Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Minimize average tasks processing time in satellite mobile edge computing systems via a deep reinforcement learning method

Shanchen Pang[1*], Jianyang Zheng[1], Min Wang[2], Sibo Qiao[1], Xiao He[1] and Changnan Gao[1]

Abstract

Recently, the development of Low Earth Orbit (LEO) satellites and the advancement of the Mobile Edge Computing (MEC) paradigm have driven the emergence of the Satellite Mobile Edge Computing (Sat-MEC). Sat-MEC has been developed to support communication and task computation for Internet of Things (IoT) Mobile Devices (IMDs) in the absence of terrestrial networks. However, due to the heterogeneity of tasks and Sat-MEC servers, it is still a great challenge to efficiently schedule tasks in Sat-MEC servers. Here, we propose a scheduling algorithm based on the Deep Reinforcement Learning (DRL) method in the Sat-MEC architecture to minimize the average task processing time. We consider multiple factors, including the cooperation between LEO satellites, the concurrency and heterogeneity of tasks, the dynamics of LEO satellites, the heterogeneity of the computational capacity of Sat-MEC servers, and the heterogeneity of the initial queue for task computation. Further, we use the self-attention mechanism to act as a Q-network to extract high-dimensional dynamic information of tasks and Sat-MEC servers. In this work, we model the Sat-MEC environment simulation at the application level and propose a DRL-based task scheduling algorithm. The simulation results confirm the effectiveness of our proposed scheduling algorithm, which reduces the average task processing time by 22.1%, 30.6%, and 41.3%, compared to the genetic algorithm(GA), the greedy algorithm, and the random algorithm, respectively.

Keywords Satellite Mobile Edge Computing (Sat-MEC), Deep Reinforcement Learning (DRL), Task scheduling, Minimize task processing time

Introduction

Cloud Computing has become one of the key drivers of modern business and technology development [1]. Cloud computing can provide organizations and individuals with efficient, cost-effective, flexible, scalable, and secure computing resources and services, including services such as storage, databases, servers, and software. However, cloud computing faces challenges such as latency, security, bandwidth, and availability in some specific application scenarios. To overcome these challenges, Fog Computing and Edge Computing are emerging [2]. Fog computing extends the concept of cloud computing. It is closer to the place where the data is generated than cloud computing. The data, data-related processing and applications are centralized in devices at the edge of the network, instead of being stored almost entirely in the cloud. Edge computing further promotes Fog Computing's concept of "local network processing power" by enabling sensitive data to be processed on local devices or edge nodes, reducing reliance on remote data transfers and thus increasing privacy and security [3]. In addition, the edge computing paradigm sinks computing capability

*Correspondence:
Shanchen Pang
pangsc@upc.edu.cn
[1] College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China
[2] College of Information Engineering, Yangzhou University, Yangzhou 225127, Jiangsu, China

to the edge, where data from IMDs can be processed faster through edge computing, thus increasing availability in remote or unstable network environments [4].

By combining edge computing with LEO satellite to form the Sat-MEC architecture, we can not only overcome the limitations of cloud computing but also provide a feasible and efficient solution for the realization of 6G [5]. This integrated approach promises to provide fast, reliable services on a global scale to meet future communications need. We begin by introducing the readers to the difference between task offloading and task scheduling: task scheduling is deciding which tasks should be executed by which edge/fog devices in a edge/fog network. It involves determining the most suitable edge/fog device for each task based on device capabilities, compute capacity, network conditions, and task requirements. Task offloading, on the other hand, refers to transferring computational tasks from IoT devices to more powerful edge/fog devices in the network [6, 7].

In the Sat-MEC architecture, the terrestrial tasks can offload from IMDs to the Sat-MEC servers in the absence of terrestrial network services or in specific areas. In our Sat-MEC scenario, when tasks are offloaded from the ground to the over-the-top satellite, it is difficult for the limited computational resources to compute multiple terrestrial tasks at a single satellite. However, with the development of LEO satellite constellations in recent years, LEO satellites can communicate directly with each other by Inner Satellite Link(ISL). The computational resources of the satellites can be shared by ISL. Task scheduling can be performed in the Sat-MEC servers resource pool to balance the computational load and futher to minimize latency, reduce network congestion, and ensure efficient use of resources [8].

Our objective in this work is to reduce the average processing time of tasks from ground-based IMDs by pursuing a reasonable task scheduling strategy. In our Sat-MEC scenario, we consider multiple ground IMDs simultaneously, and multiple tasks generated at the exact moment can be processed locally by the ground IMDs and collaboratively by the resource pool of the Sat-MEC servers. In our research, we found that for the simultaneous scheduling of multiple tasks in complex dynamic environments, the sequential order of the task scheduling also affects the average processing time of the tasks, Therefore, the simultaneous scheduling problem of simultaneous scheduling of multiple tasks, not only the problem of matching tasks with Sat-MEC servers but also the problem of task scheduling sequence should be considered. To solve this NP-hard problem, we are searching for a method to schedule tasks that can find an optimal solution in the high-dimensional feature space of the problem.

Swarm intelligence algorithms, such as GA and Particle Swarm Optimization(PSO), are efficient in finding global or near-global optimal solutions in the domain of high-dimensional and nonlinear problems [9]. However, in our scenario, the diversity of tasks on the ground and the heterogeneity of resources on Sat-MEC servers lead to complexity in the solution space. GA and PSO are challenging in complex and dynamic offloading or scheduling problems. Therefore we need find an algorithm that has the strong ability to explore the huge problem space and obtain a near-global optimal solution.

The stochastic process model provides a practical perspective when considering different approaches for scheduling algorithms. In particular, Markov properties provide an essential framework for understanding decision-making in task scheduling. Markov property means that the probability distribution of a stochastic process for a future state depends only on the current state, given the present state and all past states [10]. A Markov chain is a collection of discrete random variables with the Markov property, and a Markov Decision Process(MDP) introduces the concepts of decision, action, reward, and debriefing based on the Markov chain. As we have already discussed, Markov attributes and MDP provide a robust framework for describing decision-making in uncertain environments. Intelligence must continually learn how to make the best decisions in such environments by interacting with the environment. Reinforcement learning is specifically designed to address this aspect. Traditional reinforcement learning techniques have difficulty when dealing with complex environments and large state spaces. This is where DRL comes in, combining the capabilities of deep learning with the principles of reinforcement learning to handle more complex scenarios [11].

Through continuous experimentation and exploration, the agent learns to find the optimal strategy that maximizes cumulative rewards. In the task scheduling process in our Sat-MEC scenarios, the over-the-top LEO satellite receives tasks and the characteristics of tasks, collects the characteristics information of Sat-MEC servers from neighboring satellites by ISL, and makes the task scheduling process accordingly, which consistents with the Markov chain and reinforcement learning model. Therefore, we use DRL, a combination of deep and reinforcement learning as the model for our scheduling algorithm. Furthermore, we use the self-attention mechanism as a Q-network for extracting complex dynamic features. Ultimately, the exploration strategy under high-dimensional space is obtained through continuous interaction with the environment, and ultimately, task scheduling in dynamic and complex environments is realized [12]. The main contributions in this paper are fourfold.

- We formalize the task offloading issue in the Sat-MEC scenario as a MDP, using the Satellite Tool Kit (STK) to model the spatial information of the LEO satellite. Additionally, we used Python to model the simulation of the Sat-MEC environment, where we consider task heterogeneity and server computational resource heterogeneity.
- To address the challenge of finding solutions in a high-dimensional state space, we resort to function approximators based on deep neural networks and integrate the self-attention mechanism into the Q-network. We proposed a DRL-based on the Double Deep Q-Network (DDQN) [13], i.e., SATDRL. This algorithm thoroughly considers the global information of tasks and servers and enables simultaneous offloading decisions for multiple tasks arriving at the same time. Thus, it facilitates learning the optimal computation offloading strategy within the Sat-MEC scenario.
- We have performed numerical experiments based on PyTorch 1.9 [14] to verify the theoretical research of this paper. The results show that our proposed offloading algorithm outperforms the three basic methods, which could reduce the average task processing time by 22.1%, 30.6%, and 41.3%, compared to the GA, the greedy algorithm, and the random algorithm, respectively. Whcih the SATDRL algorithm achieves the best computational scheduling performance and is capable of making good scheduling decision action in a high-dimensional space under different environmental constraints..

Related works

In recent years, the Sat-MEC scenarios has received a lot of attention with the rapid development of LEO satellite related technologies and artificial intelligence. Some works considered different approaches such as numerical computation, game theory, genetic algorithms, and deep learning to achieve their different goals, including channel transmission stability, minimizing task execution energy consumption, minimizing task processing time, and improving computational resource utilization. In this section, we will introduce the relevant literature and methods below.

Wang et al. [15] modeled a game-theoretic-based computational offload system in a satellite edge computing scenario. Intermittent ground satellite communication due to satellite orbit is considered in their system model. And proposed an iterative algorithm to search for the Nash equilibrium of the game to reduce the average cost of the device. Li et al. [16] introduce a concept termed 'LEO-MEC', which involves the installation of edge servers on LEO satellites, they addressed two main issues in their research, the issue of service request scheduling decisions, which directly affects the system's resource utilization, and the problem of service placement, which has an intimate correlation with the scheduling decisions. They have attained superior outcomes by utilizing the OPTI toolbox to tackle these issues compared to other benchmark algorithms. Wang et al. [17]developed a computational latency model for transmission delay and computational delay for a scenario where a task set consisting of multiple independent tasks with high quality of service Quality of Service (QoS) requirements is offloaded to a satellite edge computing cluster. The satellite edge computing scenario is proposed using a GA-based offloading algorithm for QoS enhancement. Tang et al. [18] proposed a hybrid cloud and edge computing low-orbiting satellite (CECLS) network with a three-tier computing architecture and investigated computational offloading decisions in this framework to minimize the total energy consumption of ground users, and using a binary variable relaxation method to transform the original nonconvex problem into a linear programming problem, they propose a distributed algorithm based on Alternating Direction Multiplier Method (ADMM) to approximate the optimal solution with low computational complexity. The network can provide heterogeneous computing resources for ground users and enable ground users to access computing services worldwide. Zhu et al. [19] consider a satellite-ground cooperative edge computing architecture where tasks can be performed by SatEC servers or urban TCs (ground stations). The offload location decision and bandwidth allocation is a MIP (certificate linear programming) problem model-free learning, they propose a DRTO algorithm based on the current channel state to make the offload decision. Meanwhile, DRTO can improve its offloading strategy by learning the real-time trend of channel state to adapt to the high dynamic time complexity of satellite-terrestrial networks. Yu et al. [20] present a context that integrates edge computing within the structure of Edge Computing-enabled Satellite-Aerial-Ground Integrated Networks (EC-SAGINs). Particularly in exceptional scenarios, this can offer Internet of Vehicles services to users in remote regions. Furthermore, the authors propose an advanced scheme of pre-classification alongside a decision-making algorithm predicated on Deep Imitation Learning (DIL). This enables the satellite to execute tasks as rapidly as feasible while ensuring minimal utilization of resources. Mao, et al. [21]also consider combining SAGIN with edge computing to achieve rational resource allocation and task offloading. Cassar et al. [22]proposed an edge computing platform that leverages the computing-as-a-service capabilities of low-orbiting satellites to implement an in-orbit computing continuum for equal access to computing, thoroughly improving the utilization of computing resources.

There has also been a lot of excellent work in recent years on task scheduling and offloading of for MEC sysytem with using Device-to-Device(D2D) or broker approach. He et al. [23] maximize the number of D2D-enabled devices by integrating D2D-MEC techniques in order to further increase the computational power of cellular networks. Seng et al. [24] proposed a GS-based user matching algorithm in a D2D-enabled MEC system to find a match between an offloading requester's computational task and an edge server or user. Zanzi et al. [25] proposed a smart online aggregated reservation (SOAR) framework for MEC brokers to minimize their cost of reserving resources in a MEC environment that supports brokers for multiple users without the knowledge of future demands. Zhang et al. [26] considered inseparable tasks in their considered Sat-MEC system. They proposed a greedy algorithm for task allocation and designed different task allocation strategies for different kinds of tasks to reduce the average cost of task computation. Chai et al. [27]considered task dependencies in their considered Sat-MEC scenario formed a DAG-directed acyclic graph of tasks with dependency properties, and provided a low-complexity multi-task dynamic offloading framework. They proposed an RNN-based offloading algorithm that achieves a lower long-term cost of the offloading system. Liu et al. [28]proposed a new support wireless power transmission (WPT) in Space-Air-Ground Power Internet of Things (SAG -PIoT) architecture to solve the problem of limited battery capacity and difficulty in replacing the IoT devices, and based on this, combined with Liapunov optimization method proposed a joint online optimization algorithm for task allocation and system multiple resource allocation to minimize the long term time averaged network operation cost. Hussein et al. [29]considered two different scheduling algorithms based on two swarm intelligence algorithms, Ant Colony Optimisation (ACO) and PSO, to balance the IoT tasks on the fog nodes efficiently and to improve the QoS of the IoT applications and the utilization of the fog nodes, taking into account the cost of communication and the response time. Javanmardi et al.[30], in their proposed offloading problem in an IoT scenario, considered multiple characteristics of fog nodes and tasks such as CPU processing power, memory size and bandwidth, and CPU requirements. They proposed a Particle Swarm Optimisation (PSO) algorithm combined with fuzzy logic to improve global search capability. Zhang et al. [31]designed a satellite peer offloading scheme that defines the multi-hop satellite peer offloading (MHSPO) problem as a global optimization problem and transforms the Lyapunov-based entire network cost minimization problem into several sub-problems carried out on individual satellites. Their scheme efficiently balances the imbalanced workloads and improves the resource utilization of the satellite network. Matrouk et al. [32]. proposed a mobile-aware proximal policy optimization algorithm (MAPPO) deployed at the gateway to perform the history-aware switching process, which improves network performance and accuracy and reduces latency. The tasks are classified and scheduled through a two-process modular neural network. The authors' approach reduces latency and offload time and improves system throughput. Chen et al. in [33], employ DRL to address the co-optimization problem of computation offloading and resource allocation within MEC systems. Similarly. Seid et al. [34] propose an optimal system performance achieving model-free collaborative computing offloading and resource allocation strategy based on DRL. Also, zheng et al. [35]proposed a LEO network architecture using centralized resource pooling based on satellite resource pooling, and designed a combined allocation of fixed channel pre-allocation and dynamic channel scheduling based on reinforcement learning. The system allocates the channel resources by Q-learning algorithm and trains the optimal channel allocation strategy. Shakarami et al. [12] consider modeling the offloading decision problem between different execution environments in a multi-user/multi-server environment with heterogeneous services and edge/cloud platforms, using multivariate linear regression and DNN modeling as a hybrid model to obtain optimal offloading results.

Liu et al. [36] conducted an in-depth study and found the optimal offloading probability and optimal transmission rate based on M/M/1 queueing theory to minimize energy consumption, execution delay, execution delay, and price cost. Li et al. [37] developed a system model consisting of M/M/1 and M/M/c queues to capture the task execution process of an IMD, an MEC server, and a remote cloud server, respectively, and solved a joint optimization problem regarding task offloading delay and energy consumption. Chen et al. [38] considered the emergent task computation queue idleness in edge servers. They proposed a computation task mechanism consisting of edge servers, cloud centers, and edge devices based on task co-scheduling. Sharif et al. [39] assigned different priorities to different tasks, performed priority-based task scheduling and resource allocation according to the urgency of the task, and decided the priority of each task. Zhou et al. [40] investigated the joint impact of task prioritization and mobile computing services on MEC networks by measuring system performance through the computational utility of multiple users, where the effect of a wide range of task prioritization was considered. A DRL algorithm was used to learn practical solutions through continuous interactions between the AGENT and the system environment. Guo et al. [41] optimize system bandwidth and computational power resources based on federated learning and DRL to ensure that higher priority tasks are

allocated higher bandwidth and computational resources. Wang et al. [42] identified the task type as energy sensitive task. They adopted the idea and method of the matching game to task offloading and matching problems based on minimizing energy consumption. Additionally, there are some papers that do not take into account task prioritization or the nature of the task, for example, Chen et al. [33] studied the QoS-aware computation offloading problem for IoT devices in low-orbit satellite edge computing based on non-cooperative competition among IoT devices, they proposed a distributed QoS-aware computation offloading algorithm to improve the QoS of IoT devices.

