Link Quality-Aware Computation Offloading in Satellite Edge Computing Networks

Abstract—Satellite Edge Computing Networks (SECN) face significant challenges in computation offloading due to the dynamic variations in satellite link quality. The rapid advancement of satellite constellation technology has enhanced the role of satellite networks in providing global coverage and supporting terrestrial networks. However, satellite link quality is affected by multiple factors, including bit error rate, link stability, and latency, which introduce complexities in computation offloading. To address these challenges, a computation offloading model is developed by incorporating link bit error rate, link duration, and link latency, and is formulated as a Markov Decision Process (MDP). A deep reinforcement learning (DRL)-based approach is proposed, featuring a Deep LSTM Attention Q-Learning algorithm (QCO-DQL) that integrates long short-term memory (LSTM) networks and an attention mechanism to minimize link quality costs. Experimental results demonstrate that QCO-DQL outperforms existing methods in terms of link bit error rate, duration, and latency, effectively adapting to dynamic satellite environments and providing optimized computation offloading strategies for users.

Index Terms—satellite network, edge computing, link quality, DRL, LSTM.

1 Introduction

In recent years, the rapid development of satellite constellations, exemplified by Starlink, has significantly enhanced global communication capabilities. Satellite networks have emerged as a crucial supplement to terrestrial networks, offering wide-area coverage and seamless connectivity. Due to their extensive coverage and seamless integration, satellite networks have gained considerable attention from both academia and industry. The proliferation of computationally intensive applications such as speech recognition, facial recognition, intelligent transportation, and 3D gaming has heightened the demand for robust computing services within satellite networks. Inspired by the paradigm of Mobile Edge Computing (MEC), the concept of Satellite Edge Computing (SEC) has emerged, where edge computing resources are deployed on Low Earth Orbit (LEO) and Geostationary Earth Orbit (GEO) satellites. This architecture extends computational capabilities to the network edge, enabling users worldwide to access computing services with reduced latency and improved responsiveness.

Satellite Edge Computing Networks (SECNs) integrate satellite communication and edge computing technologies, utilizing satellites as relay nodes for computing tasks. Within SECNs, different link types—such as user-to-LEO, LEO-to-LEO, and LEO-to-GEO—exhibit varying distances, transmission delays, and packet loss rates, leading to heterogeneity in link quality. The dynamic nature of LEO satellites, characterized by their high mobility compared to terrestrial networks, further complicates network conditions, necessitating sophisticated computation offloading strategies.

Satellite link quality is subject to frequent variations due to atmospheric interference, satellite mobility, and environmental conditions. High-energy particles such as cosmic rays and solar winds can increase the bit error rate (BER) of satellite links, particularly in regions with weaker geomagnetic fields (e.g., polar areas). Moreover, inter-satellite distances dynamically change, leading to intermittent link disruptions when two adjacent satellites exceed their communication range.

The Satellite Edge Computing Network (SECN) leverages satellites as communication relay nodes, deploying edge computing resources on Low Earth Orbit (LEO) and Geostationary Orbit (GEO) satellites. However, the dynamic nature of satellite links poses significant challenges for computation offloading. Unlike terrestrial networks, satellite links are subject to high mobility, varying distances, and fluctuating link qualities, which can lead to increased packet loss rates, reduced link stability, and higher latency. These factors collectively impact the efficiency and reliability of computation offloading in SECN environments.

Existing research on computation offloading in SECNs has primarily focused on optimizing latency, energy consumption, or a balance between the two. However, these studies often overlook the critical issue of fluctuating satellite link quality. For instance, high-energy particles such as solar winds and cosmic rays can increase the bit error rate (BER) in satellite links, particularly in regions with weaker Earth's magnetic fields like the polar areas. Additionally, the fast movement of LEO satellites means that the network environment is highly dynamic, with frequent changes in link conditions. These challenges necessitate a comprehensive approach to computation offloading that accounts for the dynamic nature of satellite links.

In this paper we study the Link Quality-aware Computation Offloading (LQCO) problem in the SECNs. Specifically, we propose a novel computation offloading strategy for SECNs that jointly considers *link error rate*, *link duration* and *link latency*. In addition, we make the first attempt at considering the influence of LEO satellites' high-speed movement on link quality. In the SECNs architecture, the GEO satellite acts as the cloud, LEO satellites act as edge computing nodes, and the ground users are the end-users. Please note that since the movement of users is extreme slowly compared to that of the LEO satellites and does not affect the LQCO strategy when offloading tasks

in the SECN, it is negligible in this paper. Besides, all the offloading tasks submitted by users can be computed on LEO satellites or GEO satellites. Major contributions are as follows:

- We propose a computational offloading model that jointly considers link error rates, link duration, and link latency variations caused by satellite mobility. By integrating a link quality analysis module, our approach ensures more reliable task execution under dynamic network conditions.
- To tackle the high-dimensional and dynamic nature of SECNs, we develop a deep reinforcement learning (DRL) framework utilizing Long Short-Term Memory (LSTM) networks and attention mechanisms. This model effectively captures temporal dependencies in satellite link variations and dynamically adjusts offloading decisions to optimize performance.
- Unlike existing static offloading methods, our proposed strategy dynamically adapts to real-time changes in link quality, network topology, and resource availability. This enables more efficient task allocation and reduces the likelihood of computation failures due to degraded links.
- We conduct extensive simulations and comparative analysis against baseline methods, including conventional RL-based offloading, heuristic algorithms, and random allocation strategies. Our results demonstrate significant improvements in task completion time, link stability, and energy efficiency.

The remainder of this paper is organized as follows. Section II provides a detailed review of related work. Section III presents the system model and problem formulation. Section IV describes our proposed approach, including the MDP formulation and the LQCO-DQL algorithm. Section V evaluates the performance of our approach through simulations. Finally, Section VI concludes the paper and outlines future work.

2 RELATED WORK

In a satellite environment, MEC is deployed on LEO satellites, and due to the fast movement of LEO satellites, the entire compute offloading process is in a dynamic network environment. In addition, compared with the ground scenario, the types of links in the satellite edge computing network are more diverse.

Because delay and energy consumption are two important indexes to measure the superiority of computational offloading in the modeling of satellite edge computing networks, many scholars have considered their effects separately or simultaneously. Considering the impact of energy consumption, Song et al. [41] used LEO satellites to minimize energy consumption for ground users. Tang et al. [42] studied a computational offloading decision that minimizes the total energy consumption of ground users under the coverage time and computing capacity limits of each LEO satellite. You et al. [43] established a convex optimization problem model with the goal of minimizing user energy consumption and derived an unloading priority function. This function generates a priority for the user based on the user's channel gain and local compute power consumption. Song Zhengyu et al. [44] proposed a task migration and resource allocation algorithm based on LEO satellite collaborative edge computing to minimize the total energy consumption of users. Considering the impact of delay, Cheng et al. [45] comprehensively considered the resource partitioning strategy of satellite edge computing networks and the establishment of collaborative networks in emergency situations, and took different user sensitivities to delay and connection time as resource allocation factors to realize dynamic resource scheduling in emergency situations. Fang et al. [46] proposed a dynamic offload strategy that jointly optimizes mission offload decision-making, computing resource allocation, and transmit power control, thereby minimizing the total task delay for ground users. Zhai et al. [47] proposed a computational offload scheme for resource sharing through collaboration among multiple satellites, and proposed a solution algorithm based on the greedy strategy to optimize the computational delay of all tasks under the limitation of transmission and computational power. Considering the impact of delay and energy consumption, Cui et al. [48] co-optimized coupling user association, offload decision, computing and communication resource allocation for a MEC enhanced network with multiple satellites and multiple cloud computing centers to minimize delay and energy consumption costs. Wang et al. [49] proposed a satellite edge computing network with bilateral computing power, which deployed MEC on ground base stations and LEO satellites to optimize the user's delay and energy consumption. Li et al. [50] proposed a combined offload decision and resource allocation strategy to reduce system delay and energy consumption. Wang et al. [51] proposed a method to jointly optimize computing resource allocation, wireless resource allocation, and offload decision-making, thereby reducing system delay and energy consumption and reducing running time.

