2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall) | 979-8-3315-1778-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VTC2024-Fall63153.2024.10758053

# Joint Optimization of Computing and Routing in LEO Satellite Constellations with Distributed Deep Reinforcement Learning

Shaohua Xia[1], Jiangtao Luo*[2], Yongyi Ran[3]
*School of Communication and Information Engineering*
*Chongqing University of Posts and Telecommunications*, Chongqing, China
1805519393@qq.com[1], luojt@cqupt.edu.cn[2], ranyy@cqupt.edu.cn[3]

***Abstract*—Earth observation satellites in low earth orbit (LEO) collect a large amount of image data daily, while space-to-ground links have become the major bottleneck for data transmission due to the limited bandwidth. Existing approaches focus on exploring more efficient routing strategies to achieve better data transmission but still struggle to keep pace with the surging volume of observed data. However, the advancement of onboard computing power has opened the possibility of processing data on satellites to reduce the transmitted data volume. This paper proposes a distributed deep reinforcement learning (DRL) algorithm to improve transmission efficiency by jointly optimizing computing and routing. Aiming to minimize task latency while considering the limitations of satellite storage resources, the problem is modeled as a partially observable Markov process (POMDP). An algorithm based on dueling double deep Q-Network (Dueling-DDQN) is proposed to achieve dynamic decision-making utilizing local and neighboring resource states. Furthermore, a method for dynamic optimization of backhaul destinations is proposed, using the pre-trained Q-network to estimate action values across multiple candidate destination satellites, thus enabling further optimization of data transmission without additional training. Simulation results indicate that the proposed algorithm achieves the lowest latency across various task loads compared to baseline methods.**

***Index Terms*—LEO satellite network, computing and routing, deep reinforcement learning, distributed algorithm.**

## I. Introduction

LEO satellite Constellations consisting of hundreds of satellites have been launched for frequent high-resolution Earth observations. Improvements in sensing technology, such as hyperspectral image (HSI) [1], have greatly enhanced data precision while resulting in a dramatic increase in the volume of observation data. Currently, each observation satellite produces approximately 1 TB data daily [2], which is transmitted in terms of raw data and takes up a lot of satellite bandwidth. Additionally, these data can not always be transmitted back to the ground timely due to the non-uniform distribution of ground stations. Therefore, how to transmit observation data efficiently to the ground has emerged as a crucial concern in Earth observation.

Many efforts have been devoted to exploring more efficient transmission mechanisms to address the above problems. Designing better routing strategies for LEO satellite constellations is one of the most direct ways to optimize the transmission throughput and delay [3], [4]. However, such approaches can not essentially reduce the volume of data to be transmitted and are struggling to keep pace with the surging increase in earth observation data. With the development of onboard computing capabilities, it is promising to reduce the data transmission burden by processing data onboard before transmitting [5]. For example, data can be compressed onboard to reduce the required transmission bandwidth. Also, tasks such as object detection and disaster forecasting [6] only need to transmit a few bytes of results after being processed onboard. Therefore, routing and onboard processing (i.e., computing) can be considered jointly to improve the transmission of Earth observation data.

However, joint computing and routing in LEO satellite networks face three primary challenges. **First, high dynamic environment**. Frequent topology changes due to high-speed movements, solar influences [7], and variant traffic distribution necessitating real-time decision-making. **Second, uneven distribution of resources**. Limited and varied onboard computing capabilities may require offloading data to other satellites for processing, influencing routing decisions based on computing capacity. **Third, high complexity in joint optimization**. LEO satellite constellation comprises hundreds of satellites with high-dimensional state space. At the same time, the joint decision of backhaul destination, computing satellite, and transmission paths leads to high-dimensional action space.

To address these problems, we propose a DRL based distributed optimization algorithm, to improve the transmission efficiency of Earth observation data by regulating the computing and routing in LEO satellite constellations jointly. Given the high complexity and high dynamics of the LEO satellite network, a Dueling-DDQN algorithm is proposed to obtain the joint strategies for the computing and routing of observation tasks under limited satellite storage. Extensive experiments are carried out, showing that the proposed algorithm can outperform the baseline algorithms. The main contributions of this paper are as follows:

1) Propose a distributed dueling double deep Q-Network (Dueling-DDQN) algorithm to obtain the joint decision of computing and routing. The joint optimization problem is modeled as a partially observable Markov

This work is supported by National Natural Science Foundation of China (No.62171072, No.U23A20275).

decision process (POMDP) to minimize the task delays under storage capacity constraints, and Dueling-DDQN is employed to solve the optimization problem, which can cope with the issues of high computational complexity and enable dynamic decision-making utilizing local observation.

