sensors

MDPI

Article

Computation Offloading and Resource Allocation Based on P-DQN in LEO Satellite Edge Networks

Xu Yang [1], Hai Fang [1], Yuan Gao [1], Xingjie Wang [2,*], Kan Wang [2] and Zheng Liu [2]

[1] Xi'an Institute of Space Radio Technology, Xi'an 710100, China; yangx@cast504.com (X.Y.); fangh@cast504.com (H.F.); gaoy69@cast504.com (Y.G.)
[2] School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China; wangkan@xaut.edu.cn (K.W.); liuzheng@xaut.edu.cn (Z.L.)
* Correspondence: 2221221101@stu.xaut.edu.cn

Abstract: Traditional low earth orbit (LEO) satellite networks are typically independent of terrestrial networks, which develop relatively slowly due to the on-board capacity limitation. By integrating emerging mobile edge computing (MEC) with LEO satellite networks to form the business-oriented "end-edge-cloud" multi-level computing architecture, some computing-sensitive tasks can be offloaded by ground terminals to satellites, thereby satisfying more tasks in the network. How to make computation offloading and resource allocation decisions in LEO satellite edge networks, nevertheless, indeed poses challenges in tracking network dynamics and handling sophisticated actions. For the discrete-continuous hybrid action space and time-varying networks, this work aims to use the parameterized deep Q-network (P-DQN) for the joint computation offloading and resource allocation. First, the characteristics of time-varying channels are modeled, and then both communication and computation models under three different offloading decisions are constructed. Second, the constraints on task offloading decisions, on remaining available computing resources, and on the power control of LEO satellites as well as the cloud server are formulated, followed by the maximization problem of satisfied task number over the long run. Third, using the parameterized action Markov decision process (PAMDP) and P-DQN, the joint computing offloading, resource allocation, and power control are made in real time, to accommodate dynamics in LEO satellite edge networks and dispose of the discrete-continuous hybrid action space. Simulation results show that the proposed P-DQN method could approach the optimal control, and outperforms other reinforcement learning (RL) methods for merely either discrete or continuous action space, in terms of the long-term rate of satisfied tasks.

Keywords: LEO satellite edge networks; offloading decision; resource allocation; hybrid action space; P-DQN

Citation: Yang, X.; Fang, H.; Gao, Y.; Wang, X.; Wang, K.; Liu, Z. Computation Offloading and Resource Allocation Based on P-DQN in LEO Satellite Edge Networks. *Sensors* **2023**, *23*, 9885. https://doi.org/10.3390/s23249885

Academic Editors: Xingwang Li, Kang An, Bin Li and Kefeng Guo

Received: 23 November 2023
Revised: 12 December 2023
Accepted: 14 December 2023
Published: 17 December 2023

1. Introduction

With the growth in global communications demand and the development of space Internet, connectivity to rural areas has become imperative for future networks. Since the traditional terrestrial network has limited coverage in remote areas, its infrastructure is vulnerable to natural disasters, e.g., earthquakes and floods, thus disrupting user communications [1]. Therefore, it is a prerequisite to support lower latency and more reliable communication in future wireless networks [2].

In the past few decades, satellite and terrestrial networks typically developed independently and competed with each other [3]. Although the terrestrial network is advantageous in terms of high-speed data transmission and low latency, its coverage is limited, covering only about 6% of the Earth's surface and about 20% of the land area [4]. In contrast, satellite networks are not subject to regional restrictions and can cover the globe, meeting the Internet needs in remote areas, sea and air. Besides, satellite networks have higher

survival when disasters occur, especially in earthquakes, yet also face the challenge of long-distance transmission.

Therefore, both industry and academia are promoting the integration of terrestrial and satellite communications, to achieve seamless coverage and high-quality service anytime and anywhere. It is apparent that global seamless communication will be an important component in 6G networks, and thus both academia and industry have begun to discuss its requirements, application scenarios, and potential solutions [5–8].

The 6G network will form a three-dimensional coverage of global communication through the interaction of satellite and terrestrial networks, forming a seamless three-dimensional coverage on a global scale, and is expected to provide heterogeneous services and seamless network coverage [9,10]. The integrated satellite-terrestrial network architecture can integrate the information of both networks, thereby ensuring wider network coverage and higher performance [11,12]. Yet, when providing ubiquitous and reliable services, the integrated satellite–terrestrial network also faces challenges, especially in meeting the growing quality of service (QoS) requirement. That is, with the rapid development of computing-intensive and -sensitive applications, the network has to offer a variety of computing services. More especially, users can offload part of or all computing tasks to the data center [13–15]. The data center, nevertheless, is typically built in remote areas, incurring high transmission cost and service latency, and thus failing in meeting the QoS requirements, e.g., high data rate, low latency, and low processing energy consumption [15–18].

As compared to terrestrial ones, low earth orbit (LEO) satellite networks are typically deployed in the space area with an orbital altitude of 500 to 2000 km. Different from high-orbit and medium-orbit satellites, the LEO satellite's channel fading and service latency would be greatly reduced. Further, since the LEO satellite network is closer to the ground, it has lower backhaul latency and smaller channel fading, free from the ground terrain [19]. The traditional LEO satellite network is usually limited by the finite onboard capacity. Fortunately, emerging mobile edge computing (MEC) can provide services with low latency, high reliability, high security, and high flexibility by deploying computing and storage resources closer to users [20–22]. Assisted by the MEC, LEO satellite edge networks are expected to deploy MEC servers on satellites and cooperate with cloud computing data centers to further reduce energy consumption and task response latency, forming an end-edge-cloud multi-level processing architecture for different business types. That is, MEC servers on satellites can act as edge nodes to provide computing services for the ground terminal, typically with limited capacity. It is also likely that ground terminals offload their tasks to cloud computing data centers [23]. Yet, due to the lack of reliable connectivity to data centers through terrestrial networks, e.g., in remote areas, some tasks have to be forwarded to data centers via the visible LEO satellite. In addition to the cloud-edge-end hierarchical architecture we are investigating, there is a rising trend in integrated continuum architectures. For instance, Trakadas et al. in [24] introduced the meta-operating system reference architecture (RAMOS) to tackle the data surge resulting from IoT proliferation, aiming to establish a dynamic, distributed, and trusted continuum for future data-intensive applications at the edge. Yet, creating a continuum from IoT to the edge and cloud still poses an ongoing challenge [25].

Therefore, we aim to propose a joint computing offloading and resource allocation method in the LEO satellite edge network, based on the parameterized deep Q-network (P-DQN) reinforcement learning (RL), to capture the dynamics in network conditions. The main contributions are listed as follows:

- To better simulate the real LEO network, the dynamic and changeable LEO satellite scenario is defined. The wireless channel with time-varying characteristics is modeled, the communication and computing models under three different offloading strategies are constructed, and the service latency model is obtained.
- The joint computing offloading and resource allocation problem in the LEO satellite edge network is built. Constraints on offloading decisions on processed tasks,

on remaining available computing resources, and on power control on both LEO satellites and the cloud server are respectively inferred, followed by the optimization problem formulation.

- For the highly dynamic LEO satellite edge network and the discrete-continuous hybrid action space, an MDP model with parameterized actions is constructed to capture the dynamics in computing offloading, resource allocation, and power control, and the P-DQN RL method is used to maximize the number of accessed tasks.

The rest is organized as follows. In Section 2, a brief summary of existing works on computation offloading and resource allocation in LEO satellite networks is provided. In Section 3, the system model is proposed and the optimization problem is established. In Section 4, the problem is further characterized by the parameterized Markov process and solved by the P-DQN. In Section 5, simulation experiments are conducted to verify the algorithm performance. Finally, Section 6 concludes the work and provides an outlook on future endeavors.

2. Related Work

In ground edge networks, there have been many works on joint computation offloading and resource allocation. Yan et al. in [26] introduced the multi-user edge network scenario, considered the task dependency among users, and formulated it as a mixed-integer program, to optimize both task offloading and power control decisions, for the minimization of a weighted sum of energy consumption and latency. Likely, a multi-user multi-task network scenario was presented in [27], by formulating it as a mixed integer program and considering the service caching, computation offloading, and resource allocation, to minimize the weighted sum of latency and energy consumption. Besides, Wu et al. in [28] introduced a multi-cell MEC-assisted network, developed an analytical model to decouple power control and computing resource allocation, and proposed heuristics. Tan et al. in [29] studied the multi-user cooperative MEC network based on orthogonal frequency division multiple access (OFDMA), and formulated the collaborative decision making, computation offloading, and resource allocation as a mixed nonlinear program. In particular, to minimize the total energy consumption of devices, a two-stage alternating framework is proposed to decompose the collaborative problem into two layers, of which the first one is the offloading decision generation method based on an ant colony system, and the second one is the resource allocation method based on deep Q network, to obtain the optimal power control, subcarrier assignment, and computing resource allocation, given offloading decisions. Acknowledging the importance of energy efficiency (EE) optimization, Ruan et al. in [30] focused on the energy-efficient power allocation in cognitive satellite–terrestrial networks. Besides, optimal power allocation schemes for both non-real-time and real-time applications were addressed in [31], for optimizing the EE of cognitive satellite users. Spantideas et al. in [32] introduced a power configuration algorithm based on deep Q-Learning for 5G cells, thereby optimizing both EE and throughput adequacy. Likewise, the joint power allocation and user association in wireless heterogeneous networks using the DRL was proposed in both [33,34]. However, due to technical limitations, EE optimization is not covered in the current work, and would be encapsulated in future work.