In terms of model solving, since reinforcement learning and game theory can adapt to the dynamic changing satellite network environment, and convex optimization algorithm can obtain the best unloading decision, many scholars have adopted reinforcement learning, game theory or convex optimization algorithm to solve the model. In terms of reinforcement learning solution, Xu et al. [52] proposed a DRL-based computational resource allocation method for joint communication. Different from the traditional Q-Learning method, DRL uses deep neural networks to estimate state action values. Based on the satellite edge computing network architecture, Zhu et al. [53] proposed a low computational complexity deep reinforcement learning-based approach to solve the computational offloading problem, which only makes offloading decisions based on the current channel state. Liao et al. [54] proposed a learning-based task offloading and resource allocation algorithm based on the sky-earth integrated electric iot architecture to solve the joint optimization problem of task offloading and computing resource allocation. Wu et al. [55] consider a dynamic network in which users can leave or join the network at any location, and propose a DQL-based hierarchical online learning algorithm that interactively learns the utility of each offload decision with the MEC network. Tang et al. [56] proposed a SAGIN computational unloading method based on reinforcement learning based on the integrated

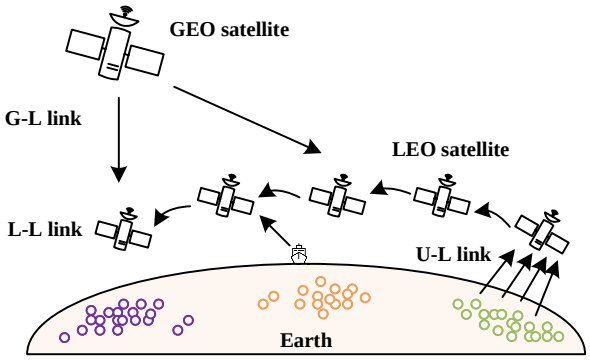


Fig. 1: Architecture of SECNs.

network of heaven, earth and space, taking into account the fast movement of nodes and frequent changes of link state. In terms of using game theory to solve the problem, Wang et al. [57] proposed a multi-access edge computing network architecture based on LEO satellites. In order to obtain the solution of the task unloading feedback, a difference game model of the architecture was established. Zhang et al. [58] studied the joint computing and communication resource management of satellite networks in order to minimize the computing delay of computationally intensive applications. They also considered two different satellite edge computing scenarios and proposed a matching theory based on game theory to obtain an approximate optimal solution. Wang et al. [59] proposed a new LEO satellite edge computing system model, defined delay and energy consumption as system costs, then decomposed the problem into two subproblems, and used game theory to solve the computational offloading problem under optimal resource allocation. Wang et al. [60] proposed a game theory method to optimize the computational offload strategy in satellite edge computing, and designed a computational offload game framework to solve the computational offload problem based on the queuing theory to calculate the response time and energy consumption of the task as indicators of optimization performance. In terms of solving with convex optimization algorithm, Tang et al. [42] used ADMM to study the computational offloading problem aiming at minimizing the total energy consumption of users based on the satellite edge computing network architecture. Wei et al. [62] used the theory of successive convex optimization to analyze a computational offloading problem of LEO satellite networks combining cloud computing and edge computing.

To sum up, in the study of satellite edge computing network, MEC is mainly deployed on LEO satellites, and the collaboration between satellites is used to complete computing offloading. However, most scholars regard the satellite edge computing network as a static scenario, and do not consider the impact of satellite movement on computing offloading in the actual scenario. In addition, few scholars comprehensively consider the role of different types of link quality in computing offload.

3 System Model

Fig. shows the architecture of SECN, which includes three layers: GEO layer, LEO layer and ground layer. **GEO Layer:** Similar to this layer includes one GEO satellite regarded as the cloud center because GEO satellites have large payloads and expansive coverage areas and their positions are relatively fixed.

TABLE 1: NOTATION

Notation	Description
$\mathcal{M}$	Set of satellites
$\mathcal{I}$	Set of users
$\gamma_{i,\tilde{m}}$	Coverage angle of the accessible LEO satellite $\tilde{m}$
$t^R_{i,\tilde{m}}$	Time that accessible LEO satellite $\tilde{m}$ covers user i
$r^U_{i,\tilde{m}}$	Transmission rate from user i to its accessible LEO satellite $\tilde{m}$
q_i	User i's computation task
LL^U_i	Local latency of user i
$T^{S'}_{i,m}$	Offloading latency of user i in LEO satellite m
$T^G_{i,g}$	Offloading latency of user i in GEO satellite g
LE^U_i	Local energy consumption of user i
$E^S_{i,m}$	Offloading energy of user i in LEO satellite m
$E^G_{i,g}$	Offloading energy of user i in GEO satellite g
f^U_i	Computing power of user i
f^S_m	Computing power of LEO satellite m
f^G_g	Computing power of GEO satellite g
$a_{i,m}$	Offloading decision of user i
t_{total}	Total latency of network
e_{total}	Total energy consumption of network
λ	Weight coefficient of indicator
Z_m	Maximum resources provided by LEO satellite m
$a'_{i,m}$	Continuous variables of $a_{i,m}$

LEO Layer: It consists of LEO satellites equipped with MEC servers. In this paper, we only focus on the collaboration and computation offloading among the co-orbiting LEO satellites, which are called L-L links (i.e., inter-satellite communication), same to Besides, LEO satellites can communicate with GEO satellites via G-L (GEO satellite to LEO satellite) links

Ground Layer: It is the user layer in which each user's device has computing capability. Due to the long distance and transmission power limitations, it is hard for users to directly communicate with the GEO satellite. Note that each ground area is covered by at least one LEO satellite. Then, for each user, we call the LEO satellite that covers it and is closest to it as its *accessible LEO satellite*. Thus, similar to the previous works we allow users to offload their tasks to the accessible LEO satellites directly via U-L (User to LEO satellite) links, or to other LEO satellites (L-L link) and/or the GEO satellite (G-L link) via their accessible LEO satellites as intermediate relays.

Consider a scenario where one GEO satellite provides coverage for co-orbiting LEO satellites and ground users. Let $\mathcal{M} = \{1, \cdots, m, \cdots, M, g\}$ be a set of satellites, including M LEO satellites and one GEO satellite g, and $\mathcal{I} = \{1, \cdots, i, \cdots, I\}$ be a set of ground users. Assume that each user i generates a computation task in each time slot, and the computation task of the user is represented as $q_i = \{o_i, x_i\}$. Where o_i (bits) indicates the data amount of the task, $x_i = o_i * w_i$ represents the CPU cycles required for each bit of the task, and w_i (cycles/bit) indicates the task intensity [70]. Note that compared with LEO satellites, the GEO satellite move slowly and cover a wide range. Thus we assume that it is static with respect to the ground The main symbols used in the rest of this paper are summarized in Table

3.1 Link Error Rate Model

The link error rate (LER) of SECN consists of the U-L link error rate, L-L link error rate, and G-L link error rate.

U-L LER. LER refers to the probability of data loss in the satellite link. A high link error rate may cause the link to be unavailable or unable to transmit data. Similar to [] [], we use the bit error rate (BER) to measure the LER. Let us define the LER between the user i and its access LEO satellite $\hat{m}$ as $\mathcal{P}^{U}_{i,\hat{m}}$, which can be expressed as,

$$\mathcal{P}^{U}_{i,\hat{m}} = 1 - (1 - \alpha_{i,\hat{m}})^{o_i}, \tag{1}$$

where $\alpha_{i,\hat{m}}(bit)$ indicates the BER between user i and it's access LEO satellite $\hat{m}$. Influenced by the complex transmission environment between the user and the satellite, the LER is often high [] [].

L-L LER. It indicates the error rate of the task from the user's access LEO satellite $\hat{m}$ to its offload LEO satellite m. For convenience, we assume an ideal scenario, where all erroneous bits remain uncorrected. The L-L link error define as $\mathcal{P}^{S}_{\hat{m},m}$ and calculated as follows,

$$\mathcal{P}^{S}_{\hat{m},m} = 1 - (1 - \alpha_{\hat{m},m})^{o_i \alpha_{i,\hat{m}} (\alpha_{\hat{m},m})^{n_{\hat{m},m}}}, \tag{2}$$

where $n_{\hat{m},m}$ and $\alpha_{\hat{m},m}$ represent the hops between the user's access LEO satellite $\hat{m}$ and the BER of U-L link, respectively. The $o_i \alpha_{i,\hat{m}} (\alpha_{\hat{m},m})^{n_{\hat{m},m}}$ denote the amount of the task satellite m received, this implies that as $n_{\hat{m},m}$ increases, the hops between user's access LEO satellite and the offload LEO becomes greater, resulting in more severe data distortion, which aligns with practical situations.