2) Propose a method for dynamic optimization of backhaul destinations, using the pre-trained Q-network to estimate action values across multiple candidate destination satellites. By selecting the action with the highest value estimation among different destinations, this method achieves further optimization of data transmission without additional training.
3) Develop a system-level simulation environment to model the satellite observation task at the constellation scale. Different algorithms are evaluated under varying traffic loads. Simulation results show that the proposed algorithm has significant advantages in average delay compared to baseline algorithms.

## II. Related Works

### A. Routing Strategies for LEO Satellite Networks

Many routing strategies have been proposed to optimize data transmission on satellites. Tang et al. [3] proposed source-based and destination-based multipath cooperative routing algorithms to deliver data flow along multiple paths dynamically. In [8], routing schemes using a time-aggregated graph (TAG) were proposed to guarantee service transmission QoS. Additionally, work in [4] presented a load-balanced collaborative offloading (LBCO) strategy for balanced traffic distribution in data offloading. These centralized approaches rely on global information from LEO networks, which is not always suitable for LEO satellite networks due to their extensive communication overhead and delay.

Researchers have devised distributed methods to overcome the limitations inherent in centralized approaches. Zhang et al. [9] represented satellite networks as grid graphs, utilizing their unique topology to enhance distributed resource awareness. Study in [10], [11] introduced distributed routing algorithms based on DRL, where every satellite represents an agent observing information within single-hop neighbors.

### B. Computing and Transmission Joint Optimization

Some existing works investigated the computing and transmission joint optimization for satellite networks. Work in [12] investigated scenarios where satellites offer task processing capabilities to remote Internet of Things (IoT) devices and formulate an energy-efficient computation offloading and resource allocation algorithm. The study in [13] formulated a stochastic computation offloading problem to jointly optimize communication and computing resource allocation and computation offloading decisions and proposed a method combining deep reinforcement learning and Lyapunov optimization to solve this problem. In [14], an intelligent computing offloading scheme based on the deep deterministic policy gradient (DDPG) is proposed to allocate computing and transmission resources.

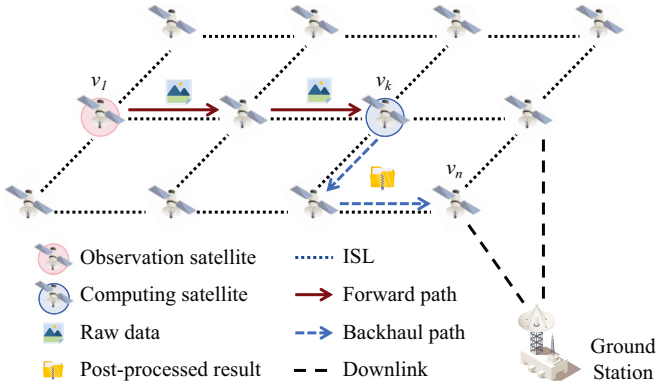



Fig. 1. The scenario of an observation task to be computed onboard and transmitted back to the ground.

These studies optimized the allocation of various resources. However, they primarily focused on terrestrial tasks and only considered satellites providing offloading services for their coverage areas, neglecting the allocation challenges of onboard computing tasks at the scale of the entire constellation.

## III. System Model

### A. Network Model

A constellation of LEO satellites forms a network with a bidirectional graph structure $G = \{V, E\}$, where $V$ denotes the set of satellites and $E$ the active inter-satellite links (ISLs). $V = \{v_1, v_2, ..., v_N\}$ represent the satellites, with each satellite $v_i$ characterized by its state at any time $t$ as computing capacity $c_i$ in floating point operations per second (FLOPs), occupied storage $m_i(t)$, and queue length for computing tasks $q_i^c(t)$.

The edges $E$ comprise directed ISLs between satellites. An edge $e_{i,j} \in E$ represents a directed link from $v_i$ to $v_j$, characterized at time $t$ by the link rate $r_{i,j}$, transmission queue length $q_{i,j}^t$, and delay $D_{i,j}(t)$.

### B. Task Model

Observation tasks for processing and transmission in LEO satellites are described by a tuple $(s, d, s')$, where $s$ is the initial data size, $d$ is the computational demand, and $s'$ is the post-processing data size.

These tasks can be divided into two primary categories: compression and inference. Compression tasks focus on reducing the raw data size and generally require lower computational power. The compressed data $s'$ is significantly reduced in volume and will be transmitted to ground stations. Inference tasks involve applying machine learning models for target recognition or disaster detection functions, demanding higher computational resources. The results of inference $s'$ are minimal, often at most a few kilobytes.

