Satellite Edge Computing With Collaborative Computation Offloading: An Intelligent Deep Deterministic Policy Gradient Approach

Hangyu Zhang, Rongke Liu, *Senior Member, IEEE*, Aryan Kaushik, *Member, IEEE*, and Xiangqiang Gao

***Abstract*—Enabling a satellite network with edge computing capabilities can complement the advantages further of a single terrestrial network and provide users with a full range of computing service. Satellite edge computing is a potentially indispensable technology for future satellite-terrestrial integrated networks. In this article, a three-tier edge computing architecture consisting of the terminal–satellite–cloud is proposed, where tasks can be processed at three planes and intersatellites can cooperate to achieve on-board load balancing. Facing varying and random task queues with different service requirements, we formulate the objective problem of minimizing the system energy consumption under the delay and resource constraints, and jointly optimize the offloading decision, communication, and computing resource allocation variables. Moreover, the distribution of resources is based on the reservation mechanism to ensure the stability of the satellite-terrestrial link and the reliability of computation process. To adapt to the dynamic environment, we propose an intelligent computation offloading scheme based on the deep deterministic policy gradient (DDPG) algorithm, which consists of several different deep neural networks (DNNs) to output both discrete and continuous variables. Additionally, by setting the selection process of legal actions, the simultaneous decisions on offloading locations and allocating resources under multitask concurrency is realized. The simulation results show that the proposed scheme can effectively reduce the total energy consumption of the system by ensuring that the task is completed on demand, and outperform the benchmark algorithms.**

***Index Terms*—Computation offloading, deep deterministic policy gradient (DDPG), intersatellite cooperative computing, resource allocation, satellite edge computing.**

Manuscript received 27 May 2022; revised 9 December 2022; accepted 23 December 2022. Date of publication 30 January 2023; date of current version 9 May 2023. This work was supported in part by the Beijing Natural Science Foundation under Grant L202003, and in part by the Civil Aerospace Technology Advance Research Project of the 13th Five-Year Plan under Grant D030301. *(Corresponding author: Rongke Liu.)*

Hangyu Zhang is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: zhanghangyu@buaa.edu.cn).

Rongke Liu is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China, and also with the Shenzhen Institute of Beihang University, Shenzhen 518038, China (e-mail: rongke_liu@buaa.edu.cn).

Aryan Kaushik is with the School of Engineering and Informatics, University of Sussex, BN1 9RH Brighton, U.K. (e-mail: aryan.kaushik@sussex.ac.uk).

Xiangqiang Gao is with the China Academy of Space Technology, Xi'an 710100, China, and also with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: xggao@buaa.edu.cn).

Digital Object Identifier 10.1109/JIOT.2022.3233383

I. Introduction

In RECENT years, the fifth-generation (5G) mobile communication technology has been deployed in different scenarios, such as vehicle-to-everything (V2X) [1] and industrial Internet [2]. Compared to the previous generations of communication standards, 5G has the characteristics of employing large bandwidth, lower latency, and wide connection, and providing enhanced mobile broadband service for users [3]. Some advanced wireless communication technologies have been addressed for 5G systems, such as optimizing energy efficiency gains and low-complexity solutions [4], [5], [6]. However, due to the limitations of existing economic and technological development, users located in disaster areas, marine, or other remote areas are still unable to establish communication directly with the terrestrial network [7].

In response to the above shortcomings, satellite communication network (SCN) has emerged as the potential core technology. It has full coverage capability and strong invulnerability while providing reliable and stable communication services regardless of the geographical conditions [8]. The satellite network can supplement the deficiencies of the terrestrial network in different scenarios. Therefore, the integrated satellite-terrestrial network (STN) has become one of the core research directions of the future sixth-generation (6G) network architectures [9]. In the current STN serving Internet of Things (IoT), the terminals forward data via satellites to the terrestrial cloud data center for processing. Compared with satellite nodes, cloud has higher computing capacity and sufficient supply of energy [10]. However, the cloud platform is usually far from the user terminals, which leads to higher communication latency. Meanwhile, with the rapid development of various computation-intensive and latency-sensitive applications, such as smartphones, wearable devices, and virtual reality, many more applications raise higher demand for security, real-time, and intelligence [11].

In the terrestrial communication network, mobile-edge computing (MEC) provides a new paradigm for IoT applications to reduce the pressure on the network caused by the long distance and excessive data between the cloud and terminals [12]. The main idea of MEC is to deploy computing and storage resources at the edge of the network in order to obtain better service performance and Quality-of-Service (QoS), such as lower latency, less bandwidth consumption, and better confidentiality. Borrowing from the mind of MEC

in the terrestrial network, edge computing has been introduced into the satellite network [13], [14], [15]. Satellites deploying MEC servers need to be as close to the user as possible, so they are usually located in low-Earth orbit (LEO) and the space is regarded as an edge. Instead of communicating with terrestrial base stations, users obtain reliable computing service directly from satellites. The satellites implement on-board processing tasks rather than acting only as a relay transponder, which will be an indispensable paradigm for the future integrated STN [16]. However, compared to the terrestrial MEC, satellites are restricted by dynamic change in orbit and scarcity of on-board resources. It is urgent to improve the ability of the satellite MEC network for fast data computing and efficient utilization of resources.

Based on this, existing research has implemented intersatellite link (ISL) through the optical or visible light communication system to achieve fast, reliable, and efficient satellite network [17]. Intersatellite cooperation using ISL makes full use of the available resources in space. On the one hand, it increases system capacity and coverage area and further meets the needs of users. On the other hand, it reduces the resource consumption of a single satellite and improves the overall viability of the satellite network. Therefore, the intersatellite cooperative technology has now been widely discussed and has broad application prospects [18], [19], [20]. However, considering the dynamic and random task queues in a realistic scenario, it leads to a huge state and action space of the satellite MEC system enabled by cooperative computing. The decision on offloading locations and resource allocation of nodes are very difficult at this time. To the best of our knowledge, there is little research focusing on the above-mentioned issues.

In the existing MEC systems, most of the previous works adopt convex approximation [21], [22], the game-based methods [6], [13], or the meta-heuristic algorithms [14], [23] to solve the computation offloading problem. These traditional approaches can solve long-term optimization problems, but cannot obtain prior information in the dynamic MEC environment and may only obtain approximate optimal solutions for complex nonconvex problems. Therefore, many researchers have turned to apply deep reinforcement learning (DRL) methods to solve the corresponding problems [24], [25], [26]. DRL can not only provide solutions to large-scale complex problems by exploiting the function approximation properties of deep neural network (DNN) but also receive reward feedback from the environment to dynamically adjust its policy in a model-free framework [27]. However, some value-based DRL algorithms need to discretize the continuous space, such as DQN [28], which limits the optimization accuracy of computation offloading problems. On the other hand, some policy-based DRL algorithms, such as actor–critic (AC) [29], need to choose actions according to a specific distribution when applied to continuous action spaces and are not well suited for hybrid action decisions based on discrete and continuous action spaces.

To complement the relevant research works, we consider IoT users in remote or disaster areas without cellular coverage, who can be served by the LEO satellite MEC network. However, due to the tidal problem of computations with large fluctuations in the spatial and temporal distribution of actual service demand, as well as the limitations of the satellite with dynamic and constrained resources, it leads to resource shortages and service timeouts. At this time, through a comprehensive analysis of delay, energy consumption, and load among computing nodes, overload requests of the access satellites can be offloaded to other satellite nodes or cloud center to improve the overall performance of the network. In such a computation offloading process, the joint optimization of discrete offloading decisions and continuous resource allocation is involved.

Further, facing the actual environment where application requirements are time-varying and tasks arrive randomly, we propose a deep deterministic policy gradient (DDPG)-based computation offloading scheme. DDPG is an improved version that combines DQN and AC, which does not need to discretize the action space and directly uses DNN to estimate the best action. On the one hand, the DDPG algorithm can avoid dimensional explosion and adapt to the complex state space. On the other hand, the proposed intelligent DDPG algorithm can solve the legal action selection problem of discrete and continuous variables. Following that, our main contributions are summarized as follows.

1) We propose a three-tier satellite edge computing architecture and jointly optimize offloading decision and resource allocation. Intersatellite computing resources can be shared through ISL. Concurrent multiuser tasks are decided synchronously on demand whether to be processed locally, at the satellite edge or cloud immediately or to wait for the next scheduling.
2) This article investigates the objective optimization problem of minimizing the system energy consumption while satisfying the delay and resource constraint to ensure that the task requirements are met. Moreover, resource allocation is accomplished based on a reservation mechanism to reduce the communication and computation overhead.
3) This study adopts a computation offloading strategy based on the DDPG algorithm in order to adapt to time-varying service requirements while addressing the problem of simultaneous learning of discrete variables (offloading decision) and continuous variables (resource allocation). Extensive simulation results show that the proposed scheme has better convergence performance and achieves better results than the benchmark algorithms.

The remainder of this article is organized as follows. Section II introduces the related works. Section III presents the system model and problem formulation. In Section IV, the computation offloading problem is described and constructed as a Markov decision process (MDP). Then, a DDPG-based solution is proposed. Simulation results of the proposed scheme are shown and discussed in Section V. Finally, this article is summarized in Section VI.

II. Related Works

A. Computation Offloading in Satellite Edge Computing

Satellite edge computing is an emerging research field and how to make the optimal strategy in a hierarchical computation offloading architecture is one of the main challenges. Zhang et al. [13] investigated the joint computing and communication resource allocation problem for SCN to minimize the execution latency of the computation-intensive applications. A game-theoretic and the many-to-one matching theory-based scheme was proposed to obtain the optimal solution. Song et al. [14] proposed a novel MEC framework to explore the task processing capability of satellites for IoT mobile devices, and the optimization problem was decomposed into two-layered subproblems corresponding to the space and ground segments. Moreover, an energy-efficient computation offloading and resource allocation algorithm was proposed. In [21], a hybrid cloud and edge computing LEO satellite network was presented, which can provide users with heterogeneous computing resources. A distributed computation offloading scheme based on the alternating direction method of multipliers (ADMMs) was proposed for minimizing the energy consumption of users.

It can be known from the literatures listed above that due to the complexity of the computation offloading problem in the satellite MEC environment, most studies do not consider intersatellite cooperative computing and usually use the traditional optimization methods to obtain suboptimal solutions, which reduces the utilization of on-board resources and accuracy of edge computing strategies. Therefore, our proposed network model is aimed at the ISL-enabled satellite MEC system and applies intelligent algorithms to obtain optimal policies, and contributes further to existing research literature.