G-L LER. Same as L-L LER, the G-L LER represent the error rate of the task from the user's access LEO satellite $\hat{m}$ to the GEO satellite g. Let us define the G-L LER as $\mathcal{P}^{G}_{\hat{m},g}$, which is calculated as follows,

$$\mathcal{P}^{G}_{\hat{m},g} = 1 - (1 - \alpha_{\hat{m},g})^{o_i \alpha_{i,\hat{m}} \alpha_{\hat{m},g}}, \tag{3}$$

where $o_i \alpha_{i,\hat{m}} \alpha_{\hat{m},g}$ represent the amount of the task satellite g received, $\alpha_{\hat{m},g}$ represent the BER of G-L link. Since the amount of data returned by the task is generally small, the failure rate of the task return is ignored.

3.2 Link Duration Model

The link duration signifies the period during which data transmission is sustained without interruption. It consists of U-L link duration, L-L link duration, and G-L link duration.

U-L Link Duration. The U-L link duration is the time that the user is within the coverage area of its access LEO satellite. Let $\mathcal{D}^{U}_{i,\hat{m}}$ be the duration of the U-L link, according to [], [], $\mathcal{D}^{U}_{i,\hat{m}}$ is calculated by,

$$\mathcal{D}^{U}_{i,\hat{m}} = \frac{u^{U}_{i,\hat{m}}}{v^{U}_{i,\hat{m}}}, \tag{4}$$

where $v^{U}_{i,\hat{m}}$ denotes the linear velocity of the user's access LEO satellite $\hat{m}$, $u^{U}_{i,\hat{m}}$ indicates the communication arc length between user i and its access LEO satellite $\hat{m}$.

L-L Link Duration. According to [], the L-L link might be disconnected due to the phase change of adjacent LEO satellites, as shown in Fig., using the Walker constellation as an example, the adjacent LEO satellite are disconnected near the pole. (phase Angle variation diagram). Within the time interval t, the duration of the LEO satellite relative to the access LEO satellite is defined as $\mathcal{D}^{S}_{m,m+1}$, and then $\mathcal{D}^{S}_{m,m+1}$ is calculated as follows,

$$\mathcal{D}^{S}_{m,m+1} = \begin{cases} \varphi^{D\_S}_{m,m+1} - t, & 0 \le t < \varphi^{D\_S}_{m,m+1}, \\ 0, & \varphi^{D\_S}_{m,m+1} \le t < \Phi, \\ \frac{T}{2} - t + \varphi^{D\_S}_{m,m+1}, & \Phi \le t < \frac{T}{2}, \end{cases} \tag{5}$$

where $\Phi = \varphi^{D\_S}_{m,m+1} + \varphi^{W\_S}_{m,m+1}$, $\varphi^{D\_S}_{m,m+1}$ and $\varphi^{W\_S}_{m,m+1}$ indicates the duration of LEO satellite disconnection and the time when LEO satellite disconnection, respectively.

G-L Link Duration. Similar to the U-L link duration, we use the coverage time of the GEO satellite to the LEO satellite to represent the duration of the G-L link. Let us define the G-L link duration as $\mathcal{D}^{G}_{\hat{m},g}$, $\mathcal{D}^{G}_{\hat{m},g}$ is calculated by,

$$\mathcal{D}^{G}_{\hat{m},g} = \frac{u^{G}_{\hat{m},g}}{v^{G}_{\hat{m},g}}, \tag{6}$$

where $v_{\hat{m},g}$ denotes the linear velocity of the GEO satellite g, and $u_{\hat{m},g}$ denotes the communication arc length between the user's access LEO $\hat{m}$ satellite and the GEO satellite g.

3.3 Link Latency Model

The latency of SECN consists of U-L latency, L-L link latency, and G-L link latency. *U-L link latency.* U-L link latency consists of three parts: U-L transmission latency, U-L propagation latency and LEO satellite computation latency.

U-L transmission latency. It represents the time required for the user to transmit its tasks to its access LEO satellite. The U-L transmission latency is defined as $t^{T\_S}_{i,\hat{m}}$ the latency for the user i to transmit its accessible satellite $\hat{m}$. Calculated by the following formula,

$$t^{T\_S}_{i,\hat{m}} = \frac{o_i}{r^{S}_{i,\hat{m}}}, \tag{7}$$

where $r^{S}_{i,\hat{m}}$ denotes the transmission rate from user i to its access LEO satellite $\hat{m}$.

U-L propagation latency. It denotes the time required for the user to propagate its task to its access satellite. Let $t^{P\_S}_{i,\hat{m}}$ be the U-L propagation latency, then $t^{P\_S}_{i,\hat{m}}$ is calculated by,

$$t^{P\_S}_{i,\hat{m}} = \frac{d^{S}_{i,\hat{m}}}{c}, \tag{8}$$

where $d^{S}_{i,\hat{m}}$ represents the distance between the user i and its access LEO satellite $\hat{m}$, c represents the speed of light.

Computation latency. The computation latency that user i offload its tasks to LEO satellite m is denoted as $t^{C\_S}_{i,m}$, which is calculated by,

$$t^{C\_S}_{i,m} = \frac{x_i}{f^{S}_{m}}, \tag{9}$$

where f^{S}_{m} (GHz) denotes the computing power of the LEO satellite m.

Let us define U-L link latency as $t^{S}_{i,\hat{m}}$, according to Eqs.(8)-(8), $t^{S}_{i,\hat{m}}$ is calculated as,

$$t^{S}_{i,\hat{m}} = t^{T\_S}_{i,\hat{m}} + 2t^{P\_S}_{i,\hat{m}} + t^{C\_S}_{i,m}. \tag{10}$$

L-L link latency L-L link latency includes two parts: L-L transmission latency and L-L propagation latency. *U-L transmission latency.*According to U-L link, we define the L-L transmission latency and propagation latency as $t^{T\_S}_{m,m+1}$, $t^{P\_S}_{m,m+1}$, respectively. Then $t^{T\_S}_{m,m+1}$ and $t^{P\_S}_{m,m+1}$ are calculated as

$$t^{T\_S}_{m,m+1} = \frac{o_i}{r^S_{m,m+1}}, \tag{11}$$

$$t^{P\_S}_{m,m+1} = \frac{d^S_{m,m+1}}{c}, \tag{12}$$

where $r_{m,m+1}$ indicates the transmission rate between adjacent LEO satellites, and $d^S_{\hat{m},m+1}$ indicates the distance between adjacent LEO satellites.

Let us define the L-L link latency as $t^S_{\hat{m},m}$, according to Eqs.(8)-(8), $t^S_{\hat{m},m}$ is calculated as,

$$t^S_{\hat{m},m} = (t^{T\_S}_{m,m+1} + t^{P\_S}_{m,m+1}) n_{\hat{m},m}. \tag{13}$$

where $n_{\hat{m},m}$ denotes the hops between user i's access LEO satellite and its offloading LEO satellite. When the user offloads its task to its access LEO satellite, the latency of the L-L link does not exist, that is $t^S_{\hat{m},m+1} = 0$.

G-L link latency G-L link latency includes three parts: G-L transmission latency, G-L propagation latency and computation latency.

G-L transmission latency. G-L transmission latency denotes the latency of the user transmitting its task to the GEO satellite. We define G-L transmission latency as $t^{L\_G}_{i,g}$, $t^{T\_G}_{i,g}$ is calculated by,

$$t^{T\_G}_{i,g} = \frac{o_i}{r^G_{\hat{m},g}}, \tag{14}$$

where $r^G_{\hat{m},g}$ indicates the transmission rate from user i's access LEO satellite $\hat{m}$ to the GEO satellite g.

G-L propagation latency. It defined as the time latency for the user to propagate its tasks to the GEO satellite, including U-L propagation latency and G-L propagation latency, we define it as the G-L propagation latency as $t^{P_G}_{i,g}$, $t^{P_G}_{i,g}$ is calculated by,

$$t^{P\_G}_{i,g} = \frac{d^G_{\hat{m},g}}{c}, \tag{15}$$

where $d^G_{\hat{m},g}$ indicates the distance between the user i's access LEO satellite $\hat{m}$ to the GEO satellite g.