### C. Delay Model

As shown in Fig. 1, assuming a observation task is generated at $v_1$, with the destination satellite being $v_n$, the

pre-computation transmission path $P_1 = \{v_1, ..., v_k\}$, the computing satellite being $v_k$, the backhaul path being $P_2 = \{v_k, ..., v_n\}$, the task arrival time at each satellites are $t_1$, $t_2$, ..., $t_{k-1}$, $t_{k+1}$, ..., $t_n$ excluding the computing satellite $v_k$, $t_k$ represent the time when the task finishes computation at satellite $v_k$, and $t'_k$ is the time when the task arrive $v_k$. Assuming the speed of light is $\nu$. Thus, the task process involves four types of delays in the system:

**Propagation delay** $T_p$, which is primarily determined by the distance of each link:

$$T_p = \sum_{i=1}^{n-1} \frac{D_{i,i+1}(t_i)}{\nu} + \frac{D_n^g(t_n)}{\nu} \tag{1}$$

where $D_n^g$ represents the distance from $v_n$ to the connected ground station.

**Transmission delay** $T_t$, which is determined by the task data volume and link rate:

$$T_t = \sum_{i=1}^{k-1} \frac{s}{r_{i,i+1}} + \sum_{i=k}^{n-1} \frac{s'}{r_{i,i+1}} \tag{2}$$

**Queuing delay** $T_q$, which includes queuing delays for both computing and transmission, determined by the transmission queue of each link and the computing queue:

$$T_q = \sum_{i=1}^{n-1} \frac{q_{i,i+1}^t(t_i)}{r_{i,i+1}} + \frac{q_k^c(t'_k)}{c_k} + \frac{q_n^g(t_n)}{r_n^g} \tag{3}$$

where $r_n^g$ represents the downlink rate from $v_n$ to the ground station and $q_n^g(t_n)$ represents the queue length of downlink in $v_n$ at time $t_n$.

**Computing delay** $T_c$, which is determined by the computation requirements of the task and the satellite computing capacity:

$$T_c = \frac{d}{c_{v_k}} \tag{4}$$

### D. Problem Formulation

In the transmission of observation tasks, we aim to allocate transmission paths $P_1, P_2$ and the computing satellite $v_k$ under the constraint of storage capacities, minimizing the overall task delay as much as possible. Therefore, the problem can be formulated as follows:

$$\begin{aligned} \min_{\substack{P_1=\{v_1,\ldots,v_k\},\\ P_2=\{v_k,\ldots,v_n\}}} \quad & \sum_{i=1}^{n-1} \frac{D_{i,i+1}(t_i)}{\nu} + \frac{D_n^g(t_n)}{\nu} + \sum_{i=1}^{k-1} \frac{s}{r_{i,i+1}} + \frac{d}{c_k} \\ & + \sum_{i=k}^{n-1} \frac{s'}{r_{i,i+1}} + \sum_{i=1}^{n-1} \frac{q_{i,i+1}^t(t_i)}{r_{i,i+1}} + \frac{q_k^c(t'_k)}{c_k} + \frac{q_n^g(t_n)}{r_n^g} \\ \text{s.t.} \quad & v_1, v_2, \ldots, v_k \in \{v_i \mid m_i(t_i) + s \le M\}, \\ & v_{k+1}, \ldots, v_n \in \{v_i \mid m_i(t_i) + s' \le M\}, \\ & v_n \in \{v_i \mid D_i^g(t_i) \le D_{max}\} \end{aligned} \tag{5}$$

These constraints specify that using storage $m_i(t)$ on satellites $v_i$ along the transmission paths must not exceed the maximum storage resource value $M$. Additionally, the destination satellite $v_n$ must be within the coverage area of any ground station. $D_{max}$ represents the maximum allowable distance for the satellite-to-ground link, calculated based on the orbital altitude and the beam coverage angle.

## IV. Deep Reinforcement Learning For Joint Computing and Routing

A DRL algorithm is proposed to address the issue of distributed decision-making for joint computing and routing in complex satellite networks. Given the inherent limitations in obtaining complete system information in satellite networks, the POMDP framework is employed for dynamic optimization under conditions of partial visibility.

### A. POMDP

*1) State Space:* When a task arrives, the task information $S_t$ and local resource information $S_r$ along with the destination state $S_d$ will serve as input state to the DRL agent, i.e., $S = \{S_r, S_t, S_d\}$.