B. Enabling Intersatellite Collaboration for Satellite Edge

In order to increase the system capacity and extend the network life, a few recent works have investigated satellite edge computing architectures that consider intersatellite cooperation. Li et al. [22] considered the deployment of MEC servers with computing and storage resources in LEO network with ISL and modeled the joint request scheduling and service placement problem as a mixed integer linear programming optimization problem. The problem was solved by the open-source OPTI toolbox to obtain optimal decisions. In [30], a satellite MEC integration architecture for high-speed STN was proposed while using dynamic network function virtualization (NFV) techniques to integrate computing resources. The authors also presented a collaborative computation offloading task scheduling model to improve QoS of the mobile users. Wang et al. [31] proposed a novel STN architecture with double edge computing, and MEC servers were deployed on the terrestrial base station as well as on the satellite network with ISL, which solved the problem of limited computing resources of edge servers in remote areas. A minimum cost matching algorithm was used to optimize the system energy consumption and average latency on the edge servers.

For these existing works, the main research issue is the selection of offloading locations. And heterogeneous and complex task queues are not considered, which cannot solve the prioritized computing and resource allocation problem of task scheduling process in a dynamic environment. At the same time, they usually only get approximate optimal solutions when faced with nonconvex multiobjective programming. Our proposed scheme solves the problem of when and where tasks are computed on demand and can adapt to dynamic and time-varying service requirements to make optimal computation offloading scheme.

C. DRL-Based Satellite Edge Computing

In recent years, the field of machine learning has developed rapidly, the decision-making ability of reinforcement learning (RL) and the perception ability of deep learning (DL) are combined to form DRL. However, considering the complexity of satellite scenarios, there are only a few studies on the computation offloading problem in satellite MEC. In [28], a software-defined STN framework was proposed to jointly manage and orchestrate networking, caching, and computing resources. The authors described the joint resource allocation problem as a joint optimization problem and used a DQN-based approach to solve it for performance improvement. Zhu et al. [32] studied the joint offloading decision and bandwidth allocation problem in satellite MEC networks, where tasks can be executed by satellite or urban terrestrial cloud. The proposed DQN-based task offloading algorithm could accelerate the learning process and minimize the offloading cost based on current observed channel states. Cui et al. [33] presented a dynamic resource management framework to jointly optimize offloading decision, computing, and communication resource for the satellite-assisted vehicle-to-vehicle (V2V) networks. Meanwhile, an offloading decision method based on DQN was proposed, and the Lagrangian multiplier method was used to realize resource allocation.

In most of the above studies, the DQN algorithm that discretizes continuous actions is usually used to solve the continuous resource allocation problem, which cannot achieve the tradeoff between high-precision decision-making and a low-dimensional neural network. For this reason, some researchers decompose the offloading decision and resource optimization into two subproblems and solve them sequentially, which also increases the complexity of the solution process. Based on this, we propose an intelligent DDPG-based approach to directly optimize discrete and continuous variables simultaneously.

III. System Models and Problem Formulation

In this section, we first describe the network model of MEC-enabled STN. Then, we, respectively, describe the end-to-end delay and system energy consumption of computing tasks in three modes: 1) computing locally; 2) offloading to satellite network; and 3) transferring to cloud through access satellite. Finally, the optimization problem of minimizing the energy consumption with delay and resource constraints is

TABLE I
NOTATIONS

Notation	Definition
U	Number of users
S	Number of satellites
Q	Maximum length of the task queues
λ	Task arrival rate
$\tau_{u,k}^{p}$	Task generation time slot
$d_{u,k}^{p}$	Size of computation data of the tasks
$c_{u,k}^{p}$	Task workload
$T_{u,k}^{max,p}$	Maximum tolerable delay of the tasks
ρ	System time slot length
f_u	The computing capacity of users
κ	The effective capacitance coefficient
$x_{u,k}^{p}$	The offloading decision of the tasks
$f_{s',k}^{p}$	The computing resource allocated by satellite
$r_{s,k}^{p}$	The communication resource allocated by satellite
c	The speed of light
R_s	The communication capability of ISL
R_{sc}	The communication capability of satellite-cloud link
p_u	The transmission power of users
p_s	The transmission power between satellites
p_c	The transmission power between satellite and cloud
d_{us}	The distance between user and satellite
d_s	The distance between satellites
d_{sc}	The distance between satellite and cloud
f_c	The computing capacity of cloud
F	The total computing capacity of a satellite
R	The total communication capacity of a satellite

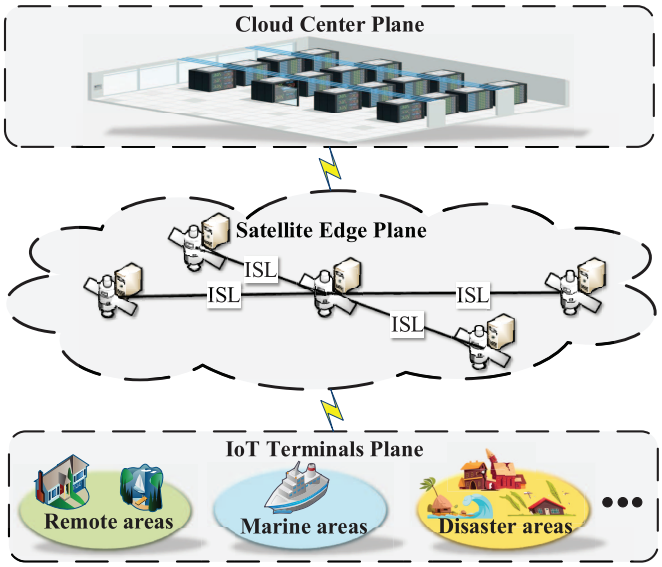

Fig. 1. Network model of the satellite MEC system.

proposed. Some key notations and related descriptions are listed in Table I.

A. Network Model

The three-tier edge computing architecture based on terminal–satellite–cloud is shown in Fig. 1. The satellite MEC system contains U users and S satellites, which can be represented as sets $\mathcal{U} = \{1, 2, \ldots, U\}$ and $\mathcal{E} = \{1, 2, \ldots, S\}$, respectively. Each satellite with a MEC server establishes ISLs with four surrounding satellites: two intraplane ISLs are between adjacent satellites on the same orbit, and two interplane ISLs are between adjacent satellites on two adjacent orbits [34]. Similar to the existing researches on satellite edge computing [6], [15], our model is based on a quasi-static scenario, that is, the network topology and associated users of satellites remain fixed during the computing service. In addition, the users are located in remote or disaster areas and are not served by terrestrial communication networks. We assume that the user $u(u \in \mathcal{U})$ accesses the satellite network only through a single satellite $s(s \in \mathcal{E})$, and each user has a task queue with a maximum length of Q that can cache tasks. The tasks can be computed locally, offloaded to the satellite nodes, or relayed to remote cloud through access satellite. Because of the existence of ISL, the tasks can be transmitted to other satellites for collaborative computing when the access satellite computing capacity is insufficient.

In the three-tier computing architecture, the computing capability of each plane is gradually increasing from the terminal to the cloud. In terms of task scheduling, lower complexity tasks can be performed by the local user. More complex tasks are suitable to be offloaded at the satellite edge. Tasks with high complexity and low real-time requirements should be placed at the remote cloud.

The whole satellite MEC system runs in a time slot mode. Specifically, the system time is divided into $T_{\max}$ time slices of equal length ρ. Tasks are decided whether to be scheduled or not, slot by slot. At the beginning of each slot, the tasks of each user arrive dynamically and are added to the end of the buffer queue. The expectation of the number of computing tasks generated by a user in each slot obeys the Poisson distribution of arrival rate λ. The kth task in the buffer queue of user u in time slot p is modeled as a four tuple $\mathcal{K}_{u,k}^{p} = \{\tau_{u,k}^{p}, d_{u,k}^{p}, c_{u,k}^{p}, T_{u,k}^{\max,p}\}$. This means that the task $\mathcal{K}_{u,k}^{p}$ containing $d_{u,k}^{p}$ bits generated in the time slot $\tau_{u,k}^{p}$ needs to be completed within time $T_{u,k}^{\max,p}$, and its workload is $c_{u,k}^{p}$ cycles/bit. The above values can be obtained through the program analyzer. Without loss of generality, the output data of a task is very small compared to its input data [35]. Therefore, we ignore the transmission delay in returning the results of the task calculation.

We assume that tasks of each user are indivisible, and will be queued according to their generation time. Then, the system can schedule several tasks at once in the queue as per the current environmental status. We use $x_{u,k}^{p}$ to represent the offloading decision of task $\mathcal{K}_{u,k}^{p}$, and $x_{u,k}^{p} \in \{0, 1, 2, \ldots, S+2\}$. The parameter $x_{u,k}^{p} = 0$ means that the task is not scheduled. While $x_{u,k}^{p} = 1, 2, \ldots, S$ means that the task is offloaded at the satellite $1, 2, \ldots, S$. $x_{u,k}^{p} = S+1, S+2$ means that the task is computed locally or processed at the cloud. If a task is scheduled, all tasks behind it are moved forward to fill its vacancies. If the task queue has reached the maximum length, the newly arrived tasks will have to be discarded, i.e., task overflow occurs. Since the tasks are independent and have different attributes, the offloading decision needs to simultaneously solve these two issues of when and where tasks are scheduled in order to meet the requirements of heterogeneous tasks.

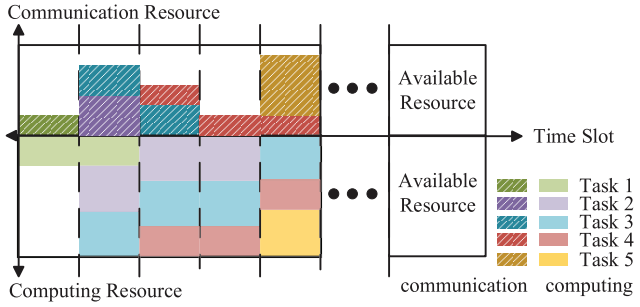

Fig. 2. Example of resource allocation.

Furthermore, we adopt a reservation-based resource allocation mechanism. Once a task is scheduled, its allocated orthogonal resources will be reserved and occupied for several time slots. Therefore, unserviced tasks have to wait for the next scheduling in order to be completed on demand in some time slots with few available resources. This mechanism is less efficient, but due to the large communication delay and the limited resources of the satellites, the mechanism is widely adopted in the satellite network as demand assigned multiple access (DAMA) in order to ensure the reliability of task completion and reduce the overhead of resources [36], [37]. Specifically, if task $\mathcal{K}_{u,k}^{p}$ is scheduled in time slot p, the estimated communication and computing time are t_1 and t_2, respectively. Then, the allocated communication resources will be occupied in (t_1/ρ) time slots and the allocated computing resources will be occupied in $(t_1 + t_2/\rho)$ time slots. That is, the communication resources are released after the transmission ends, but the computing resources remain occupied until the task computation is completed to ensure that tasks are processed or forwarded immediately when they arrive at the satellite without additional delay jitter [33]. For instance, in Fig. 2, task 1 arrives in the initial time slot and is completed through communication of one-time slot and computation of one-time slot. The resource allocation scheme for task 1 is that communication resources are occupied by one-time slot and computing resources are occupied by two-time slots.