As shown in Table 1 below, we have integrated some of the literature and explored their research methods, in which the modeling algorithms in the deep reinforcement learning/deep learning category have speedy and accurate reasoning capabilities after training is completed and deployed. On the one hand, the time complexity of the training process is very high, which makes them hard to train. On the other hand, its compelling solution space exploration capability performs well for complex scenarios and dynamic scheduling and offloading problems.

System model and problem description

In this section, we established the system architecture model for Sat-MEC scenarios. Subsequently, we delineated the communication conditions between satellites and ground stations, followed by developing the communication, task queue, and task computation models. Our proposed DRL-based approach is used to address the problem at hand (Table 2).

As shown in Fig. 1, in our work, we consider the Sat-MEC scenarios with a terrestrial satellite terminal (TST), multiple terrestrial IMDs, and an over-the-top satellite during the current time slot. Each LEO satellite, equipped with MEC server, provides computing capability for task computation. The TST serves as the access point for several IMDs, supporting TST-satellite link transmission on the Ka-band and achieving small cell coverage to facilitate the IMD-TST link on the C-band, which is divided into orthogonal sub-carriers.

We assume that the tasks generated by each IMD on the ground terminal within a timestamp are related. The tasks of the same IMDs are intelligently offloaded to the same Sat-MEC server. When tasks start executing, terrestrial IMDs often face the problem of limitations in computing power and electrical power, especially in those scenarios where satellite access is required. In some harsh environments, for example, the electrical power and computing resources of IMDs are very scarce and valuable. Therefore, we only allow terrestrial IMDs to perform partial task computation within the electrical power constraint, and the remaining tasks will be sent via the Ka-band to the over-the-top satellite at that moment for further processing. We use a prior hyperparameter θ to simulate the offloading and local execution ratio under different environments. The computation scheduling process consists of two stages. In the first stage, corresponding to the ground segment, IMD offloads tasks to the TST through OFDMA. After collecting the tasks from the relevant IMDs, the TST equipped with an independent antenna aperture offloads tasks to the over-the-top LEO satellite in the space segment. Those tasks that have yet to be offloaded to the MEC server are executed locally by IMDs. In addition, in the second stage, for each LEO satellite, we regard it as an agent with a data forwarding function. In the scenarios of this paper, we assume that when there are multiple LEO satellites within the TST line-of-sight transmission range, only the closest satellite is selected for data transmission. Due to ISL between LEO satellites, the tasks received by the over-the-top LEO satellite can be forwarded to the neighboring LEO satellites through ISL for cooperative processing. Therefore, the MEC servers deployed on these LEO satellites can simultaneously perform computational tasks from the ground-based IMD through our scheduling algorithms deployed at the agent level.

Communication model

Due to the obstruction of the Earth and its atmosphere on the satellite-terrestrial link, communication is not always possible. Considering that the offloading action of terrestrial tasks must be established in a communicable environment, the communication model will be discussed first.

The establishment of effective satellite communication links depends on two critical factors. The first factor is the unobstructed line-of-sight visibility between LEO satellites or between LEO satellites and ground stations. The second factor is sufficient transmission power for satellite communication or between satellites and ground stations.

Line-of-sight visibility

The line-of-sight visibility between communication satellites depends on the relative geometric position between the satellites and the Earth, as communication can only occur within the line-of-sight range. If the Earth blocks the line of sight between two satellites, their communication link is considered unavailable. Two satellites orbiting around the Earth, regardless of whether they are in the same or different orbits, can only communicate with each other when they are both above a horizontal plane tangent to the Earth's surface. The critical condition occurs when the connecting line between the two satellites is tangent to the Earth's surface (Fig. 2).

To perform line-of-sight visibility analysis, two critical angles, α_1 and α_2, can be defined as follows:

$$\alpha_1 = \arccos(R_e/r_{1L}), \tag{1}$$

Table 1 Related works comparison

Method	Research Area	Advantages	Drawbacks	Time/Space
Deep Learning	[19–21].	The method can learn and optimize complex strategies	The implementation of the method requires large amounts of data and computational resources and poor interpretability.	High/High
Deep Reinforcement Learning	[33, 34, 43].	The method can learn and optimize complex strategies; it is suitable for tasks that are not explicitly labeled.	The implementation of the method requires large amounts of data and computational resources and poor interpretability.	High/High
Swarm Intelligence Algorithm	[17, 29, 30].	The method can find approximate solutions to complex optimization problems; it has good robustness and adaptability.	The method may fall into a local optimum; when solving complex problems, specific parameters need to be set manually to obtain a better strategy.	Mid/Low
Mathematical Optimization	[16, 18, 44].	The method exists efficiently for convex optimization problems and provides deterministic and theoretical guarantees.	The method can be very complex when dealing with high-dimensional and non-convex optimization problems; the solution may fall into local optimal solutions for complex problems.	Low/Mid
Dynamic Game Theory	[15, 45, 46].	The algorithms model the interactions between multiple decision makers; equilibrium concepts (e.g., Nash equilibrium) are provided to predict the players' strategic choices under certain conditions.	The method is more complicated to find certain theoretical game equilibria in certain complex situations; it is still a challenge to deal with time-varying strategies and decisions in context.	Mid/Low
Greedy Strategy	[26, 47, 48].	The method is simple and easy to implement; optimal solutions can be obtained for some problems.	The method can only obtain approximate solutions for many problems; it is easy to fall into local optimization.	Low/Low
Lyapunov Method	[28, 31, 49].	The method provides an explicit and rigorous determination of the stability of a system; it applies to both linear and nonlinear systems; it is commonly used to stabilize target queues.	The method can be challenging to find suitable Lyapunov functions, proving the existence of upper bounds on Lyapunov drift. The method may be inapplicable or very difficult for some highly nonlinear or complex systems.	Mid/Low

Table 2 Symbol interpretation

α_1 and α_2	two critical angles	R_e	Earth radius
(τ_1, γ_1)	the longitude and latitude	(τ_2, γ_2)	the geodetic coordinates
d_c	the inter-satellite distance	P_r	the received signal power
d_i	the input data size of task$_i$	λ_m	IMD task arrival rate
θ	the offloading rate(a prior parameter)	λ_{Sat}	Sat-MEC server initial task arrival rate
γ	the discount factor	L_s	the signal path loss
Φ	tasks scheduling policy	χ^j	the environment's state
Φ^*	tasks scheduling policy	χ'	the next environment state
B_0	the bandwidth on C-band	σ^2	Gaussian white noise power
f^m	computational capacity of IMD	$w(\chi, a)$	the utility function
$V(\chi)$	the optimal value of the state χ	B_{TST}	the bandwidth on the Ka-band
c_i	the number of CPU cycles for task$_i$ computation	f^s	the number of cpu cycles per second
EIRP	equivalent isotropically radiated power	G_r	the gain of satellite receiving antenna
b_i^{up} and b_i^{down}	the data size of task$_i$ uplink and downlink	$V(\chi, \Phi)$	the state value function of the Agent
S_{TST}	the signal power from TST to satellite	G_t	the gain of the satellite transmitting antenna
Q_s	the initial task queue backlog at current time	T_i^{up} and T_i^{down}	transmission time for task$_i$ uplink and downlink
$\Phi_{(a)}$	the corresponding tasks scheduling methods	a_n(t)	the action of task$_i$ being scheduled to satellite s.
f^s	the number of cpu cycles per second for processing tasks on LEO satellite s	N_{TST}	the interference experienced by the over-the-top satellite on the sub-carrier

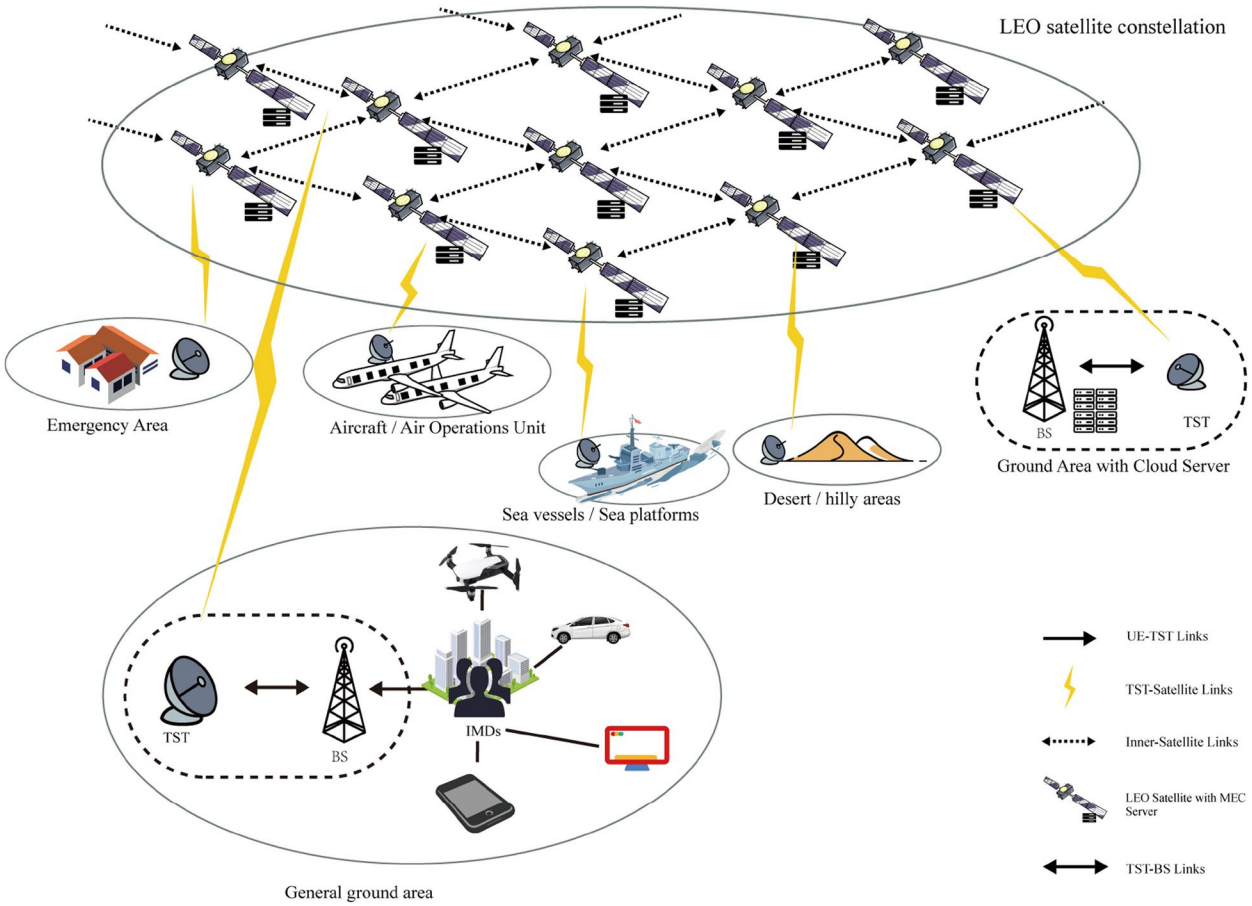

Fig. 1 Sat-MEC system architecture

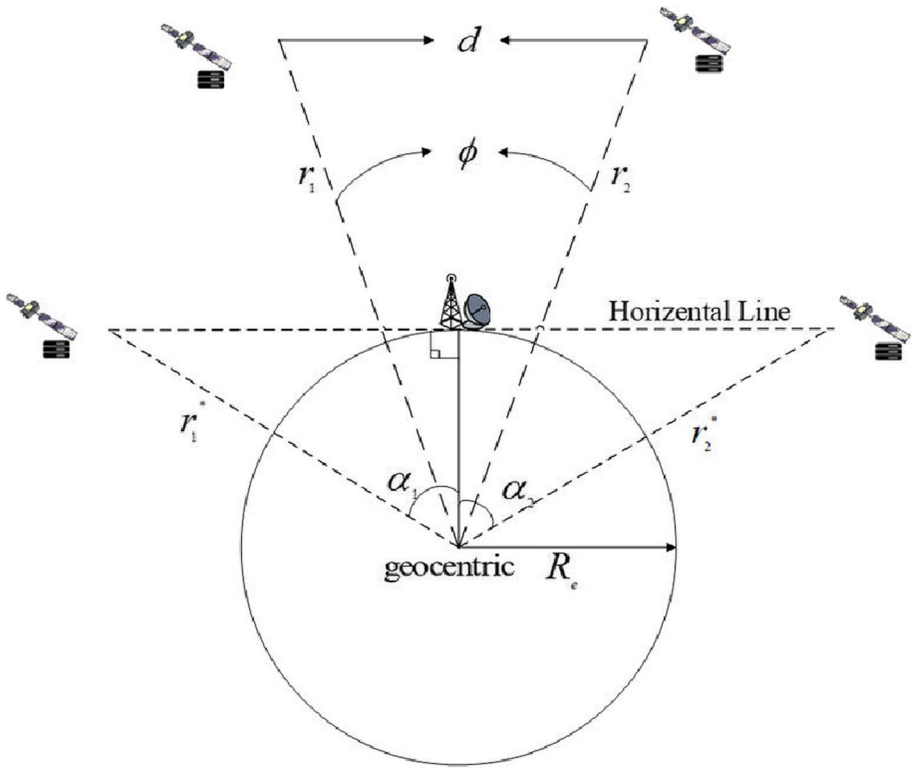

Fig. 2 Link transport capability evaluation-1

$$\alpha_2 = \arccos(\mathrm{R_e}/\mathrm{r_{2L}}), \tag{2}$$

where R_e represents the Earth's radius, r_{1L} and r_{2L} denote the respective distances between the Earth's center and the two satellites as they simultaneously pass through the tangent horizontal plane. The angle between the lines connecting the two communication satellites and the Earth's center can be denoted as ϕ_1:

$$\phi_1 = \arccos \quad \left(\left(\mathrm{r_1^2} + \mathrm{r_2^2} - \mathrm{d_c^2}\right)/2 * \mathrm{r_1} * \mathrm{r_2}\right). \tag{3}$$

The spatial coordinates of the two LEO satellites can be calculated by utilizing (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2). The calculated distances from the two LEO satellites to the Earth's center are r_1 and r_2, respectively. The inter-satellite distance is denoted as d_c, and the visibility function determining whether the two LEO satellites are mutually visible can be expressed as:

$$\varphi_1 = \alpha_1 + \alpha_2 - \phi_1, \tag{4}$$

The fact that φ_1 is present indicates that the two LEO satellites are visible and meet the line-of-sight visibility requirement for link communication. Otherwise, it signifies they are not visible.