- The resource state $S_r$ can be represented by resource information of the current satellite and neighbors. $S_r = \{m_i(t), q_i^c(t), C, M, E\}$, where $m_i(t)$ represents the occupied storage of current satellite, $q_i^c(t)$ represents the total computing requirement in the computing queue of the current satellite. $C = \{q_{\eta_1}^c(t), ..., q_{\eta_K}^c(t)\}$ represents computing state of neighbor satellites(in our experiment, K=4). $M = \{m_{\eta_1}(t), ..., m_{\eta_K}(t)\}$ represents occupied storage of neighbor satellite. $E = \{q_{i,\eta_1}^t(t), ..., q_{i,\eta_K}^t(t)\}$ is the length of transmission queue towards neighbor satellites. All these values are normalized to their respective maximums. When the number of neighbors is less than $K$, the missing positions are filled with a value of 1, signifying that the link in that direction is unreachable.
- The task state $S_t = \{s, d, s', x_c\}$, where $s$ is the data size of the task, $d$ is computation demand, $s'$ is post-computation data size, and $x_c$ is a binary variable indicating whether the task has been computed.
- The destination direction state $S_d = \{l_{\eta_1}, ..., l_{\eta_K}\}$, where $l_{\eta_j}$ represents the hops from the $j^{th}$ neighbor to the destination satellite, normalized by dividing by the diameter of the network graph. If the number of neighbors is less than $K$, the gaps are filled with a value of 2, indicating that the link in that direction is unreachable.

*2) Action Space:* The action space can be represented as $\mathbf{A} = \{A_{\eta_1}, ..., A_{\eta_K}, A_c\}$, where $A_{\eta_j}$ denotes pushing the task to the transmission queue towards the $j^{th}$ neighbor, and $A_c$ denotes pushing the task to the computing queue of the current satellite.

*3) Reward:* The reward function $R$ for task allocation in satellite networks is defined based on various conditions to optimize task efficacy while minimizing delays and resource

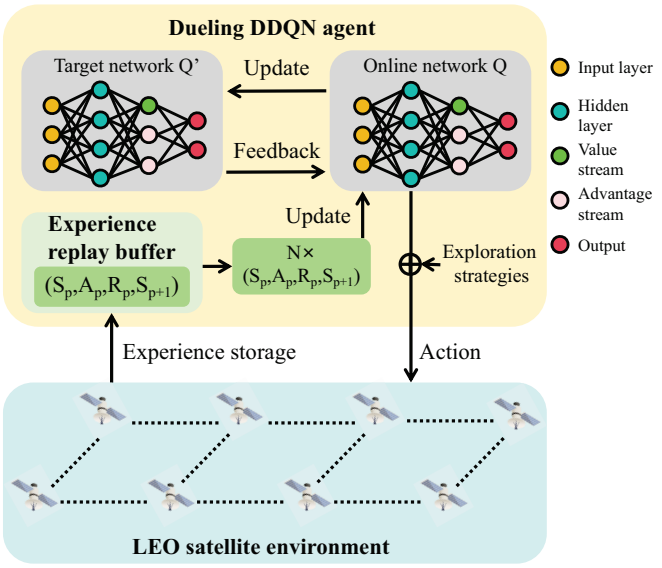



Fig. 2. Dueling-DDQN-based joint optimization of computing and routing for LEO satellites in training.

usage. Each condition is associated with a specific transmission or processing event, with corresponding rewards or penalties:

1) When the task reaches the destination ground station, the reward is calculated as:

$$R = \begin{cases} \beta_s + \beta_d \cdot (t_b - t_\tau), & \text{if } x_c = 1 \\ \beta_d \cdot (t_b - t_\tau), & \text{if } x_c = 0 \end{cases} \tag{6}$$

where $x_c$ donates whether the task is computed. $\beta_s$ is the reward for successful transmission, and $\beta_d$ is the delay penalty. Here, $t_b$ is the task start time, and $t_\tau$ is the current decision time.

2) When packet loss occurs during transmission. The reward is:

$$R = -\beta_l + \beta_d \cdot (t_b - t_\tau), \tag{7}$$

where $\beta_l$ is the penalty for packet loss.

3) For a normal one-step transition:

$$R = \begin{cases} \beta_d \cdot (t_L - t_\tau), & \text{if } m_i < m_r \\ -\beta_m + \beta_d \cdot (t_b - t_\tau), & \text{if } m_i >= m_r \end{cases} \tag{8}$$

where $m_r$ is the reserved storage space. $t_L$ representing the time at the last decision step, $\beta_m$ is the penalty for exceeding the storage threshold.