B. Local Computing Model

For local computing (LC), user u must use its own computing resources to process tasks. The end-to-end delay includes waiting time and computing time in this mode. Assuming that the computing capacity of each user is the same and fixed as f_u cycles/s, the time of the task $\mathcal{K}_{u,k}^{p}$ that is decided to be executed locally is

$$T_{u,k,p}^{\text{local}} = \rho\left(p - \tau_{u,k}^{p}\right) + \frac{d_{u,k}^{p} c_{u,k}^{p}}{f_u} \tag{1}$$

where $\rho(p - \tau_{u,k}^{p})$ represents the waiting time of task $\mathcal{K}_{u,k}^{p}$.

To describe the energy consumption of performing tasks locally, we use the widely adopted model of energy consumption per computing cycle as $e = \kappa f^2$ [21], [35], where κ is the energy consumption coefficient depending on the effective switching capacitance of the chip architecture and f is the CPU frequency. So the energy consumption of the user u processing task $\mathcal{K}_{u,k}^{p}$ locally is

$$E_{u,k,p}^{\text{local}} = \kappa f_u^2 d_{u,k}^{p} c_{u,k}^{p}. \tag{2}$$

C. Satellite Edge Computing Model

For the mode of offloading tasks to the satellite, the end-to-end delay includes four parts: 1) waiting delay; 2) transmission delay; 3) propagation; and 4) computing delay. At the same time, any two satellites can communicate with each other through multihop routing in the satellite MEC network with ISL, without the relay of the terrestrial station. Therefore, the problem formulation will be classified into two forms due to the difference in offloading decision satellites. When the task is offloaded at the access satellite, there is no ISL transmission. When the task is offloaded at the nonaccess satellite, the transmission link includes multihop ISL. Let $r_{s,k}^{p}$ represents the communication capacity of the uplink between the user and the access satellite s, and $f_{s',k}^{p}$ represents the computing resources allocated by the decision satellite s'. When the resources of access satellite are sufficient, access satellite s is the offloading decision satellite s'. At this time, the end-to-end delay of offloading task $\mathcal{K}_{u,k}^{p}$ is

$$t_{u,k,p}^{\text{sat}} = \rho\left(p - \tau_{u,k}^{p}\right) + \frac{d_{u,k}^{p}}{r_{s,k}^{p}} + \frac{d_{u,k}^{p} c_{u,k}^{p}}{f_{s',k}^{p}} + \frac{2d_{us}}{c} \tag{3}$$

where d_{us} represents the distance between the user and the satellite, and c represents the speed of light with a value of 3×10^5 km/s.

When intersatellite cooperative computing tasks are required, the task transmission links involve x-hop ISL, and $s \neq s'$. Therefore, the delay of offloading tasks to the satellite edge network can be expressed as

$$T_{u,k,p}^{\text{sat}} = \begin{cases} t_{u,k,p}^{\text{sat}}, & s = s' (\text{no ISL}) \\ t_{u,k,p}^{\text{sat}} + x\frac{d_{u,k}^{p}}{R_s} + x\frac{2d_s}{c}, & s \neq s' (x\text{-hop ISL}) \end{cases} \tag{4}$$

where d_s represents the distance between the satellites, and R_s represents the communication capacity of ISL.

At the same time, the energy consumption of offloading tasks to the satellites includes two parts: 1) communication and 2) computing energy consumption. Let p_u is the transmission power of the user, and p_s is the transmission power between satellites. The energy consumption without ISL is

$$e_{u,k,p}^{\text{sat}} = p_u \frac{d_{u,k}^{p}}{r_{s,k}^{p}} + \kappa {f_{s',k}^{p}}^2 d_{u,k}^{p} c_{u,k}^{p}. \tag{5}$$

Therefore, the energy consumption of offloading tasks to the satellite edge network can be expressed as

$$E_{u,k,p}^{\text{sat}} = \begin{cases} e_{u,k,p}^{\text{sat}}, & s = s' (\text{no ISL}) \\ e_{u,k,p}^{\text{sat}} + x p_s \frac{d_{u,k}^{p}}{R_s}, & s \neq s' (x\text{-hop ISL}). \end{cases} \tag{6}$$

D. Cloud Computing Model

When the computing capacity of the satellites is insufficient to complete the task, the user can offload tasks to the terrestrial cloud through the relay of access satellite. Assuming that the

computing capacity of cloud is fixed as f_c cycles/s, and the end-to-end delay, including four parts is

$$T^{\text{cloud}}_{u,k,p} = \rho\left(p - \tau^p_{u,k}\right) + \frac{d^p_{u,k}}{r^p_{s,k}} + \frac{d^p_{u,k}}{R_{sc}} + \frac{d^p_{u,k} c^p_{u,k}}{f_c} + \frac{2(d_{us} + d_{sc})}{c} \tag{7}$$

where d_{sc} represents the distance between satellite and cloud, and R_{sc} represents the communication capacity of the satellite-cloud link.

Since the energy supply of cloud is adequate [38], we refer to the problem formulations in [21] and [39], which ignore the computing energy consumption at this time. Let p_c represents the transmission power between the satellite and the cloud. In cloud computing mode, the communication energy consumption of the two links is

$$E^{\text{cloud}}_{u,k,p} = p_u \frac{d^p_{u,k}}{r^p_{s,k}} + p_c \frac{d^p_{u,k}}{R_{sc}}. \tag{8}$$

E. Optimization Objective

In summary, according to different offloading decisions in the three-tier satellite MEC network, the end-to-end delay and system energy consumption of processing task $\mathcal{K}^p_{u,k}$ are expressed, respectively, as

$$T^p_{u,k} = \begin{cases} 0, & x^p_{u,k} = 0 \\ T^{\text{sat}}_{u,k,p}, & x^p_{u,k} = 1, 2, \ldots, S \\ T^{\text{local}}_{u,k,p}, & x^p_{u,k} = S+1 \\ T^{\text{cloud}}_{u,k,p}, & x^p_{u,k} = S+2 \end{cases} \tag{9}$$

$$E^p_{u,k} = \begin{cases} 0, & x^p_{u,k} = 0 \\ E^{\text{sat}}_{u,k,p}, & x^p_{u,k} = 1, 2, \ldots, S \\ E^{\text{local}}_{u,k,p}, & x^p_{u,k} = S+1 \\ E^{\text{cloud}}_{u,k,p}, & x^p_{u,k} = S+2. \end{cases} \tag{10}$$

Based on the above discussion, processing tasks locally consumes less energy, but may not complete tasks on time. Satellite edge collaborative computing is beneficial to reduce delay, but increases the energy consumption of satellites. Processing tasks at cloud center takes less computing time and energy consumption but increases the communication time. Therefore, the offloading decision and resource allocation of each time slot have a great influence on the task completion. In this article, we intend to minimize the system energy consumption during long-term task processing subject to delay and resource constraints. The objective optimization problem can be formulated as

$$\min_{x^p_{u,k}, f^p_{s',k}, r^p_{s,k}} \sum_{p=1}^{T_{\max}} \sum_{u=1}^{U} \sum_{k \in \Omega_p} E^p_{u,k} \tag{11}$$

$$\text{s.t.} \quad T^p_{u,k} \le T^{\max,p}_{u,k} \quad \forall u, k, p \tag{11a}$$

$$x^p_{u,k} \in \{0, 1, 2, \ldots, S+2\} \quad \forall u, k, p \tag{11b}$$

$$\sum_{k \in \Omega^p_{s'}} f^p_{s',k} \le F - \sum_{k' \in \psi^p_{s'}} f^p_{s',k'}, s' \in \mathcal{E} \quad \forall p \tag{11c}$$

$$\sum_{k \in \Omega^p_s} r^p_{s,k} \le R - \sum_{k' \in \phi^p_s} r^p_{s,k'}, s \in \mathcal{E} \quad \forall p \tag{11d}$$

$$0 \le f^p_{s',k} \le F, s' \in \mathcal{E} \quad \forall k, p \tag{11e}$$

$$0 \le r^p_{s,k} \le R, s \in \mathcal{E} \quad \forall k, p. \tag{11f}$$

In (11), Ω_p and Ω^p_s represent the total set of tasks scheduled and the tasks scheduled by satellite s in time slot p, respectively. ψ^p_s and ϕ^p_s represent the set of tasks occupying computing and communication resources of satellite s in time slot p, respectively. F and R are the total computing and communication resources of each satellite.

The constraints in (11) can be explained as follows: equation (11a) indicates that the end-to-end delay of the task cannot exceed the service demand time. Equation (11b) indicates that the offloading decision of task $\mathcal{K}^p_{u,k}$ is either not scheduled or processed at local, satellite network, or cloud center. Equation (11c) means that the total computing resources allocated to all scheduled tasks must not exceed the available computing capacity of the current server. Equation (11d) means that the total communication resources allocated to all scheduled tasks must not exceed the available communication capacity of the current satellite. Equations (11e) and (11f), respectively, mean that each satellite must allocate a non-negative computing and communication resource to the task associated with it in time slot p.

According to the previous section, $f^p_{s',k}$ and $r^p_{s,k}$ are continuous variables, $x^p_{u,k}$ is a discrete variable, and the objective function is nonlinear with respect to the variables. Therefore, the problem defined in (11) is a mixed-integer nonlinear programming (MINLP), and it is difficult to find the optimal solution using traditional optimization methods. Next, We propose a DRL-based method to solve the problem.

IV. Computation Offloading Scheme via Deep Reinforcement Learning

In order to find the optimal computation offloading strategy, many value-based DRL algorithms have recently been applied to edge computing research [25], [26], [28], [33]. The main challenge of the value-based method is how to handle the continuous action space, such as DQN. Currently, the usual method is to quantize the action. However, coarse discretization loses a large amount of behavioral information, and too fine discretization increases the dimensionality of the action space. On this basis, we propose a DDPG-based computation offloading scheme to optimize both discrete and continuous variables. In this section, we first model the problem as an MDP to further describe the computation offloading process. Next, an intelligent DDPG-based algorithm is presented to solve the optimization problem. Finally, we give the complexity analysis of the proposed method.

A. MDP

Since the offloading decision of the time slot p will be influenced by the offloading decision of the time slot $p-1$, the computation offloading process in the above dynamic scenario can be constructed as an MDP. Define MDP as

$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where the elements, respectively, represent the state space, action space, state transition matrix, reward function, and discount factor of the optimization problem. Each element in the MDP is described as follows.