Computing latency. The computing power of GEO satellite is set as f^G_g (GHz), and the computing latency $t^{C\_G}_{i,g}$ of the user i in GEO satellite g is calculated by,

$$t^{C\_G}_{i,g} = \frac{x_i}{f^G_g}, i \in \mathcal{I}. \tag{16}$$

3.4 Offloading Model

when user offload its task to the LEO satellite m, we define the network's LER, the duration and the latency as $\mathcal{P}^S_{i,m}$, $\mathcal{D}^S_{i,m}$, and $\mathcal{T}^S_{i,m}$, respectively. According to Eqs.(4), (4), $\mathcal{P}^S_{i,m}$ is calculated as,

$$\mathcal{P}^S_{i,m} = \mathcal{P}^U_{i,\hat{m}} + (1 - \mathcal{P}^U_{i,\hat{m}}) \mathcal{P}^S_{m,m+1}. \tag{17}$$

According to Eqs.(4),(4), $\mathcal{D}^S_{i,m}$ and $t^S_{i,m}$ is calculated as follow,

$$\mathcal{D}^S_{i,m} = \mathcal{D}^U_{i,\hat{m}} + \mathcal{D}^S_{m,m+1} n_{\hat{m},m}. \tag{18}$$

$$\mathcal{T}^S_{i,m} = t^S_{i,\hat{m}} + t^S_{\hat{m},m}. \tag{19}$$

Since the calculated result data is generally much smaller than the original data, the transmission latency of overcomes are ignored. However, the distance between the user and the LEO satellite is long, so the propagation latency cannot be ignored.

When the user offloads its task to the GEO satellite g, we define the network's LER as $\mathcal{P}^G_{i,g}$, the duration as $\mathcal{D}^G_{i,g}$ and the latency as $\mathcal{T}^G_{i,g}$, respectively. According to Eqs.(),(), the $\mathcal{P}^g_{i,g}$ is calculated as follow,

$$\mathcal{P}^G_{i,g} = \mathcal{P}^U_{i,\hat{m}} + (1 - \mathcal{P}^U_{i,\hat{m}}) \mathcal{P}^S_{\hat{m},g}. \tag{20}$$

According to Eqs.(),(), Eqs.(),(), Eqs.(),(), the $\mathcal{D}^g_{i,g}$ and $t^g_{i,g}$ is calculated as follow,

$$\mathcal{D}^G_{i,g} = \mathcal{D}^U_{i,\hat{m}} + \mathcal{D}^G_{\hat{m},g}. \tag{21}$$

$$\mathcal{T}^G_{i,g} = t^S_{i,\hat{m}} + t^G_{\hat{m},g}. \tag{22}$$

As mentioned above, the propagation latency of the G-L link cannot be ignored. In addition, Adaptive Modulation and Coding technology and adaptive transmission technology of power control are adopted to offset short-term attenuation or interference that may occur in the link[2]. Adaptive transmission technology uses an efficient spectrum transmission scheme under normal conditions and switches to a power-saving scheme in bad weather based on historical selection and link quality to offset attenuation or interference [][].

4 PROBLEM FORMULATION

4.1 Link Quality Analysis

Mobility of LEO satellites affects link quality by influencing the duration and latency of the satellite. To specify the variation in link quality, we discuss four scenarios, as shown in fig.. When the calculation time required by the user task is less than the LEO satellite coverage time, the calculation offloading of the task can be completed within the initial coverage area of the LEO satellite and returned directly to the user. Otherwise, the LEO satellite will be far away from the user, so the computing unloading task completed by the satellite should be returned to the user through other LEO satellites as intermediate relays. In this case, the delay and link stability of the user's computing unloading process will change, resulting in a change in link quality cost. In addition, the effect of satellite movement on the link failure rate is ignored because the data returned is small. In order to clarify the impact of LEO satellite high-speed motion on link stability and delay, four scenarios are divided according to whether the calculation time required for offloading the task is greater than the time required for the current LEO satellite to cover the task, as shown in Figure 4.1.

As shown in fig.(4), in that case, the user i's computation outcome can be returned directly. When $t = 1$, the computation result is returned to the user through the U-L link. The link duration includes U-L link duration $\mathcal{D}_{i,m_3}$ and the link latency include U-l link latency t^S_{i,m_3}.

As shown in Fig.(4), in that case, the user i's computation outcome must be returned through the m_3, m_2. When $t = 3$,

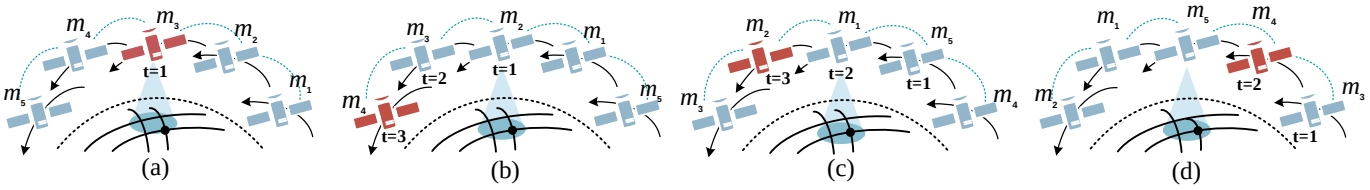


Fig. 2: Four scenarios of LEO satellites' mobility.

the link duration includes U-L link duration $\mathcal{D}^U_{i,m_2}$ and L-L link duration $\mathcal{D}^U_{m_2,m_4}$, the link latency include U-L link latency t^S_{i,m_2} and L-L link latency $t^S_{m_2,m_3}$, $t^S_{m_3,m_4}$.

As shown in Fig.(4), in that case, the user i's computation outcome must be returned through the m_6. When $t = 2$, the link duration includes U-L link duration $\mathcal{D}^U_{i,m_6}$, L-L link duration $\mathcal{D}^S_{m_6,m_4}$, $\mathcal{D}^S_{m_4,m_3}$, the link latency include U-L link latency t^S_{i,m_6} and L-L link latency $t^S_{m_6,m_4}$, $t^S_{m_4,m_3}$.

Based on the aforementioned analysis, let us define $\tilde{m}$ to indicate user i's accessible LEO satellite when offloading its task. Then, given user i and LEO satellite m for offloading user i's task, we use $D^S_{i,m}$ to represent the influence of LEO satellite mobility on link duration, $D^S_{i,m}$ is calculated as,

$$D^S_{i,m} = \begin{cases} \mathcal{D}^S_{i,m}, & \tilde{m} = m', \\ \mathcal{D}^S_{i,m} + \mathcal{D}^S_{m,m+1} n_{m',\hat{m}}, & \tilde{m} \neq m'. \end{cases} \tag{23}$$

We use to represent the influence of LEO satellite mobility on link latency, $T^S_{i,m}$ is calculated as,

$$T^S_{i,m} = \begin{cases} \mathcal{T}^S_{i,m}, & \tilde{m} = m', \\ \mathcal{T}^S_{i,m} + (t^{T\_S}_{m,m+1} + t^{P\_S}_{m,m+1}) n_{m',\hat{m}}, & \tilde{m} \neq m'. \end{cases} \tag{24}$$

Since the size of the computation results is extremely small, we ignore the link error rate incurred by returning the computation results.