Figure 3 illustrates the analysis of the satellite-to-ground link, where the distance from the satellite to the center of the Earth can be determined through the calculation of the satellite and Earth center coordinates. The negative impact of terrain, ground objects, and ground noise on effective communication cannot be established when the antenna elevation angle is zero, according to empirical evidence. Moreover, the minimum elevation angle required for effective communication can vary significantly among different Earth stations due to their location, topography, and environmental factors. As a consequence, the geographical region delimited by the boundary line defined by the antenna's minimum elevation angle ξ is commonly referred to as the communication coverage area of the satellite. The maximum angle of visibility α_1 can be expressed as:

$$\alpha_1 = 90^\circ - \xi - \arcsin\left((R_e/r_1)\cos\xi\right). \tag{5}$$

Given the longitude and latitude (τ_1, γ_1) of a ground station and the orbital six elements, the geodetic coordinates (τ_2, γ_2) of the sub-satellite point can be calculated for a certain time. Based on this, the angle ϕ_2 between the line connecting the satellite and the center of the Earth and the line connecting the center of the Earth and the ground station can be donated as:

$$\phi_2 = \arccos[\cos(\tau_2 - \tau_1)\cos\gamma_1\cos\gamma_2 + \sin\gamma_1\sin\gamma_2] \tag{6}$$

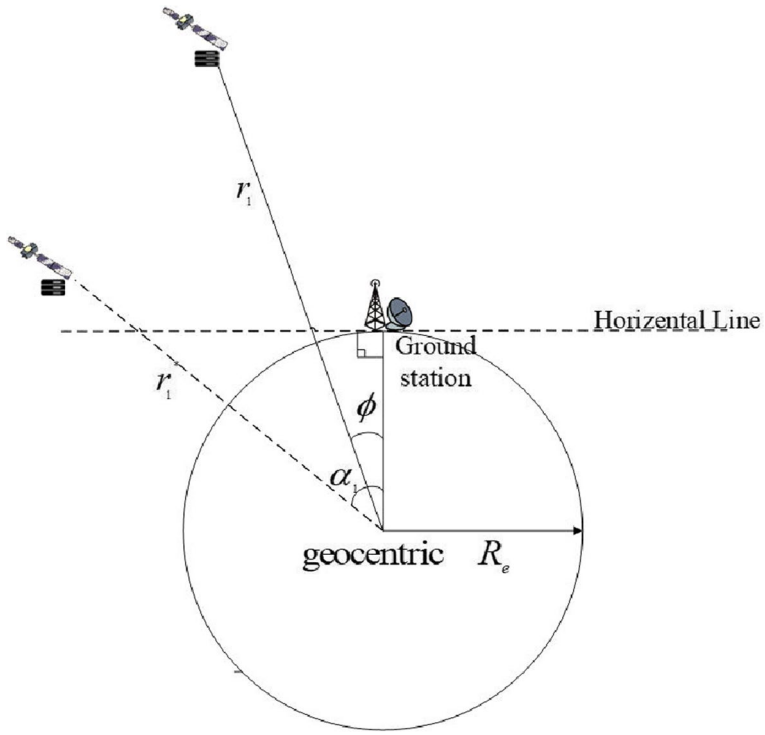

Fig. 3 Link transport capability evaluation-2

The visibility function of the satellite-to-ground link is expressed as:

$$\varphi_2 = 2(\alpha_1 - \phi_2) \tag{7}$$

For the ground station, when φ_2 >0 indicates the satellite is visible from the station. The instance of $\varphi_2 = 0$ signifies either the satellite's rise or set from time, with a change from negative to positive indicating rise and vice versa indicating set. Based on this, the visibility time of a LEO satellite to the ground station can be calculated, and by performing similar calculations for each satellite in the constellation, the coverage time of LEO satellites to the ground station can be obtained. In the Sat-MEC scenarios, we give the definition of an over-the top satellite: a satellite that establishes communication with the TST at the current moment. There are three scenarios in which the TST establishes communication with the over-the-top satellite. 1. The TST is covered by the service range of only one LEO satellite, then the TST establishes communication with this satellite. 2. The TST is covered by several LEO satellites, and we choose the LEO satellite that is closest to the TST. 3. The ground TST is covered by multiple satellites, and the ground TST is the closest to multiple LEO satellites at an equal radius. At this time, we can calculate the coverage time (service time) of the satellites and select the LEO satellite with the longest coverage time (service time) for communication.

Transmission power passability

Establishing a communication link is necessary to have line-of-sight visibility. The power requirements for signal transmission and reception must be met by the distance between inter-satellites or between satellites and ground stations. Suppose the distance is too great. Even if line-of-sight visibility exists between them, the signal loss from satellite transmission may be too significant for the receiving antenna to pick up the signal correctly, thus rendering the communication impossible. To demonstrate, consider inter-satellite links as an illustration. The free-space electromagnetic wave propagation model is the basic model for the inter-satellite link channel, where the received signal power by the satellite antenna can be expressed as:

$$P_r = EIRP + G_r - L_s - L_{As}(dBW), \tag{8}$$

$$EIRP = P_t(dBW) + G_t(dB), \tag{9}$$

Where G_t is the gain of the satellite transmitting antenna in the direction of the communication satellite, P_t is the signal power emitted by the antenna, EIRP is the effective omni-directional radiated power of the satellite transmitting system, and G_r is the gain of the satellite receiving antenna in the direction of the communication satellite. L_{As} is the signal atmospheric loss between the links, L_s is the signal path loss, so the free space propagation loss formula [50]:

$$L_s = 32.45 + 20\log d_c + 20\log f, \tag{10}$$

where f denotes the operating frequency of the communication signal, measured in GHz, d_c refers to the distance between the communication satellites, measured in kilometers. r_1 and r_2 represent the distances from the two satellites to the center of the earth, measured in km, which can be determined based on the three-dimensional coordinates of the two satellites. The angle between the line connecting the two communication satellites and the earth's center can be calculated using Eq. (6), which denotes ϕ_1. Equations (8), (9), and (10) demonstrate that, under constant gain of the satellite receiving antenna and system losses, as the propagation distance and frequency increase, the propagation path loss will also increase, leading to a rapid reduction in received power. Satellite communication requires that the received signal power P_r exceed the sensitivity of the receiver P_{rmin}, which can be expressed as:

$$P_r \geqslant P_{rmin}(dBW) \tag{11}$$

Data transmission model

Tasks generated by ground IMDs cannot be communicated directly with LEO satellites due to different frequency bands. Ground IMDs need to rely on the TST to offload data to the over-the-top satellite at the current time. Ground IMDs have three stages for offloading and scheduling tasks:

(1) IMDs to the TST data transmission During the first stage of computational offloading, the data transmission rate of the task from IMD to TST can be expressed through Shannon's formula [51] as below:

$$r_i^{IMD-TST} = B_0 \log_2\left(1 + \frac{S_i}{N_0}\right), \tag{12}$$

where S_i is signal power from IMD_i to TST on sub-carrier k, B_0 is the bandwidth of each sub-carrier on C-band, and σ^2 is the additive Gaussian white noise power.

(2) TST to over-the-top LEO satellite data transmission In this stage, the data transmission rate of the task generated by IMD_i from the TST to the over-the-top satellite can be expressed by Shannon's formula as:

$$r_i^{TST-Sat} = B_{TST} \log_2\left(1 + \frac{S_{TST}}{N_{TST}}\right), \tag{13}$$

where S_{TST} is the signal power from TST to satellite, B_{TST} is the bandwidth of each sub-carrier on the Ka-band. N_{TST} is the interference experienced by the over-the-top satellite on the sub-carrier .When tasks are offloading in the Ka-band, the antenna of TST usually has good directivity. Therefore, TST can ensure low off-axis antenna gain and tolerate co-channel interference when it chooses an over-the-top satellite to offload tasks [52].

(3) Scheduling by ISL The ISL for LEO satellites utilizes ka-band point beam inter-satellite antennas with 4 point beams per satellite. The link between two satellites is established by scanning and aligning the point beams. In this work, a mesh link is used for the space network of the LEO satellite constellation, and the mesh link allocation method is to establish four links for each satellite, with two satellites in the same orbit and one satellite in each of two adjacent different orbits, Each LEO satellite is an agent,the ISL shown in Fig. 4 below.

ISL uses point-beam inter-satellite antennas, with each point-beam antenna employing TDMA for data transmission. We still use electromagnetic waves for communication modeling of ISL, and the transmission rate of the ISL link is given by the Shannon formula:

$$r_i^{Sat-Sat^*} = B_{Sat} \log_2\left(1 + \frac{S_{Sat}}{N_{Sat}}\right). \tag{14}$$

Task queue model

In the scenarios of the Sat-MEC, we considered two distinct task queues, including the task queue designated for IMDs and the initial task queue for Sat-MEC servers.

In this paper, tasks are generated by IMDs. Our approach involves a stochastic task arrival model, in which only a few tasks arrive at this moment and the number of arrivals follows a Poisson distribution [53]. However, considering the correlation of tasks generated by the same IMD, we assume that tasks generated by the same IMD can only be offloaded to the same Sat-MEC server. We, therefore, consider the same IMD-generated offloading task as a whole task when it is scheduled in Sat-MEC's agent.

We let task_i denote the tasks arriving at IMD_i at the current moment, as we defined above, treat the tasks generated by the same IMD as a whole task for scheduling in Sat-MEC, and represent the whole task generated by IMD_ias a 3-tuple set $\{t, d_i, c_i\}$ denotes the time slot when task$i$ arrives [42]. d_i is the input data size of Task_i [54], which is independently generated and satisfies a random distribution with the arriving rate λ_m, a practical constraint of the problem, especially for those delay-sensitive tasks. In addition, c_i is the number of CPU cycles needed to process input data bits and assume it obeys a random distribution within a specific range [55, 56], which can better represent the heterogeneity of the taks. Note that the system controller quickly obtains d_i and c_i. In addition, we consider

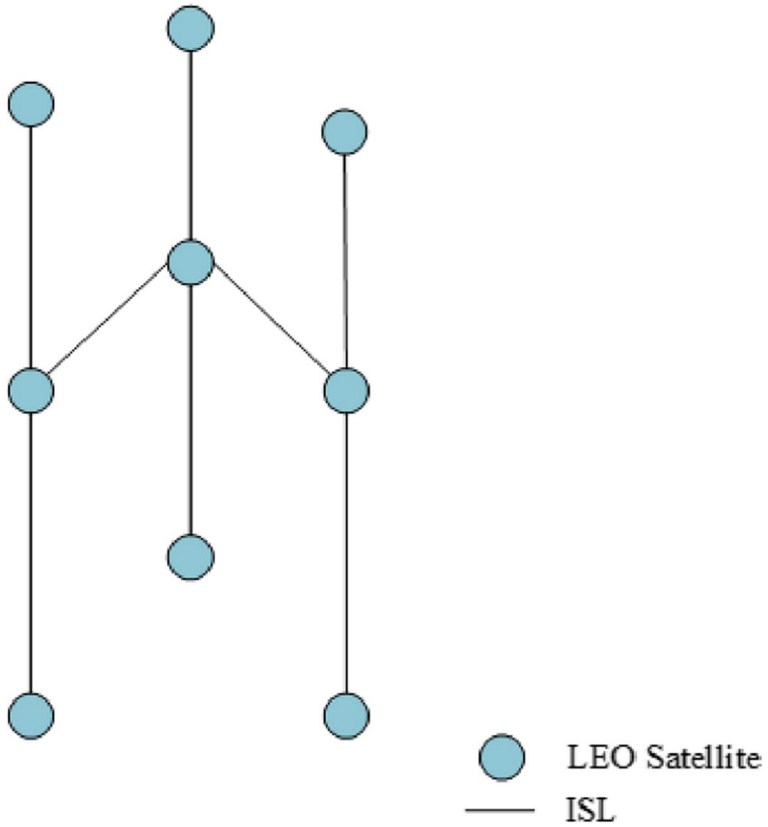

Fig. 4 The construction of ISL between LEO satellites

the initial task computing queue of the Sat-MEC servers. When tasks are offloaded to LEO satellites at time *t*, some satellites may already have tasks in their task computation queue, and there is an initial task computing queue backlog for LEO satellites [57]. We use a Poisson distribution with an immediate arrival rate of λ_{Sat} to simulate the initial queue backlog for each LEO satellite Sat-MEC server.

Notably, task transmission waiting time and Sat-MEC server waiting time occur during the transmission of tasks by ISL and during the processing of tasks on the Sat-MEC server, which we simulate in detail below.

As shown in Fig. 5 above, tasks are sent from an over-the-top LEO satellite via ISL to a target LEO satellite for task processing. In the above Fig. 5 example, three tasks are offloading from the over-the-top LEO satellite MEC server to the Sat*-MEC server in order: Task1, Task2, and Task3. The following equation gives their required offloading time:

$$
\begin{aligned}
T_{Task_1+Task_2+Task_3} = & \frac{\theta d_1}{R_i^{Sat-Sat*}} + \frac{\theta c_1}{f_{Sat^*}} + \frac{\theta d_1}{R_i^{Sat-Sat^*}} + \frac{\theta d_2}{R_i^{Sat-Sat^*}} + \left(\frac{\theta c_1}{f_{Sat^*}} - \frac{\theta d_2}{R_i^{Sat-Sat^*}}\right)^+ \\
& + \frac{\theta c_2}{f_{Sat^*}} + \frac{\theta d_1}{R_i^{Sat-Sat^*}} + \frac{\theta d_2}{R_i^{Sat-Sat^*}} + \frac{\theta d_3}{R_i^{Sat-Sat^*}} + \left\{\left(\left(\frac{\theta c_1}{f_{Sat^*}} - \frac{\theta d_2}{R_i^{Sat-Sat^*}}\right)^+ \right.\right. \\
& \left.\left. + \frac{\theta c_2}{f_{Sat^*}} - \frac{\theta d_3}{R_i^{Sat-Sat^*}}\right)^+ + \frac{\theta c_3}{f_{Sat^*}}\right\}.
\end{aligned} \tag{15}
$$

In the equation above, we compute an instance where three tasks are delegated to the Sat*-MEC server. Here, θ represents the offloading ratio, denoting the proportion of tasks offloaded. In this particular case, we presume the initial task computation queue of the Sat*-MEC server to be empty. The notation $()^+$ implies that if the value within the parentheses falls below 0, it should be considered 0.

In our proposed scenario, we introduce an algorithm for concurrently scheduling decisions for multiple tasks. The

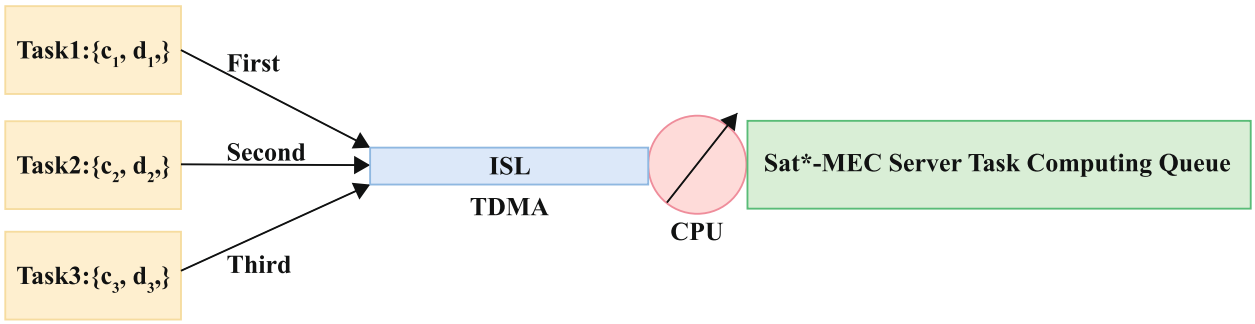

Fig. 5 Tasks processing waiting model

scheduling decision involves determining the sequence of tasks to be scheduled to the same server. As an example, Task 2 is scheduled to be the second task scheduled to the Sat*-MEC server. Before it can be transmitted, it needs to wait for Task 1. Upon Task 2's arrival at the Sat*-MEC server and assuming an initial task computation server queue backlog of zero, the waiting duration for Task 2 at the Sat*-MEC server is calculated to be $\left(\frac{\theta c_1}{f_{Sat*}} - \frac{\theta d_2}{R_i^{Sat-Sat*}}\right)^+$.

The objective of our work here is to show the reader that in the case of scheduling multiple tasks to the server at the same time, especially in the transmission mode of TDMA, the scheduling order of the multi-task scheduling server affects the average processing time of the tasks due to the heterogeneity of the tasks. Therefore, when we make decisions on task scheduling, we need to consider not only the characteristics of the current task and the characteristics of the servers, but also the characteristics of other tasks arriving at the same time, and only by considering the characteristics of multiple servers and multiple tasks arriving at the same time can we theoretically realize the optimal scheduling solution.

Here we define the time expended, excluding task transmission time and computation time, during the execution of task$i$ as T_i^{waste}, where $T_i^{waste} = \left(\frac{\theta c_1}{f_{Sat*}} - \frac{\theta d_2}{R_i^{Sat-Sat*}}\right)^+$. In our work, the total goal is to minimize the average task scheduling time, therefore, it is very important for our ultimate objective to analyze the characteristics of multi-tasking and multi-Sat-MEC servers simultaneously.

Task computing model

Due to the heterogeneity of the ground environment, when the terminal IMDs are in a city or a region with sufficient power and computing capability, tasks should primarily rely on local execution; however, when the terminal device is in a desert, hilly, or natural disaster area, the harsh environment of the terminal region, which makes the tasks more processed on the Sat-MEC server. The ratio of tasks offloaded to the LEO is set as the task offloading rate θ,θ is a priori hyper-parameter, which we aim to model the differences in task scheduling in different situations by varying its value in our work. When the IMD computation capability is lacking, the task prefers to offload to the Sat-MEC server. In some work [58], solar-wind hybrid energy system is utilized in non-urban areas to generate electricity to feed the IMD and TST to ensure that their power is available for transmitting data, Our work is still more in the search for an algorithm with high exploratory capability in a high-dimensional dynamic feature space, focusing on the problem of average latency of tasks and realizing an optimal scheduling algorithm. Therefore, we assume in our work that IMD and TST will not fail to work due to lack of electrical energy. In this work, the processing tasks include two cases: non-offloaded local computation and offloaded Sat-MEC computation.