Moreover, unmanned aerial vehicle (UAV)-assisted MEC begins to emerge, bringing more sophisticated computation offloading issues. Li Bin et al. in [35] utilized the double deep Q-network algorithm to investigate the task offloading problem in UAV-enabled MEC with the digital twin, by optimizing the mobile terminal user association, UAV trajectory planning, transmission power distribution, and computing resource allocation, thereby minimizing the system energy consumption. Likewise, the UAV-assisted MEC was also proposed to support resource-intensive applications in [36]. More precisely, by introducing the digital twin-empowered MEC network with multiple UAVs and one ground base station, the multi-agent proximal policy optimization is used to save energy.

As compared to ground edge networks and UAV-assisted MEC, the research on computation offloading and resource allocation in LEO satellite edge networks is still in the preliminary stage. Considering the high dynamics in the LEO satellite environment, how to offload tasks to nodes with abundant resources, and how to allocate resources to those offloaded tasks have become the challenges. Qiu et al. in [37] proposed a software-defined space-ground integrated network framework for the management and orchestration of caching and computing resources, using deep Q-learning methods. Xu et al. in [38] proposed a satellite-assisted maritime network architecture on edge computing, using deep Q learning to minimize the total service latency. In aforementioned both works, although cloud servers with substantial computing resources are mentioned, both network models only include the LEO satellite layer and base station layer, and the explicit incorporation of cloud servers into the model is missing. In contrast, we aim to propose the multi-tier cloud-edge-end architecture, encompassing computing-capable end users, MEC-assisted LEO satellites, and the cloud server with rich resources, thereby providing users with a wider range of offloading options and access opportunities.

Furthermore, Cheng et al. in [39] used the deep reinforcement learning (DRL) method to learn the optimal offloading decision dynamically in an air-ground integrated edge network, meanwhile proposing heuristics to solve the mixed integer program of joint computing resource allocation and task scheduling. Cui et al. in [40] respectively used the Lagrange multiplier and DRL methods to optimize the service latency, provided the resource allocation is given. Wang et al. in [41] likely decomposed the joint problem into two sub-problems, using the Lagrange multiplier method for communication and computing resource allocation provided the computation offloading is preserved. The cooperative offloading problem in LEO satellite Internet of Things (IoTs) was studied in [42], where LEO satellites forward tasks to ground MEC servers, and the weighted latency and energy minimization problem is designed as a partially observable MDP (POMDP), followed by the multi-agent DRL framework.

In brief, existing works have extensively explored the challenges of computation offloading and resource allocation in either ground-edge networks [26–29] or UAV-assisted MEC [35,36]. Yet, unlike previous studies on satellite edge networks [39–42], we consider the more precise description of dynamic characteristics of the network, including the relative position variation between LEO satellites and ground terminals, together with time-varying channel fading. Further, in existing RL works addressing computation offloading in LEO satellite edge networks, it is common to decompose the original problem into two sub-problems [37–42]. In contrast, we resort to the P-DQN method, thereby handling the hybrid action space and offering a more integrated solution, without the intricate problem decomposition.

3. System Model and Problem Description

3.1. LEO Satellite Edge Network Model

The LEO satellite edge network model is shown below in Figure 1.

As shown in Figure 1, the network includes multiple ground terminals, several LEO satellites equipped with edge servers, and one cloud server. Let $\mathcal{L} = \{1, 2, \ldots, L\}$ be the satellite set, with L as the total number of satellites. Designate that LEO satellites use Ka-band (27–40 GHz) to provide access for ground terminals, and each satellite only takes charge of terminals within its coverage. Moreover, let $\mathcal{K} = \{1, 2, \ldots, K\}$ be the division of K regions, and then let $\mathcal{I}_k = \{1, 2, \ldots, I_k\}$ be the set of terminals in region k. Assume that all terminals within one coverage access the same satellite (each terminal, yet, can only access one satellite at one time), while each satellite can serve multiple terminals simultaneously. Since terminals within the same coverage have similar distances, the channel states (between terminals and LEO satellites) are almost identical, and the time division multiple access (TDMA) can work to avoid multi-user interference within the region. Next, let M be the cloud server. Due to the long distance and sophisticated geographical conditions, a

direct connection between M and terminals cannot be established. Thus, the service flow from terminals must be forwarded through LEO satellites to M.

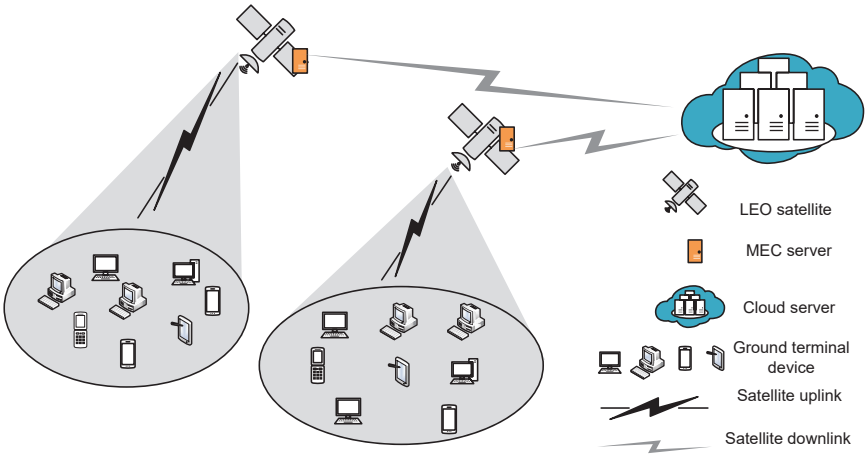

Figure 1. LEO satellite edge networks.

To better depict dynamics in the network due to the random arrival of tasks, the system time can be discretized into successive time slots with equal length, i.e., $\mathcal{T} = \{1, 2, \ldots, T\}$. Let $R_{k,i}^{\bar{t}}$ be the new arrival task of terminal i in region k during time slot $\bar{t} \in \mathcal{T}$, and $l_k \in \mathcal{L}$ be the satellite covering region k. Further, all tasks are latency sensitive, and denote $t_{\max}$ as the maximum tolerance latency per task and $c_{k,i}^{\bar{t}}$ as the size per packet to be processed by the task, respectively. Besides, only consider tasks surviving for several time slots, the duration of which is far less than the continuous coverage duration of LEO satellites for one region (which is set as about 9 min in [43]). As such, almost all tasks considered can be terminated (accessed or failed) within the LEO satellite covering the duration, and thus the handover and its associated unstable connections are reasonably ignored. Note that, due to the insufficient computing capability and limited resources in ground terminals, a larger processing latency would be incurred locally. In contrast, when offloading tasks to LEO satellites, a relatively larger propagation latency is preferred.

Thus, there are three options for the terminal to process task $R_{k,i}^{\bar{t}}$, and the latency depends on the computation offloading mode. Let $\mathcal{X}_{k,i}^{\bar{t},t} = \{x_{k,i}^{\bar{t},t,1}, x_{k,i}^{\bar{t},t,2}, x_{k,i}^{\bar{t},t,3}\}$ be the offloading set of task $R_{k,i}^{\bar{t}}$ at slot t (originating at $\bar{t}$ and not yet completely terminating until t), where $x_{k,i}^{\bar{t},t,1}, x_{k,i}^{\bar{t},t,2}, x_{k,i}^{\bar{t},t,3} \in \{0, 1\}$ and $x_{k,i}^{\bar{t},t,1} + x_{k,i}^{\bar{t},t,2} + x_{k,i}^{\bar{t},t,3} = 1$, i.e., each task $R_{k,i}^{\bar{t}}$ can only choose one processing method. In particular, $x_{k,i}^{\bar{t},t,1} = 1$, $x_{k,i}^{\bar{t},t,2} = 1$ and $x_{k,i}^{\bar{t},t,3} = 1$ denote that the task is processed locally, offloaded to the satellite, and offload to the cloud server, respectively.

3.2. Channel Model

Due to the high mobility of LEO satellites, the relative position between satellites and ground terminals changes rapidly, and so do the free space loss, atmospheric fading, and many other factors involved in the satellite–terrestrial link [44]. As shown in Figure 1, the link from terminals to the LEO satellite is named as satellite uplink, and that from the LEO satellite to the cloud server is named as the satellite downlink. Note that when returning processing results to terminals, the transmission latency (and thus the link) is ignored due to the small-sized result. Thus, when offloading task $R_{k,i}^{\bar{t}}$ to satellites, the data transmission only goes through the satellite uplink; when offloading it to the cloud server through the LEO satellite, the data transmission must go through the satellite uplink first and then through the satellite downlink.

As stated in Section 3.1, the channel state of all terminals in the same satellite coverage is almost identical, so the satellite uplink channel state $g_k^{t,L}$ in region k at slot t can be defined as

$$g_k^{t,L} = G_s^E \cdot G_k^{t,L} \cdot G_r^L, \tag{1}$$

where G_s^E denotes the transmit antenna gain of terminals, $G_k^{t,L}$ is the channel fading between region k and the associated satellite at slot t, and G_r^L represents the receiving antenna gain of satellites. In particular, the channel fading between terminals and satellites generally includes free space path loss, atmospheric fading, and small-scale fading (obeying the Rician distribution) [45], i.e.,

$$G_k^{t,L} = \left(\frac{c}{4\pi d_k^{t,L} f_e}\right)^2 \cdot \Phi\left(\alpha^{t,L}\right) \cdot \psi, \tag{2}$$

where c represents the light speed, $d_k^{t,L}$ is the distance between region k and its access satellite at slot t, f_e denotes the carrier frequency, $\Phi\left(\alpha^{t,L}\right)$ specifies the atmospheric fading, and ψ is the Rician distributed small-scale fading. More precisely, atmospheric fading $\Phi\left(\alpha^{t,L}\right)$ is expressed as

$$\Phi\left(\alpha^{t,L}\right) = 10^{\left(\frac{3\delta}{10\sin\alpha^{t,L}}\right)}, \tag{3}$$

where $\sin\alpha^{t,L} = H/d_k^{t,L}$, H is the orbital altitude of LEO satellites, and δ is the attenuation through rain and clouds, separately. The Rayleigh fading channel models (e.g., in [46,47]) is not used throughout.