4.2 LQCO Problem

We use $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,M}\}$ to represent the user's offloading decision for each LEO satellite, $a_{i,m} \in \{0,1\}$ indicates whether the user's computing tasks are offloaded to LEO satellite m. If both $a_{i,m}$ are zero, then the task is computed in the GEO satellite.Use $Q^S_{i,m}$ and $Q^G_{i,g}$ respectively to represent the link quality cost of the user offloading the task to the LEO satellite, and the link quality cost of the user offloading the task to the GEO satellite, respectively, $Q^S_{i,m}$ is calculated as:

$$Q^S_{i,m} = \varpi_\pi \mathcal{P}^S_{i,m} + \varpi_\gamma D^S_{i,m} + \varpi_t T^S_{i,m}, \tag{25}$$

where ϖ_π, ϖ_γ and ϖ_t represent the weights of latency, link error rate, and link duration, respectively. Similarly, the $Q^G_{i,g}$ is calculated as,

$$Q^G_{i,g} = \varpi_\pi \mathcal{P}^G_{i,g} + \varpi_\gamma \mathcal{D}^G_{i,g} + \varpi_t \mathcal{T}^G_{i,g}. \tag{26}$$

According to equations (4.3), (4.24) and (4.25), the cost of SECN link quality is calculated as,

$$f(a_{i,m}) = \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{M}} [a_{i,m} Q^S_{i,m} + (1 - a_{i,m}) Q^G_{i,g}]. \tag{27}$$

On the premise that satellite computing resources are constrained and link quality costs are minimized, the computational offloading problem is expressed in the following form,

$$\min_{A_i} \quad f(a_{i,m}), \tag{28}$$

$$\text{s.t.} \quad \sum_{m \in \mathcal{M}} a_{i,m} \leq 1, i \in \mathcal{I}, \tag{28a}$$

$$a_{i,m} \in \{0,1\}, \forall i, m, \tag{28b}$$

$$\sum_{i \in \mathcal{I}} a_{i,m} x_i \leq Z_m, m \in \mathcal{M}, \tag{28c}$$

Among them, $f(a_{i,m})$ is the link quality cost. (C1-C2) Ensure that each offload task can only be processed locally or on one satellite; (C3) represents the computing resource constraint, where, represents the maximum computing resource of the satellite; Problem (4.27) involves 0-1 integer variables and is NP-hard in the case of matrix indeterminate. Next, this chapter reformulates the optimization problem as an MDP and designs a DRL algorithm to obtain the optimal strategy.

5 APPROACH DESIGN

5.1 Markov Process

According to the problem in (28), we form the offloading problem in SECN as a Markov decision process (MDP). More details about our MDP are given as below.

State $\mathcal{S}$ In time slot t, we define the state $s(t) \in \mathcal{S}$ of the environment as $S(t) = \{o(t), \mathcal{P}(t), \mathcal{D}(t), \mathcal{T}(t)\}$, where $o(t) = \{o_1(t), \ldots, o_i(t), \ldots, o_I(t)\}$ represents the task size of users, $t^R(t) = \{t^R_{1,\hat{m}}(t), \ldots, t^R_{i,\hat{m}}(t), \ldots, t^R_{I,\hat{m}}(t)\}$ represents the task size of users denotes the remaining coverage time of LEO satellite m to user i,

Action $\mathcal{A}$: In time slot t, we define the action $a(t) \in \mathcal{A}$ as users' offloading decisions, such as $a(t) = \{a_1(t), \ldots, a_i(t), \ldots, a_I(t)\}$. Since each user has $M + 1$ possible offloading locations, the entire action space contains $(M + 1)^I$ actions. However, it may cause disaster of dimensionality when I and M are large.

Reward $\mathcal{R}$: In time slot t, the access satellite will get an immediate reward $R(t) \in \mathcal{R}$ after taking action $a(t) \in \mathcal{A}$ according to $S(t) \in \mathcal{S}$. To minimize users' task offloading cost, the reward function $R(t)$ can be defined according to Eq. (28) as the negative of the offloading cost for all users, and expressed as follows,

$$R(t) = -f(a_{i,m}(t)). \tag{29}$$

Due to the mobility of LEO satellites and the large number of users, the SECN is dynamic and high-dimensional, and it is difficult to make offloading decisions by traditional methods. Therefore, we use LC-LSDQ to solve the offloading problem.

5.2 Reinforcement Learning Process

In traditional Q-learning, the action value function is put into the Q-table, and the action with the maximum function can be selected. However, in the satellite environment, the state space and the action space are high dimensional, and it is very difficult to put all the action value functions into the Q-table. Therefore, we uses deep networks to evaluate action value functions.

Fig 4.2 shows the user's interaction with the environment. In the decision stage, the user perceives state $S(t)$ from the environment, i.e., task state, coverage time state, transmission rate state, and computation resource state. According to the given policy, the selected action $a(t)$ works in the current environment, that is, the user makes a offloading decision. The environment $S(t)$ is then transformed into a new state $S(t+1)$.

The policy function is defined as $\iota : S(t) \times A(t) \to [0,1]$, where $\iota(S(t), A(t))$ is in sate $S(t)$, the probability of selected $A(t)$. There are two value functions to represent the feedback of each decision, including the state value function $V^{\iota}(s)$ and the action state value function $Q^{\iota}(s,a)$, the $Q^{\iota}(s,a)$ is expressed as follows,

$$Q^{\iota}(s,a) = E^{\iota}[\sum_{k=0}^{\infty} \gamma_q^k R(t+k+1|S(t)=s, A(t)=a)]. \quad (30)$$

Where $E^{\iota}[*]$ is the mathematical expectation under the state transition probability P and $\gamma_q^k \in [0,1]$ representing the discount factor weighing the immediate reward and the long-term reward.

In addition, DQL uses time-difference methods to evaluate $Q(s,a)$, i.e.,

$$\begin{aligned} Q^{\iota}(s,a) \leftarrow & Q^{\iota}(s,a) \\ & + \alpha_q(R(t) + \gamma_q^k \max_a Q^{\iota}(s_0, a_o) - Q^{\iota}(s,a)). \end{aligned} \quad (31)$$

where α_q is the learning rate. Additionally, we use a deep network to evaluate the action value function, i.e., $Q(s,a,w) \approx Q(s,a)$, where w is the set of weights and biases in the deep network. In each learning iteration, the deep network is trained to minimize the loss function $L(w)$, which is expressed as,

$$\begin{aligned} L(w) = E[(R(t+1) & \\ & + \gamma_q^k \max_a Q^{\iota}(s(t+1), a(t+1), w(t+1)) \\ & - Q^{\iota}(s(t), a(t), w(t))^2]. \end{aligned} \quad (32)$$

The empirical playback mechanism and fixed target deep network mechanism in DQL are very suitable for satellite edge computing networks [53].

(1) Experience playback. Due to the uncertainty and complexity of the satellite environment, the experience playback mechanism can store a large number of historical experiences and randomly select a set of experiences to train the deep network. This allows the algorithm to learn from past experience rather than just relying on current experience, which improves the duration and generalization of the algorithm. In addition, the experience playback mechanism can help the algorithm make full use of the limited LEO satellite resources and improve the utilization efficiency of resources, so that users can make more efficient offloading decisions.

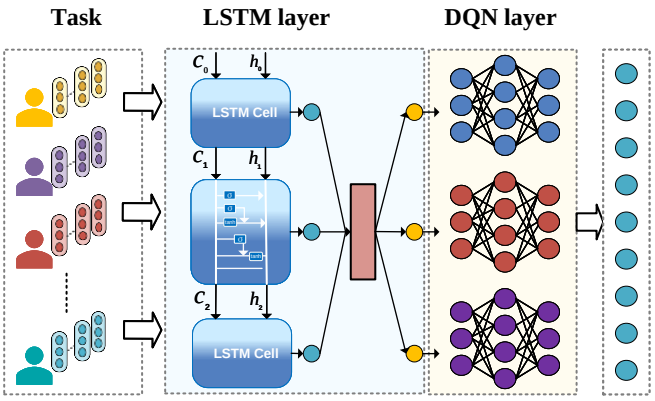


Fig. 3: Four scenarios of LEO satellites' mobility.

(2) Fixed target deep network. In satellite edge computing network, because of the dynamic change of environment and the uncertainty of link quality, maintaining a fixed target network can make the algorithm better cope with the environment change and avoid the induration in the training process. Therefore, the fixed target deep network mechanism can be effectively applied to the computing offload task in satellite edge computing network, and improve the performance and duration of the algorithm.

Therefore, DQL is shown as algorithm 4.1, where $\varepsilon - greedy$ policy [10] is used for exploration.