1) State Space: In each time slot, the system observes the current state to obtain environmental information. We, respectively, use the task queue state of each user $Q_u(u \in \mathcal{U})$, available computing and communication resource state of the satellite network $F_s(s \in \mathcal{E})$ and $R_s(s \in \mathcal{E})$ to represent the environmental exploration information. Among them, the user queue Q_u represents the complete information of each task. If the queue is not full, it is supplemented with 0. Therefore, the system state $s_p \in \mathcal{S}$ in time slot p contains $U * Q * 4 + S * 2$ elements. Then, s_p is defined as follows:

$$s_p = \left\{Q_1^p, \ldots, Q_U^p, F_1^p, \ldots, F_S^p, R_1^p, \ldots, R_S^p\right\}. \tag{12}$$

2) Action Space: After obtaining s_p, corresponding discrete and continuous actions will be generated. The discrete offloading decision as: $x_u = \{x_{u,1}, x_{u,2}, \ldots, x_{u,Q}\}$, where $x_{u,k} \in \{0, 1, 2, \ldots, S+2\}(u \in \mathcal{U}, k \in \{1, 2, \ldots, Q\})$. $x_{u,k}$ represents when and where each task is scheduled, which, respectively, indicate that the task is decided not to be scheduled, offloaded at satellites, computed locally, and processed at cloud. Moreover, the system must also allocate continuous computing resource $f_u = \{f_{u,1}, f_{u,2}, \ldots, f_{u,Q}\}(u \in \mathcal{U})$ and communication resource $r_u = \{r_{u,1}, r_{u,2}, \ldots, r_{u,Q}\}(u \in \mathcal{U})$ for each user u. To unify the output range of resources and reduce the errors caused by different dimensions, we limit the value interval of $f_{u,k}$ and $r_{u,k}$ as [0, 1]. At this time, the meanings of $f_{u,k}^p$ and $r_{u,k}^p$, respectively, represent the percentage of resources allocated to task $\mathcal{K}_{u,k}^p$ occupying the currently available computing and communication resources. Therefore, the action $a_p \in \mathcal{A}$ in time slot p contains $U * Q * 3$ elements. Then, a_p is defined as follows:

$$a_p = \left\{x_1^p, \ldots, x_U^p, f_1^p, \ldots, f_U^p, r_1^p, \ldots, r_U^p\right\}. \tag{13}$$

To exclude unreasonable part in the action space, we assume that the resource allocation is determined by the offloading decision, i.e., the offloading decision has a higher decision priority than the resource allocation. Therefore, since the strategy outputs both discrete and continuous variables, it will generate illegal actions. The action selection process of task $\mathcal{K}_{u,k}^p$ is shown in Fig. 3. When the offloading decision $x_{u,k}^p$ is 0 or $S+1$, no satellite is required to allocate resources for the task at this time, and the actions $f_{u,k}^p$ and $r_{u,k}^p$ will default to 0. When $x_{u,k}^p$ is $S+2$, no satellite is required to allocate computing resources for the task at this time, and $f_{u,k}^p$ will be 0 by default. If $r_{u,k}^p$ is too small at this time, it will be regarded as an illegal action. When $x_{u,k}^p$ is 1 to S, the satellite needs to allocate resources $f_{u,k}^p$ and $r_{u,k}^p$ at the same time. If $f_{u,k}^p$ or $r_{u,k}^p$ is too small, it will be regarded as an illegal action.

3) State Transition Probability Matrix: The transition probability of the system from s_p to s_{p+1} can be expressed as $q(s_{p+1}|s_p, a_p)$. Because s_{p+1} is affected by a_p, and the state space $\mathcal{S}$ and action space $\mathcal{A}$ are huge, it is difficult to accurately model $q(s_{p+1}|s_p, a_p)$. Therefore, we adopt a model-free computation offloading method.

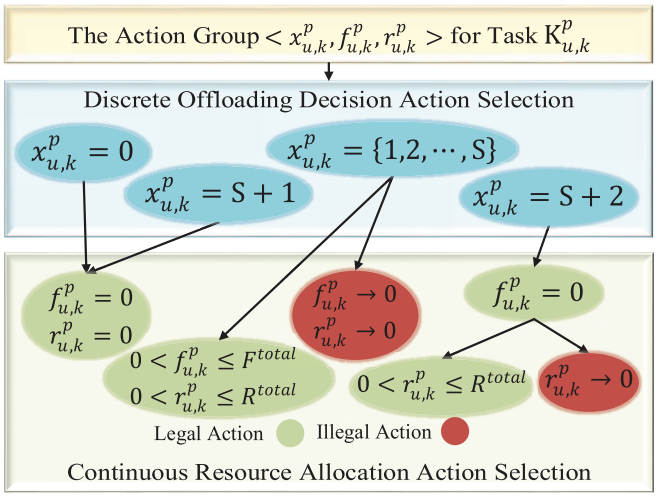

Fig. 3. Action selection process.

4) Reward Function: After performing action a_p under state s_p in time slot p, the environment enters a new state s_{p+1} and returns the corresponding reward r_p. The designed reward function should be based on the cost of the system state transition process, which is not only related to the objective function but also to the corresponding constraints. The specific formula of the reward function r_p is expressed as

$$r_p = \sum_u \sum_{k \in \Omega_p} \left(\Re - \omega E_{u,k}^p - \varrho_1 - \varrho_2 - \varrho_3\right) \tag{14}$$

where $\Re$ is a constant that makes the reward tend to be positive. $E_{u,k}^p$ is the optimization objective in (11), and ω is the scaling coefficient. ϱ denotes punishment and consists of three parts. The first part ϱ_1 is the punishment caused by task queue overflow, and the second part ϱ_2 is the punishment for task processing timeout. Both cases reduce the QoS of users, resulting in penalties. The third part ϱ_3 is the punishment when the strategy makes illegal actions, which encourages system to perform correct actions.

B. DDPG

The framework of the computation offloading process based on DDPG is shown in Fig. 4. Compared to DQN with the Q network, DDPG approximates the policy network and the Q-value network by using two separate frameworks, respectively, called as the actor-network and the critic network. Their functions are that the actor-network is responsible for executing decisions, and the critic network is responsible for guiding the correctness of the behavior. Both networks also contain two subnetworks named the online networks μ, Q and the corresponding target networks μ', Q', and their structure is the same. The parameters of the above four neural networks are θ^μ, θ^Q, $\theta^{\mu'}$, and $\theta^{Q'}$, respectively. The fixed target network updates parameters more slowly than the online network, which solves the problem of nonstability of the training process. The placement of the experience replay buffer can reduce the correlation between samples by random sampling at the same time. In addition, compared to DQN which takes actions

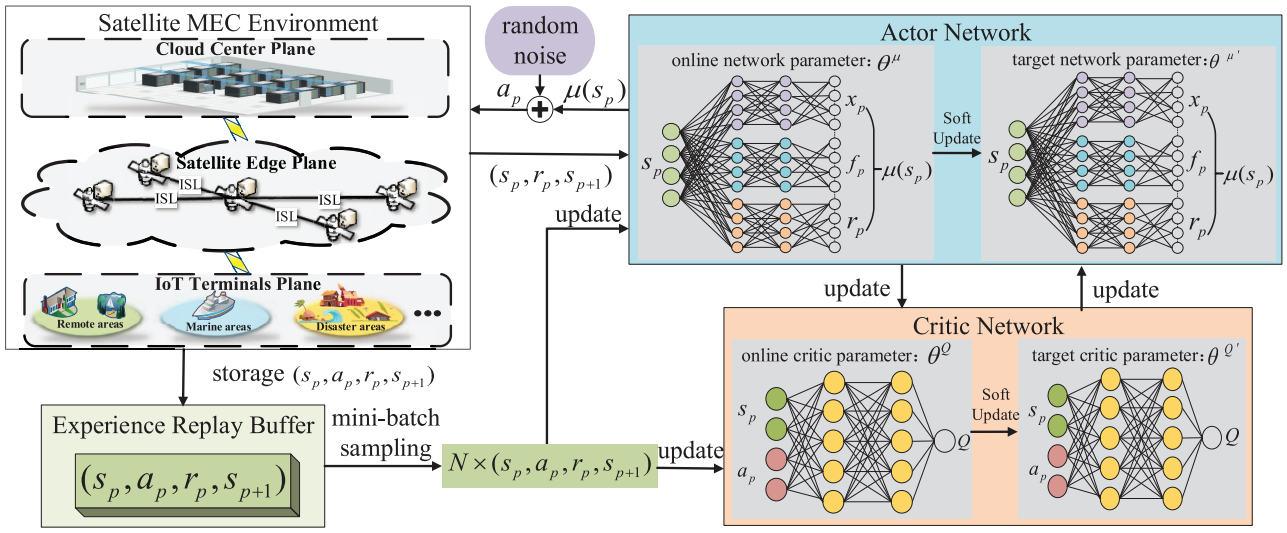

Fig. 4. DDPG-based computation offloading scheme for satellite MEC.

based on a probability, the advantage of DDPG as a deterministic algorithm is that its policy network can directly output unique actions under the same state, which reduces the number of required samples and greatly improves the efficiency of the algorithm. In our proposed scheme, the agent can directly output the unique multidimensional actions $<x_p, f_p, r_p>$ trained by several different DNNs without frequent sampling.

Specifically, the critic network is similar to the DQN network. When the action policy μ is fixed, it outputs $Q^{\mu}(s_p, a_p)$ in time slot p updated by the Bellman equation according to the current observed state s_p and the specific action a_p

$$Q^{\mu}(s_p, a_p) = E_{r_p, s_{p+1} \sim E}\left[r(s_p, a_p) + \gamma Q^{\mu}(s_{p+1}, \mu(s_{p+1}))\right] \tag{15}$$

where $r(s_p, a_p)$ represents the reward for performing action a_p at state s_p. γ represents the discount factor in the Bellman equation and indicates the impact of future awards on current awards, without loss of generality $\gamma \to 1$. E is the expected distribution corresponding to s_{p+1} and r_p.

Therefore, the loss function L of the critic network is

$$L\left(\theta^{Q}\right) = E\left[\left(Q\left(s_p, a_p | \theta^{Q}\right) - y_p\right)^2\right] \tag{16}$$

where y_p can be described as

$$y_p = r(s_p, a_p) + \gamma Q'\left(s_{p+1}, \mu'\left(s_{p+1} | \theta^{\mu'}\right) | \theta^{Q'}\right). \tag{17}$$

The actor-network outputs the specific action a through the deterministic policy μ with the current state s, and maximizes the Q value of the critic network. Assuming that J is the reward function obtained by the network, we define the policy gradient ∇J with the following formula proved in [40]:

$$\nabla_{\theta^{\mu}} J \approx E_{s_p \sim \rho^{\mu}}\left[\nabla_a Q\left(s, a | \theta^{Q}\right)|_{s=s_p, a=\mu(s_p)} \times \nabla_{\theta^{\mu}} \mu\left(s | \theta^{\mu}\right)|_{s=s_p}\right] \tag{18}$$

where ρ^{μ} represents the distribution of state s_p under the current policy μ. It updates the parameters Q^{μ} by applying the chain rule to the Bellman equation.