Further, the channel state of satellite downlink between LEO satellites and the cloud server M for region k at slot t is represented as $g_k^{t,M}$ as

$$g_k^{t,M} = G_s^L \cdot G_k^{t,M} \cdot G_r^M, \tag{4}$$

where G_s^L is the transmit antenna gain of LEO satellites, $G_k^{t,M}$ is the channel fading between LEO satellites and the cloud server for region k at slot t , and G_r^M is the receiving antenna gain of cloud server. Likewise, the channel fading of satellite downlink also includes free space path loss, atmospheric fading and Rician distributed small-scale fading, respectively, i.e.,

$$G_k^{t,M} = \left(\frac{c}{4\pi d_k^{t,M} f_l}\right)^2 \cdot \Phi\left(\alpha^{t,M}\right) \cdot \psi, \tag{5}$$

where $d_k^{t,M}$ is the distance between M and satellite l_k, f_l is the carrier frequency of LEO satellites, and $\Phi\left(\alpha^{t,M}\right)$ is the atmospheric fading, i.e.,

$$\Phi\left(\alpha^{t,M}\right) = 10^{\left(\frac{3\delta}{10\sin\alpha^{t,M}}\right)}, \tag{6}$$

with $\sin\alpha^{t,M} = H/d_k^{t,M}$.

3.3. Latency and Satisfied Task Model

First, when locally processing task $R_{k,i}^{\bar{t}}$ on terminals, the total service latency includes only the processing latency, i.e., $T_{k,i}^{\bar{t},\mathrm{tol}} = c_{k,i}^{\bar{t}}/C_{k,i}^{\bar{t},E}$, where $c_{k,i}^{\bar{t}}$ represents the packet size of $R_{k,i}^{\bar{t}}$, and $C_{k,i}^{\bar{t},E}E$ is the remaining constant computing resource always allocated to $R_{k,i}^{\bar{t}}$, respectively. If $T_{k,i}^{\bar{t},\mathrm{tol}} \leq t_{\max}$, then $R_{k,i}^{\bar{t}}$is satisfied; otherwise, it fails.

Second, when offloading $R_{k,i}^{\bar{t}}$ to satellite l_k, the total service latency consists of processing, transmission, and propagation ones. The processing latency is $c_{k,i}^{\bar{t}}/C_{k,i}^{\bar{t},L}$, where $C_{k,i}^{\bar{t},L}$

is the computing resource always allocated to $R_{k,i}^{\bar{t}}$ by l_k, and the propagation latency is $2d_k^{\bar{t},L}/c$. Further, the transmission rate in the satellite uplink becomes

$$s_{k,i}^{\bar{t},t,L} = W_k^L \log_2 \left(1 + \frac{P_{k,i}^{\bar{t},L} g_k^{t,L}}{\sum_{m=1,m\neq k}^{K} \sum_{j=1}^{I} P_{m,j}^{\bar{t},L} g_m^{t,L} + \sigma^2} \right), \tag{7}$$

where W_k^L represents the link bandwidth allocated to region k, $s_{k,i}^{\bar{t},t,L}$ is the transmission rate of $R_{k,i}^{\bar{t}}$ (originating at slot $\bar{t}$ and not yet transmitted completely until slot t) at t, $P_{k,i}^{\bar{t},L}$ is the transmit power allocated to $R_{k,i}^{\bar{t}}$ and we assume that $P_{k,i}^{\bar{t},L}$ does not vary across slots for simplicity, $\sum_{m=1,m\neq k}^{K} \sum_{j=1}^{I} P_{m,j}^{\bar{t},L} g_m^{t,L}$ is the interference power caused by other regions except k at slot t, and σ^2 is the noise power, respectively. Since the channel state varies across slots, the transmission rate also changes. Besides, the remaining data (not yet transmitted) of $R_{k,i}^{\bar{t}}$ becomes

$$z_{k,i}^{\bar{t},t} = \begin{cases} c_{k,i}^{\bar{t}} - s_{k,i}^{\bar{t},t,L} \cdot \tau, & t = \bar{t} \\ z_{k,i}^{\bar{t},t-1} - s_{k,i}^{\bar{t},t,L} \cdot \tau, & t > \bar{t} \end{cases}, \tag{8}$$

where τ is the size per slot. At the beginning of each slot t, judgments are made depending on the state of $R_{k,i}^{\bar{t}}$. Define variables $\eta_{k,i}^{\bar{t},t}$ to represent the condition of $R_{k,i}^{\bar{t}}$ (originating at slot $\bar{t}$) at current slot t, i.e.,

$$\eta_{k,i}^{\bar{t},t} = \begin{cases} 1, & \text{if } T_{k,i}^{\bar{t},\text{tol}} \leq t_{\max} \\ 0, & \text{if } T_{k,i}^{\bar{t},\text{tol}} > t_{\max} \text{ or } T_{k,i}^{\bar{t},\text{genr}} \geq t_{\max} \\ -1, & \text{if } T_{k,i}^{\bar{t},\text{genr}} < t_{\max} \end{cases}, \tag{9}$$

where $T_{k,i}^{\bar{t},\text{tra}}$ is the traversed latency from $\bar{t}$ to t of unfinished task $R_{k,i}^{\bar{t}}$. $\eta_{k,i}^{\bar{t},t} = 1$, $\eta_{k,i}^{\bar{t},t} = 0$ and $\eta_{k,i}^{\bar{t},t} = -1$ indicates that $R_{k,i}^{\bar{t}}$ is exactly finished and judged to be satisfied, that $R_{k,i}^{\bar{t}}$ is judged to fail, and that the judgment has to be postponed to next slot, all at slot t. The explicit judgment on satisfied conditions of tasks are shown in Algorithm 1.

Third, when offloading $R_{k,i}^{\bar{t}}$ to the cloud server M, the total service latency consists of processing, transmission, and propagation ones as well. The processing latency is $c_{k,i}^{\bar{t}}/C_{k,i}^{\bar{t},M}$, where $C_{k,i}^{\bar{t},M}$ is the computing resource always allocated to $R_{k,i}^{\bar{t}}$, and the propagation latency is $(2d_k^{\bar{t},L} + 2d_k^{\bar{t},M})/c$. Further, the transmission rate in the satellite downlink becomes

$$s_{k,i}^{\bar{t},M} = W_k^M \log_2 \left(1 + \frac{P_{k,i}^{\bar{t},M} g_k^{t,M}}{\sum_{m=1,m\neq k}^{K} \sum_{j=1}^{I} P_{m,j}^{\bar{t},M} g_m^{t,M} + \sigma^2} \right), \tag{10}$$

where W_k^M represents the link bandwidth allocated to region k, and $\sum_{m=1,m\neq k}^{K} \sum_{j=1}^{I} P_{m,j}^{\bar{t},M} g_m^{t,M} + \sigma^2$ is the interference power caused by other regions except k at slot t. Assume that the satellite works in the full-duplex mode when forwarding the data from terminals to M via satellites, the transmission rate of $R_{k,i}^{\bar{t}}$ becomes $s_{k,i}^{\bar{t},t,M'} = \min\{s_{k,i}^{\bar{t},t,L}, s_{k,i}^{\bar{t},t,M}\}$. Since the channel state varies across slots, the transmission rate also changes, and the judgment on satisfied conditions resembles Algorithm 1, with a slight difference on the data rate calculation.

Algorithm 1 Judgment on satisfied conditions of task $R_{k,i}^{\bar{t}}(\bar{t} \leq t)$ at slot t

Input: unfinished task $R_{k,i}^{\bar{t}}(\bar{t} \leq t)$
Output: judgment result $\eta_{k,i}^{\bar{t},t}$

```
Initialize η_{k,i}^{t̄,t} = −1
while η_{k,i}^{t̄,t} = −1 do
    Calculate the size of remaining data z_{k,i}^{t̄,t}
    if z_{k,i}^{t̄,t} ≤ 0 then
        Obtain the total service latency: T_{k,i}^{t̄,tol} = (c_{k,i}^{t̄} / C_{k,i}^{t̄,L}) + (t − t̄ + 1) · τ + (2d_k^{t̄,L} / c)
        if T_{k,i}^{t̄,tol} ≤ t_max then
            η_{k,i}^{t̄,t} = 1
        else
            η_{k,i}^{t̄,t} = 0
        end if
    else
        Calculate the traversed latency: T_{k,i}^{t̄,tra} = (t − t̄ + 1) · τ + (d_k^{t̄,L} / c)
        if T_{k,i}^{t̄,tra} ≥ t_max then
            η_{k,i}^{t̄,t} = 0
        else
            η_{k,i}^{t̄,t} = −1
        end if
    end if
end while
```