These conditions collectively aim to ensure that tasks are processed and transmitted efficiently while managing the limited resources of satellite networks effectively. In our experiment, $\beta_s$, $\beta_d$, $\beta_l$, and $\beta_m$ are 1, 0.05, 1, and 0.25, respectively.

### B. Dueling-DDQN

DDQN [15] represents a notable technique in deep reinforcement learning aimed at addressing the overestimation issue prevalent in traditional Q-learning algorithms through the introduction of dual Q-networks. Dueling architecture further enhances this approach by decomposing the Q-value into separate estimations of the state-value and advantage functions, enabling more robust policy learning and improved performance [16].

As shown in Fig. 2, the core framework of Dueling-DDQN comprises two main components: an online network $Q$ for selecting optimal actions and a target network $Q'$ for evaluating these actions. Both networks have the same structure, consisting of some hidden layers to extract features from the state $S$, followed by two streams: one for estimating the state-value function $V(S)$ and another for estimating the advantage function $\mathcal{A}(S, A)$. The Q-value is then computed by combining these two streams as follows:

$$Q(S, A) = V(S) + \left( \mathcal{A}(S, A) - \frac{1}{|\mathbf{A}|} \sum_{A' \in \mathcal{A}} \mathcal{A}(S, A') \right) \tag{9}$$

where $|\mathbf{A}|$ denotes the number of possible actions.

The online network $Q$ selects actions $A$ based on the current state of the environment $S$ and updates its parameters every step to adapt to environmental changes rapidly. In contrast, the target network $Q'$ updates its parameters at a lower frequency to maintain stability in the learning process.

The target value $y$ is calculated using the target network $Q'$. For each experience tuple $(S, A, R, S')$, the target value $y$ is defined as the current reward $R(S, A)$ plus the product of the discount factor $\gamma$ and the predicted value of target network $Q'$ for the next state $S'$ and best next action $A'$. Specifically, $\gamma$ is a value between 0 and 1 that determines the degree to which the algorithm prioritizes short-term versus long-term rewards. A lower value of $\gamma$ makes the agent emphasize immediate rewards more, whereas a higher value encourages the agent to consider long-term rewards in the future. This is expressed as:

$$y = R(S, A) + \gamma \cdot \max_{A' \in \mathbf{A}} (Q'(S', A', \theta')) \tag{10}$$

where $\theta'$ represents the weights of target network $Q'$.

The loss function $L(\theta)$ is defined based on the temporal difference loss, measuring the difference between the predicted value of the online network and the target value. Where $\theta$ represents the weights of the online network $Q$. It is defined as:

$$L(\theta) = [y - Q(S, A, \theta)]^2 \tag{11}$$

Minimizing this loss function allows Dueling-DDQN to learn the optimal policy effectively.

The parameters of the online network are updated using gradient descent to minimize the loss function. The formula for updating is:

$$\theta = \theta - \alpha \cdot \nabla_\theta L(\theta) \tag{12}$$

Where $\alpha$ is the learning rate, and $\nabla_\theta L(\theta)$ represents the gradient of the loss function concerning the parameters of the online network. During the update process, the algorithm computes the gradient of the loss function concerning the

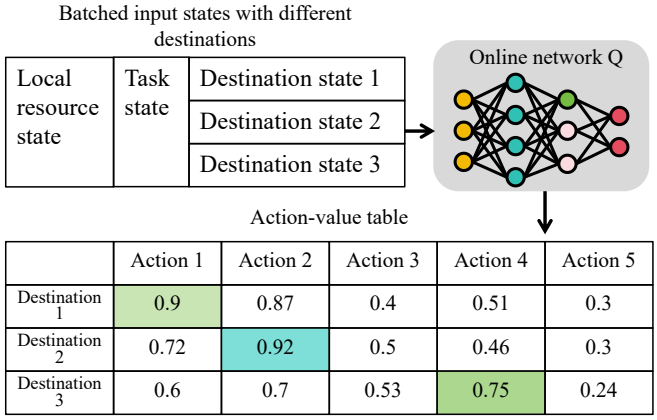
**Algorithm 1** Dueling-DDQN for joint optimization of computing and routing