Local computing:

F^m denotes the maximum computational capacity of IMD and f^m denotes the number of CPU cycles per second for processing tasks on IMD, non-offloading subtasks of task$_i$ executed on IMD with task CPU cycles needed $(1-\theta)\cdot c_i$. For each task, the processing latency incurred on IMD, which is calculated in Eq. (16):

$$T_i^{loc} = \frac{(1-\theta)c_i}{f^m} \tag{16}$$

Sat-MEC computing:

The offloaded tasks are received by the over-the-top satellite in the current region and further distributed by ISL to other adjacent LEO satellites (in the same or a different orbit) for co-processing according to our proposed scheduling algorithm. f^s denotes the number of CPU cycles per second for processing tasks on LEO satellite s, $\theta \cdot d_i$ denotes the size of the mission data offloaded to the satellite, and $\theta \cdot c_i$ denotes the computing load of task$_i$, which is the necessary central processing unit CPU cycles for executing task$_i$. The processing time of a task$_i$ on a LEO satellite is calculated by:

$$T_i^{sat} = T_i^{up} + T_i^{queue} + T_i^{down}, \tag{17}$$

$$T_i^{up} = \frac{\theta \cdot c_i}{R_i^{IMD-TST}} + \frac{\theta \cdot c_i}{R_i^{TST-Sat}} + \frac{\theta \cdot c_i}{R_i^{Sat-Sat^*}} \tag{18}$$

current moment of IMD. The steps of task offloading execution are divided into task offloading from IMD to over-the-top LEO satellite, over-the-top LEO satellite executing scheduling according to our scheduling policy for selecting LEO satellite for scheduling purposes for task execution, and the waiting and execution time of the task in the Sat*-MEC server, as well as the time of the backhaul.

$$\begin{aligned}(P1) &= \min_{\theta,\mathbf{Sat}^*} \sum_{i=1}^{\mathcal{I}} \max\left(\mathrm{T_i^{loc}}, \mathrm{T_i^{sat}}\right) \\ &= \min_{\theta,\mathbf{Sat}^*} \sum_{i=1}^{\mathcal{I}} \max\left(\mathrm{T_i^{loc}}, \mathrm{T_i^{up}} + \mathrm{T_i^{queue}} + \mathrm{T_i^{down}}\right) \\ &= \min_{\theta,\mathbf{Sat}^*} \sum_{i=1}^{\mathcal{I}} \max\left(\left(\frac{(1-\theta)\mathrm{c_i}}{\mathrm{f^m}}, \frac{\mathrm{b_i^{up}}}{\mathrm{R_i^{IMD-TST}}} + \frac{\mathrm{b_i^{up}}}{\mathrm{R_i^{TST-Sat}}} + \frac{\mathrm{b_i^{up}}}{\mathrm{R_i^{Sat-Sat^*}}} + \frac{\mathrm{Q_s} + \theta \cdot \mathrm{c_i}}{\mathrm{f^s}} + \mathrm{T_i^{waste}}\right)\right)\end{aligned} \tag{21a}$$

$$T_i^{down} = \frac{b_i^{down}}{R_i^{IMD-TST}} + \frac{b_i^{down}}{R_i^{TST-Sat}} + \frac{b_i^{down}}{R_i^{Sat-Sat^*}}, \tag{19}$$

$$T_i^{queue} = \frac{Q_s + \theta \cdot c_i}{f^s}, \tag{20}$$

where T_i^{up} and T_i^{down} denote transmission time for task$_i$ uplink and downlink. let T_i^{queue} denote the waiting and processing time for task$_i$ in the Sat-MEC task computation queue, Q_s is the initial task queue backlog at the current time in the Sat-MEC server of satellite s, and f_s donates CPU clock frequency of LEO satellite s. b_i^{up} and b_i^{down} denote data size of task$_i$ uplink and downlink , where we assume $b_i^{up} = d_i$. $R_i^{IMD-TST}$, $R_i^{TST-Sat}$, $R_i^{Sat-Sat^*}$ denote the transmission rates from the IMDs to the TST, the TST to the overhead satellite, and the over-the-top satellite to the target satellite respectively. Sat^* donates target satellite,which means the satellite which the task transfers through ISL and finally reaches, the task will be offloaded and compute on Sat^*-MEC server.

$$s.t. f_m \in (0, F_m] \tag{21b}$$

$$s.t. \theta \in (0, 1] \tag{21c}$$

$$Sat^* \in [Sat_0, Sat_1, \cdots, Sat_s] \tag{21d}$$

Since the above optimization problem in (P1) is non-convex and NP-hard, we use a DRL-based approach to achieve a feasible solution. In the next section, we model the formulated optimization problem as a MDP problem [59], where the action selection aims to maximize the reward function. In the Sat-MEC scenario, the over-the-top satellite acts as an agent to select an action to schedule tasks and then receive a reward at time slot t. The state space, action space, and reward function will described in next section.

Problem description

In the proposed Sat-MEC Scenario, our objective is to minimize the average tasks processing time of all IMDs generated. According to the Communication Model and the Task Computing Model, the optimization problem can be formulated as (P1), where T_i^{loc} denotes the time when the task$_i$ is executed locally. T_i^{sat} denotes the time when the task is offloaded from the local via TST to the over-the-top satellite and execute at Sat*-MEC at the current moment. The scheduling decision is executed via the over-the-top satellite, and the task is processed in the $Sat*$-MEC server. Therefore, for each task$_i$ generated by IMD$_i$, we take the maximum value of local execution and task offload execution as the task processing time at the

Methods

State space

In this paper, as depicted in Fig. 6, the system controller is installed at the broker level and is responsible for the communication and coordination between the Sat-MEC servers. It receives task requests from the ground and analyzes other neighboring Sat-MEC servers' resource availability and computational capacity. Integrating the DRL into the broker can enhance decision-making and optimize the task schedule.

We set the sensorial information of the over-the-top LEO satellite at moment t as the state $S_n(t) \in S$. The components of $S_n(t)$ including tasks state and Sat-MEC servers state, the task state indicated by the data size of the task and the number of CPU cycles required for the task computation, which received from the TST, the Sat-MEC server states indicated by Sat-MEC server computation capacity and Sat-MEC server initial task computation queue backlog, that information could receive from Control Channel

and make scheduling decision at over-the-top LEO satellite as shown in Fig. 6. We have the task state matrix and the Sat-MEC server state matrix below for further discussion.

$$S(t) = \left\{ \mathbf{S(t)^{Task}}, \mathbf{S(t)^{Sat}} \right\} \tag{22}$$

$$S(t)^{Task} = \begin{pmatrix} S(t)_1^{Task} \\ S(t)_2^{Task} \\ \vdots \\ S(t)_i^{Task} \\ \vdots \\ S(t)_n^{Task} \end{pmatrix} = \begin{pmatrix} d(t)_1, c(t)_1 \\ d(t)_2, c(t)_2 \\ \vdots \\ d(t)_i, c(t)_i \\ \vdots \\ d(t)_n, c(t)_n \end{pmatrix}_{n\times 2} \tag{23}$$

$$\mathbf{S(t)^{Sat}} = \begin{pmatrix} Sat(t)_1^q, Sat(t)_1^c, Sat(t)_1^{loc}, Sat(t)_1^{trans} \\ Sat(t)_2^q, Sat(t)_2^c, Sat(t)_2^{loc}, Sat(t)_2^{trans} \\ Sat(t)_3^q, Sat(t)_3^c, Sat(t)_3^{loc}, Sat(t)_3^{trans} \\ Sat(t)_4^q, Sat(t)_4^c, Sat(t)_4^{loc}, Sat(t)_4^{trans} \\ Sat(t)_5^q, Sat(t)_5^c, Sat(t)_5^{loc}, Sat(t)_5^{trans} \end{pmatrix}_{5\times 4} \tag{24}$$

The i-th line of $s(t)^{Task}$ is $s(t)_i^{Task}$, which is the characteristic of the arrival task$_i$, $d(t)_i$ is the size of the arrival task i, and $c(t)_i$ is the number of CPU cycles required to compute the task$_i$. The line of $s(t)^{Sat}$ is $s(t)_s^{Sat}$, let s donates the satellite number, where each of them has four feature values. $Sat(t)_s^q$ denotes the initial backlog of the task computation queue at time t for $Sat(t)_s$, $Sat(t)_s^c$ denotes the computational capacity of Sat-MEC server n (number of CPU cycles/second), $Sat(t)_s^{loc}$ denotes the euclidean distance of satellite n from the receiving satellite (over-the-top satellite), $Sat(t)_s^{trans}$ denotes the channel capacity of the ISL transmission between satellite n and over-the-top satellite.

Action space

Based on the current moment t, the over-the-top LEO satellite as the agent senses the environment information at the current moment t and processes the tasks from the ground based on the agent scheduling algorithm, choosing tasks to schedule to other satellites connected through the ISL or to processing them at the over-the-top satellite, as the transmission queue shown in Fig. 6. Formally, we define the vector $a_n(\mathrm{t}) = \{\mathrm{x_{si}(t)}, \forall \mathrm{s} \in \mathbf{S}, \forall \mathrm{i} \in \mathbf{N}\}$, which represents the action of task$_i$ being scheduled to satellite s.

Computation task scheduling

In this section, we combine deep reinforcement learning and the self-attention mechanism to form practical and feasible algorithms to approach the optimal task scheduling algorithm using the the self-attention mechanism to represent the Q-network.

The task scheduling policy $\mathbf{\Phi}$ can be defined as : $\mathbf{\Phi} : \mathcal{X} \rightarrow \mathcal{Y}$. More precisely, the Agent identifies an action $\mathbf{\Phi}\left(\chi^j\right) = \Phi_{(a)}\left(\chi^j\right) = a_n(\mathrm{t}) \in \mathcal{Y}$ according to $\mathbf{\Phi}$ after

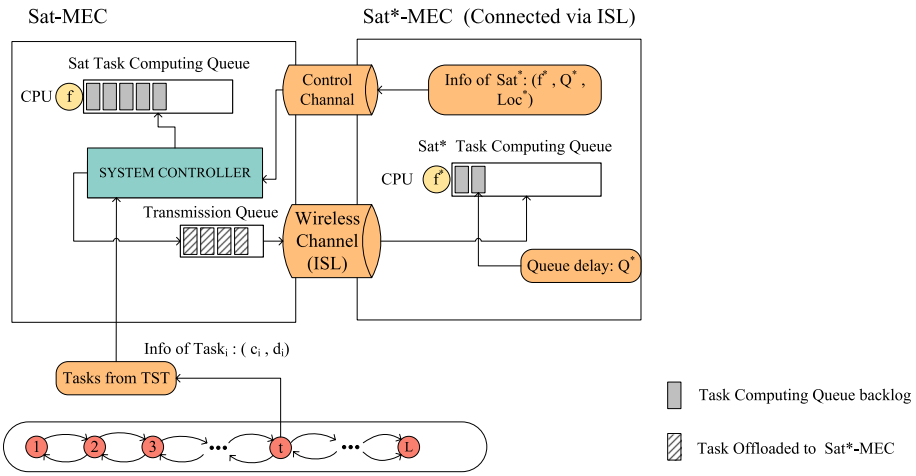

Fig. 6 Tasks scheduling decision process on LEO satellites

observing environment's state $\chi^j \in \mathcal{X}$ at the onset of the scheduling desicion epoch j, where $\mathbf{\Phi} = (\Phi_{(a)})$, with $\Phi_{(a)}$ as the corresponding tasks scheduling methods.

Given the tasks scheduling desicion policy $\mathbf{\Phi}$, the $\{\chi^j : j \in \mathbb{N}_+\}$is a controlled Markov chain characterized by the next enviroment state transition probability:

$$\begin{aligned} Pr\left\{\chi^{j+1} \mid \chi^j, \Phi\left(\chi^j\right)\right\} = & Pr\left\{d(t+1)_i \mid d(t)_i, \mathbf{\Phi}\left(\chi^j\right)\right\} \cdot \Pr\left\{c(t+1)_i \mid c(t)_i, \mathbf{\Phi}\left(\chi^j\right)\right\} \\ & \cdot \prod_{\mathrm{n}\in\mathrm{N}} Pr\left\{\mathrm{Sat(t+1)_n^q} \mid \mathrm{Sat(t)_n^q}, \mathbf{\Phi}\left(\chi^j\right)\right\} \\ & \cdot Pr\left\{\mathrm{Sat(t+1)_n^c} \mid \mathrm{Sat(t)_n^c}, \mathbf{\Phi}\left(\chi^j\right)\right\} \\ & \cdot Pr\left\{\mathrm{Sat(t+1)_n^{trans}} \mid \mathrm{Sat(t)_n^{trans}}, \mathbf{\Phi}\left(\chi^j\right)\right\} \\ & \cdot Pr\left\{\mathrm{Sat(t+1)_n^{loc}} \mid \mathrm{Sat(t)_n^{loc}}, \mathbf{\Phi}\left(\chi^j\right)\right\} \end{aligned} \tag{25}$$

Moreover, we establish the utility linked to each epoch. $\{w(\chi^j, \Phi(\chi^j)) : j \in \mathbb{N}_+\}$over the series of environment states$\{\chi^j : j \in \mathbb{N}_+\}$, The Agent's anticipated utility over the extended duration, given the initial environment state, χ^1 could be formulated as follows.

$$V(\chi, \mathbf{\Phi}) = \mathrm{E}_\Phi\left[(1-\gamma)\cdot\sum_{\mathrm{j}=1}^{\infty}(\gamma)^{\mathrm{j}-1}\cdot \mathrm{w}\left(\chi^j, \Phi\left(\chi^j\right)\right) \mid \chi^1 = \chi\right], \tag{26}$$

Here, we denote the environment state as $\chi = S(t) \in \mathcal{X}$, the discount factor as $\gamma \in [0,1)$, and $(\gamma)^{j-1}$ represents the discount factor of the $(j-1)$th order. The function $V(\chi, \Phi)$ is also referred to as the state value function of the Agent, corresponding to environment state χ under strategy Φ.

The objective of the agent is to develop a task scheduling methods. $\Phi^* = \Phi(a)^*$, which Optimal the extended-term utility $V(\chi, \Phi)$ for any starting environment state χ, leading to the following formalization:

$$\mathbf{\Phi}^* = \arg\max_{\Phi} V(\chi, \mathbf{\Phi}), \forall \chi \in \mathcal{X}. \tag{27}$$

The function $V(\chi)$ represents the optimal value of the state χ under the policy Φ^*. This function applies to all environment states χ belonging to the set $\mathcal{X}$.

The optimal method to achieve the environment state value function can be derived by solving the Bellman equation [60] for:

$$V(\chi) = \max_a\{(1-\gamma)\cdot w(\chi, a) + \gamma \cdot \sum_{\chi'} \Pr\{\chi' \mid \chi, \mathrm{a}\} \cdot \mathrm{V}(\{\chi'\})\} \tag{28}$$

where $w(\chi, a)$ denotes the utility obtained when executing the action a from the current network state χ resulting in the next environment state χ'. Here, $\chi' = S(t+1) \in \mathcal{X}$.

However, the conventional approach to solving the equation above is typically based on value iteration or policy iteration [61], which requires comprehensive knowledge of statistics such as computational task arrivals, initial server queue backlogs, and channel state transitions. We can use a non-policy learning approach which means useing Q values instead of using V values. One advantage of non-policy Q-learning is its agnosticism towards an existing knowledge of environment state transition statistics [61]. $\forall \chi \in \mathcal{X}$, so, the state-value function $V(\chi)$ can be derived directly from

$$V(\chi) = \max_a Q(\chi, a), \tag{29}$$

where

$$Q(\chi, a) = (1-\gamma)\cdot w(\chi, a) + \gamma \cdot \sum_{\chi'} \Pr\{\chi' \mid \chi, \mathrm{a}\} \cdot \mathrm{V}(\chi'). \tag{30}$$

Replacing Eq. (29) in Eq. (28) gives the following:

$$Q(\chi, a) = (1-\gamma)\cdot w(\chi, a) + \gamma \cdot \sum_{\chi'} \Pr\{\chi' \mid \chi, \mathrm{a}\} \cdot \max_{(\mathrm{a}')} \mathrm{Q}(\chi', (\mathrm{a}')). \tag{31}$$

In the above equation, we let $a' \in \mathcal{Y}$ donate the task scheduling action under the environment state χ'. In a practical environment, the number of computed tasks arrival and the number of cpu cycles required for computation per task is not available in advance. By employing the Q-learning technique, the agent endeavors to acquire knowledge about $Q(\chi, a)$, iteratively, based on a review of the environment state $\chi = \chi^j$ at the current decision epoch j, the executed scheduling action $a = a^j$, the utility achieved $w(\chi, a)$, and the environment state χ' obtained at the subsequent epoch $j+1$. The updated rules are as follows:

$$, Q^{j+1}(\chi, a) = Q^j(\chi, a) + \alpha^j\left((1-\gamma)\cdot w(\chi, a) + \gamma \cdot \max_{a'} Q^j\left(\chi', a'\right) - Q^j(\chi, a)\right), \tag{32}$$

where α^j denotes the dynamically adjusting learning rate, it can be observed that Eq. (32) reveals the limited scalability of the traditional Q-learning rule. Given the discrete nature of the Q function representation, Q-learning encounters challenges when applied to high-dimensional scenarios characterized by significantly large network states or action spaces, as the traditional Q-table learning process becomes prohibitively slow. In the scenarios of our work, the composition of the environmental states has a very high dimensionality. As a result, the convergence of the Q-learning process within a fixed number of scheduling decision periods becomes unattainable.