Algorithm 1 *ATO-SLA*

Input: $O(t), Q^{local}(t), Q^{edge}(t), E^{CF}(t), G, T, Y, Z$
Output: $X(t)$

```
Randomly initialize the network parameters θ and μ.
Initialize the actor network parameter: x(t).
for t = 1, 2, ..., G do
    for i = 1, 2, ..., T do
        Collect x(t) into the LSTM network.
        According to Eq. (4), obtain the data output from
        the LSTM layer and then fed into the DQN.
        According to Eq. (4), obtain the reward.
        Based on Equation (10), select a and obtain the
        reward r
        Update Q according to Eq.().
    end for
end for
Input the system state of time slot t into the trained actor
network, and then the offloading decision X(t) can be
obtained.
```

5.3 LSTM

The LSTM (Long Short-Term Memory) architecture is a special type of Recurrent Neural Network (RNN) structure. RNNs can utilize historical information to process and predict sequential data. The introduction of LSTM has addressed the long-standing issue of vanishing gradients in RNNs. Currently, it has achieved significant progress in the field of natural language processing [20]. The structure of LSTM is shown in Figure 4. In Figure 4, the forget gate determines which information to discard, the input gate decides how much new information to incorporate, and

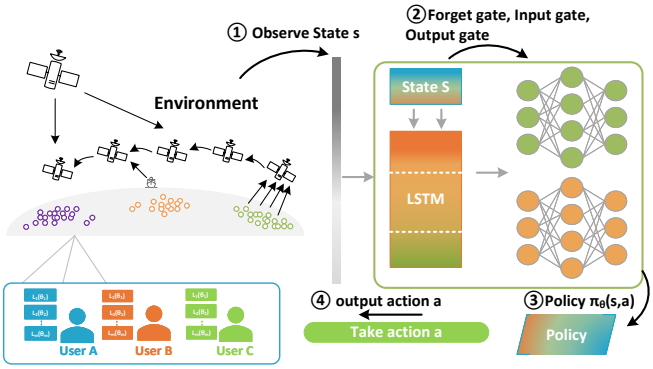


Fig. 4: Four scenarios of LEO satellites' mobility.

the output gate determines the data to be output from the current LSTM unit to the next unit.

Eq.(14) represents the forget gate. The forget gate determines which information should be forgotten from the previous state of the satellite edge computing network. It is calculated by applying the sigmoid function to the weighted sum of the previous output and the current input.

$$F_t = sigmoid(W_F \times [h_{t-1}, x_t] + b_F), \tag{33}$$

where W_F represents the weight matrix of forget gate, h_{t-1} denotes the output of LSTM unit at time step $t-1$, b_F and x_t represents the bias term matrix of forget gate and the input at the current time step, respectively.

Eq.(16) represents the input gate. The input gate determines the relevance of the new input values. Similarly to the forget gate, the input gate is calculated by applying the sigmoid function to the weighted sum of the previous output and the current input.

$$I_t = sigmoid(W_I \times [h_{t-1}, x_t] + b_I). \tag{34}$$

where W_I represents the weight matrix of input gate, b_I represents the bias term matrix of input gate.

The new state of the satellite edge computing network added to the unit and its candidate value are denoted by $\overline{C_t}$ and $C(t)$, respectively. The $\overline{C_t}$ is computed by applying the hyperbolic tangent function $\tanh$ to the weighted sum of the previous output and the current input.

$$\overline{C_t} = \tanh(W_C \times [h_{t-1}, x_t] + b_C). \tag{35}$$

$$C_t = F_t C_{t-1} + I_t * \overline{C_t}. \tag{36}$$

where W_C represents the weight matrix of satellite edge computing network state, and b_C represents the matrix of satellite edge computing network state bias term.

The output gate is denoted by h_t, and it is computed by applying the sigmoid function to the weighted sum of the previous output and the current input.

$$h_t = sigmoid(W_H \times [h_{t-1}, x_t] + B_H) \times \tanh(C_t). \tag{37}$$

where W_H represents the weight matrix of output gate, B_H represents the bias term matrix of output gate.

The LSTM structure employs three gates to determine the retention level of the input data sequence, enabling the prediction of the future based on historical information. By incorporating the LSTM structure into the algorithm, the experience from historical information is leveraged to help users estimate the behaviors of other users. This is particularly beneficial in scenarios with unstable link quality, as it enables the derivation of better offloading solutions and enhances the performance of the MJADA algorithm.

6 PERFORMANCE EVALUATION

We conduct the experiment on a device with an Intel Core i5-12600K processor (2.5 GHz, 4 cores) and $16GB$ of RAM. Our Python 3.10 implementation simulates a satellite edge computing network with 5 LEO satellites and 1 GEO satellite. Similar to [], [], the number of users varied from 20 to 100 with a step of 20. Each user's task size is randomly generated from $[2, 10]$Mb. Following in [], [], three primary communication error were considered in our scenarios: U-L LER $\alpha_{i,\hat{m}}$ of 0.1, L-L LER $\alpha_{\hat{m},m}$ of 0.05, and G-L LER $\alpha_{\hat{m},g}$ of 0.07. The weighting factors ϖ_π, ϖ_γ, and ϖ_t for Eq.() were set to 0.3, 0.4, and 0.3, respectively.

In addition, We set the input dimension to 10 for all neural network models. The LSTM model contains a single hidden layer with 64 neurons. For the DNN model, we implement two hidden layers, each with 300 neurons and ReLU activation functions, plus one output layer. All models were trained using the Adam optimizer with a learning rate of 0.0005. The learning rate decreases by 10% every 100 epochs. To ensure model convergence, we run the training for 1000 epochs. Following [], [], we determine the GEO and LEO satellites parameters by combining previous research with our experimental results. Table 4 lists these parameter settings.

Benchmarks. Four representative methods are used to compare the performance with LQ-LSDQ.

- *LQ-DQ*: This method uses a DNN model with three fully connected layers. It processes current satellite states and task requirements to determine offloading decisions but does not consider temporal features.
- *LQ-LS*: This approach employs an LSTM model to focus on temporal patterns in satellite movement and link quality. It makes sequential offloading decisions based on historical satellite state information.
- *CE*: This baseline approach offloads all tasks to LEO satellites. It represents a fixed strategy without considering dynamic network conditions or task requirements.
- *CG*: This method offloads all tasks to GEO satellites. Similar to CE, it uses a fixed strategy but targets GEO satellites instead of LEO satellites.

Performance Metrics. In general, we employ three metrics to evaluate the effectiveness of all the approaches.

- *Average cost*. According to Section 4, the average cost combines both LER, communication duration and communication latency. The LER depends on data transmission between users and satellites, while the communication latency relates to task processing and data transmission. We calculate the average cost as Eq(4).
- *Average latency*. The average latency measures the total time from task initiation to completion, including both transmission and processing times.

TABLE 2: SYSTEM PARAMETERS

Parameter	Value
Distance between adjacent LEO satellites $d^S_{m,m+1}$	$4481km$
User i's distance to LEO satellites $d^U_{i,\tilde{m}}$	$784km$
Radius of the Earth re	$6371km$
Computational density of the task w_i	$3Mcycles/bit$
LEO satellite $\tilde{m}$'s linear velocity $v_{i,\tilde{m}}$	$27000km/h$
Speed of light c	$3 \times 10^8 m/s$
User i's transmit power $p^U_{i,\tilde{m}}$	$23dBm$
Channel bandwidth $b^U_{i,\tilde{m}}$ between user i and accessible satellite $\tilde{m}$	$20MHz$
Channel gain $h^U_{i,\tilde{m}}$	$-150dB$
Noise power $\sigma_{i,\tilde{m}}$	10^{-9}
Transmission rate between adjacent LEO satellites $r^S_{m,m+1}$	$40Mbps$
Computing power of local users f^U_i	$200MHz$
LEO satellite m_1's computing power f^S_{m1}	$1GHz$
LEO satellite m_2's computing power f^S_{m2}	$1.5GHz$
LEO satellite m_3's computing power f^S_{m3}	$2GHz$
LEO satellite m_4's computing power f^S_{m4}	$2.5GHz$
LEO satellite m_5's computing power f^S_{m5}	$3GHz$
Transmission rate from accessible satellite $\tilde{m}$ to the GEO satellite $r^G_{\tilde{m},g}$	$20Mbps$
Distance between accessible satellite and GEO satellite $d^G_{\tilde{m},g}$	$31305km$
Computing power of GEO satellite f^G_g	$50GHz$
User i's energy factor ε_i	$5 \times 10^{-26} J/Hz^3/s$
LEO satellite m's energy factor ε_m	$5 \times 10^{-32} J/Hz^3/s$
Threshold for ϑ_1, ϑ_2	10^{-6}
Weighting factor between latency and energy consumption λ	0.5

- *Task Success Rate, TSR [], []*. We define task success rate as the ratio of completed tasks to total submitted tasks. A task is considered successful when it meets both latency and reliability requirements. The success rate is calculated as,

$$TSR = \frac{N_{success}}{N_{total}}, \tag{38}$$

where $N_{success}$ is the number of successfully completed tasks and N_{total} is the total number of submitted tasks. We set the success threshold based on both completion time (must be within deadline) and quality of service requirements.