*Initialization*:
Initialize $Q(\theta)$ with random weights and copy them to $Q'(\theta')$. Establish replay buffer $B$ and set initial exploration probability $\epsilon$.
**for** each training epoch **do**
  Reset the initial state $S_0$ and set marker $p = 0$
  **for** each task within the epoch **do**
    **while** task not completed or lost **do**
      **if** a sampled probability is less than $\epsilon$ **then**
        Select an action $A_p$ randomly
      **else**
        Select $A_p$ that maximizes $Q(S_p, A_p, \theta)$
      **end if**
      Execute $A_p$, observe $S_{p+1}$ and $R_p$
      Store $(S_p, A_p, R_p S_{p+1})$ in the replay buffer $B$
      $p \leftarrow p + 1$
    **end while**
  **end for**
  **for** each training iteration **do**
    Sample a batch of experiences from $B$
    $y = R(S, A) + \gamma \cdot \max_{A' \in \mathbf{A}}(Q'(S', A', \theta'))$
    Calculate $L(\theta) = [y - (Q(S, A, \theta))]^2$
    Update weights of network $\theta \leftarrow \theta - \nabla_\theta L(\theta)$
  **end for**
  Update exploration probability $\epsilon$
  **if** current epoch is a multiple of the target update period **then**
    Copy parameters from the online network $Q(\theta)$ to the target network $Q'(\theta')$
  **end if**
**end for**
Store the parameters of $Q(\theta)$

parameters of the online network and adjusts them accordingly to optimize the action selection.

### C. Optimization of Destination Satellite Selection Based on Q-Network for Data Backhaul

Unlike regular data transmission tasks, the backhaul ground station and the destination satellite for observation data are changeable. Observation data can be transmitted to any ground destination and subsequently uploaded to cloud servers. Due to the high-dimensional and highly dynamic nature of satellite networks, optimizing the choice of return ground stations and destination satellites becomes a complex problem. Given local resource status information, task information, and destination direction information, the pre-trained Q-network of Dueling-DDQN can be used to estimate the action-value function. Although the task destination is the satellite nearest to the ground station during training, tasks with different destinations can also be estimated by the Q-network using relative direction information in the inference stage.




| | Action 1 | Action 2 | Action 3 | Action 4 | Action 5 |
|---|---|---|---|---|---|
| Destination 1 | 0.9 | 0.87 | 0.4 | 0.51 | 0.3 |
| Destination 2 | 0.72 | 0.92 | 0.5 | 0.46 | 0.3 |
| Destination 3 | 0.6 | 0.7 | 0.53 | 0.75 | 0.24 |

Fig. 3. During inference, the nearest satellites within the coverage of ground stations will be selected as candidate destinations. The direction information of these destination satellites will be encoded and concatenated with the resource state and task state to form a batched input for Q-network. An action-value table for different destinations and actions will be generated, where the destination and action corresponding to the highest value will be adopted.

Since the predictability of satellite trajectories, any satellite can compute the current coverage relationship between other satellites and ground stations locally. Thus, as shown in Fig. 3, it is possible to consider different destination satellites and select the best action corresponding to the highest estimated value from all candidate destination satellites as the next action choice. This enables dynamic optimization selection of the destination satellite. Then, the optimized action:

$$A^* = \arg \max_{S \in \mathbf{S}, A \in \mathbf{A}} Q(S, A, \theta) \tag{13}$$

where $\mathbf{S}$ denotes the state set with different candidate destination satellites.

## V. Simulation Results and Analysis

### A. Experiment Setup

To evaluate the performance of different algorithms in the LEO satellite constellation, a system-level simulator is developed in *Python*. The discrete event library *SimPy* is used to build computing and transmission queues. *Skyfield* is used for satellite ephemeris calculations. The DRL network was constructed using *PyTorch*.

Table I lists the configurations of constellation parameters, satellite resources, and tasks used in the simulation. Owing to the high computation cost and runtime in constellation-scale simulation, a low storage resource limit (1.5GB) was adopted. This approach was employed to evaluate the storage resource planning capability of the algorithm in an acceptable simulation time. Eleven ground stations were adopted worldwide. The arrival time of observation tasks follows a Poisson distribution, and the duration time follows an exponential distribution.

### B. Comparing Algorithms

The proposed algorithm is in comparison with two other algorithms:

TABLE I
CONFIGURATION OF SIMULATION ENVIRONMENT

| Parameter | Value |
|---|---|
| Number of orbital planes | 12 |
| Number of satellites per plane | 24 |
| Orbital altitude | 500 km |
| Orbital inclination | 60° |
| Beam coverage angle | 45° |
| Data size per task | 25-75 MB |
| Compression ratio | 9-11 |
| Computation demand for compression | 1200-2000 FLOP/Byte |
| The data size of inference output | 5 KB |
| Computation demand for inference | 2400-4000 FLOP/Byte |
| Link failure rate | 2.6% |
| ISL transmission rate | 1.2 Gbps |
| Downlink transmission rate | 3 Gbps |
| Onboard computing capacity | 50 GFLOPS |
| Satellite storage limit | 1.5 GB |
| Learning rate ($\alpha$) | 0.0002 |
| TD target decay ($\gamma$) | 0.99 |
| Size of experience replay buffer | 500000 |
| Size of mini-batch | 1024 |