Therefore, we proposed the tasks scheduling method to optimize with a DRL-based framework.

As Fig. 7 illustrates, we use the self-attention mechanism as the Q-network, and the Q-network input is the total number of tokens of tasks and Sat-MEC servers. First, the tasks and servers form a set of tokens by embedding two different kinds of tokens. Then, the self-attention mechanism operation between tokens is performed to output the matching score between tasks and servers, and the selection of task scheduling solution is performed.

For example, if 20 tasks reach the over-the-top satellite at time t, 20 tasks will be offloaded to 5 Sat-MEC servers. Firstly, tasks characteristics are mapped to the token by W1, servers characteristics are mapped to the token by W2, and the 25 tokens mapped into the task-server similarity score matrix are formed by the self-attention mechanism. At this time, the dimension of the similarity matrix should be 25*25*number of channels [62].

Further, the similarity matrix is embedded and downscaled to form a matrix of 25*25*1. In our 25*25*1 matrix, the ith task's destination is the ith row, and the maximum value of the 21st-25th columns is the ith task's destination. When multiple tasks are selected to offload to the same server, the maximum value of the number of rows of tasks corresponding to that server column in the comparison matrix is used as the priority offload, and the offload solution for 20 tasks is output at once.

In addition, inspired by the successful modeling of optimal state action Q-functions using deep neural networks [63], we used a double DQN to solve the large-scale network state space $\mathcal{X}$ [13]. Specifically, the Q function expressed in Eq. (30) is approximated as $Q(\chi, a) \approx Q((\chi, a); \boldsymbol{\lambda})$, where $(\chi, a) \in \mathcal{X} \times \mathcal{Y}$ and $\boldsymbol{\lambda}$ denotes the vector of parameters associated with the DQN.During this time, the DQN parameters $\boldsymbol{\lambda}$ can be learned iteratively rather than finding the optimal Q function. In the Sat-MEC system we are considering, the SATDRL for stochastic computational scheduling is shown in Fig. 7.

It is assumed that the Sat-MEC server employs a replay memory of a limited capacity M for storing past experiences $\mathbf{m}^j = (\boldsymbol{\chi}^j, a^j, w(\boldsymbol{\chi}^j, a^j), \boldsymbol{\chi}^{j+1})$ During the learning process of SATDRL, the transition between two consecutive decision epochs j and $j+1$ involves the occurrence of events that are crucial for the system's experience accumulation. where $(\chi^j, \chi^{j+1}) \in \mathcal{X}$ and $a^j \in \mathcal{Y}$. The collection of

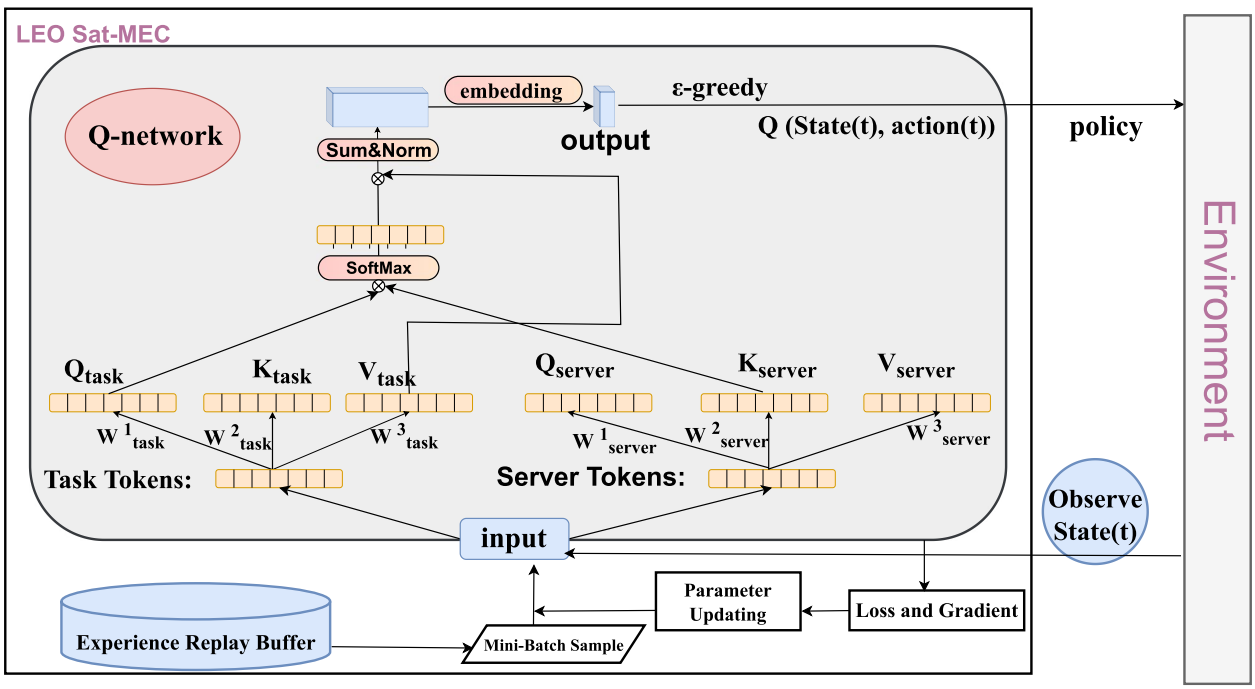

Fig. 7 Our proposed DRL framework in the Sat-MEC scenario

experiences, denoted as $\mathcal{M}^j = \{\mathbf{m}^{j-M+1}, \ldots, \mathbf{m}^j\}$, represents the experience pool. The Agent utilizes both a DQN and a target DQN to optimize its learning process, $Q\left(\chi, a; \boldsymbol{\lambda}^j\right)$ and $Q\left(\chi, a; \boldsymbol{\lambda}^j_{target}\right)$, with parameters $\boldsymbol{\lambda}^j$ at the tasks scheduling decision epoch j and $\boldsymbol{\lambda}^j_{target}$ at a past epoch before decision epoch $j, \forall(\chi, a) \in \mathcal{X} \times \mathcal{Y}$. Based on the experience replay method proposed by [64], the Agent employs a strategy known as mini-batch sampling. During each decision epoch j, the Agent randomly selects a subset $\widetilde{\mathcal{M}}^j \subseteq \mathcal{M}^j$ from the historical experience pool $\mathcal{M}^j$ to perform online training of the DQN. In other words, the parameters $\boldsymbol{\lambda}^j$ are adjusted to minimize the loss function, as specified by Eq. (33), with the condition that $a' \in \mathcal{Y}$.

The loss function $L_{(\mathrm{SATDRL})}(\boldsymbol{\lambda}^j)$ represents the mean-squared error of the Bellman equation at the tasks scheduling decision epoch j. It replaces $Q^j(\chi, a)$ and its corresponding target $(1-\gamma) \cdot w(\chi, a) + \gamma \cdot \max_{a'} Q^j(\chi', a')$ with $Q(\chi, a; \boldsymbol{\lambda}^j)$ and $_{(1-\gamma) \cdot w(\chi, a) + \gamma \cdot Q(\boldsymbol{\chi}', \arg\max_{a'} Q(\boldsymbol{\chi}', a'; \boldsymbol{\lambda}^j); \boldsymbol{\lambda}^j_{\mathrm{target}})}$, respectively.

By computing the derivative of the loss function $L_{(\mathrm{SATDRL})}(\boldsymbol{\lambda}^j)$ in relation to the DQN parameters λ^j, we can derive the gradient following the expression presented in Eq. (34). Algorithm 1 provides a comprehensive overview of the implementation of the SATDRL algorithm by the Agent for the purpose of task scheduling in our proposed Sat-MEC scenarios.

$$L\left(\boldsymbol{\lambda}^j\right) = \mathrm{E}\left[\left((1-\gamma) \cdot \mathrm{w}(\boldsymbol{\chi}, \mathrm{a}) + \gamma \cdot \mathrm{Q}\left(\boldsymbol{\chi}', \arg\max_{\mathrm{a}'} \mathrm{Q}\left(\boldsymbol{\chi}', \mathrm{a}'; \boldsymbol{\lambda}^j\right); \boldsymbol{\lambda}^{\mathrm{j}}_{\mathrm{target}}\right) - \mathrm{Q}\left(\boldsymbol{\chi}, \mathrm{a}; \boldsymbol{\lambda}^{\mathrm{j}}\right)\right)^2\right] \tag{33}$$

1: **Algorithm input:** A *DQN* and a target *DQN* with two sets $\boldsymbol{\lambda}^j$ and $\boldsymbol{\lambda}^j_{target}$ with their random parameters, $j = 1$.
2: **Algorithm initialization:** For $j = 1$, we initialize the replay memory $\mathcal{M}^j$ with a fixed capacity of $M \in \mathbb{N}_+$, and we create a mini-batch $\overline{\mathcal{M}}^j$ of size $\tilde{M} < M$ for performing experience replay.
3: **Environment Interaction:** At the onset of tasks scheduling desicion epoch j, the Agent receives the environment state information $\chi^j \in \mathcal{X}$, including tasks characteristic and Sat-MEC servers characteristic, which is taken as an input to the Q-network (self-attention) with parameters $\boldsymbol{\lambda}^j$, and then chooses a scheduling action $a^j \in \mathcal{Y}$ stochastic with probability ϵ or $a^j = \arg\max_{a \in \mathcal{Y}} Q\left(\chi^j, a\right); \boldsymbol{\lambda}^j\right)$ with probability $1 - \epsilon$.
4: **Policy Execution:** Upon processing the chosen task scheduling action c^j, the Agent experiences an utility $w\left(\chi^j, c^j\right)$ and receives the next environment state information $\chi^{j+1} \in \mathcal{X}$ during the succeeding tasks scheduling decision epoch $j+1$.
5: **Experience Storage:** The Agent stores the latest transition $\mathbf{m}^j = \left(\chi^j, (c^j), w\left(\chi^j, c^j\right), \chi^{j+1}\right)$ in the replay memory $\mathcal{M}^j$ at the system controller of the over-the-top satellite.
6: **Parameter Update:** By randomly selecting a mini-batch of transitions $\overline{\mathcal{M}}^j \subseteq \mathcal{M}^j$ from the replay memory, the Agent updates the DQN parameters $\boldsymbol{\lambda}^j$ the gradient update equation (34).
7: **Target Network Synchronization:** At regular intervals, the Agent resets the target *DQN* parameters as $\boldsymbol{\lambda}^{j+1}_{target} = \boldsymbol{\lambda}^j$, while in other cases, $\boldsymbol{\lambda}^{j+1}_{target} = \boldsymbol{\lambda}^j_{target}$.
8: **Decision Epoch Advancement:** The marks for the task offload decision era are updated via $j \leftarrow j + 1$.
9: **Termination Check:** until our scheduled stop conditions are met.
10: **return:** The Q network parameter $\boldsymbol{\lambda}$ in relation to the agent DQN network.

Algorithm 1 SATDRL algorithm for minimizing average tasks processing time in proposed Sat-MEC framework

$$\nabla^j_{\lambda} L(\lambda^j) = E\left[\left((1-\gamma) \cdot w(\chi, a) + \gamma \cdot Q(\chi', \arg\max_{a'} Q(\chi', a'; \lambda^j); \lambda^j_{target}\right) - Q(\chi, a; \lambda^j)) \cdot \nabla_{\lambda^j} Q(\chi, (c, e); \lambda^j)\right] \tag{34}$$

Experiment results

Experimental settings

In this section, we will evaluate the performance of our proposed algorithm, i.e., SATDRL, in the context of task scheduling. We will also verify the superiority of our proposed algorithm through various experiments. These include a convergence analysis, a comparative analysis with the utility function values of other algorithms, and a discussion on the offloading ratio θ.

We modeled the LEO satellites and the ground IMDs environment using Python and modeled the satellite in STK software. We did this to derive the time-series 3D coordinates of the satellites and to use them as a vehicle for the simulation environment, but not to implement the satellite's functionality, such as orbital dynamics signal fading. In the process of simulation, considering that our approach does not add or change the packet or header information at the network, there is practically no actual medium and protocol stack involved, including the delays brought by the broker approach, such as the time to execute scheduling decisions, the time for protocol conversion, and the time for data transcoding and classification. Therefore, we chose to perform the simulation at the application level without going deeper into the TCP/IP layers or modifying the underlying network parameters. The results of generating packet requests using any network do not differ significantly from the reported results, so we can focus on the scheduling algorithms themselves and the performance and effectiveness of the scheduling algorithms at the application level. In the experimental phase of the simulation, we use the self-attention mechanism to act as a Q-network for extracting the tasks and Sat-MEC servers characters in the high-dimensional space for training the SATDRL better.

Within the Sat-MEC scenario, where terrestrial IMDs generate tasks that can be processed by both local and Sat-MEC servers co-processing, we default to satellites in the same or in different orbits that could connect to the over-the-top satellite via ISL, with other settings as shown in the System Model and described in the Table 3: Simulation Parameters. Simultaneously, we consider variations in the offloading rate, denoted as θ, and the number of IMDs.

To validate the effectiveness and feasibility of our proposed method, we utilized STK software to generate a comprehensive dataset [65], simulating 636 LEO satellites registered under One-Web LEO satellites. This dataset spans over a period of 10 hours, presenting geocentric inertial coordinates within a 3D framework, sampled at a frequency of 0.05Hz. The bandwidth for satellite-ground and inter-satellite communications are 20 and 100 MHz, respectively. ISL links utilize point-beam inter-satellite antennas, with each satellite equipped with four point beams to establish inter-satellite links. ISL communication is carried out through a time division multiple access system [66]. For satellite to satellite communication, using a free-space path loss model (citing the previous free-space loss equation) that models small-scale fading on Ka-band as Rician fading, we assume that the expected overall atmospheric fading due to rainfall, gas fading, cloud fading, and scintillation is 5.2 dB when TST communicates with an over-the-top satellite [67]. The polarization loss and antenna misalignment loss are 0.1 and 0.35 dB, respectively [68] (Fig. 8).

Table 3 Simulation parameters

Parameters	Default Values
Ka-band carrier frequency	30GHz
Number of IMDs	[10,20,30,40,50,60,70]
Number of Sat-MEC server	5
d_i	[0.2-1]MB
c_i	[1-4] Gcycles
f^m	0.3 Gcycles/s
f^s	[10-15] Gcycles/s
N_0	-174dBm
B_0	500MHz
B_{TST}	800MHz
B_{Sat}	100MHz
R_e	6371Km
λ_m	5
λ_{Sat}	8
θ	[0-1]
Replay memory capacities M,N	6000
Mini-batch sizes $\overline{M}$	200
Genetic Algorithm's Population Size	200
Genetic Algorithm's Number of Generations	2000
Genetic Algorithm's Crossover Probability	0.5
Genetic Algorithm's Mutation Probability	0.1

The attributes of our tasks and Sat-MEC servers capture multi-dimensional heterogeneity, which includes diversity in data size of the tasks, variability in the number of CPU cycles required for the task computation, differences in the computational capability of Sat-MEC servers, heterogeneity in the initial backlog of tasks in the Sat-MEC server queue, and irregularity in Sat-MEC temporal information. To address task scheduling decisions in heterogeneous environments, we propose a DRL-based scheduling decision algorithm to minimize the average tasks execution time. To demonstrate the algorithm's adaptability to diverse data, we set attribute values within certain boundaries for task and server feature configurations, as illustrated in the associated table.