3.4. Problem Formulation

Since multiple tasks may arrive at one region at the same slot, they (offloaded to the same satellite) would compete for computing resources. It is prerequisite to jointly optimize computing offloading and resource allocation per slot, to maximize the average number of satisfied tasks over the long run, i.e.,

$$\begin{aligned} \max \ & \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{i=1}^{I} \sum_{\bar{t}=1}^{t} \eta_{k,i}^{\bar{t},t} \\ \text{s.t.} \quad & \text{C1:} \quad \textstyle\sum_{i=1}^{I} C_{k,i}^{\bar{t},L} \leq \omega_k^{t,L}, \forall k, \forall \bar{t}, t \\ & \text{C2:} \quad 0 \leq P_{k,i}^{\bar{t},L} \leq x_{k,i}^{\bar{t},t,2} \cdot P_{k,i,\max}^{L}, \quad \forall k, \forall i, \forall \bar{t}, t \\ & \text{C3:} \quad \textstyle\sum_{k=1}^{K} \sum_{i=1}^{I} C_{k,i}^{\bar{t},M} \leq \omega^{t,M}, \quad \forall \bar{t}, t \\ & \text{C4:} \quad 0 \leq P_{k,i}^{\bar{t},M} \leq x_{k,i}^{\bar{t},t,3} \cdot P_{\max}^{M}, \quad \forall k, \forall i, \forall \bar{t}, t \\ & \text{C5:} \quad 0 \leq C_{k,i}^{\bar{t},L} \leq x_{k,i}^{\bar{t},t,2} \cdot \omega_k^{t,L}, \quad \forall k, \forall i, \forall \bar{t}, t \\ & \text{C6:} \quad 0 \leq C_{k,i}^{\bar{t},M} \leq x_{k,i}^{\bar{t},t,3} \cdot \omega^{t,M}, \quad \forall k, \forall i, \forall \bar{t}, t \end{aligned} \tag{11}$$

where $\omega_k^{t,L}$ and $\omega^{t,M}$ are, respectively, the remaining amount of computing resources on l_k and M, both at slot t, and $P_{k,i,\max}^{L}$ and $P_{\max}^{M}$ are the constant maximum transmitting power budget per terminal and per satellite. C_1 denotes that the sum computing resources allocated to offloaded task does not exceed the remaining ones of l_k, C_2 denotes that the transmit power on $R_{k,i}^{\bar{t}}$ does not exceed the budget, C_3 specifies that the sum computing resources allocated to tasks (offloaded to M) are below the remaining capacity, and C_4 is analogous to C_2, except that the power is from l_k to M. Further, C_5 (C_6) indicates that only when $x_{k,i}^{\bar{t},t,2} = 1$ ($x_{k,i}^{\bar{t},t,3} = 1$) holds, $C_{k,i}^{\bar{t},L}$ ($C_{k,i}^{\bar{t},M}$) can take positive values; otherwise, $C_{k,i}^{\bar{t},L} = 0$ ($C_{k,i}^{\bar{t},M} = 0$) must hold.

4. P-DQN-Based Approach

Traditional RL methods such as DQN, actor-critic, and asynchronous actor-critic (A3C) are designed to handle discrete action spaces. The DDPG, on the other hand, is tailored for dealing with continuous actions. To adapt above RL methods to the discrete-continuous hybrid action space, there are two approaches, i.e., either discretizing the hybrid action space, or relaxing it into a continuous one, which would result in a high-dimensional action space. In this work, we use one prevailing architecture, namely P-DQN, which is directly appropriate for hybrid action space without any approximation or relaxation. In particular, existing P-DQN frameworks (which are predominantly used in the game control [48]), are enabled to address the computation offloading in LEO satellite edge networks. To adapt the classical P-DQN, we integrate offloading decisions with resource allocation into one hybrid action space. More precisely, the MDP with parameterized action space has to be constructed, followed by assessing the satisfaction of tasks and establishing the deferred reward function. This type of parameterized action space facilitates the maximization of satisfied task numbers.

4.1. MDP with Parameterized Action Space

The parameterized action MDP (PAMDP) model is an extension of standard MDP [49]. Note that the MDP is represented by a quadruple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, which are the state space, the action space, the state transition probability set, and the reward function set, separately. In contrast, the PAMDP redefines the discrete-continuous hybrid action space in the MDP, as follows:

$$\mathcal{A} = \{(v, \boldsymbol{n}_v) \mid \boldsymbol{n}_v \in \mathcal{N}_v \text{ for all } v \in \mathcal{V}\}, \tag{12}$$

where $\mathcal{V} = \{1, \ldots, V\}$ is the set of discrete actions, and $\mathcal{N}_v$ is the set of continuous parameters of each $v \in \mathcal{V}$. A high-level action v is first preserved from $\mathcal{V}$, and then the low-level parameter $\boldsymbol{n}_v \in \mathcal{N}_v$ associated with v is selected. In particular, the PAMDP model paper can be established as follows:

- **State space:** For each $\boldsymbol{s}^t \in \mathcal{S}$, define $\boldsymbol{s}^t = \left\{\mathcal{Y}^t, \mathcal{Z}^t, \left\{\omega_k^{t,L}\right\}_{k\in\mathcal{K}}, \omega^{t,M}\right\}$, where $\mathcal{Y}^t$ and $\mathcal{Z}^t$, respectively, represent the sets of new arrival tasks and being already processed ones.
- **Parameterized action space:** Define the parameterized action as $\mathcal{A} = \left\{\boldsymbol{a}_{k,i}^{\bar{t},t}\right\}$, where $\boldsymbol{a}_{k,i}^{\bar{t},t} = \{x_{k,i}^{\bar{t},t,1}, x_{k,i}^{\bar{t},t,2}(C_{k,i}^{\bar{t},L}, P_{k,i}^{\bar{t},L}), x_{k,i}^{\bar{t},t,3}(C_{k,i}^{\bar{t},M}, P_{k,i}^{\bar{t},M})\}, \bar{t} \le t$. In particular, $x_{k,i}^{\bar{t},t,1}$, $x_{k,i}^{\bar{t},t,2}$, and $x_{k,i}^{\bar{t},t,3}$ are three types of offloading decisions. For $x_{k,i}^{\bar{t},t,1}$, and the task is processed locally without parameters; for $x_{k,i}^{\bar{t},t,2}$, the task is offloaded to the LEO satellite, the parameters are $C_{k,i}^{\bar{t},L}$ and $P_{k,i}^{\bar{t},L}$; and for $x_{k,i}^{\bar{t},t,3}$, the task is offloaded to the cloud server, and the parameters become $C_{k,i}^{\bar{t},M}$ and $P_{k,i}^{\bar{t},M}$.
- **Transition probability:** A model-free RL architecture is used since both state and action spaces are high-dimensional and we cannot give the precise state transfer.
- **Reward function:** To judge all tasks in $\mathcal{Z}^t$ per slot, the temporal reward function per task can be defined as $r_{k,i}^{\bar{t},t}$. In particular, when the task is completed in the current slot, $r_{k,i}^{\bar{t},t}$ takes the large positive value; when the task is judged to be transmitted continuously, $r_{k,i}^{\bar{t},t}$ is temporarily set to be zero; and when the task fails, $r_{k,i}^{\bar{t},t}$ is finally set to be negative.

4.2. P-DQN Training

The original Bellman equation in Q-learning and DQN is expressed as follows:

$$Q(\boldsymbol{s}, \boldsymbol{a}) = \mathop{\mathbb{E}}_{r_t, \boldsymbol{s}_{t+1}}\left[r_t + \gamma \max_{a' \in \mathcal{A}} Q(\boldsymbol{s}_{t+1}, \boldsymbol{a}') \middle| \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}\right]. \tag{13}$$

In the hybrid action space $\mathcal{A}$, for each $\boldsymbol{a} \in \mathcal{A}$, the action value function is defined as $Q(\boldsymbol{s}, \boldsymbol{a}) = Q(\boldsymbol{s}, v, \boldsymbol{n}_v)$. If v_t is the discrete action selected at slot t and $\boldsymbol{n}_{v_t}$ is its associated continuous parameter, then the Bellman equation can be written as

$$Q(\boldsymbol{s}_t, v_t, \boldsymbol{n}_{v_t}) = \mathop{\mathbb{E}}_{r_t, \boldsymbol{s}_{t+1}} \left[r_t + \gamma \max_{v \in \mathcal{V}} \sup_{\boldsymbol{n}_v \in \mathcal{N}_v} Q(\boldsymbol{s}_{t+1}, v, \boldsymbol{n}_v) \middle| \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = (v_t, \boldsymbol{n}_{v_t}) \right]. \tag{14}$$

First, solve $\boldsymbol{n}_v^* = \arg\sup_{\boldsymbol{n}_v \in \mathcal{N}_v} Q(\boldsymbol{s}_{t+1}, v, \boldsymbol{n}_v)$ for all $v \in \mathcal{V}$ and then take the maximum $Q(\boldsymbol{s}_{t+1}, v, \boldsymbol{n}_v^*)$. When Q is fixed, for any $\boldsymbol{s}$ and v, $\arg\sup_{\boldsymbol{n}_v \in \mathcal{N}_v} Q(\boldsymbol{s}, v, \boldsymbol{n}_v)$ can be regarded as the mapping $\boldsymbol{n}_v^Q : \mathcal{S} \to \mathcal{N}_v$, and then (14) becomes

$$Q(\boldsymbol{s}_t, v_t, \boldsymbol{n}_{v_t}) = \mathop{\mathbb{E}}_{r_t, \boldsymbol{s}_{t+1}} \left[r_t + \gamma \max_{v \in \mathcal{V}} Q\left(\boldsymbol{s}_{t+1}, v, \boldsymbol{n}_v^Q(\boldsymbol{s}_{t+1}) \right) \middle| \boldsymbol{s}_t = \boldsymbol{s} \right]. \tag{15}$$