Finally, DDPG uses a small constant τ to achieve the soft update of the target network parameters

$$\begin{aligned} \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \\ \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}. \end{aligned} \tag{19}$$

The flow of computation offloading based on DDPG is shown in Algorithm 1. The detailed execution and training process is as follows: At each episode, the online actor-network first outputs multidimensional actions according to the current state s_p, and a random selection process is added before each action is executed for increasing the exploration of the policy with respect to the environment. Specifically, for discrete actions, a ε-greedy policy is taken to select offloading decision in the same way as the DQN. For continuous actions, Ornstein–Uhlenbeck noise mechanism (OU-noise) is added directly to the resource capacity according to [41]. As the training process tends to be stable, the amplitude of action exploration will be reduced according to the following formula:

$$\begin{aligned} \varepsilon &\leftarrow \varepsilon + \Delta \\ \sigma &\leftarrow \sigma - \Delta \end{aligned} \tag{20}$$

where σ implies the disturbance amplitude of OU-noise, and Δ denotes the range of the randomness decrease at each iteration.

Next, the satellite MEC environment performs actions a_p and updates the state s_{p+1} while returning rewards r_p, at which time the agent stores the set (s_p, a_p, r_p, s_{p+1}) in the experience replay buffer R. Then, the agent updates the parameters θ^{μ} and θ^{Q} of the online network μ and Q using the gradient descent of the extracted mini-batch of N samples from R. Finally, in order to prevent the divergence of the training process and ensure the stability of the algorithm, the scheme softly updates the target network μ' and Q' with a constant τ close to 0.

Algorithm 1 Proposed DDPG-Based Computation Offloading

Randomly initialize actor network $\mu(s|\theta^{\mu})$ and critic network $Q(s, a|\theta^{Q})$ with weights θ^{μ} and θ^{Q}.
Initialize target network μ' and Q' with weights $\theta^{\mu'} \leftarrow \theta^{\mu}$, $\theta^{Q'} \leftarrow \theta^{Q}$.
Initialize replay buffer $R = \emptyset$.
for *episode* $= 1$ to $E_{\max}$ **do**
  Initialize a OU-noise object O for action exploration.
  Reset environment state s_0 and reward $r = 0$.
  for $p = 1$ to $T_{\max}$ **do**
    Select an offloading decision according to ε-greedy policy and add O to resource capacity based on $\mu(s_p|\theta^{\mu})$, generate action a_p.
    Execute action a_p and observe reward r_p and new state s_{p+1}.
    Store transition (s_p, a_p, r_p, s_{p+1}) in R.
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R.
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$.
    Update the critic network $Q(s, a|\theta^{Q})$ by minimizing the loss L with the samples:

$$L \approx \frac{1}{N}\sum_{i}\left(y_i - Q(s_i, a_i|\theta^{Q})\right)^2. \tag{21}$$

    Update the actor network $\mu(s|\theta^{\mu})$ by using the sampled policy gradient:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_{i}\nabla_a Q(s, a|\theta^{Q})|_{s=s_i, a=\mu(s_i)} \times \nabla_{\theta^{\mu}}\mu(s|\theta^{\mu})|_{s=s_i}. \tag{22}$$

    Soft update according to equation (19).
  end for
  Reduce the exploration according to equation (20).
end for

During the testing stage, each network is loaded with the optimal parameters learned during the training stage. The online actor-network directly outputs computation offloading strategy based on current environmental observation. After the action is performed, the environment feedbacks the reward and enters a new state. There is no action random selection process, sample storage, and parameter update process in this stage.

C. Complexity Analysis

In this part, we analyze the time complexity of the proposed scheme, which is mainly determined by the training process of DDPG in Algorithm 1. DDPG consists of four neural networks. Assuming that the actor-network contains X fully connected layers, and the critic network contains Y fully connected layers. We consider adding the bias in each layer, then the time complexity of DDPG can be expressed as

$$\begin{aligned} &p_a u_i + 2 \times \sum_{x=0}^{X-1} u_x^{\text{actor}} u_{x+1}^{\text{actor}} + 2 \times \sum_{y=0}^{Y-1} u_y^{\text{critic}} u_{y+1}^{\text{critic}} \\ &= O\left(\sum_{x=0}^{X-1} u_x^{\text{actor}} u_{x+1}^{\text{actor}} + \sum_{y=0}^{Y-1} u_y^{\text{critic}} u_{y+1}^{\text{critic}}\right) \end{aligned} \tag{23}$$

where u_i represents the neural unit number in the ith layer, and p_a represents the corresponding parameters determined by the activation layer function.

Since DDPG is an extended version of DQN, it is also meaningful to compare the complexity of the proposed algorithm with that of the DQN. We assume that the DQN has Z layers and has v_i neurons in the ith layer. Since the network structure of DQN is similar to that of the critic network, the time complexity of DQN is $O(\sum_{z=0}^{Z-1} v_z v_{z+1})$. Here, $u_Y^{\text{critic}} = 1$ in DDPG, and v_Z in DQN is positively related to the dimension of action space. In the action space of our proposed problem $v_Z = (D_x D_f D_r)^{U \times Q}$, where D_x represents the dimension of discrete variables, and D_f and D_r represent the dimensions of continuous variables. v_Z becomes larger as the DQN discretizes the continuous action more finely, and is exponentially related to the number of users U and the length of the task queues Q. Therefore, under the same requirements, DDPG has lower computational complexity than DQN, and the superiority of the DDPG-based approach is verified in the following numerical results.

V. Numerical Results

In this section, the proposed computation offloading scheme based on DDPG is analyzed via simulations. First, we describe the simulation parameters and DDPG network architecture, respectively. Then, we conduct an analysis to illustrate the influence of parameters in DDPG on the training process and results. Finally, the performance of the optimal strategy trained by DDPG is evaluated by comparing it with the baseline algorithms under different environmental parameters.

A. Simulation Setup

Due to economic, technical, and hardware constraints, it is difficult to apply the proposed intelligent computation offloading approach in real satellite scenarios. Therefore, we establish a simulated LEO satellite MEC environment as are carried out in [14] and [21], and use simulation experiments to verify the performance of the proposed method. For our experimental simulation, hardware environment is a CPU-based server, and this server has 8 GB 1867-MHz DDR3, 2.7-GHz Intel Core i5, and 256G memory. Software environment is Python 3.6 with Tensorflow 2.0.0.

In order to reduce the complexity of the simulation model, we simplify the number of users and satellites by referring to [13] and [28]. The reason for this is that for the number of satellites, any large-scale constellation can be logically divided into several small satellite networks [42], [43]. We assume $S = 5$ and model a small-scale satellite subnetwork in this article. When the scenario is extended to an actual large-scale satellite network, it can be composed of several proposed subnetworks. Tasks are centrally controlled, scheduled and allocated resources within a subnet, and independent computation offloading strategies are deployed in parallel among several subnets in a decentralized methodology, which effectively reduces the difficulty of state synchronization and resource management in the large-scale satellite network and

ensures the ability to process tasks in real-time for each subnetwork. Undoubtedly, the focus of this article is on the computation offloading approach within a small-scale satellite network, but it still has adaptability and superiority in practical applications.

For the number of users, we assume $U = 3$ and $Q = 5$. That is, for some satellites, there is no offloading requirement of users. For some other satellites, there may be at most $U * Q$ tasks to be processed. This is a more reasonable simulation of an unbalanced task arrival rate scenario in a centralized burst service demand area under a small-scale satellite network coverage, and better reflects the superiority of an ISL-enabled satellite network. These parameters can simply be modified to adapt to other satellite MEC scenarios.

Meanwhile, Algorithm 1 runs with the number of episodes $E_{\max} = 3000$ and the length of time slots $T_{\max} = 200$ to train the model. In these time slots, tasks arrive dynamically and concurrently and are stored in the queue, waiting for system scheduling. The size of memory replay buffer is $R = 60\,000$, the mini-batch sample size is $N = 256$, and the discount factor is $\gamma = 0.99$. OU-noise O follows the distribution with a mean of 0 and initial variance $\sigma = 0.5$. The value of ε is initialized to 0.5, and soft update factor is $\tau = 0.001$. Further, we assume that the computing capacity of ground users is 200 MHz, and the computing resources of a satellite and cloud are 3 GHz and 10 GHz, respectively [21]. The communication capacity of ISL is 10 Gb/s, which is deployed based on the laser link [17]. And the communication resources of user-satellite and satellite-cloud are 200 Mb/s and 300 Mb/s, respectively [31]. The transmit powers of user-satellite, intersatellite, and satellite-cloud links are, respectively, 2 W, 30 dBW, and 20 dBW [33]. Table II summarizes the main parameter settings in the simulation.

The architectures of the actor-network and the critic network in DDPG are shown in Table III. For the actor-network, we use two fully connected layers to fit the behavior strategy. And the Softmax function and the Sigmoid function are used to activate the output layers for discrete and continuous actions, respectively, to ensure the bounds of the output actions. For the critic network, we use a four-layer fully connected network, and each layer is activated by the function ReLU to approximate the nonlinear Q value of the output.

Moreover, we compare the proposed intelligent DDPG algorithms with five benchmark algorithms which are as follows:

1) Local Computing: All arriving tasks are performed locally by the user in each time slot, regardless of whether their service requirements are met or not.

2) Computation Without ISL Assistance (No-ISL): This computation offloading method follows the same DDPG network structure as the proposed scheme. The only difference is that there is no collaborative computing between satellites in the satellite MEC network, and each task can be offloaded to the access satellite or cloud center except for local execution.

3) DQN [44]: In order to compare with the most commonly used DRL algorithm at present, we also implement the computation offloading strategy based on the DQN, which realizes resource allocation by discretizing the continuous action space. Specifically, the continuous computing and communication resources are defined as $f^p_{u,k} \in \{0, (f_{\max}/L - 1), \ldots, f_{\max}\}$ and $r^p_{u,k} \in \{0, (r_{\max}/L - 1), \ldots, r_{\max}\}$, where the value of discrete division L is set to 5. Moreover, we reduce the dimensionality of the action space by excluding the illegal actions in order to decrease the learning difficulty of the DQN algorithm. In addition, the network architecture of the DQN algorithm is the same as that of the critic network of DDPG, and the ε-greedy policy is also used in the process of exploration.

