCT), Tianjin, China, 13–16 October 2021; pp. 906–912. [\[CrossRef\]](#)
46. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*; Dy, J., Krause, A., Eds.; PMLR: Cambridge, MA, USA, 2018; Volume 80, pp. 1587–1596.
47. Gong, H.; Ge, H.; Ma, S.; Sun, A.; Chen, X.; Liu, L. Task Offloading Strategy Based on TD3 Algorithm in Cloud-Edge Collaborative MEC. In Proceedings of the 2022 4th International Conference on Natural Language Processing (ICNLP), Abu Dhabi, UAE, 7–11 December 2022; pp. 452–459. [\[CrossRef\]](#)
48. Hazarika, B.; Singh, K.; Biswas, S.; Li, C.P. DRL-Based Resource Allocation for Computation Offloading in IoV Networks. *IEEE Trans. Ind. Informatics* **2022**, *18*, 8027–8038. [\[CrossRef\]](#)
49. Chen, N.; Zhang, S.; Qian, Z.; Wu, J.; Lu, S. When Learning Joins Edge: Real-Time Proportional Computation Offloading via Deep Reinforcement Learning. In Proceedings of the IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 4–6 December 2019; pp. 414–421. [\[CrossRef\]](#)

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.