Similar to DQN, the neural network $Q(\boldsymbol{s}, v, \boldsymbol{n}_v; \boldsymbol{w})$ is used to approximate $Q(\boldsymbol{s}, v, \boldsymbol{n}_v)$ [50], with $\boldsymbol{w}$ as the network weight. For such $Q(\boldsymbol{s}, v, \boldsymbol{n}_v; \boldsymbol{w})$, the deterministic policy network $\boldsymbol{n}_v(\cdot; \boldsymbol{\theta}) : \mathcal{S} \to \mathcal{N}_v$ can approximate $\boldsymbol{n}_v^Q(\boldsymbol{s})$, with $\boldsymbol{\theta}$ as the policy network weight. That is, given $\boldsymbol{w}$, for each $v \in \mathcal{V}$, $\boldsymbol{\theta}$ is obtained as

$$Q(\boldsymbol{s}, v, \boldsymbol{n}_v(\boldsymbol{s}; \boldsymbol{\theta}); \boldsymbol{w}) \approx \sup_{\boldsymbol{n}_v \in \mathcal{N}_v} Q(\boldsymbol{s}, v, \boldsymbol{n}_v; \boldsymbol{w}), \tag{16}$$

where $\boldsymbol{w}$ is estimated by minimizing the mean square Bellman error by the gradient descent. In the t-th step (slot) and with the multi-step algorithm with j, the j-th step's target becomes

$$y_t = \sum_{i=0}^{j-1} \gamma^i r_{t+i} + \gamma^j \max_{v \in \mathcal{V}} Q(\boldsymbol{s}_{t+j}, v, \boldsymbol{n}_v(\boldsymbol{s}_{t+j}, \boldsymbol{\theta}_t); \boldsymbol{w}_t). \tag{17}$$

Then, the least squared loss function is used to train $\boldsymbol{w}$ as follows:

$$L_t(\boldsymbol{w}) = \frac{1}{2} [Q(\boldsymbol{s}_t, v_t, \boldsymbol{n}_{v_t}; \boldsymbol{w}) - y_t]^2. \tag{18}$$

In particular, the objective is to find $\boldsymbol{\theta}$ maximizing $Q(\boldsymbol{s}, v, \boldsymbol{n}_v(\boldsymbol{s}; \boldsymbol{\theta}); \boldsymbol{w})$ when $\boldsymbol{w}$ is fixed, whereas conventional loss functions typically require minimization. Therefore, a negative sign is added in (19) to formulate it as a loss function, allowing us to simultaneously maximize the objective while minimize the loss function, as follows:

$$L_t(\boldsymbol{\theta}) = -\sum_{v=1}^{V} Q(\boldsymbol{s}_t, v, \boldsymbol{n}_v(\boldsymbol{s}_t; \boldsymbol{\theta}); \boldsymbol{w}_t)/V. \tag{19}$$

Next, the data in the experience replay pool is used to obtain the stochastic gradient $\nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$ and $\nabla_{\boldsymbol{\theta}} L_t(\boldsymbol{\theta}_t)$, and both weights are updated as follows:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \beta_t \nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t), \tag{20}$$

and

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \zeta_t \nabla_{\boldsymbol{\theta}} L_t(\boldsymbol{\theta}_t), \tag{21}$$

where β_t and ζ_t, respectively, denote the step sizes for updating parameters $\boldsymbol{w}$ and $\boldsymbol{\theta}$.

Till now, the joint computation offloading and resource allocation algorithm for LEO edge networks with P-DQN is listed in Algorithm 2, together with the flowchart in Figure 2. Apparently, the P-DQN is an online and off-policy RL method.

Algorithm 2 Joint computation offloading and resource allocation with P-DQN

Input: step sizes $\{\beta_t\}_{t=1}^{T}$ and $\{\zeta_t\}_{t=1}^{T}$, exploration rate ε and batch size U.

Initialize the simulation environment parameter settings.
Initialize the network parameters $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$, and empty the replay buffer D.
for $t = 1, 2, ..., T$ **do**
  Calculate action parameters (computing resources and power): $\boldsymbol{n}_v(\boldsymbol{s}_t, \boldsymbol{\theta}_t) \rightarrow \boldsymbol{n}_v$.
  Select the action $\boldsymbol{a}_t = (v_t, \boldsymbol{n}_{v_t})$ according to the ϵ-greedy strategy:

$$\boldsymbol{a}_t = \begin{cases} (v_t, \boldsymbol{n}_{v_t}) \text{ such that } v_t = \arg\max_{v \in \mathcal{V}} Q(\boldsymbol{s}_t, v, \boldsymbol{n}_v; \boldsymbol{w}_t), & \text{with probability } 1-\varepsilon \\ \text{sample an action in } \mathcal{A} \text{ randomly.} & \text{with probability } \varepsilon \end{cases}$$

  Execute $\boldsymbol{a}_t$ and store the tuple $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ into D.
  Randomly sample U tuples $(\boldsymbol{s}_u, \boldsymbol{a}_u, r_u, \boldsymbol{s}_{u+1})$ from D, $u = 1, 2, \ldots, U$.
  Compute the target Q value:

$$y_u = \begin{cases} r_u, & \text{if is\_end is true} \\ r_u + \max_{v \in \mathcal{V}} \gamma Q(\boldsymbol{s}_{u+1}, v, \boldsymbol{n}_v(\boldsymbol{s}_{u+1}, \boldsymbol{\theta}_t); \boldsymbol{w}_t). & \text{otherwise} \end{cases}$$

  Use the experience replay to obtain the stochastic gradients $\nabla_{\boldsymbol{w}} L_t(\boldsymbol{w}_t)$ and $\nabla_{\boldsymbol{\theta}} L_t(\boldsymbol{\theta}_t)$.
  Update the parameters $\boldsymbol{w}$ and $\boldsymbol{\theta}$ based on (20) and (21).
end for

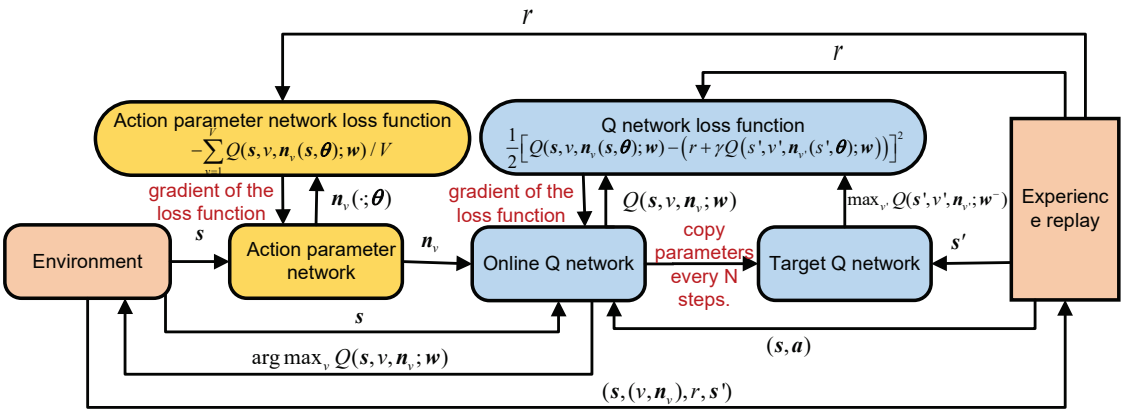

Figure 2. Flowchart for joint computation offloading and resource allocation with P-DQN.

5. Simulations and Results Analysis

5.1. Parameter Settings

Experiments are conducted on a simulation server equipped with one NVIDIA GeForce RTX 3060 graphics card, one 12th generation 6-core processor, and two 8 GB RAM modules. The software environment involves Python 3.9.13, PyTorch 1.13.0, and satellite tool kit (STK 11.6). Detailed parameter settings are presented in Table 1.

In particular, to acquire position information of LEO satellites across 100 time slots, three LEO satellites are simulated at an orbital height of 900 Km using STK, thereby obtaining position information reports. The CPU's computing power is configured at 1000 cycles/bit [51], making the computing resources on the LEO satellite and cloud server reach 3×10^7 bit/s and 1×10^8 bit/s, respectively [40]. Therefore, the magnitudes of computing resources in the figures below may appear smaller than those in Table 1, but in reality, equivalent, with the only difference in units.

Table 1. Simulation parameters.

Parameters	Value
Length per time slot	0.1 s
Slot number per episode	100
Task size	$[8 \times 10^4, 1.2 \times 10^5]$ bits
Number of LEO satellites	3
LEO satellite orbit altitude	900 Km
Maximum transmit power per LEO satellite	100 w
Maximum transmission power per ground terminal	20 w
Carrier frequency of Ka-Band	30 GHz
Link bandwidth	25 MHz
Number of terminals in region 1	14
Number of terminals in region 2	12
Number of terminals in region 3	8
Noise power spectral density	−174 dBm/Hz
Computing resources per LEO satellite	3×10^{10} cycle/s
Computing resources of cloud server	1×10^{11} cycle/s
Maximum tolerance latency	260 ms
Discounting factor	0.9

Furthermore, for the Q network, the input layer's dimensionality equals the sum of state space and parameter action dimensions, while the output layer's dimensionality matches that of the action space. Hidden layers comprise three fully connected ones with 256, 128, and 64 neurons, respectively. Moreover, rectified linear unit (ReLU) activation functions are utilized for each hidden layer for nonlinear mappings. Next, for the parameterized policy network, its input layer matches with the dimensionality of state space, hidden layers mirror those of the Q network, and the output layer's dimensionality corresponds to that of the parameter action, utilizing the hyperbolic tangent (tanh) function as its activation.