4) Double DQN [45]: Due to the maximization operation of DQN, it is easy to cause the estimation of Q value to be too optimistic. In order to reduce overestimation, double DQN (DDQN) is proposed. On the basis of natural DQN, the accuracy of algorithm learning is improved by decoupling

TABLE II
SIMULATION PARAMETERS

Parameters	Value
Number of users U	3
Number of satellites S	5
Maximum length of task queues Q	5
Task arrival rate λ	1
Size of computation data of tasks $d^p_{u,k}$	[10, 100] KB
Task workload $c^p_{u,k}$	[1, 1.5] Kcycles/bit
Maximum tolerable delay of tasks $T^{max,p}_{u,k}$	[0.05, 0.1] sec
System time slot length ρ	0.05
The computing capacity of users f_u	200 MHz
The effective capacitance coefficient κ	10^{-28}
The communication capability of ISL R_s	10 Gbps
Satellite-cloud communication capability R_{sc}	300 Mbps
The transmission power of users p_u	2 W
The transmission power between satellites p_s	30 dBW
The transmission power of satellite-cloud link p_c	20 dBW
The distance between user and satellite d_{us}	1000 km [37]
The distance between satellites d_s	800 km
The distance between satellite and cloud d_{sc}	2000 km
The computing capacity of cloud f_c	10 GHz
The computing capacity of a satellite F	3 GHz
The communication capacity of a satellite R	200 Mbps
The reward scaling coefficient ω	10
The reward correction factor $\Re$	3
The punishment factor $\varrho_1, \varrho_2, \varrho_3$	1

TABLE III
DDPG ARCHITECTURE

Net	Layer	Units	Activation
Actor	Input	Shapes of State	
	Fully connected	64	
	Fully connected	32	
	Output	Shapes of Discrete Action	Softmax
		Shapes of Continuous Action	Sigmoid
Critic	Input	Shapes of State and Action	
	Fully connected	128	ReLU
	Fully connected	128	ReLU
	Fully connected	64	ReLU
	Fully connected	64	ReLU
	Output	1	

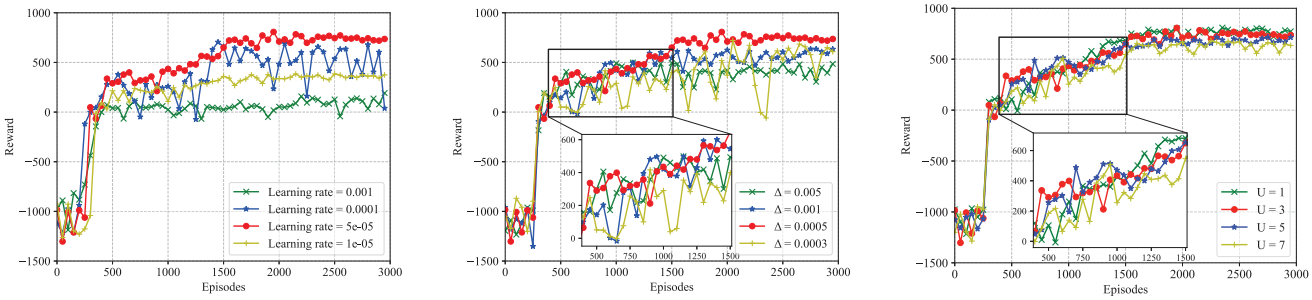

Fig. 5. Convergence performance comparison. (a) Under different learning rates. (b) Under different exploration times. (c) Under the different number of users.

the action selection and evaluation process of the target value function.

5) Dueling DQN [46]: Aiming at the problem that the size of value function has nothing to do with the action in some states, dueling DQN (DuDQN) improves DQN from the network structure. The algorithm splits the original action value function $Q(s, a)$ into two parts, the state value function $V(s)$ and the advantage function $A(s, a)$, and uses the neural network to fit them in the output layer, which improves the network model performance.

B. Parametric Analysis

In this part, we compare the impact of learning rate, exploration time and the number of users on the convergence performance of the proposed algorithm through a series of simulation experiments. Since the training process of the actor-network depends on the output Q-value of the critic network, the critic network should have a faster learning speed than the actor-network in order to achieve a better guidance implication. Therefore, we assume that the learning rate of the critic network is twice that of the actor-network, and only the learning rate of the actor-network is described below. The influence of learning rate on the convergence performance of the proposed strategy is shown in Fig. 5(a).

It can be observed that when the learning rate is 0.001, the training effect is not good as the strategy can be approximately regarded as a greedy algorithm. Under the learning rate of 0.0001, the result falls into the local optimal and fluctuates sharply with increasing iterations, but the suboptimal solution cannot be avoided. When the learning rate decreases to 0.00005, the convergence performance is the best. The strategy remains optimal and the fluctuations are slight as the training progresses. When the learning rate is further set to 0.00001, the reward is almost no fluctuation, but the growth rate is slow. Thus, it can be concluded that if the learning rate is too large, the result will soon saturate at a worse value. Within a reasonable range, with the decrease of the learning rate, the reward can increase faster but fluctuate more. Conversely, if the learning rate is too small, although the result deviation is reduced, the network is updated more slowly and more iterations are needed for training. Therefore, the learning rate should be chosen properly, neither too large nor too small. The most suitable learning rate could be 0.00005.

The ratio of the exploration process to the total process can be described indirectly by Δ. When Δ is larger, the random exploration time in the training process is shorter, otherwise it is longer. The influence of Δ on the convergence performance of the proposed algorithm is illustrated in Fig. 5(b). As shown in various curves, when Δ is set to 0.005 or 0.001, the exploration time ends immediately and the curve converges quickly, but the result remains locally optimal for the rest of the episode. At Δ of 0.0005, the problem obtains the optimal solution and remains stable after a certain amount of fumbling. When Δ is further reduced to 0.0003, the curve jiggles sharply due to the random exploration of the strategy, and the training process cannot converge to a steady state. According to the above analysis, if the exploration time is too short, the trained strategy may lose the opportunity to find the optimal solution, and we cannot obtain the best results. On the contrary, when the exploration time is too long, the strategy is in a chaotic groping state for a long time, which leads to the instability of the training process. Thus, Δ is still not too large or too small, and the most appropriate value could be $\Delta = 0.0005$.

In addition, we appropriately adjust the number of users in the satellite MEC environment to demonstrate the adaptability and scalability of the proposed strategy under varying loads. Since the dimensions of the system state and action are related to the amount of tasks, the number of neurons in the input and output layers of the algorithm needs to be modified accordingly. The convergence performance of the algorithm under different numbers of users is shown in Fig. 5(c). It can be seen that under the different number of users, the learning process of the intelligent DDPG-based strategy can achieve a good convergence effect, and the average reward of the task can be maximized. However, due to changes in the network structure, more neurons mean more parameters and longer learning time. Therefore, as mentioned above, the simulation is based on the unbalanced load scenario in a small-scale STN. In order to compromise between the number of users and the complexity of the model, we defaulted to the number of users $U = 3$.

C. Performance Comparison

The system performance under different algorithms is shown in Fig. 6. Among them, Fig. 6(a) shows the convergence performance comparison, and each strategy achieves a stable computation offloading process. Fig. 6(b)

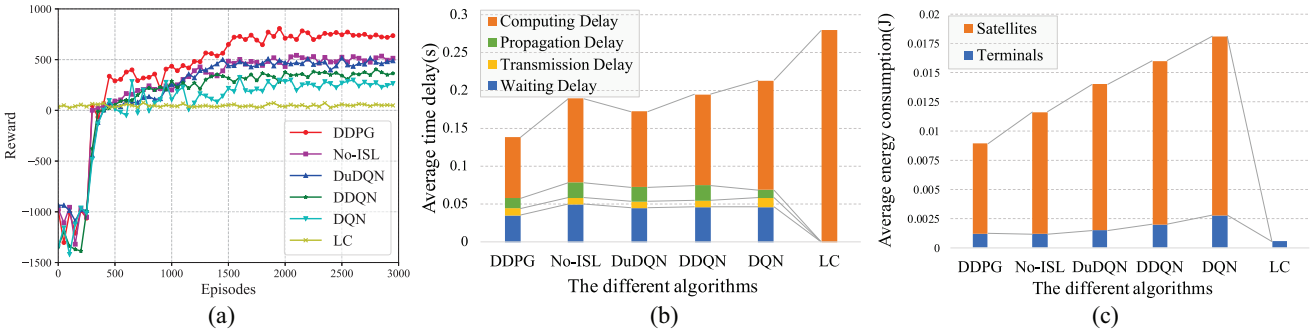

Fig. 6. System performance comparison under different algorithms. (a) Convergence performance. (b) Average time delay. (c) Average energy consumption.

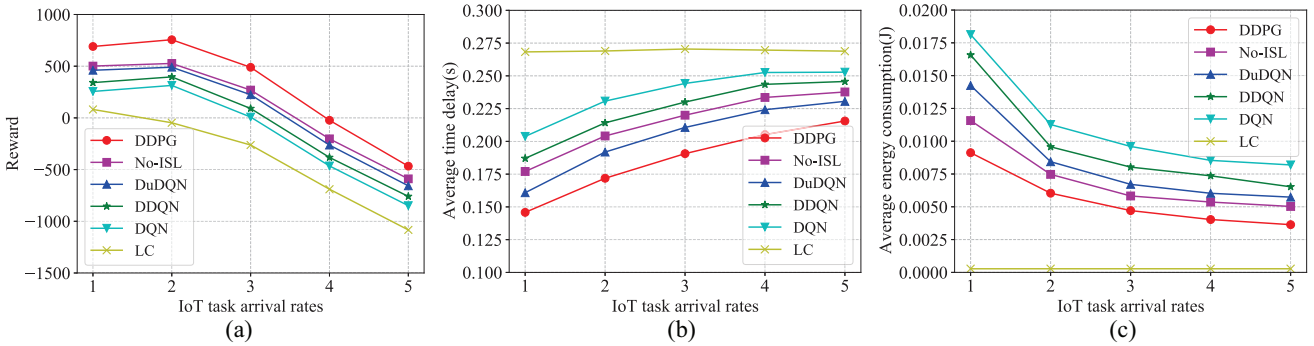

Fig. 7. Performance of each algorithm under different task arrival rates. (a) Reward. (b) Average time delay. (c) Average energy consumption.

and (c) show the average time delay and energy consumption with multicomponents, respectively. There is no difference in the order of magnitude, so all components of the results deserve attention and cannot be ignored. It can be observed that the DDPG obtains the optimal strategy and maximizes the reward, demonstrating the effectiveness of the proposed scheme. While for the No-ISL, due to the lack of load balancing capability, the system energy consumption and the end-to-end delay are increased. Therefore, the reward of the No-ISL is lower than that of the DDPG. Moreover, due to the discretization of the action space by DQN, many behavior attempts are lacking at this time, which makes it difficult for DQN to accurately find the optimal offloading strategy. So the delay and energy consumption are increased in this case. Both DDQN and DuDQN improve the performance of natural DQN, but their performance is still inferior to DDPG which explores on continuous action space. For LC, since most tasks cannot be completed on time, it has the lowest reward and is only the terminal energy consumption caused by computing locally.