For performance comparisons, we simulate three baseline strategies:

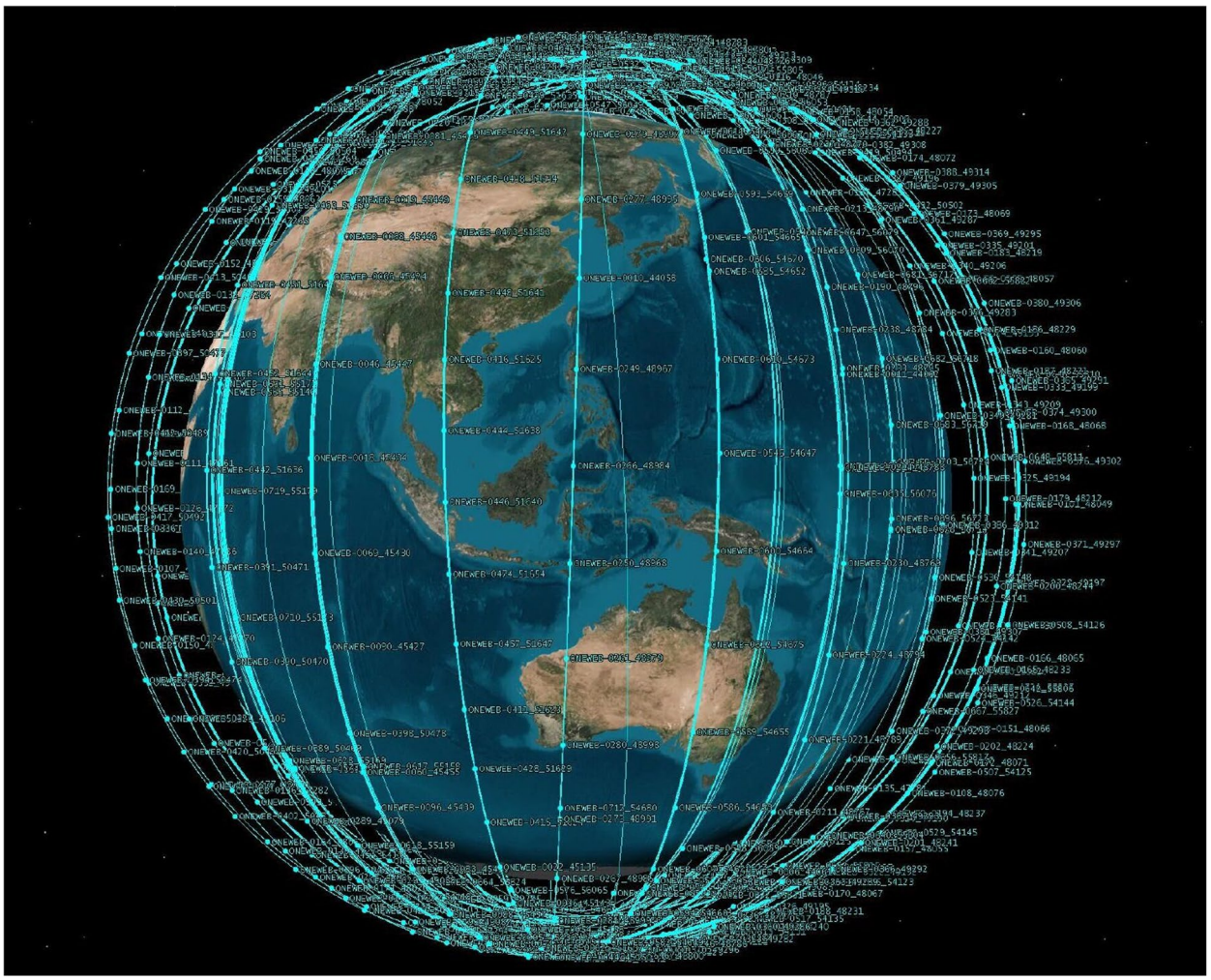

Fig. 8 Modeling of 636 LEO satellites under OneWeb satellite with STK

(1) Random Algorithm: When the overhead satellite receives N tasks at moment t, these tasks are randomly allocated to S satellites.

(2) Greedy Algorithm: For each task received by the overhead satellite at moment t, a greedy approach is employed. Each task is offloaded for computation to the Sat-MEC server that minimizes its execution time.

(3) GA : Before assigning tasks, a genetic algorithm is run to ascertain the optimal solution for task-to-service offloading within a certain number of iterations. Key parameters illustrate as Table 3.

Experiment analysis

In this subsection, we undertake a comprehensive exploration of our proposed algorithm through experiments conducted under diverse settings, aiming to corroborate its effectiveness. We commence this section with an examination of the convergence performance of the algorithms, providing an insight into their stability and reliability. Subsequently, we delve into a comparative study where the merits of DRL are juxtaposed against three baseline algorithms. This comparison seeks to underscore the disparities in the performance of each algorithm concerning task scheduling. A meticulous discussion and analysis will follow, shedding light on the intricacies and nuances of each algorithm's operation and outcomes.

Convergence performance

This experiment aims to verify the convergence of our proposed algorithm, SATDRL, for task scheduling in the Sat-MEC scenarios. Our proposed algorithm's convergence performance is demonstrated in Figs. 9 and 10 when the offloading ratio θ is 0.5 and the number of IMDs is 50. which also illustrates the change in the reward and loss functions as the training epochs increase in our proposed

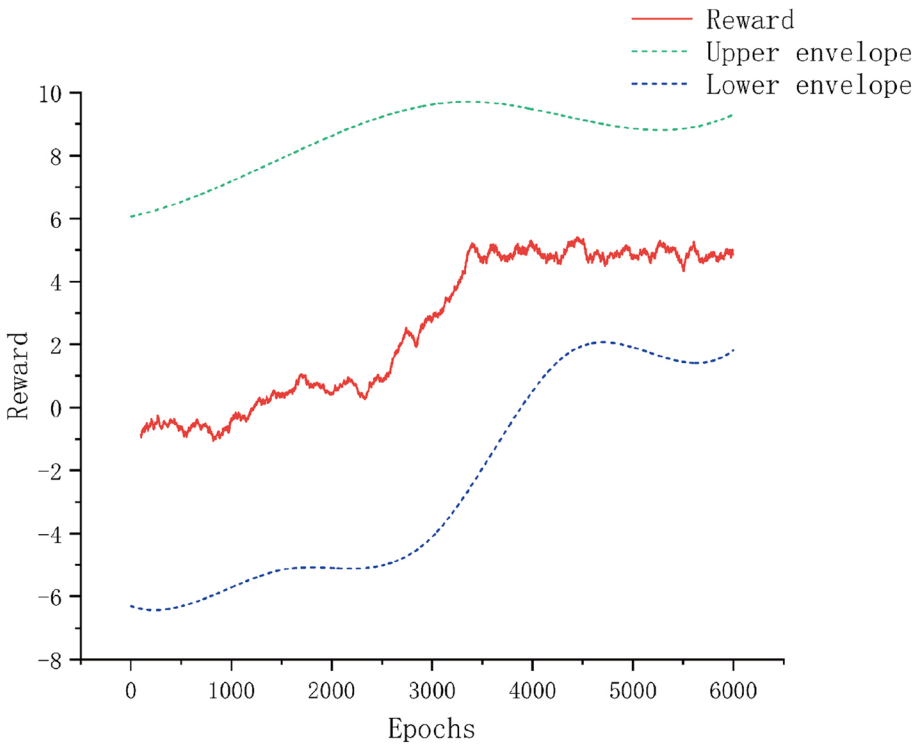

Fig. 9 The reward of the proposed algorithm (SATDRL)

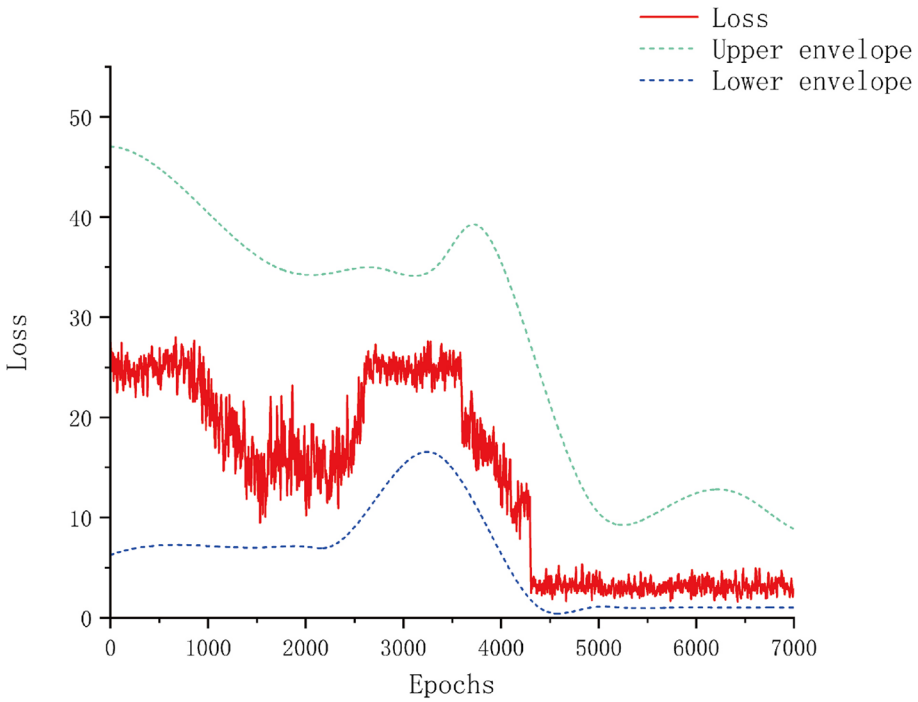

Fig. 10 The loss of the proposed scheduling algorithm (SATDRL)

algorithm. We also used an envelope to illustrate the magnitude of oscillation during the algorithm's convergence. It was discovered that the algorithm's oscillation amplitude is quite significant. This is due to the heterogeneity of tasks and servers in our environment: the heterogeneity of task data sizes, the number of CPU cycles required for computing tasks, the computing capability of Sat-MEC servers, and the initial task computation queues in the Sat-MEC server. The high-dimensional and unstable state space in our environment leading to algorithm convergence and oscillations post-convergence difficulties. In addition to the complexity of the characteristics of the data itself that makes it difficult for the model to converge, we have two more obvious loss decreases occurring for what the convergence image shows,which we explain to the readers below:

1. The initial convergence means that the model found a relatively good strategy at this stage, similar to the greedy approach, which only considers the matching relationship between tasks and servers and does not learn the effect of the task scheduling sequence on the scheduling result. But then, when exploring the state space more deeply, the model enters a re-exploration, gradually avoiding the idea of local optimality of the greedy algorithm, leading to a rise in loss.

2. The self-attention mechanism as the Q-network of DDQN in DRL. In the beginning, when the weights of self-attention are randomized, the model may perform relatively well in the early stage because it only relies on the local characteristics of the loss function for optimization and inevitably falls into the local optimum. However, as training progresses, there may be a period of oscillation as the model begins to adjust these weights to capture more complex scheduling patterns. Following this, it takes enough training for the weights to gradually stabilize, leading to a quadratic decrease in loss.

3. The DDQN approach for training, and although DDQN is more stable than traditional DQN, it may still oscillate in high-dimensional dynamic space environments. When the model converges initially, it is based on the existing knowledge the target network provides. However, as the target network is updated, the policy may be revised with the new knowledge, resulting in a transient rise in loss.

The above reasons are unavoidable, and no algorithm can search for the global optimum in a high-dimensional dynamic environment and have good convergence performance. Our goal in combining the self-attention mechanism and DRL approach is to expand the model's generalization ability, try to avoid overfitting the model, and learn a deeper scheduling strategy.

SATDRL algorithm performance

Figure 11 shows the impact of varying numbers of IMDs on various algorithms when the offloading ratio θ is 0.5.

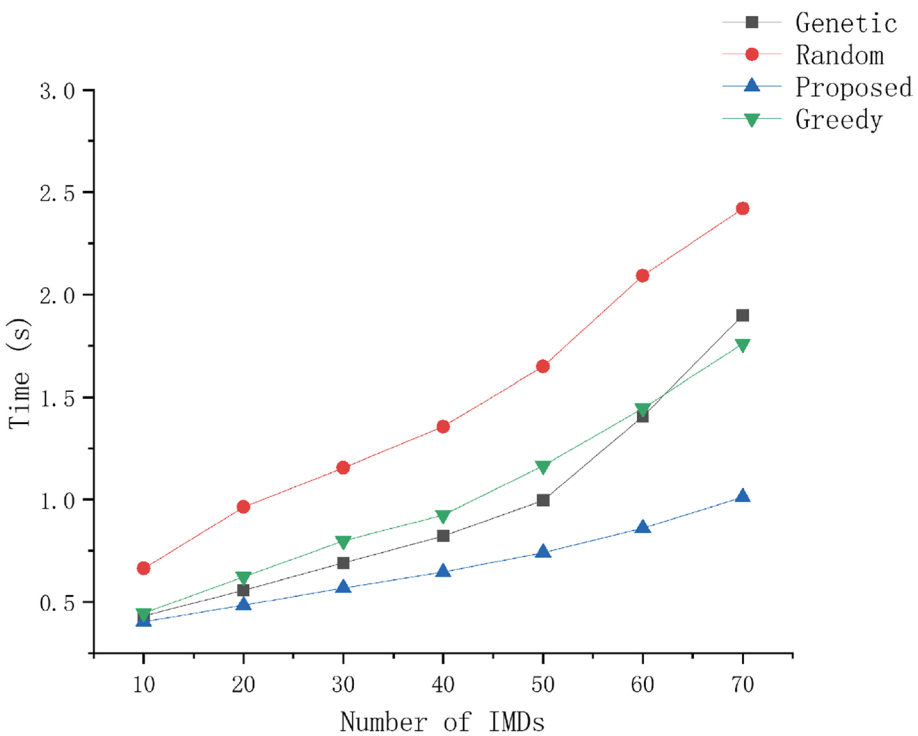

Fig. 11 The effection of the change in the number of IMDs on the average tasks processing time

We can observe that when the number of IMDs is relatively small (10, 20), there is not a significant difference between the greedy algorithm, the GA, and the algorithm we proposed. This situation is largely because the GA based on pseudo-random range searching, is highly likely to find reasonably good sub-optimal solutions when the solution space isn't particularly large. As for the greedy algorithm, under circumstances with fewer tasks, the offloading algorithm generated through the greedy strategy can sometimes provide a satisfactory sub-optimal solution.

As the number of IMDs increases and the solution space grows rapidly, the DRL-based task scheduling algorithm, which can still find high-quality solutions in a high-dimensional space, outperforms the other three methods at all times, and as the problem characteristic dimension grows, DRL demonstrates its advantage even more. In addition, we can observe an interesting state from Fig. 11, the solution quality of the greedy algorithm starts to outperform the genetic algorithm when the number of IMDs is greater than 60. According to our analysis, we believe that this situation is rooted in the fact that the two algorithms are different in their nature. The swarm intelligence algorithms, like GA, need to set more hyper-parameters in high-dimensional spaces to increase their explore-ability, especially in dynamically changing environments, while the greedy algorithm has been based on the idea of greedy strategy, although in higher dimensional spaces, its greedy strategy can also guarantee a lower bound on the solution.

Next, we illustrate the distribution of solutions for different numbers of IMDs.

As depicted in Fig. 12, the boxplot represents the distribution of average task processing time following scheduling under various algorithmic strategies, with an offloading rate of 0.5 and a variable number of IMDs. The boxplot shows the maximum, upper quarter, median, lower quarter, and minimum values from top to bottom.

Additionally, the small hollow square within the boxplot represents the mean value of the data. It is not hard to find out that when the number of IMD is 10, 20, and 30, there is almost no difference between the performance of our algorithm and the greedy algorithm, GA, compared to the pride, and those three scheduling algorithms are able to provide good scheduling solutions. However, as the number of IMDs increase, the GA and the greedy algorithm have difficulty in searching for the optimal solution in the high-dimensional solution space. At this time, our SATDRL scheduling algorithm still provides a high-quality scheduling solution. In addition, the box-and-line diagram can show the quality of the scheduling scheme and the degree of discretization of the solution. In Fig. 12, it is obviously that the quality of our proposed SATDRL scheduling scheme is the best compared to the other three schemes, and the degree of discretization of the solution is about the same as that of the greedy algorithm, which indicates that our proposed scheduling algorithm can output high-quality scheduling solution with greater accuracy.

Also, we found that the greedy algorithm outperforms the genetic algorithm when the number of IMDs exceeds 60. In the face of high-dimensional dynamic solution space, the GA must adjust or add its hyper-parameter to adapt. The greedy algorithm, by pursuing local optimization, ensures to some extent the quality of the overall solution. The DRL scheduling algorithm relies on a large amount of training data, extensive computational resources, and model training time to have strong exploratory capability in the high-dimensional dynamic space to find a high-quality solution.