5.2. Performance Analysis

To validate the effectiveness of the proposed P-DQN method, the following four baseline methods are listed as follows:

(1) Random offloading (RO): Randomly offloading tasks locally, to LEO satellites and to the cloud server [52].
(2) Average resource allocation (ARA): Computing resources on both LEO satellites and the cloud server are evenly shared among offloaded tasks [40].
(3) DQN offloading (DQNO): The DQN is only used for the task offloading [52].
(4) Deep deterministic policy gradient (DDPG) resource allocation (DDPGRA): The DDPG is used to allocate both computing and power resources for already offloaded tasks.

Figure 3 compares the average reward per episode of the proposed method at different learning rates, set to be 0.001, 0.0001, and 0.00001, respectively. Figure 3 clearly shows that the learning rate variation significantly impacts the converged average return and convergence performance. With the learning rate of 0.001, the proposed method begins to converge at episode 10 or so, but next falls into the local optimum with the lower return value. With 0.00001, it does not converge until episode 80 or so. Further, with 0.0001, the proposed method not only demonstrates the improved average return per episode, but

also shows a relatively fast convergence rate. It is inferred that a high learning rate can lead to quick convergence but increases the risk of being trapped in the local optimum. Conversely, the lower learning rate results in a smaller step size, thus slowing down the convergence rate to the optimum.

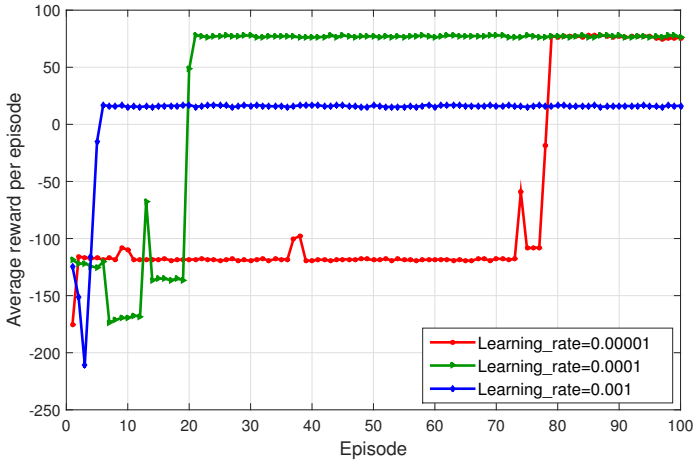

Figure 3. Average reward under different learning rates.

Figure 4 compares the average return per episode of the proposed method for different batch sizes (32, 64, and 128). The batch size of 32 results in slower convergence and lower average return. Conversely, the batch size of 128 results in a faster convergence rate, but is easily trapped into the local optimum. For 64, both convergence and return values are acceptable.

There, in the sequel, we pick the learning rate of 0.0001 and a sample batch size of 64 for method comparison.

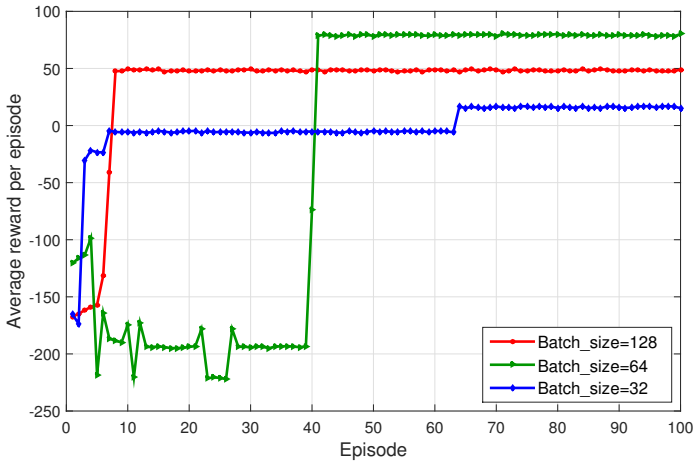

Figure 4. Average return under different batchsize settings.

Figure 5 compares the rate of satisfied tasks under different computing resources budgets in the cloud server. Both RO-ARA and RO-DDPGRA methods use random offloading, thus resulting in a lower access rate when available computing resources are less. Although the rate is improved as computing resources become more, the improvement is limited. This is because that the RO method would not offload more tasks to the cloud server, even with substantial resources. In contrast, both the proposed method and DQNO-ARA tend to prioritize the task of offloading to servers with ample resources. Yet, since the proposed method uses the parameterized continuous resource optimization and DQNO-ARA is just the equal one, the former one obtains the higher rate. Further, as computing resources are gradually increased, the values of both methods continue to grow, yet with

the narrowed gap. It is intuitive that more available computing resources would weaken the role of dynamic resource allocation.

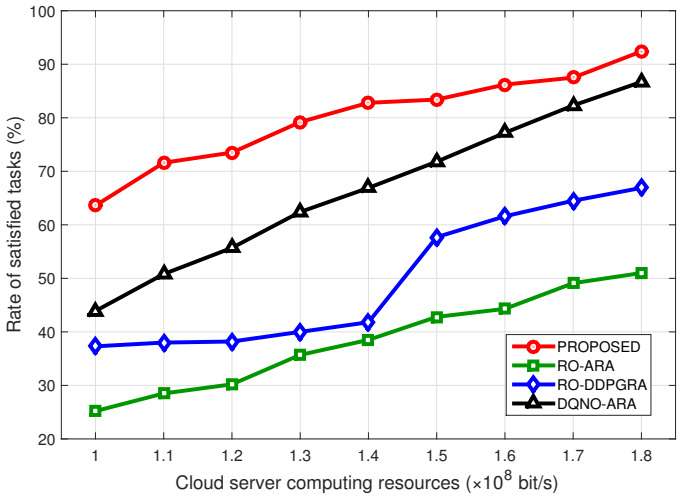

Figure 5. Rate of satisfied tasks under different computing resources budgets in the cloud server.

Figure 6 illustrates the rate of satisfied tasks at different computing resource budgets in LEO satellites. When available resources are less, all methods exhibit relatively lower satisfied task rates. Both RO-ARA and RO-DDPGRA use random offloading, showing an approximately linear growth with increased resources. By comparison, proposed method and DQNO-ARA consistently achieve higher satisfied rates, surpassing the other two RO methods. Likely, due to the exploitation in dynamic resource allocation, proposed method outperforms DQNO-ARA, nevertheless with the narrowed gap with more resources.

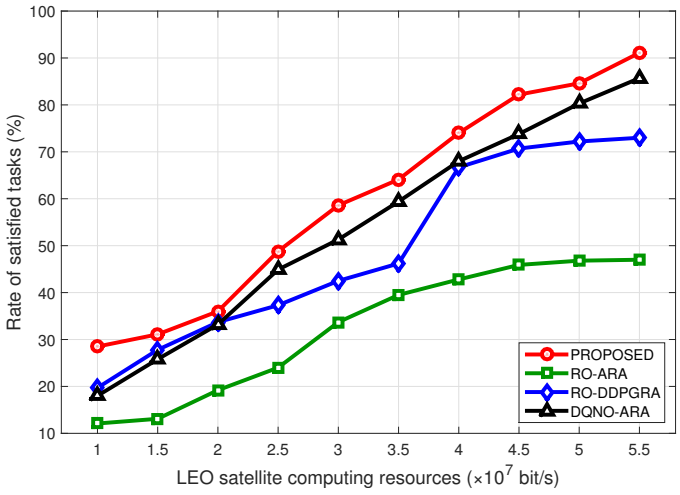

Figure 6. Rate of satisfied tasks under different computing resource budgets of LEO satellites.

In Figure 7, the rate of satisfied tasks under different maximum tolerance latency are compared. As the latency tolerance increases, the rates of all methods show an approximately linear growth. Given smaller tolerance, the rates of three other methods are almost identical except for the RO-ARA method. However, as the latency tolerance increases, proposed method exhibits significant advantages. In particular, at the tolerance of 290 ms, proposed method is improved by 7%, 22%, and 27% over three other ones, respectively. Note that even at the tolerance of 290 ms, the rates of satisfied tasks of RO-ARA and RO-DDPGRA are still relatively lower, due to the fact that some tasks are randomly offloaded locally. As such, there tasks cannot be completed within the tolerance, owing to the limited computing resources in ground terminals.

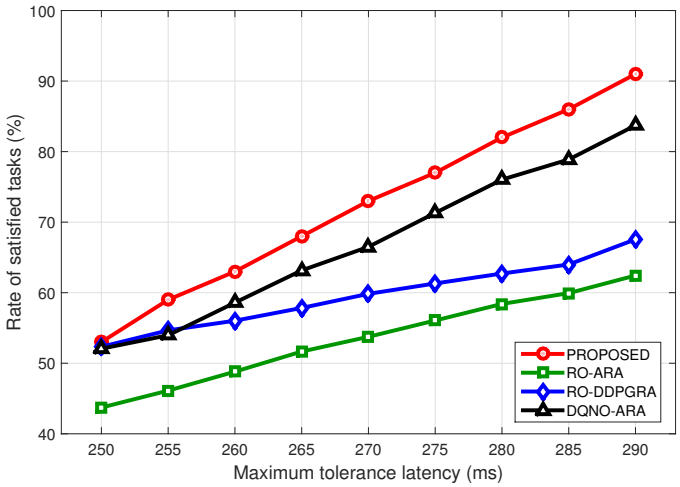

Figure 7. Rate of satisfied tasks under different maximum tolerance latency settings.