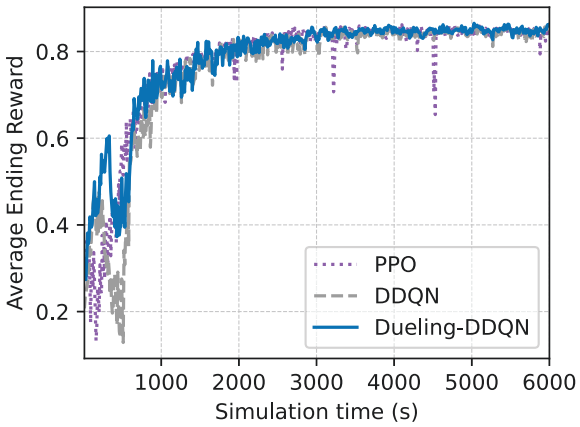



Fig. 4. Average reward changing curve during DRL training.

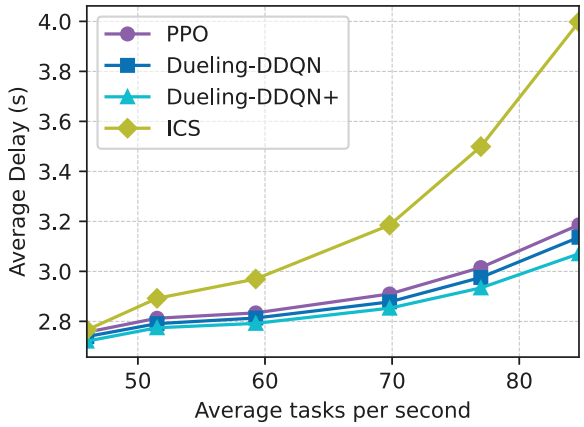



Fig. 5. Comparison of average delay for different algorithms under various traffic loads.

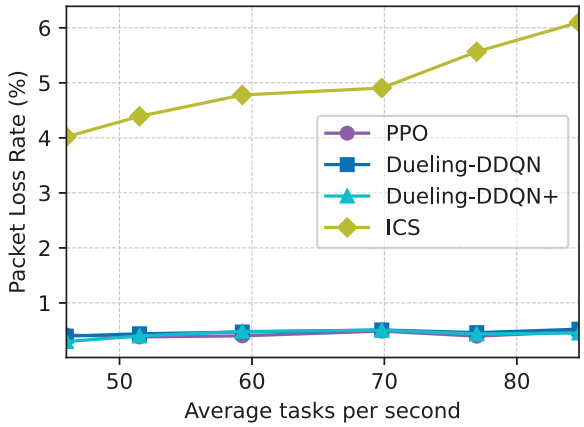



Fig. 6. Comparison of packet loss rate for different algorithms under various traffic loads.

- Proximal Policy Optimization (PPO) [17]. A DRL algorithm based on the Actor-Critic architecture and policy optimization methods. It improves performance by iteratively updating the policy and value function using a clipped objective function to maintain stable learning, prevent large policy updates, and ensure efficient exploration and exploitation.
- Ideal centralized solution (ICS). The centralized approach adopts the algorithm proposed in [18], which plans routing as an optimal algorithm that searches for all feasible solutions. The global information used for decision-making in this algorithm was obtained instantly from the simulator, ignoring all latencies in information updates.

### C. Rewards in DRL Training

As shown in Fig. 4, the Dueling-DDQN algorithm exhibited a significantly faster convergence speed in the early training stages than the PPO and DDQN algorithms. After reward convergence, the rewards of the three algorithms were close, with Duelin-DDQN being slightly higher than those of DDQN and exhibiting much less fluctuation than the PPO algorithm. The Dueling-DDQN demonstrated the fastest convergence speed and the best training performance.