Under we proposed the Sat-MEC scenario, the effectiveness of terrestrial IMDs often depends on geographical factors. The specific geographical context in which these devices are located leads to different performance levels and constraints. We use the offload rate to measure IMD's computational power and electricity. In certain instances where the over-the-top satellite communicates with TST, in situations such as the interruption of ground communication, power failures, or the emergence of urgent circumstances, terrestrial IMDs are relegated to processing minimal tasks or not processing any tasks at all. Figure 13 delineates our experimentation with diverse θ values. When θ equals 0, all tasks are executed locally, while with θ equal to 1, all tasks are subjected to offloading for execution. Figure 13 illustrates the average processing time of the task as a function of the unloading rate when the number of IMDs is 10, 20, 30, 40, 50, 60 and 70. As well as demonstrates the performance comparison of our proposed SATDRL with GA, Greedy Algorithm, and Randomized Algorithm.

Figure 13 illustrates that the change of the result with different offloading ratio from 0.3-0.7 when we keep the number of IMDs at average. More specifically, we show the differences in results produced by changes in offloading rates for different numbers of IMDs in Fig. 14.

As Fig. 14 illustrates, in the scenarios characterized by varying IMD quantities, the increase in the offloading ratio significantly reduces the average task processing duration. However, as the offloading ratio continues its ascent, the task processing duration in the satellite begins to surpass that of the terrestrial counterparts. It can be observed that the average processing time for fully offloaded tasks is shorter than that for tasks executed entirely locally. Nevertheless, with the increasing number of IMDs, the average computational time within the satellite also increases.

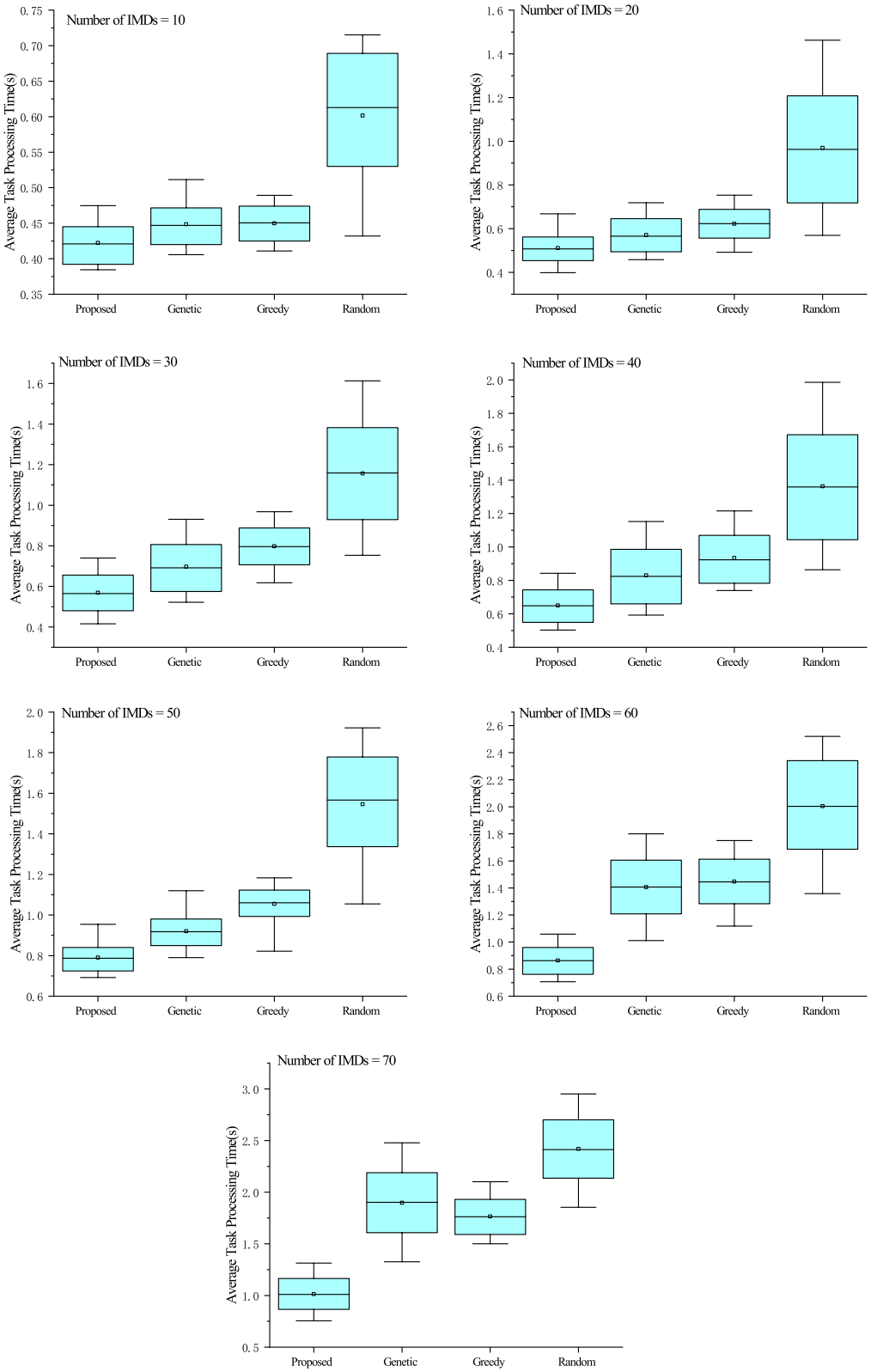

Fig. 12 The effection of different number of IMDs on average tasks processing time

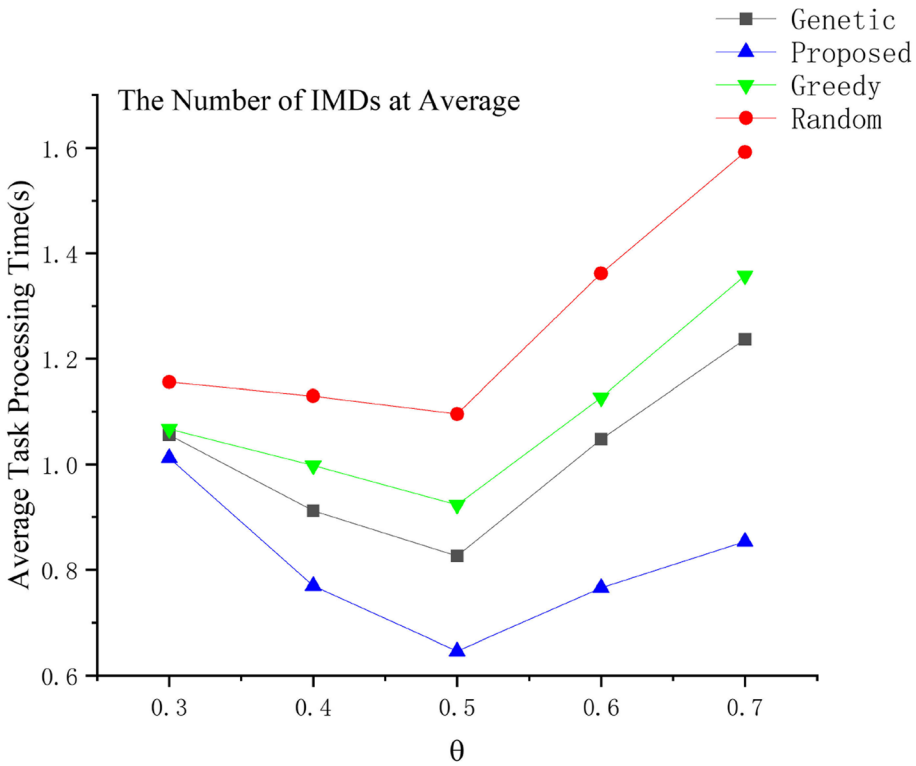

Fig. 13 The effection of different offloading rates (0.3-0.7) in keeping the number of IMDs at average on the average tasks processing time

By observing Fig. 14, we can easily find that when the number of IMDs increases to a certain number (50, 60, 70), the feature space of tasks and servers also increases, which makes it difficult for the GA to effectively search for the optimal scheduling scheme within the high-dimensional feature space. In contrast, the greedy algorithm, relying on its local optimal strategy, can guarantee the lower bound of the scheduling scheme and exceeds the GA when the number of IMDs is 60 and 70. In the face of the high-dimensional dynamic solution space, the GA may need to artificially set more hyper-parameters to increase the searching capability of its algorithm in the solution space. it's worth noting that, in our work, it is not inferred that swarm intelligence algorithms, such as GA and PSO, cannot solve the problem in high dimensional space. Because in our baseline algorithms, we are not adding specific parameters to GA for the scenario of this problem. We believe that algorithms, such as GA and PSO, can theoretically achieve the same performance as DRL by analyzing the characteristics of a particular scene, adding specific hyper-parameters, and training on the hyper-parameters.

Our proposed SATDRL algorithm demonstrates remarkable exploration performance in high dimensional dynamic environments, especially as the offloading rate varies. Our simulations and experiments, which are primarily conducted at the application level, found that the SATDRL maintains robust adaptability and superiority compared to the other three scheduling decisions within the Sat-MEC environment. It's noteworthy that our scheduling algorithms do not account for the actual medium and protocol stack, and we have not made alterations or modifications to the TCP/IP layers to ascertain the impact of our approach. Furthermore, our simulator does not employ precise network parameters, which means that the outcomes of our experiments are independent of the nuances introduced by generating packet requests in any specific network environment. Hence, while our primary objective is to identify a competent scheduling approach at the application layer, there is an implicit indication that the DRL strategy might exhibit commendable stability across the broader network context.

When using the DRL approach to solve the task offloading or scheduling problem in industrial environments, first, we model the environment encapsulating these devices and their interconnected landscape. In this context, states might encompass aspects like the device's battery level, the quality of network connectivity, and the queue of pending tasks. Informed by these states, the DRL agent then determines the optimal execution strategy for tasks: either processing them locally on the device

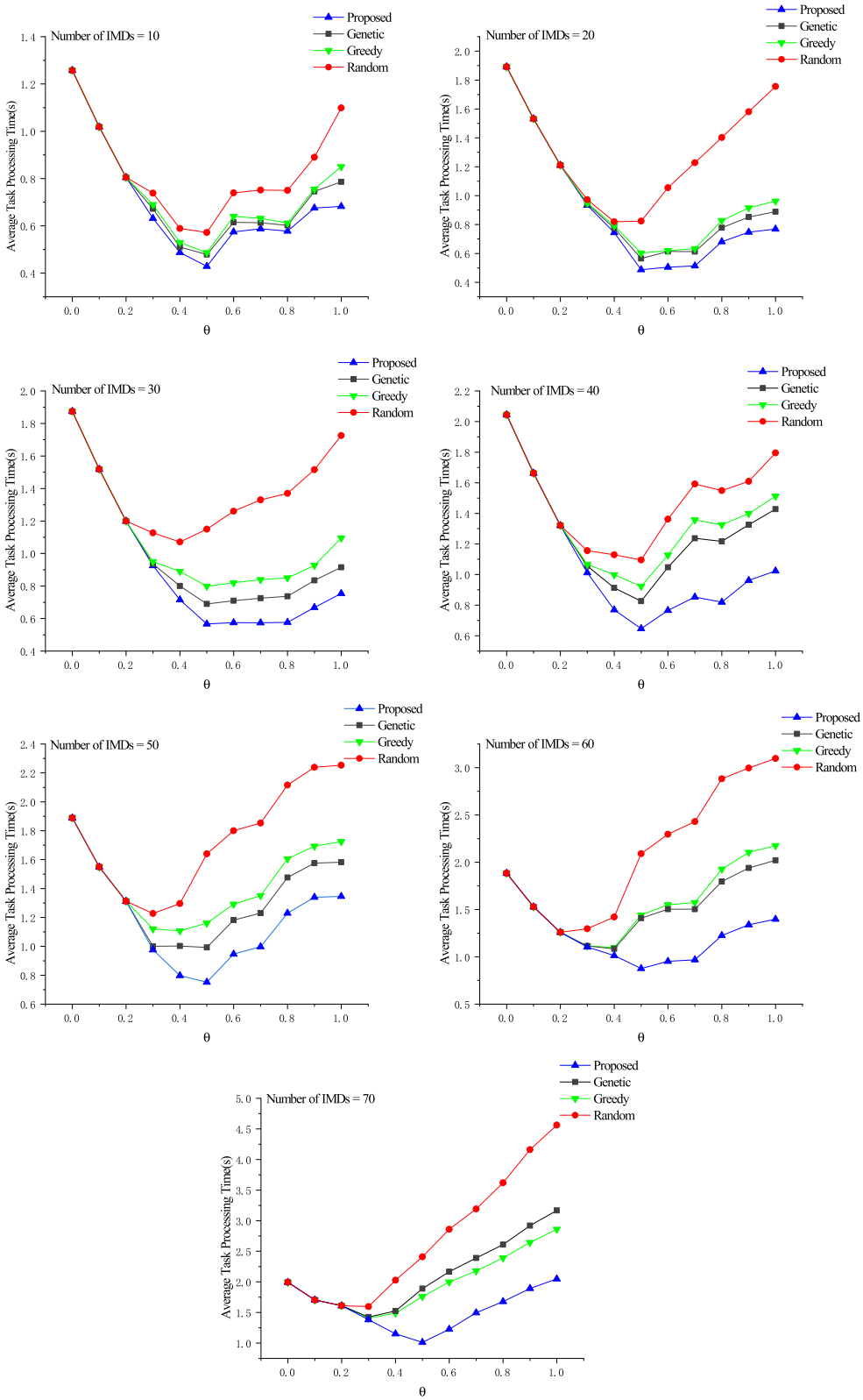

Fig. 14 The effection of offloading ratio θ on each algorithm for different number of IMDs

or offloading them to adjacent IoT devices or centralized servers. This decision-making is driven by a reward mechanism meticulously designed around metrics like task completion speed, energy consumption, and task accuracy. Reward can be designed based on the speed of task completion, energy consumption, and task accuracy. For example, fast task completion and low energy consumption may be rewarded positively, while incorrect task processing or delays may be rewarded negatively. By adopting DRL algorithms such as DQN or PPO, we then train and evaluate these models using either real-world or simulated datasets. These trained models can be deployed upon rigorous validation onto IoT devices, guiding them in real-time task offloading or scheduling decisions. Given IoT devices inherent resource constraints, optimizing the model computational footprint is imperative, potentially through techniques like model compression or employing domain-specific neural architectures. By adhering to this paradigm, we could ensure the judicious use of resources and pave the way for a more resilient and adaptive industrial IoT ecosystem.

Conclusion

In our work, we consider the scenarios of Sat-MEC system, where MEC servers are equipped on LEO satellites. The tasks generated by IMDs can be executed locally or offloaded to the Sat-MEC servers. In order to reduce the average task processing time, we emphasize the design of a task scheduling algorithm. This algorithm considers heterogeneity in the data size and the number of CPU cycles required for task computation generated by IMDs, the Sat-MEC server computational capability, and the task queue state of Sat-MEC servers. The task computation scheduling problem is formalized as a MDP. Further, we propose an online computational scheduling algorithm based on double DQN, wherein a self-attention mechanism is the Q-network, named SATDRL.

Our scheduling algorithm aims to approximate the optimal scheduling decision. After our simulations at the application level, compared to the three benchmark algorithms, our proposed algorithm can rely on a large amount of training data and extensive computational resources in an environment of constant interaction and trial and error, depending on the network depth and numerous parameters, so that it can learn a better scheduling strategy in a complex and dynamic environment than other three methods, our simulation experiments demonstrate that SATDRL reduces the average task processing time by 22.1%, 30.6%, and 41.3%, compared to the GA, the greedy algorithm, and the random algorithm, respectively.

DRL stands out due to its exceptional adaptability to dynamic environments and its capacity for abstract generalization in the context of task offloading within IoT fog computing networks. However, computational intensity and reliance on substantial-high-quality training data may restrict its applicability in real-time or resource-limited scenarios. In contrast, Swarm intelligence algorithms offer computational efficiency and ease of implementation, typically providing rapid solutions. However, they may encounter challenges related to local optima and may not to adapt to rapidly changing environments as fluidly as DRL. DRL is more suitable for complex and dynamic task offloading or scheduling problem, where large amounts of training data and computational resources are available. On the other hand, Swarm intelligence algorithms may be a more efficient choice for more straightforward problems or resource-constrained environments. The decision to choose between DRL and Swarm intelligence algorithms is based on considerations of computational resources, response time requirements, and environmental dynamism.