Figure 8 further compares the rate of satisfied tasks under different numbers of terminals. As the number increases, the rates for all methods show a declining trend. In particular, given more terminals, the RO-DDPGRA method surpasses the DQNO-ARA, suggesting that more existing terminals would make the resource allocation dominate computation offloading. More precisely, when the terminal number reaches 40, except for the RO-ARA, the rates of three other methods become close. That is, given the excessive terminal number, both computation offloading and resource allocation begin to take no effects on the performance. Nevertheless, given either more or smaller existing terminals, proposed method always outperforms benchmark ones.

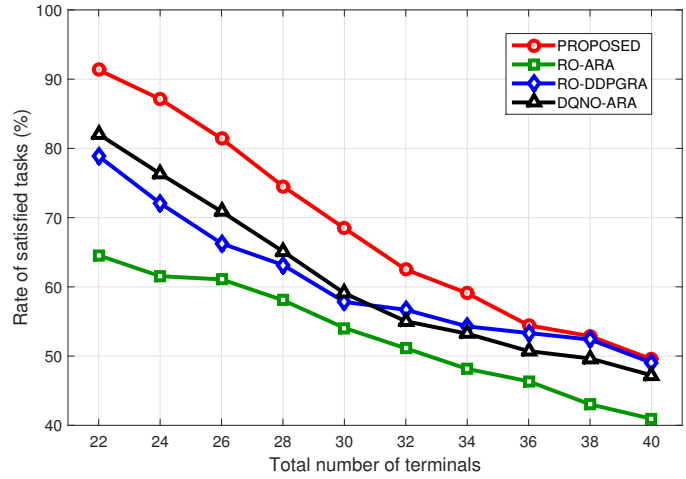

Figure 8. Rate of satisfied tasks under different terminal numbers.

Figures 9 and 10 illustrate the proportion of offloaded tasks under different approaches, and the rate of satisfied tasks in terminals, satellites and the cloud server, respectively. Both figures reveal that in RO-ARA and RO-DDPGRA methods, both employing random offloading, the number of offloaded tasks is equal to that of locally executed ones. In Figure 10, the rate of satisfied tasks in terminals is consistently below 40%. However, the RO-DDPGRA method shows around 10% higher rate for tasks offloaded to satellites, and 8% higher for tasks offloaded to the cloud server, both over the RO-ARA, owing to RO-DDPGRA. In particular, RO-DDPGRA adopts the DDPG, which can deterministically optimize resources to tasks offloaded to the same satellite (or cloud server), while RO-ARA only evenly distributes resources. In contrast, both PROPOSED and DQNO-ARA methods have significantly lower rates of tasks locally executed, and noticeably higher rates of tasks offloaded. Moreover, PROPOSED exhibits approximately 8% (and 6%) higher

satisfaction for tasks offloaded to satellites (and the cloud server), respectively, than DQNO-ARA, due to the difference in the resource allocation. Likewise, PROPOSED can allocate resources in line with the parameterized continuous action, while DQNO-ARA can only distribute resources equally. Note that RO-DDPGRA achieves higher satisfaction for tasks offloaded than the DQNO-ARA; yet, due to the randomness in the offloading decisions, the overall rate of satisfied tasks of the RO-DDPGA is constrained and falls below that of the DQNO-ARA method.

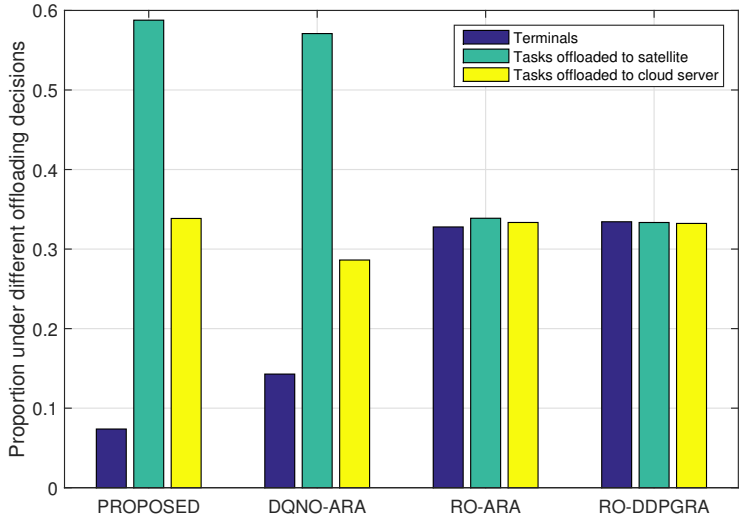

Figure 9. Proportion of offloaded tasks under different approaches.

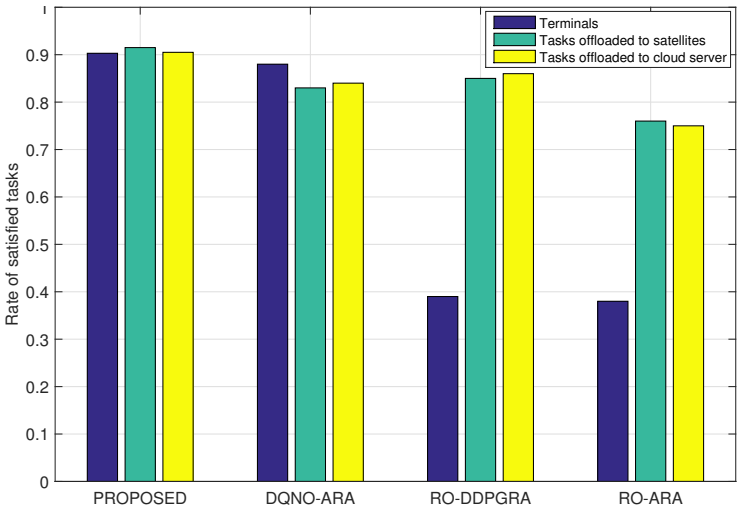

Figure 10. Rate of satisfied tasks in terminals, satellites, and cloud server.

6. Conclusions and Future Work

This work has investigated the application of P-DQN RL method to the joint computation offloading and resource allocation problem in LEO satellite edge networks. Unlike the discrete action space-based method (e.g., DQN) and the continuous action space-based one (e.g., DDPG), the P-DQN method takes effects in the mixed discrete-continuous space, without approximating the mixed space into the discrete one or relaxing it into the continuous one. Thus, this work considered the time-varying channel characteristics, and formulated the power control, computation offloading, and resource allocation to maximize the rate of satisfied tasks over the long run. To solve it, the PAMDP model was used to capture the dynamics in LEO satellite edge networks, using the parameterized continuous action. Finally,

the effectiveness of proposed method was verified through simulations, showing that it not only has a faster convergence rate, but also outperforms existing methods in terms of the rate of satisfied tasks. In future work, we will consider issues such as unstable connections between terminals and satellites (i.e., satellite handovers), and long-run optimization of EE.

Author Contributions: Conceptualization, X.Y. and H.F.; methodology, X.Y. and Y.G.; software, X.W.; validation, X.Y., H.F. and K.W.; formal analysis, Y.G.; investigation, K.W.; resources, Z.L.; data curation, X.Y. and H.F.; writing—original draft preparation, X.W.; writing—review and editing, X.Y. and H.F.; visualization, Z.L.; supervision, K.W.; project administration, Y.G.; funding acquisition, X.Y. and H.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was in part supported by the National Key Research and Development Program of China under Grant 2020YFB1808003, in part by the National Natural Science Foundation of China under Grant 61801379, and in part by the Natural Science Foundation of Shaanxi Province of China under Grant 2020JQ-647.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Acknowledgments: The authors would like to thank the editor and all reviewers for their valuable comments and efforts on this article.

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

References

1. Qu, Z.; Zhang, G.; Cao, H.; Xie, J. LEO satellite constellation for Internet of Things. *IEEE Access* **2017**, *5*, 18391–18401. [CrossRef]
2. Chien, W.; Lai, C.; Hossain, M.; Muhammad, G. Heterogeneous space and terrestrial integrated networks for IoT: Architecture and challenges. *IEEE Netw.* **2019**, *33*, 15–21. [CrossRef]
3. 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]
4. Chen, S.; Liang, Y.C.; Sun, S.; Kang, S.; Cheng, W.; Peng, M. Vision, requirements, and technology trend of 6G: How to tackle the challenges of system coverage, capacity, user data-rate and movement speed. *IEEE Wirel. Commun.* **2020**, *27*, 218–228. [CrossRef]
5. Wang, G.; Zhou, S.; Zhang, S.; Niu, Z.; Shen, X. SFC-based service provisioning for reconfigurable space-air-ground integrated networks. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1478–1489. [CrossRef]
6. Fu, S.; Gao, J.; Zhao, L. Integrated resource management for terrestrial-satellite systems. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3256–3266. [CrossRef]
7. Zhu, X.; Jiang, C.; Kuang, L.; Ge, N.; Guo, S.; Lu, J. Cooperative transmission in integrated terrestrial-satellite networks. *IEEE Netw.* **2019**, *33*, 204–210. [CrossRef]
8. Kapovits, A.; Corici, M.I.; Gheorghe-Pop, I.D.; Gavras, A.; Burkhardt, F.; Schlichter, T.; Covaci, S. Satellite communications integration with terrestrial networks. *China Commun.* **2018**, *15*, 22–38. [CrossRef]
9. Yang, Y.; Ma, M.; Wu, H.; Yu, Q.; Zhang, P.; You, X.; Wu, J.; Peng, C.; Yum, T.-S.P.; Shen, S.; et al. 6G network AI architecture for everyone-centric customized services. *IEEE Netw.* **2022**, 1–10. [CrossRef]
10. Zhang, S.; Liu, J.; Guo, H.; Qi, M.; Kato, N. Envisioning device-to-device communications in 6G. *IEEE Netw.* **2020**, *34*, 86–91. [CrossRef]
11. Lakew, D.S.; Tran, A.T.; Masood, A.; Dao, N.N.; Cho, S. A Review on Satellite-Terrestrial Integrated Wireless Networks: Challenges and Open Research Issues. In Proceedings of the International Conference on Information Networking (ICOIN), Bangkok, Thailand, 11–14 January 2023; pp. 638–641.
12. Sun, Y.; Peng, M.; Zhang, S.; Lin, G.; Zhang, P. Integrated satellite-terrestrial networks: Architectures, key techniques, and experimental progress. *IEEE Netw.* **2022**, *36*, 191–198. [CrossRef]
13. Kumar, K.; Liu, J.; Lu, Y.H.; Bhargava, B. A survey of computation offloading for mobile systems. *Mob. Netw. Appl.* **2013**, *18*, 974–983. [CrossRef]
14. Chen, X. Decentralized computation offloading game for mobile cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *26*, 129–140. [CrossRef]
15. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
16. Du, J.; Zhao, L.; Feng, J.; Chu, X. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Trans. Commun.* **2017**, *66*, 1594–1608. [CrossRef]