Experiment Sets. To assess the approaches across diverse scenarios, we designed five experimental configurations by varying key parameters, as detailed in Table 2.The varied parameters include the number of edge servers (S), number of subtasks (U), sensitive distance (Rsen), maximum transmit power (p max), and choice probability. The experiment under each set is repeated 100 times and the average is taken as the final result.

6.1 Convergence analysis

Fig.5 illustrates the loss variations of *LQ-LSDQ*, *LQ-DQ*, and *LQ-LS* during training. As the number of epochs increases, *LQ-LSDQ* shows a rapid loss reduction in the early training stages and stabilizes around epoch 120. In contrast, *LQ-DQ* and *LQ-LS* exhibit a more gradual decline, with *LQ-LS* converging significantly slower than the other two. *LQ-LSDQ* integrates LSTM and DQN, leveraging LSTM's strong memory capability for time-series data while using DQN to optimize offloading decisions. *LQ-DQ* lacks LSTM's sequential memory, which may limit its ability to handle long-term dependencies in link quality variations. Although *LQ-LS* processes time-series data, it lacks reinforcement learning-based decision-making, making it less effective in optimizing offloading under dynamic link quality compared to *LQ-LSDQ* and *LQ-DQ*.

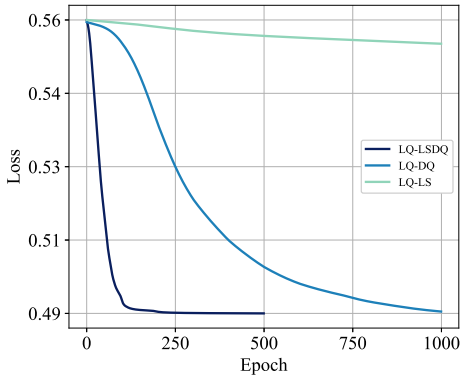
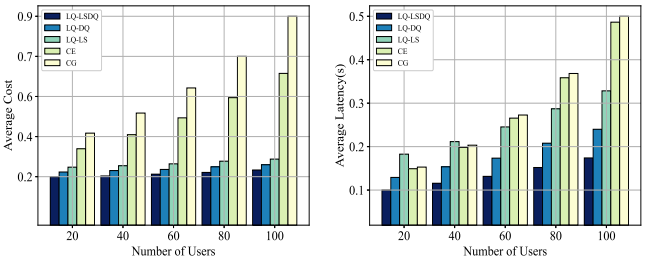


Fig. 5: Convergence analysis.



(a) Costs $v.s.$ Number of Users (b) Latency $v.s.$ Number of Users

Fig. 6: Impact of users.

6.2 Effect of Number of Users (Set #1)

Fig.6 shows the impact of different user numbers on the average cost and latency across five methods. In **Fig.6(a)**, as the number of users increases, the average cost rises for all methods. *LQ-LSDQ* achieves the lowest average cost, especially when the user count reaches 100, demonstrating superior cost control. *LQ-DQ* incurs slightly higher costs than *LQ-LSDQ* but still outperforms the two traditional methods (*CE* and *CG*). *LQ-LS* has higher costs than *LQ-LSDQ* and *LQ-DQ*, while *CE* and *CG* perform the worst, with costs rising sharply as the user count increases.

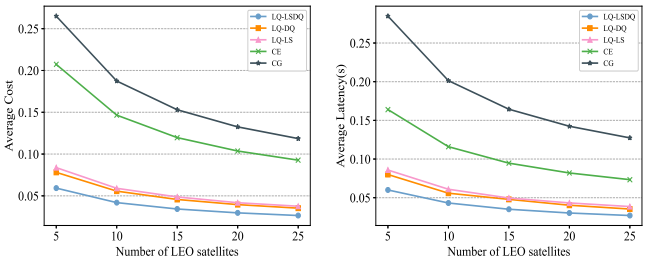
Fig.6(b) shows that average latency increases with the number of users. *LQ-LSDQ* achieves the lowest latency, particularly at 100 users, where it significantly outperforms other methods. *LQ-DQ* exhibits slightly higher latency than *LQ-LSDQ* but remains more efficient than the other methods. When the user count is low (e.g., 20 or 40), *LQ-LS* experiences higher latency due to its lower processing capability and decision-making efficiency. However, as the number of users increases, *LQ-LS* gradually achieves lower latency than *CE* and *CG*. This improvement occurs because *LQ-LS* better adapts to the time-series variations in link quality, leveraging its ability to handle time-dependent data. In contrast, *CE* and *CE* lack dynamic decision-making capabilities, leading to faster latency growth in multi-user scenarios.

6.3 Effect of Number of LEO Satellites (Set #2)

Fig.7 illustrates the impact of different LEO satellite numbers on the average cost and latency across five methods.

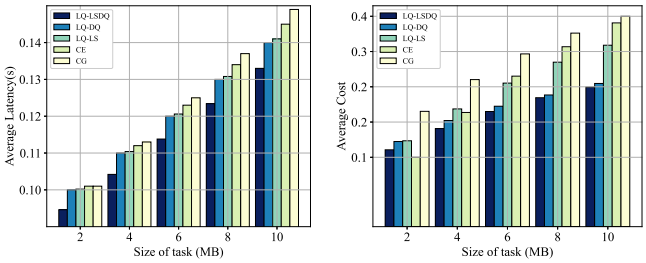
TABLE 3: Experiment Sets

	Number of users	Number of LEO satellites	Size of task	BER of U-L link
Set #1	20,40,60,80,100	15	6	0.15
Set #2	60	5,10,15,20,25	6	0.15
Set #3	60	15	2,4,6,8,10	0.15
Set #4	60	15	6	0.05,0.10,0,15,0.20,0.25



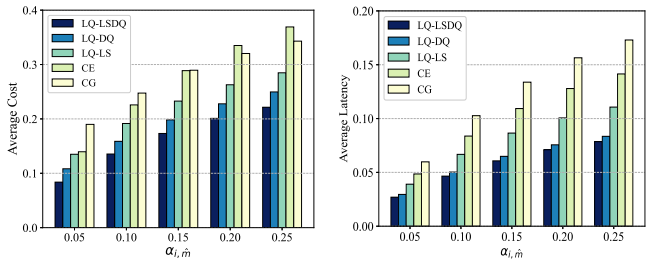
(a) Costs $v.s.$ LEO Satellites (b) Latency $v.s.$ LEO Satellites

Fig. 7: Impact of LEO satellites.



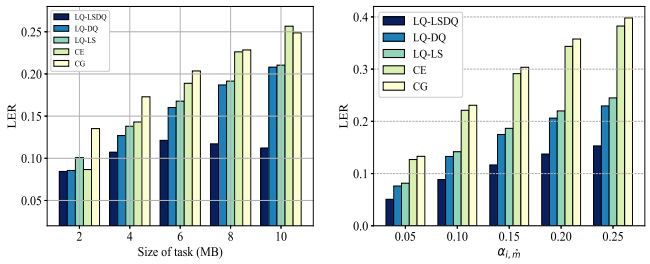
(a) Costs $v.s.$ Size of task (b) Latency $v.s.$ Size of task

Fig. 8: Impact of size of task.



(a) Costs $v.s.$ $\alpha_{i,\hat{m}}$ (b) Latency $v.s.$ $\alpha_{i,\hat{m}}$

Fig. 9: Impact of $\alpha_{i,\hat{m}}$.



(a) LER $v.s.$ Size of task (b) LER $v.s.$ $\alpha_{i,\hat{m}}$

Fig. 10: Influence on LER.