Table IV lists the running time of various algorithms. It can be seen that the training time is related to the neural network structure, and more parameters and layers require more learning time. But there is little difference in the time they spend in the testing phase, implying the feasibility of the intelligent approach for actual deployment.

Fig. 7 shows the performance of different algorithms under different task arrival rates. It can be observed from Fig. 7(a), when λ is small, the DDPG still has the largest reward and outperforms other algorithms. The average delay and energy consumption results in Fig. 7(b) and (c) demonstrate that the proposed approach ensures that the task is completed on time while accomplishing the optimal computation offloading strategy to minimize energy consumption. As λ increases, more tasks are buffered in the queue per time slot. At this time, the rewards of the DDPG and DQN series algorithms are increased because their environments have greater on-board computing capability. However, the No-ISL forwards the computation-intensive tasks that exceed the satellite load to the cloud center for processing, which increases end-to-end delay, so the reward growth is not significant. The average delay and energy consumption of LC is constant, but its penalty increases as the number of tasks arriving increases. When λ increases further, the results of each algorithm are unsatisfactory, indicating that the satellite MEC network with limited resources is unable to provide computing assistance for all tasks in an emergency situation where a large number of heterogeneous IoT applications exist in a certain region. Among them, the DDPG has the best strain ability and gets better rewards in relative terms.

TABLE IV
RUNNING TIME

Algorithm	Training stage	Testing stage
DDPG	28112.20s	21.53s
No-ISL	27657.73s	22.23s
DuDQN	13720.91s	20.03s
DDQN	14068.75s	19.47s
DQN	13804.88s	19.78s
LC	/	8.05s

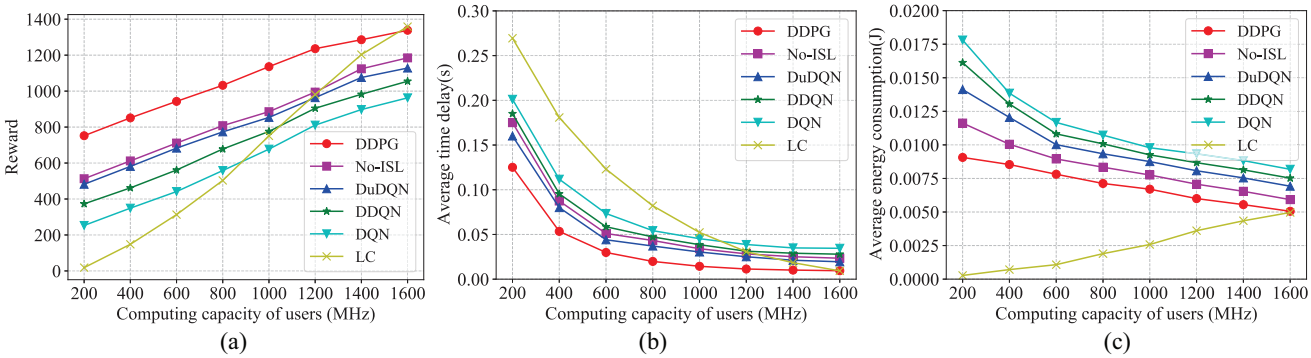

Fig. 8. Performance of each algorithm under different computing capacities of users. (a) Reward. (b) Average time delay. (c) Average energy consumption.

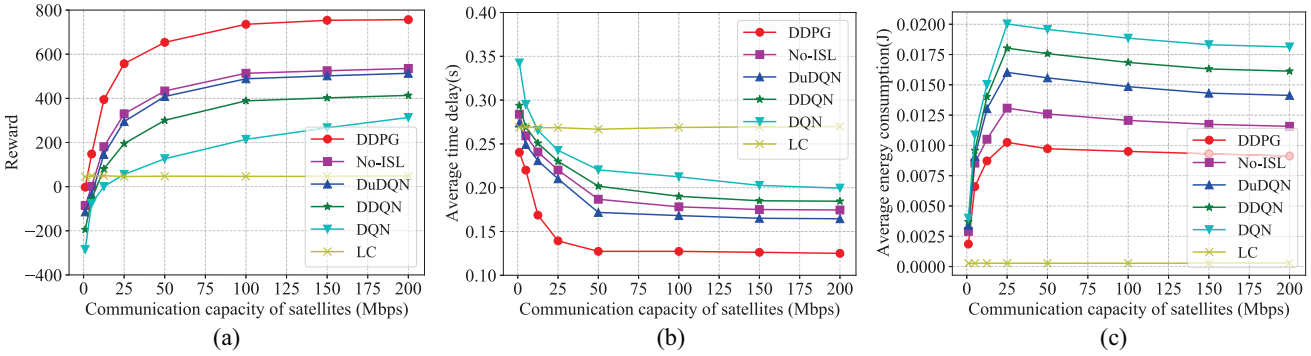

Fig. 9. Performance of each algorithm under different satellite communication resources. (a) Reward. (b) Average time delay. (c) Average energy consumption.

Furthermore, we compare the variations of the computation offloading strategies based on various algorithms when the satellite MEC environment has different capability, and carry out the following experiments, respectively.

First, we compare the results of each algorithm by setting different computing capabilities of users f_u. It can be observed in Fig. 8(a), when f_u is small, the penalty of LC is very large because the tasks cannot be processed in a certain time. Satellite edge computing significantly lightens the burden for users at this time, and all computation offloading algorithms get better rewards. According to the results of Fig. 8(b) and (c), the DDPG with optimal strategy ensures that the task is completed on time while minimizing energy consumption. On the other hand, with the gradual increase of f_u, the reward of tasks to be computed locally increases. There also have more tasks that can be processed locally on time for other algorithms, so the average delay and energy consumption of all methods are reduced at this time. When f_u increases to a certain level, the rewards of LC are higher than that of computation offloading. Comparing these different algorithms, the results of DDPG gradually overlap with that of LC, while the DQN series algorithms obtain suboptimal results due to the lack of action cognition.

Next, we compare the mitigation ability of edge computing environment in the face of tasks with large computation offloading demands by setting different satellite resource capabilities. At this time, the results of LC are independent of the satellite capacity.

Fig. 9 shows the performance of each strategy when the satellites have different communication resources R. It can be observed from Fig. 9(a), when R is small, the rewards of each algorithm are very low. This is due to the fact that most tasks cannot be offloaded immediately and have to be processed locally or wait in the queue for the next scheduling. These two second-best choices also entail penalties for computation timeout and queue overflow. Meanwhile, as the results of average delay and energy consumption in Fig. 9(b) and (c) show, there is a large delay for each algorithm, and only a small portion of tasks can be offloaded to satellite or cloud for processing. As R gradually increases, more and more tasks can be transmitted to the satellite through satellite-terrestrial link. At this time, more tasks can be completed on demand, so the system penalties are reduced and the energy consumption for transmission and computation are correspondingly increased. When R increases further, the amount of computation that the satellite edge can undertake is limited due to the restricted on-board computing resources, so some nonlatency-sensitive tasks are forwarded to the cloud for processing, where average energy consumption is slightly reduced and the reward of each strategy remains maximized. Comparing the different algorithms, it can be observed that the satellite edge network with cooperative computing at sufficient communication resources has greater load balancing capacity and makes full use of resources to process tasks, thus, gaining more rewards.

Finally, the performance of each algorithm when satellites have different computing resources F is shown in Fig. 10. As

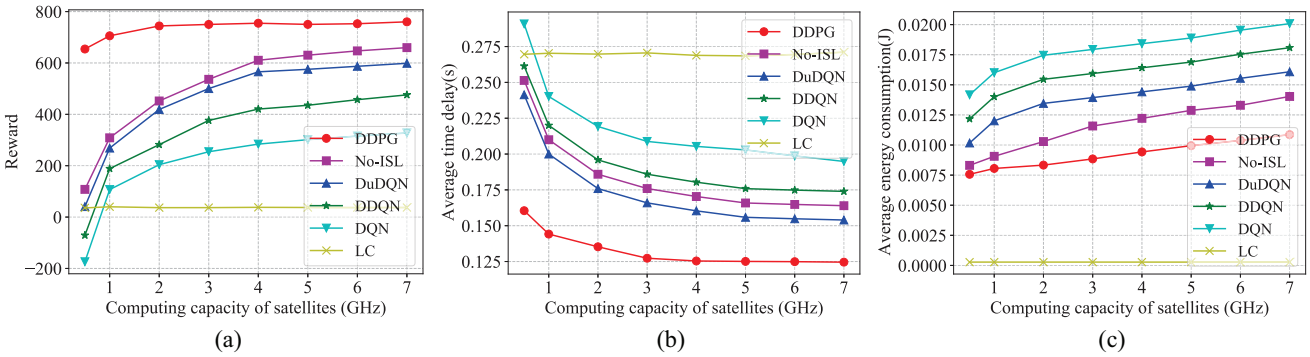

Fig. 10. Performance of each algorithm under different satellite computing resources. (a) Reward. (b) Average time delay. (c) Average energy consumption.

can be observed from the three results, when the computing resources of a single satellite are insufficient, the DDPG can expand the system capacity through ISL, and its results still have obvious advantages. At this time, No-ISL can only perform a part of the tasks at the satellite edge due to the lack of on-board computing resources, and most tasks are forwarded to the remote cloud by the access satellite, so the delay is larger and the reward is smaller. The DQN series algorithms achieve worse rewards than LC due to its poor strategy. When F gradually increases, it represents that more computations can be undertaken at satellites. Tasks can be completed on demand, so the reward of each algorithm is increased by different degrees. Among them, the curve of No-ISL surges, but it is still lower than that of DDPG. However, when F is further increased, the size of the data transmitted to the satellite network is fixed due to the limited communication capacity of the satellite-terrestrial uplink, so the curves remain stable at this time. Comparing the results of different algorithms, DDPG is always optimal, and DQN series algorithms still underperforms. No-ISL requires F to be increased to a certain level, and its result can be close to that of DDPG.

VI. Conclusion

In this article, we provide computing service based on LEO satellite edge network and remote cloud for IoT users who cannot communicate directly with terrestrial networks. In the presented three-tier satellite MEC architecture, intersatellites are enabled by cooperative computing, and users can offload tasks on demand to reduce the burden of LC. We jointly optimize offloading decision and resource allocation to minimize the system energy consumption under meeting the task requirement time and adopt the resource reservation distribution mechanism to ensure the stability of the computation. Moreover, we propose an intelligent DDPG-based computation offloading scheme to adapt to the dynamic task arrival queue environment. The algorithmic network and action selection are adjusted to achieve simultaneous training of discrete and continuous actions. The proposed approach achieves reliable and timely computing tasks and exhibits superior performance in reducing energy consumption. In our simulation results, the proposed DDPG-based approach improves the reward by nearly 3× and reduces the average energy consumption by nearly 50% compared with the DQN-based strategy.