17. Wang, C.; Yu, F.R.; Liang, C.; Chen, Q.; Tang, L. Joint computation offloading and interference management in wireless cellular networks with mobile edge computing. *IEEE Trans. Veh. Technol.* **2017**, *66*, 7432–7445. [CrossRef]
18. Chang, Z.; Zhou, Z.; Ristaniemi, T.; Niu, Z. Energy efficient optimization for computation offloading in fog computing system. In Proceedings of the IEEE Global Communications Conference (IEEE GLOBECOM), Singapore, 4–8 December 2017; pp. 1–6.
19. Zhu, X.; Jiang, C. Integrated satellite-terrestrial networks toward 6G: Architectures, applications, and challenges. *IEEE Internet Things J.* **2021**, *9*, 437–461. [CrossRef]
20. Brik, B.; Frangoudis, P.A.; Ksentini, A. Service-oriented MEC applications placement in a federated edge cloud architecture. In Proceedings of the IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
21. Mehrabi, M.; You, D.; Latzko, V.; Salah, H.; Reisslein, M.; Fitzek, F.H.P. Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey. *IEEE Access* **2019**, *7*, 166079–166108. [CrossRef]
22. Li, J.; Wang, R.; Wang, K. Service Function Chaining in Industrial Internet of Things With Edge Intelligence: A Natural Actor-Critic Approach. *IEEE Trans. Industr. Inform.* **2023**, *19*, 491–502. [CrossRef]
23. Hao, Y.; Chen, M.; Hu, L.; Hossain, M.S.; Ghoneim, A. Energy efficient task caching and offloading for mobile edge computing. *IEEE Access* **2018**, *6*, 11365–11373. [CrossRef]
24. Trakadas, P.; Masip-Bruin, X.; Facca, F.M.; Spantideas, S.T.; Giannopoulos, A.E.; Kapsalis, N.C.; Martins, R.; Bosani, E.; Ramon, J.; Prats, R.G.; et al. A reference architecture for cloud-edge meta-operating systems enabling cross-domain, data-intensive, ML-assisted applications: Architectural overview and key concepts. *Sensors* **2022**, *22*, 9003. [CrossRef] [PubMed]
25. Militano, L.; Arteaga, A.; Toffetti, G.; Mitton, N. The cloud-to-edge-to-IoT continuum as an enabler for search and rescue operations. *Future Internet* **2023**, *15*, 55. [CrossRef]
26. Yan, J.; Bi, S.; Zhang, Y.; Tao, M. Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 235–250. [CrossRef]
27. Zhang, G.; Zhang, S.; Zhang, W.; Shen, Z.; Wang, L. Joint service caching, computation offloading and resource allocation in mobile edge computing systems. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 5288–5300. [CrossRef]
28. Wu, F.; Leng, S.; Maharjan, S.; Huang, X.; Zhang, Y. Joint Power Control and Computation Offloading for Energy-Efficient Mobile Edge Networks. *IEEE Trans. Wirel. Commun.* **2021**, *21*, 4522–4534. [CrossRef]
29. Tan, L.; Kuang, Z.; Zhao, L.; Liu, A. Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing. *IEEE Trans. Wirel. Commun.* **2021**, *21*, 1960–1972. [CrossRef]
30. Ruan, Y.; Li, Y.; Wang, C.X.; Zhang, R.; Zhang, H. Energy efficient power allocation for delay constrained cognitive satellite terrestrial networks under interference constraints. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 4957–4969. [CrossRef]
31. Shi, S.; Li, G.; An, K.; Gao, B.; Zheng, G. Energy-efficient optimal power allocation in integrated wireless sensor and cognitive satellite terrestrial networks. *Sensors* **2017**, *17*, 2025. [CrossRef]
32. Spantideas, S.T.; Giannopoulos, A.E.; Kapsalis, N.C.; Kalafatelis, A.; Capsalis, C.N.; Trakadas, P. Joint energy-efficient and throughput-sufficient transmissions in 5G cells with deep Q-learning. In Proceedings of the IEEE International Mediterranean Conference on Communications and Networking (MeditCom), Athens, Greece, 7–10 September 2021; pp. 265–270.
33. Hsieh, C.K.; Chan, K.L.; Chien, F.T. Energy-efficient power allocation and user association in heterogeneous networks with deep reinforcement learning. *Appl. Sci.* **2021**, *11*, 4135. [CrossRef]
34. Liu, C.H.; Chen, Z.; Tang, J.; Xu, J.; Piao, C. Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2059–2070. [CrossRef]
35. Li, B.; Liu, Y.; Tan, L.; Zhang, Y. Digital twin assisted task offloading for aerial edge computing and networks. *IEEE Trans. Veh. Technol.* **2022**, *71*, 10863–10877. [CrossRef]
36. Liu, W.; Li, B.; Xie, W.; Fei, Z. Energy efficient computation offloading in aerial edge networks with multi-agent cooperation. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 5725–5739. [CrossRef]
37. Qiu, C.; Yao, H.; Yu, F.R.; Xu, F.; Zhao, C. Deep Q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5871–5883. [CrossRef]
38. 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]
39. Cheng, N.; Lyu, F.; Quan, W.; Zhou, C.; He, H.; Shi, W.; Shen, X. Space/aerial-assisted computing offloading for IoT applications: A learning-based approach. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1117–1129. [CrossRef]
40. Cui, G.; Li, X.; Xu, L.; Wang, W. Latency and energy optimization for MEC enhanced SAT-IoT networks. *IEEE Access* **2020**, *8*, 55915–55926. [CrossRef]
41. Wang, B.; Xie, J.; Huang, D.; Xie, X. A Computation Offloading Strategy for LEO Satellite Mobile Edge Computing System. In Proceedings of the International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 10–12 June 2022; pp. 75–80.
42. Lyu, Y.; Liu, Z.; Fan, R.; Zhan, C.; Hu, H.; An, J. Optimal Computation Offloading in Collaborative LEO-IoT Enabled MEC: A Multiagent Deep Reinforcement Learning Approach. *IEEE Trans. Green Commun. Netw.* **2023**, *7*, 996–1011. [CrossRef]
43. Maattanen, H.L.; Hofstrom, B.; Euler, S.; Sedin, J.; Lin, X.; Liberg, O.; Masini, G.; Israelsson, M. 5G NR Communication over GEO or LEO Satellite Systems: 3GPP RAN Higher Layer Standardization Aspects. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.

44. Zhou, D.; Sheng, M.; Wang, Y.; Li, J.; Han, Z. Machine learning-based resource allocation in satellite networks supporting internet of remote things. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 6606–6621. [CrossRef]
45. Yuan, Y.; Lei, L.; Vu, T.; Chang, Z.; Chatzinotas, S.; Sun, S. Adapting to dynamic LEO-B5G systems: Meta-critic learning based efficient resource scheduling. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 9582–9595. [CrossRef]
46. Dong, Y.; Wang, L.; Wang, J.; Hu, X.; Zhang, H.; Yu, F.R.; Leung, V.C.M. Accelerating Wireless Federated Learning via Nesterov's Momentum and Distributed Principle Component Analysis. *IEEE Trans. Wirel. Commun.* **2023**. [CrossRef]
47. Dong, Y.; Zhang, H.; Li, J.; Yu, F.R.; Guo, S.; Leung, V.C. An online zero-forcing Precoder for weighted sum-rate maximization in green CoMP systems. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 7566–7581. [CrossRef]
48. Xiong, J.; Wang, Q.; Yang, Z.; Sun, P.; Han, L.; Zheng, Y.; Fu, H.; Zhang, T.; Liu, J.; Liu, H. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv* **2018**, arXiv:1810.06394. [CrossRef]
49. Masson, W.; Ranchod, P.; Konidaris, G. Reinforcement learning with parameterized actions. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), Phoenix, AZ, USA, 12–17 February 2016; pp. 1934–1940.
50. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
51. Wang, Y.; Zhang, J.; Zhang, X.; Wang, P.; Liu, L. A computation offloading strategy in satellite terrestrial networks with double edge computing. In Proceedings of the IEEE International Conference on Communication Systems (ICCS), Chengdu, China, 19–21 December 2018; pp. 450–455.
52. Jiang, W.; Feng, D.; Sun, Y.; Feng, G.; Wang, Z.; Xia, X.G. Joint computation offloading and resource allocation for D2D-Assisted mobile edge computing. *IEEE Trans. Serv. Comput.* **2023**, *16*, 1949–1963. [CrossRef]