### D. Comparison of Average Delay

Fig. 5 illustrates the variation of average task delay with traffic load. Dueling-DDQN+ is the algorithm that optimizes the destination dynamically. All three distributed DRL algorithms showed a slight increase in average delay with increasing traffic load, while the centralized algorithm showed a significant increase. This is primarily because distributed algorithms can achieve dynamic decision-making based on real-time network states, making them more adaptable to the highly dynamic satellite network environment. In contrast, the centralized algorithm makes only a single decision per task, and as the load increases, the decision delay effect becomes more pronounced. The Dueling-DDQN+ algorithm, optimized for destination satellite selection, exhibited the lowest task

TABLE II
LOCATIONS OF GROUND STATIONS IN SIMULATION

| City | Latitude (°) | Longitude (°) |
|---|---|---|
| Dubai | 25.252 | 55.280 |
| Harbin | 45.750 | 126.650 |
| Istanbul | 41.019 | 28.965 |
| Jakarta | -6.174 | 106.829 |
| Karachi | 24.867 | 67.050 |
| Moscow | 55.752 | 37.616 |
| Nairobi | -1.283 | 36.817 |
| Sanya | 18.243 | 109.505 |
| Shanghai | 31.109 | 121.368 |
| Urumqi | 43.800 | 87.583 |
| Xian | 34.258 | 108.929 |

delay, and its advantage over other algorithms became more evident with increasing traffic load.

### *E. Comparison of Packet Loss Rate*

Fig. 6 shows the packet loss rate variation with traffic load. The ICS had a packet loss rate of 4% to 6%, indicating that centralized methods are less capable of handling link failures and sudden traffic bursts. The packet loss rates of the three distributed DRL methods were almost identical, remaining below 1%, indicating that DRL can adapt well to network dynamics within the network load capacity, and increasing the load does not cause an increase in the packet loss rate.

## VI. CONCLUSION

This paper investigates a distributed strategy towards joint optimization of computing and routing in the LEO satellite constellation based on DRL. A Dueling-DDQN method is proposed to achieve minimal task delay under storage limitations. Simulation results demonstrate the applicability of the proposed algorithm, where the overall delay and packet loss can be reduced compared to the centralized method over different traffic loads.

## VII. APPENDIX

The locations of the ground stations used in our simulations are detailed in Table II.

## REFERENCES


[1] P. Ghamisi, N. Yokoya, J. Li *et al.*, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 37–78, 2017.

[2] B. Tao, M. Masood, I. Gupta *et al.*, "Transmitting, fast and slow: Scheduling satellite traffic through space and time," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–15.

[3] F. Tang, H. Zhang, and L. T. Yang, "Multipath cooperative routing with efficient acknowledgement for leo satellite networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 179–192, 2018.

[4] P. He, J. Hu, X. Fan, D. Wu, R. Wang, and Y. Cui, "Load-balanced collaborative offloading for leo satellite networks," *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 19 075–19 086, 2023.

[5] B. Zhang, Y. Wu, B. Zhao *et al.*, "Progress and challenges in intelligent remote sensing satellite systems," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 1814–1822, 2022.

[6] D. Sowmya, P. Deepa Shenoy, and K. Venugopal, "Remote sensing satellite image processing techniques for image classification: a comprehensive survey," *International Journal of Computer Applications*, vol. 161, no. 11, pp. 24–37, 2017.

[7] J. Yong, F. Wen, Z. Hu *et al.*, "High-dynamic transmission modeling for laser inter-satellite links (lisls)," in *2022 Asia Communications and Photonics Conference (ACP)*. IEEE, 2022, pp. 590–594.

[8] T. Zhang, J. Li, H. Li *et al.*, "Application of time-varying graph theory over the space information networks," *IEEE Network*, vol. 34, no. 2, pp. 179–185, 2020.

[9] X. Zhang, Y. Yang, M. Xu, and J. Luo, "Aser: Scalable distributed routing protocol for leo satellite networks," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE, 2021, pp. 65–72.

[10] P. Zuo, C. Wang, Z. Wei *et al.*, "Deep reinforcement learning based load balancing routing for leo satellite network," in *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*. IEEE, 2022, pp. 1–6.

[11] G. Xu, Y. Zhao, Y. Ran *et al.*, "Towards spatial location aided fully-distributed dynamic routing for leo satellite networks," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 1570–1575.

[12] Z. Song, Y. Hao, Y. Liu, and X. Sun, "Energy-efficient multiaccess edge computing for terrestrial-satellite internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 14 202–14 218, 2021.

[13] Q. Tang, Z. Fei, B. Li, H. Yu, Q. Cui, J. Zhang, and Z. Han, "Stochastic computation offloading for leo satellite edge computing networks: A learning-based approach," *IEEE Internet of Things Journal*, 2023.

[14] H. Zhang, R. Liu, A. Kaushik *et al.*, "Satellite edge computing with collaborative computation offloading: An intelligent deep deterministic policy gradient approach," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 9092–9107, 2023.

[15] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[16] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] J. Cao, S. Zhang, Q. Chen, H. Wang, M. Wang, and N. Liu, "Computing-aware routing for leo satellite networks: A transmission and computation integration approach," *IEEE Transactions on Vehicular Technology*, 2023.