In future work, we will expand the complexity of the state space to be closer to a realistic satellite MEC environment, and further, optimize the structure design of the DRL algorithm to improve the rapidity and stability of computation offloading.

References

[1] M. H. C. Garcia et al., "A tutorial on 5G NR V2X communications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1972–2026, 3rd Quart., 2021.

[2] A. Mahmood et al., "Industrial IoT in 5G-and-beyond networks: Vision, architecture, and design trends," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4122–4137, Jun. 2022.

[3] L. Chettri and R. Bera, "A comprehensive survey on Internet of Things (IoT) toward 5G wireless systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 16–32, Jan. 2020.

[4] A. Kaushik, J. Thompson, E. Vlachos, C. Tsinos, and S. Chatzinotas, "Dynamic RF chain selection for energy efficient and low complexity hybrid beamforming in millimeter wave MIMO systems," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 4, pp. 886–900, Dec. 2019.

[5] A. Kaushik, E. Vlachos, C. Tsinos, J. Thompson, and S. Chatzinotas, "Joint bit allocation and hybrid beamforming optimization for energy efficient millimeter wave MIMO systems," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 1, pp. 119–132, Mar. 2021.

[6] X. Gao, R. Liu, and A. Kaushik, "Virtual network function placement in satellite edge computing with a potential game approach," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 1243–1259, Jun. 2022.

[7] M. Casoni, C. A. Grazia, M. Klapez, N. Patriciello, A. Amditis, and E. Sdongos, "Integration of satellite and LTE for disaster recovery," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 47–53, Mar. 2015.

[8] G. Giambene, S. Kota, and P. Pillai, "Satellite-5G integration: A network perspective," *IEEE Netw.*, vol. 32, no. 5, pp. 25–31, Sep./Oct. 2018.

[9] S. Chen, S. Sun, and S. Kang, "System integration of terrestrial mobile communication and satellite communication—The trends, challenges and key technologies in B5G and 6G," *China Commun.*, vol. 17, no. 12, pp. 156–171, Dec. 2020.

[10] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues," *IEEE Netw.*, vol. 34, no. 3, pp. 224–231, May/Jun. 2020.

[11] T. de Cola and I. Bisio, "QoS Optimisation of eMBB services in converged 5G-satellite networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12098–12110, Oct. 2020.

[12] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6722–6747, Aug. 2020.

[13] S. Zhang, G. Cui, Y. Long, and W. Wang, "Joint computing and communication resource allocation for satellite communication networks with edge computing," *China Commun.*, vol. 18, no. 7, pp. 236–252, Jul. 2021.

[14] Z. Song, Y. Hao, Y. Liu, and X. Sun, "Energy-efficient multiaccess edge computing for terrestrial-satellite Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14202–14218, Sep. 2021.

[15] X. Gao, R. Liu, A. Kaushik, and H. Zhang, "Dynamic resource allocation for virtual network function placement in satellite edge clouds," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2252–2265, Jul./Aug. 2022.

[16] J. Kua, S. W. Loke, C. Arora, N. Fernando, and C. Ranaweera, "Internet of Things in space: A review of opportunities and challenges from satellite-aided computing to digitally-enhanced space living," *Sensors*, vol. 21, no. 23, p. 8117, 2021.

[17] Y. Lee and J. P. Choi, "Connectivity analysis of mega-constellation satellite networks with optical inter satellite links," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 6, pp. 4213–4226, Dec. 2021.

[18] J. Liang, W. Liting, and D. Hao, "Summary of research on satellite cooperative transmission technology in space information network," in *Proc. Int. Conf. Inf. Sci., Parallel Distrib. Syst. (ISPDS)*, Xi'an, China, Aug. 2020, pp. 56–58.

[19] I. Leyva-Mayorga, B. Soret, and P. Popovski, "Inter-plane inter-satellite connectivity in dense LEO constellations," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3430–3443, Jun. 2021.

[20] S. Fu, J. Gao, and L. Zhao, "Collaborative multi-resource allocation in terrestrial-satellite network towards 6G," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7057–7071, Nov. 2021.

[21] Q. Tang, Z. Fei, B. Li, and Z. Han, "Computation offloading in LEO satellite networks with hybrid cloud and edge computing," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9164–9176, Jun. 2021.

[22] C. Li, Y. Zhang, X. Hao, and T. Huang, "Jointly optimized request dispatching and service placement for MEC in LEO network," *China Commun.*, vol. 17, no. 8, pp. 199–208, Aug. 2020.

[23] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2992–3005, Oct. 2021.

[24] H. Lu, X. He, M. Du, X. Ruan, Y. Sun, and K. Wang, "Edge QoE: Computation offloading with deep reinforcement learning for Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9255–9265, Oct. 2020.

[25] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3415–3426, Apr. 2020.

[26] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1529–1541, Jul.–Sep. 2021.

[27] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[28] C. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, "Deep *Q*-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5871–5883, Jun. 2019.

[29] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228, Jul. 2020.

[30] Z. Zhang, W. Zhang, and F.-H. Tseng, "Satellite mobile edge computing: Improving QoS of high-speed satellite-terrestrial networks using edge computing techniques," *IEEE Netw.*, vol. 33, no. 1, pp. 70–76, Jan./Feb. 2019.

[31] Y. Wang, J. Zhang, X. Zhang, P. Wang, and L. Liu, "A computation offloading strategy in satellite terrestrial networks with double edge computing," in *Proc. IEEE Int. Conf. Commun. Syst. (ICCS)*, Chengdu, China, Dec. 2018, pp. 450–455.

[32] D. Zhu et al., "Deep reinforcement learning-based task offloading in satellite-terrestrial edge computing networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Nanjing, China, Mar./Apr. 2021, pp. 1–7.

[33] G. Cui, Y. Long, L. Xu, and W. Wang, "Joint offloading and resource allocation for satellite assisted vehicle-to-vehicle communication," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3958–3969, Sep. 2021.

[34] X. Qi, B. Zhang, Z. Qiu, and L. Zheng, "Using inter-mesh links to reduce end-to-end delay in walker delta constellations," *IEEE Commun. Lett.*, vol. 25, no. 9, pp. 3070–3074, Sep. 2021.

[35] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[36] M. De Sanctis, E. Cianca, G. Araniti, I. Bisio, and R. Prasad, "Satellite communications supporting Internet of Remote Things," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 113–123, Feb. 2016.

[37] G. Cui, X. Li, L. Xu, and W. Wang, "Latency and energy optimization for MEC enhanced SAT-IoT networks," *IEEE Access*, vol. 8, pp. 55915–55926, 2020.

[38] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

[39] X. Gao, R. Liu, and A. Kaushik, "A distributed virtual network function placement approach in satellite edge and cloud computing," Apr. 2021, *arXiv:2104.02421*.

[40] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, vol. 32, Jun. 2014, pp. 387–395.

[41] C. Szepesvári, "Algorithms for reinforcement learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 4, no. 1, pp. 1–103, 2010.

[42] Z. Han, C. Xu, Z. Xiong, G. Zhao, and S. Yu, "On-demand dynamic controller placement in software defined satellite-terrestrial networking," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 2915–2928, Sep. 2021.

[43] X. Gao, R. Liu, A. Kaushik, J. Thompson, H. Zhang, and Y. Ma, "Dynamic resource management for neighbor-based VNF placement in decentralized satellite networks," in *Proc. 1st Int. Conf. 6G Netw. (6GNet)*, Paris, France, Jul. 2022, pp. 1–5.

[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, and J. Veness, "Human-level control through deep reinforcement learning." *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[45] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," Nov. 2015, *arXiv:1509.06461*.

[46] Z. Wang, N. D. Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," Nov. 2015, *arXiv:1511.06581*.

Hangyu Zhang received the B.S. degree from the School of Communication Engineering from Jilin University, Changchun, China, in 2019. She is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, Beihang University, Beijing, China.

Her research interests include the edge intelligence, satellite edge computing, and resource allocation.

Rongke Liu (Senior Member, IEEE) received the B.S. and Ph.D. degrees from Beihang University, Beijing, China, in 1996 and 2002, respectively.

He was a Visiting Professor with the Florida Institution of Technology, Melbourne, FL, USA, in 2006, The University of Tokyo, Tokyo, Japan, in 2015, and the University of Edinburgh, Edinburgh, U.K., in 2018, respectively. He is currently a Full Professor with the School of Electronic and Information Engineering, Beihang University. He received the support of the New Century Excellent Talents Program from the Minister of Education, Beijing. He has attended many special programs, such as China Terrestrial Digital Broadcast Standard. He has published over 100 papers in international conferences and journals. He has been granted over 20 patents. His research interest covers wireless communication and space information network.

Aryan Kaushik (Member, IEEE) received the M.Sc. degree in telecommunications from The Hong Kong University of Science and Technology, Hong Kong, in 2015, and the Ph.D. degree in communications engineering from The University of Edinburgh, Edinburgh, U.K., in 2019.

He is an Assistant Professor with the University of Sussex, Brighton, U.K. He has been a Research Fellow with the University College London, London, U.K., from 2020 to 21. He held visiting appointments with the Imperial College London, London, the Beihang University (Shenzhen and Beijing), Athena Research and Innovation Center, Marousi, Greece, and the University of Luxembourg, Luxembourg City, Luxembourg. His research interests are broadly in 5G Advanced/6G wireless communications, signal processing, energy efficient communications, and AI.

Dr. Kaushik has been serving as an Associate Editor for the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, IEEE COMMUNICATIONS LETTERS, *IET Signal Processing*, and *IET Networks*. He has been serving as the Lead Guest Editor for Special Issues in the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY and *IET Signal Processing*, a Tutorial Speaker at IEEE WCNC 2023, the Lead General Chair for IEEE international workshops, such as IEEE WCNC 2023, IEEE PIMRC 2022, and IEEE SECON 2022, and the Track Co-Chair at IEEE WCNC 2023. He has also been the TPC Member at the IEEE ICC from 2021 to 2023, including ICC 2023 RISSE SAC, and a Conference Champion for IEEE PIMRC 2020.

Xiangqiang Gao received the B.Sc. degree from the School of Electronic Engineering from Xidian University, Xi'an, China, in 2012, the M.Sc. degree from Xi'an Microelectronics Technology Institute, Xi'an, in 2015, and the Ph.D. degree from the School of Electronic and Information Engineering from Beihang University, Beijing, China, in 2022.

He is currently a Postdoctoral Fellow with the China Academy of Space Technology (Xi'an), Xi'an, and also with the School of Electronic and Information Engineering, Beihang University. His research interests include cloud computing, satellite edge computing, network function virtualization, and resource allocation.