When using the DRL approach to solve the task offloading or scheduling problem in industrial environments, first, we model the environment encapsulating these devices and their interconnected landscape. In this context, states might encompass aspects like the device's battery level, the quality of network connectivity, and the queue of pending tasks. Informed by these states, the DRL agent then determines the optimal execution strategy for tasks: either processing them locally on the device or offloading them to adjacent IoT devices or centralized servers. This decision-making is driven by a reward mechanism meticulously designed around metrics like task completion speed, energy consumption, and task accuracy. Rewards can be designed based on the speed of task completion, energy consumption, and task accuracy. For example, fast task completion and low energy consumption may be rewarded positively, while incorrect task processing or delays may be rewarded negatively. By adopting DRL algorithms such as DQN or PPO, we then train and evaluate these models using either real-world or simulated datasets. These trained models can be deployed upon rigorous validation onto IoT devices, guiding them in real-time task offloading or scheduling decisions. Given IoT devices inherent resource constraints, optimizing the model computational footprint is imperative, potentially through techniques like model compression or employing domain-specific neural architectures. By adhering to this paradigm, we could ensure the judicious use of resources and pave the way for a more resilient and adaptive industrial IoT ecosystem.

Although DRL has good exploration ability in high dimensional dynamic environments and has found quality solutions to achieve the method of minimizing the task execution time, there are still many issues that

we need to continue to discuss and study in our future research.

1. For the study of the energy consumption of LEO satellites, with the development of the LEO satellite constellation, the energy of LEO satellites is mainly obtained by solar energy, so data processing on the Sat-MEC servers' resource pool must consider both the residual energy of LEO satellites and computational resources.

2. Regarding the examination of task queues, a portion of existing research takes into account task priority and life-critical tasks, while another portion overlooks the consideration of task priority. However, integrating task priority is an imperative trajectory for forthcoming research endeavors, as it can exemplify the actual environment with notable fidelity. In our subsequent work, we intend to incorporate considerations of task priority to render our scenarios more reflective of real-world conditions, thereby enhancing the realism and applicability of our research outcomes.

3. In light of the discussed research, our other objective is to investigate optimal solutions in cloud computing capability in scheduling, especially considering the constraints experienced at the IMD and LEO satellite levels. When faced with such conditions, Our future work will explore avenues where tasks can be strategically relayed to ground-based cloud stations with robust computing capabilities, utilizing LEO satellite constellations. We will sharpen efficient routing algorithms for LEO satellite constellations, which will involve meticulously exploring the delicate balance between resource utilization, computational efficiency, and data transfer latency. We aim to construct adaptable and resilient models capable of efficiently operating within environments with limited computational resources. By refining the interaction between terrestrial stations and satellite constellations, endeavor to optimize both task executions and the overall performance of the system.

4.In our simulation experiments, we have considered more strategies for fine-grained task scheduling and verified them at the application level, however, considering the maturity of LEO satellite technology and the further development of cloud computing technology in the future, we will propose more comprehensive modeling environments to adapt to the changes in the types of tasks, as well as the realism and comprehensiveness of the communication links in our subsequent work.

Acknowledgements
The authors are very grateful to all those who contributed to this study in any capacity and who have contributed to the objective of this study.

Authors' contributions
Shanchen Pang, Jianyang Zheng wrote the main manuscript text and Min Wang, Sibo Qiao prepare Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. Xiao He, Changnan Gao prepared Figs. 11, 12 and 13. All authors reviewed the manuscript.

Funding
The authors received no specific funding for this study.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
We confirm that our research does not involves a survey asking real human participants to give opinions, or animals data to make.

Competing interests
The authors declare no competing interests.

Received: 8 July 2023 Accepted: 31 October 2023
Published online: 18 November 2023

References

1. Qian L, Luo Z, Du Y, Guo L (2009) Cloud computing: An overview. In: Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1, Springer, pp 626–631
2. Yi S, Li C, Li Q (2015) A Survey of Fog Computing: Concepts, Applications and Issues. In Proceedings of the 2015 Workshop on Mobile Big Data (Mobidata '15). Association for Computing Machinery, New York, 37–42. https://doi.org/10.1145/2757384.2757397
3. Shi W, Dustdar S (2016) The promise of edge computing. Computer 49(5):78–81
4. Qi Q, Tao F (2019) A smart manufacturing service system based on edge computing, fog computing, and cloud computing. IEEE Access 7:86769–86777
5. Dao NN, Pham QV, Tu NH, Thanh TT, Bao VNQ, Lakew DS, Cho S (2021) Survey on aerial radio access networks: Toward a comprehensive 6g access infrastructure. IEEE Commun Surv Tutor 23(2):1193–1225
6. Shakarami A, Ghobaei-Arani M, Shahidinejad A (2020) A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. Comput Netw 182(107):496
7. Shakarami A, Shahidinejad A, Ghobaei-Arani M (2020) A review on the computation offloading approaches in mobile edge computing: A g ame-theoretic perspective. Softw Pract Experience 50(9):1719–1759
8. Shakarami A, Ghobaei-Arani M, Masdari M, Hosseinzadeh M (2020) A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective. J Grid Comput 18:639–671
9. Usha Nandini D, Leni ES (2019) Efficient shadow detection by using PSO segmentation and region-based boundary detection technique. J Supercomput 75:3522–3533
10. Das TK, Gosavi A, Mahadevan S, Marchalleck N (1999) Solving semi-Markov decision problems using average reward reinforcement learning. Manag Sci 45(4):560–574
11. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: A survey. J Artif Intell Res 4:237–285
12. Shakarami A, Shahidinejad A, Ghobaei-Arani M (2021) An autonomous computation offloading strategy in mobile edge computing: a deep learning-based hybrid approach. J Netw Comput Appl 178:102974
13. van Hasselt H, Guez A, Silver D (2016) Deep Reinforcement Learning with Double Q-Learning. Proceedings of the AAAI Conference on Artificial Intelligence 30(1). https://doi.org/10.1609/aaai.v30i1.10295
14. Imambi S, Prakash KB, Kanagachidambaresan GR (2021) PyTorch[J]. Programming with TensorFlow: Solution for Edge Computing Applications 87–104. https://doi.org/10.1007/978-3-030-57077-4_10
15. Wang Y, Yang J, Guo X, Qu Z (2019) A game-theoretic approach to computation offloading in satellite edge computing. IEEE Access 8:12510–12520
16. Li C, Zhang Y, Hao X, Huang T (2020) Jointly optimized request dispatching and service placement for MEC in LEO network. China Commun 17(8):199–208

17. Wang H, Han J, Cao S, Zhang X (2021) Computation offloading strategy of multi-satellite cooperative tasks based on genetic algorithm in satellite edge computing. In: 2021 International Conference on Space-Air-Ground Computing (SAGC), IEEE, pp 22–28
18. Tang Q, Fei Z, Li B, Han Z (2021) Computation offloading in LEO satellite networks with hybrid cloud and edge computing. IEEE Internet Things J 8(11):9164–9176
19. Zhu D, Liu H, Li T, Sun J, Liang J, Zhang H, Geng L, Liu Y (2021) Deep reinforcement learning-based task offloading in satellite-terrestrial edge computing networks. In: 2021 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, pp 1–7
20. Yu S, Gong X, Shi Q, Wang X, Chen X (2021) EC-SAGINs: Edge-computing-enhanced space-air-ground-integrated networks for internet of vehicles. IEEE Internet Things J 9(8):5742–5754
21. Mao S, He S, Wu J (2020) Joint UAV position optimization and resource scheduling in space-air-ground integrated networks with mixed cloud-edge computing. IEEE Syst J 15(3):3992–4002
22. Cassará P, Gotta A, Marchese M, Patrone F (2022) Orbital edge offloading on mega-LEO satellite constellations for equal access to computing. IEEE Commun Mag 60(4):32–36
23. He Y, Ren J, Yu G, Cai Y (2019) Joint computation offloading and resource allocation in d2d enabled mec networks. In: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, pp 1–6
24. Seng S, Li X, Luo C, Ji H, Zhang H (2019) A d2d-assisted MEC computation offloading in the blockchain-based framework for UDNs. In: ICC 2019 - 2019 IEEE International Conference on Communications (ICC), pp 1–6. https://doi.org/10.1109/ICC.2019.8762023
25. Zang S, Bao W, Yeoh PL, Vucetic B, Li Y (2023) Soar: Smart online aggregated reservation for mobile edge computing brokerage services. IEEE Trans Mob Comput 22(1):527–540. https://doi.org/10.1109/TMC.2021.3075947
26. Zhang Y, Chen C, Liu L, Lan D, Jiang H, Wan S (2022) Aerial edge computing on orbit: A task offloading and allocation scheme. IEEE Trans Netw Sci Eng 10(1):275–285
27. Chai F, Zhang Q, Yao H, Xin X, Gao R, Guizani M (2023) Joint Multi-Task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT. IEEE Trans Veh Technol 72(6):7783–7795. https://doi.org/10.1109/TVT.2023.3238771
28. Liu J, Zhao X, Qin P, Geng S, Meng S (2021) Joint dynamic task offloading and resource scheduling for WPT enabled space-air-ground power internet of things. IEEE Trans Netw Sci Eng 9(2):660–677
29. Hussein MK, Mousa MH (2020) Efficient task offloading for IoT-based applications in fog computing using ant colony optimization. IEEE Access 8:37191–37201
30. Javanmardi S, Shojafar M, Persico V, Pescapè A (2021) FPFTS: A joint fuzzy particle swarm optimization mobility-aware approach to fog task scheduling algorithm for internet of things devices. Softw Pract Experience 51(12):2519–2539
31. Zhang X et al. Energy-Efficient Computation Peer Offloading in Satellite Edge Computing Networks. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2023.3269801
32. Matrouk KM, Matrouk AD (2023) Mobility aware-task scheduling and virtual fog for offloading in IoT-fog-cloud environment. Wirel Pers Commun 130(2):801–836
33. Chen J, Xing H, Xiao Z, Xu L, Tao T (2021) A DRL agent for jointly optimizing computation offloading and resource allocation in MEC. IEEE Internet Things J 8(24):17508–17524
34. Seid AM, Boateng GO, Anokye S, Kwantwi T, Sun G, Liu G (2021) Collaborative computation offloading and resource allocation in multi-UAV-assisted IoT networks: A deep reinforcement learning approach. IEEE Internet Things J 8(15):12203–12218
35. Zheng F, Pi Z, Zhou Z, Wang K (2020) Leo satellite channel allocation scheme based on reinforcement learning. Mob Inf Syst 2020:1–10
36. Liu L, Chang Z, Guo X, Ristaniemi T (2017) Multi-objective optimization for computation offloading in mobile-edge computing. In: 2017 IEEE symposium on computers and communications (ISCC), IEEE, pp 832–837
37. Li W, Jin S (2021) Performance evaluation and optimization of a task offloading strategy on the mobile edge computing with edge heterogeneity. J Supercomput 77(11):12486–12507
38. Chen S, Li Q, Zhou M, Abusorrah A (2021) Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm. Sensors 21(3):779
39. Sharif Z, Jung LT, Ayaz M, Yahya M, Pitafi S (2023) Priority-based task scheduling and resource allocation in edge computing for health monitoring system. J King Saud Univ-Comput Inf Sci 35(2):544–559
40. Zhou W et al (2023) Priority-Aware Resource Scheduling for UAV-Mounted Mobile Edge Computing Networks. IEEE Trans Veh Technol 72(7):9682–9687. https://doi.org/10.1109/TVT.2023.3247431
41. Guo Y, Zhao R, Lai S, Fan L, Lei X, Karagiannidis GK (2022) Distributed machine learning for multiuser mobile edge computing systems. IEEE J Sel Top Signal Process 16(3):460–473
42. Wang H, An J, Zhou H (2023) Task assignment strategy in LEO-mutli-access edge computing based on matching game. Computing 105:1571–1596. https://doi.org/10.1007/s00607-023-01151-3
43. Jain V, Kumar B (2023) Qos-aware task offloading in fog environment using multi-agent deep reinforcement learning. J Netw Syst Manag 31(1):7
44. Diao X, Zheng J, Cai Y, Wu Y, Anpalagan A (2019) Fair data allocation and trajectory optimization for UAV-assisted mobile edge computing. IEEE Commun Lett 23(12):2357–2361
45. Pang S, He X, Yu S, Wang M, Qiao S, Gui H, Qi Y (2023) A Stackelberg game scheme for pricing and task offloading based on idle node-assisted edge computational model. Simul Model Pract Theory 124(102):725
46. Zeng F, Chen Y, Yao L, Wu J (2021) A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing. Peer Peer Netw Appl 14:467–481
47. Wei F, Chen S, Zou W (2018) A greedy algorithm for task offloading in mobile edge computing system. China Commun 15(11):149–157
48. Fan Y, Wang L, Wu W, Du D (2021) Cloud/edge computing resource allocation and pricing for mobile blockchain: an iterative greedy and search approach. IEEE Trans Comput Soc Syst 8(2):451–463
49. Zhang N, Guo S, Dong Y, Liu D (2020) Joint task offloading and data caching in mobile edge computing networks. Comput Netw 182:107446
50. Phillips C, Sicker D, Grunwald D (2012) A survey of wireless path loss prediction and coverage mapping methods. IEEE Commun Surv Tutor 15(1):255–270
51. Tang Z, Zhou H, Ma T, Yu K, Shen XS (2021) Leveraging LEO assisted cloud-edge collaboration for energy efficient computation offloading. In: 2021 IEEE Global Communications Conference (GLOBECOM), IEEE, pp 1–6
52. Di B, Zhang H, Song L, Li Y, Li GY (2018) Ultra-dense LEO: Integrating terrestrial-satellite networks into 5g and beyond for data offloading. IEEE Trans Wirel Commun 18(1):47–62
53. Zhou Y, Jj Yang, Huang Z (2020) Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. Int J Prod Res 58(9):2561–2580
54. Zhang S, Liu A, Han C, Liang X, Xu X, Wang G. Multi-agent Reinforcement Learning-Based Orbital Edge Offloading in SAGIN Supporting Internet of Remote Things. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2023.3287737
55. Zhou C, Wu W, He H, Yang P, Lyu F, Cheng N, Shen X (2020) Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN. IEEE Trans Wirel Commun 20(2):911–925
56. Liu Y, Jiang L, Qi Q, Xie K, Xie S. Online Computation Offloading for Collaborative Space/Aerial-Aided Edge Computing Toward 6G System. IEEE Trans Veh Technol. https://doi.org/10.1109/TVT.2023.3312676
57. Liao H, Wang Z, Zhou Z, Wang Y, Zhang H, Mumtaz S, Guizani M (2021) Blockchain and semi-distributed learning-based secure and low-latency computation offloading in space-air-ground-integrated power IoT. IEEE J Sel Top Signal Process 16(3):381–394
58. Li W, Yang T, Delicato FC, Pires PF, Tari Z, Khan SU, Zomaya AY (2018) On enabling sustainable edge computing with renewable energy resources. IEEE Commun Mag 56(5):94–101
59. Li S, Huang J (2017) Energy efficient resource management and task scheduling for IoT services in edge computing paradigm. In: 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), IEEE, pp 846–851
60. Bardi M, Dolcetta IC et al (1997) Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations, vol 12. Springer

61. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction, vol 22447. MIT press, Cambridge
62. Zhang X, Wang G, Meng X, Wang S, Zhang Y, Rodriguez-Paton A, Wang J, Wang X (2022) Molormer: a lightweight self-attention-based method focused on spatial structure of molecular graph for drug–drug interactions prediction. Brief Bioinform 23(5):bbac296
63. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
64. Lin LJ (1992) Reinforcement learning for robots using neural networks. Carnegie Mellon University
65. Version SUM (2000) 4.2. 1 for pcs. Analytical Graphics, INC (AGI)
66. Rajan JA (2002) Highlights of GPS II-R Autonomous Navigation. Proceedings of the 58th Annual Meeting of The Institute of Navigation and CIGTF 21st Guidance Test Symposium (2002), Albuquerque, NM, pp. 354–363
67. Petranovich J (2012) Mitigating the effect of weather on ka-band high-capacity satellites. ViaSat Inc, Carlsbad
68. Saeed N, Elzanaty A, Almorad H, Dahrouj H, Al-Naffouri TY, Alouini MS (2020) Cubesat communications: Recent advances and future challenges. IEEE Commun Surv Tutor 22(3):1839–1862

Publisher's Note