Fig.7(a) shows that as the number of LEO satellites increases, the average cost decreases for all methods. *LQ-LSDQ* achieves the lowest average cost, significantly outperforming other methods, especially when the number of LEO satellites reaches 5. *LQ-DQ* incurs slightly higher costs than *LQ-LSDQ* but still performs well, with a clear downward trend as the satellite count increases. *LQ-LS* has higher costs than *LQ-LSDQ* and *LQ-DQ* but remains more efficient than traditional methods. As the number of satellites grows, its cost gradually decreases. *CE* and *CG* exhibit the highest costs, with a slower reduction compared to other methods, particularly when more satellites are available.

Fig.7(b) presents the changes in average latency as the number of LEO satellites varies. The latency trends align closely with the cost trends. *LQ-LSDQ* achieves the lowest latency, while *LQ-DQ* has slightly higher latency but benefits from a significant reduction as the number of satellites increases. *LQ-LS* experiences higher latency than *LQ-LSDQ* and *LQ-DQ* but remains lower than *CE* and *CG*. *LQ-LSDQ* efficiently manages bandwidth and computational resources in large-scale satellite systems, reducing unnecessary transmissions and processing tasks, minimizing network load, and improving resource utilization. *LQ-DQ* lacks the time-dependent processing capability of *LQ-LSDQ* but still optimizes bandwidth allocation effectively, reducing both cost and latency. *LQ-LS* is less efficient in resource optimization, leading to higher bandwidth and computational resource consumption, which results in higher costs and latency. *CE* and *CG*, relying on traditional optimization methods, struggle to handle large-scale dynamic resource demands, leading to inefficient bandwidth utilization and resource allocation.

6.4 Effect of Size of Task (Set #3)

Fig.8 illustrates the impact of task size variations from 2MB to 10MB on the average cost and latency across five methods. In **Fig.8(a)**, as the task size increases, the average cost rises for all methods. Larger tasks demand more computational and bandwidth resources, leading to higher costs. Among the methods, *LQ-LSDQ* achieves the lowest average cost, with a relatively small increase as task size grows. *LQ-DQ* incurs slightly higher costs than *LQ-LSDQ* but maintains a steady and moderate increase. In contrast, *LQ-LS* exhibits a sharper cost rise, surpassing both *LQ-LSDQ* and *LQ-DQ*. *CE* and *CG* perform the worst, with the highest costs and the steepest growth, especially when task sizes become larger. When the task size exceeds 50MB, *LQ-LSDQ* effectively allocates resources on demand by combining LSTM-based link quality prediction with DQN-driven dynamic decision-making. However, *LQ-DQ* struggles with decision efficiency as the state space expands, leading to resource wastage. *LQ-LS* demonstrates a linear cost increase with task size, indicating its lack of dynamic resource optimization.

Fig.8(b) shows that average latency increases significantly with task size, particularly when the task size reaches 10 MB. *LQ-LSDQ* achieves the lowest latency, with a relatively small increase, demonstrating strong latency control. *LQ-DQ* exhibits slightly higher latency than *LQ-LSDQ* but maintains a stable growth trend, resulting in better overall performance. However, *LQ-LS* experiences a larger latency increase, revealing performance bottlenecks when handling larger tasks. *LQ-LSDQ* benefits from temporal modeling, reducing the frequency of environment interactions, and

optimizing decision-making to prevent link congestion. On the other hand, *LQ-DQ* experiences noticeable latency fluctuations, particularly when task sizes exceed 70MB. This degradation occurs because *LQ-DQ* frequently interacts with the environment to update its Q-table, leading to a surge in communication overhead when the network becomes congested. *LQ-LS* exhibits an exponential increase in latency as task size grows, since its reliance on historical link quality prevents real-time adaptation to network fluctuations. Meanwhile, *CE* and *CG* methods suffer from the highest latency and the sharpest growth, demonstrating poor overall performance as task size increases.

6.5 Effect of BER of U-L Link (Set #4)

Fig.9 illustrates the impact of different U-L link failure rates $\alpha_{i,\hat{m}}$ on the average cost and latency across five methods. In **Fig.9(a)**, as $\alpha_{i,\hat{m}}$ increases, link quality deteriorates, leading to a rise in average cost for all methods. *LQ-LSDQ* consistently achieves the lowest cost, with the smoothest cost curve as $\alpha_{i,\hat{m}}$ increase. Its advantage becomes more pronounced when the $\alpha_{i,\hat{m}}$ exceeds 0.15, demonstrating its ability to maintain cost efficiency even under poor link conditions. *LQ-DQ* maintains a lower cost when $\alpha_{i,\hat{m}}$ is below 0.15. However, beyond this threshold, its cost rises sharply. This increase occurs because DQN's exploration efficiency declines at high failure rates, resulting in more suboptimal decisions. *LQ-LS* incurs higher costs than *LQ-LSDQ* and *LQ-DQ*, with a steeper cost increase as $\alpha_{i,\hat{m}}$ grows, since LSTM struggles to adapt to sudden link quality fluctuations, frequent data retransmissions lead to severe resource wastage. *CE* and *CG* exhibit the highest costs, with a significant upward trend as failure rates increase. Both methods fail to account for dynamic link quality changes, resulting in excessive retransmission overhead.

In **Fig.9(b)**, as $\alpha_{i,\hat{m}}$ increases, all methods experience a noticeable rise in average latency. *LQ-LSDQ* achieves the lowest latency and maintains a relatively small increase due to its temporal modeling, which reduces interaction frequency with the environment and prevents link congestion. In contrast, *LQ-DQ* exhibits large latency fluctuations, with a sharp degradation when $\alpha_{i,\hat{m}}$ exceeds 0.20. This occurs because *LQ-DQ* frequently interacts with the environment to update its Q-table, causing a surge in communication overhead under unstable link conditions. *LQ-LS* experiences higher latency, with a steep increase as $\alpha_{i,\hat{m}}$ rises. The prediction error of *LQ-LS* becomes significantly larger at high failure rates, increasing the likelihood of task scheduling failures. *CE* and *CG* suffer from an exponential latency increase since both methods rely on stable link transmissions. When $\alpha_{i,\hat{m}}$ exceeds 0.10, their performance deteriorates rapidly.

6.6 Effect of factors on LER

Fig.(10) illustrates the impact of different task sizes and U-L LER $\alpha_{i,\hat{m}}$ on the LER across five methods. LER represents the total failures caused by link disruptions. The experimental settings for **Fig.10(a)** and **Fig.10(b)** correspond to **Set#3** and **Set#4**, respectively.

In **Fig.10(a)**, as the task size increases, LER rises significantly for all methods, indicating that larger tasks require more bandwidth and computational resources, increasing the likelihood of link errors. *LQ-LSDQ* maintains the lowest LER and exhibits the smallest increase as task size grows, demonstrating strong stability. This advantage arises because *LQ-LSDQ* can learn and adapt to temporal patterns and feature interactions, allowing it to handle dynamic and complex link conditions more effectively. *LQ-DQ*, lacking temporal modeling, may perform worse than *LQ-LSDQ* in highly dynamic environments. *LQ-LS* shows a higher LER than *LQ-LSDQ* and *LQ-DQ*, with a steeper increase, especially for larger tasks. *CE* and *CG*, representing fixed policies, fail to adapt to changing link conditions and user demands, resulting in consistently higher LER.

In **Fig.10(b)**, as $\alpha_i, \hat{m}$ increases, link quality deteriorates, leading to a noticeable rise in LER for all methods. This trend highlights the increased frequency of link errors under poor link conditions. *LQ-LSDQ* consistently achieves the lowest LER, as it effectively adapts to varying link conditions and user demands, reducing overall link errors. *LQ-DQ* exhibits a slightly higher LER than *LQ-LSDQ* but still outperforms other methods. Even under high LER, *LQ-DQ* maintains a relatively low LER, indicating its strong performance in handling static features. However, it may struggle when link quality fluctuates dynamically. *LQ-LS* incurs a higher LER than both *LQ-LSDQ* and *LQ-DQ*, with a steeper increase as $\alpha_i, \hat{m}$ grows. Although *LQ-LS* benefits from temporal dependency modeling, it lacks the feature processing capabilities of DQN, which affects its performance in highly variable link conditions. *CE* experiences higher failure rates due to inherent limitations of LEO links and their increased error rates under high BER conditions. In *CG*, while GEO satellite links remain more stable, higher latency and potential congestion may degrade overall performance.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.