remote sensing

MDPI

Article

DRL-Based Load-Balancing Routing Scheme for 6G Space–Air–Ground Integrated Networks

Feihu Dong [1,2], Jiaxin Song [3], Yasheng Zhang [2,*], Yuqi Wang [3] and Tao Huang [1]

[1] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; dongfh@bupt.edu.cn (F.D.)
[2] The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang 050051, China
[3] School of Telecommunications Engineering, Xidian University, Xi'an 710071, China
* Correspondence: zys163@163.com

Abstract: Due to the rapid development of the space–air–ground integrated network (SAGIN), a satellite communication system has the advantages of wide coverage and low requirements for a geographical environment and is gradually becoming the main competitive technology for 6G. The low-earth-orbit (LEO) satellite network has the characteristics of low transmission delay, small propagation loss, and global coverage, and its exploration has become the main research object of contemporary satellite communications. However, traditional routing algorithms cannot adapt to the characteristics of the high dynamics and load-balancing requirements of LEO satellite networks. In this paper, a load-balancing routing algorithm for LEO satellites based on Deep Q-Network (DQN-LLRA) is proposed by using deep reinforcement learning. Making use of the model obtained by the DQN training, satellite nodes can select the best routing results according to the delay, bandwidth, and queue utilization of the surrounding satellite nodes. The simulation and analysis show that the path load obtained by the proposed algorithm is low. Compared with the Q-learning-based algorithm, this algorithm reduces the maximum queue utilization rate of the routing path by 5%, reduces the average queue utilization rate of the routing path by 13%, and effectively balances the load in the network.

Keywords: low earth orbit; satellite routing algorithm; deep reinforcement learning; high dynamics; load balancing

Citation: Dong, F.; Song, J.; Zhang, Y.; Wang, Y.; Huang, T. DRL-Based Load-Balancing Routing Scheme for 6G Space–Air–Ground Integrated Networks. *Remote Sens.* **2023**, *15*, 2801. https://doi.org/10.3390/rs15112801

Academic Editor: Andrzej Stateczny

Received: 17 March 2023
Revised: 8 May 2023
Accepted: 27 May 2023
Published: 28 May 2023

1. Introduction

In the last few years, the space–air–ground integrated network has developed rapidly. Satellite communication is a key link in 6G space–air–ground integrated networks (SAGIN). It can make up for the shortcomings of the 5G terrestrial network, improve network coverage, and ensure the fault tolerance of the system. It can also combine artificial intelligence, big data, the Internet of Things, and other technologies to provide users with diverse services. Satellite communication has a larger coverage area than traditional terrestrial networks, and its global adaptability is stronger. It is gradually becoming the main competitive technology for the next-generation communications. The importance of Low Earth Orbit (LEO) in the field of satellite communications cannot be overstated. When compared to the constellations of Geosynchronous Orbit and Medium Earth Orbit, LEO stands out due to its low transmission delay, low propagation loss, and its ability to cover the world. These unique features make LEO an attractive option for various applications, including internet services, global positioning systems, and remote sensing. The low transmission delay and low propagation loss of LEO makes it an ideal choice for time-sensitive applications such as real-time communication, while its global coverage ensures that it is suitable for applications that require connectivity in remote or hard-to-reach locations. Therefore, it is not surprising that LEO has gained significant attention and interest in recent years, leading

to the development of new technologies and algorithms to enhance its performance and efficiency. With the rapid development of SAGIN, the traditional ground communication network can not adapt to future development. The development of satellite communication in low earth orbit is already a promising development direction.

In LEO satellite networks, inter-satellite links (ISLs) ensure communication between satellites. Compared with the terrestrial communication network, the LEO satellite network changes its topology more frequently, has a longer inter-satellite link delay, and has a link state in the multiuser area that changes more frequently. Due to the high-speed dynamic changes, the cost of traditional path selection methods is significantly increased. Therefore, the routing protocol for terrestrial applications is difficult to use directly in the LEO satellite network. LEO satellite routing technology is also a supporting technology for the integration of remote sensing, communication, and computing for 6G SAGIN; so, it is necessary to study the routing algorithm in the low-orbit satellite network.

Most of the existing satellite routing algorithms have been developed based on terrestrial-network routing algorithms. Most of these algorithms are based on the shortest path. Due to the difference in the density of satellites at high and low latitudes and the difference in the density of user distribution [1], the load differences among satellites in the same constellation are large. In addition, with the high-speed movement of the satellites, the high-load coverage area also changes rapidly among satellites. Therefore, the traditional routing algorithm has faced difficulties in meeting the current development of satellite networks.

The cognitive performance of the deep learning algorithm is better, the decision-making ability of the reinforcement learning algorithm is stronger, and deep reinforcement learning combines the two. In deep reinforcement learning, an agent makes decisions through interaction with the environment and obtains feedback through trial and error, learning to maximize the reward and minimize the punishment [2]. Due to the powerful perception and decision-making ability of deep RL, more and more scholars have applied this type of learning in many fields such as computer vision [3,4], speech recognition, and automatic driving [4,5]. Deep reinforcement learning is also applicable in the field of LEO satellite networks. It can sense the topology changes, load changes, and network parameters, such as the delay and bandwidth in the satellite network. It can make the best decision according to the network service requirements.

This paper proposes a load-balancing routing algorithm based on deep reinforcement learning to address the challenges and opportunities of the LEO satellite network. It aims to solve the issues of traditional algorithms that cannot adapt to the high dynamics and load balancing of the network. The work focuses on the following:

(1) The satellite routing process is modeled as a Markov decision process (MDP) [6], and its state space, decision space, and reward function are defined.

(2) A Deep Q-Network (DQN)- [7] based load-balancing routing algorithm for LEO satellites (DQN-LLRA) is proposed. The algorithm relies only on the state information of the surrounding nodes of the current satellite node, and the optimal decision model is obtained through iterative training.

(3) Through numerical simulation, our algorithm reduces the maximum queue utilization and the average queue utilization of the routing path, avoids the occurrence of congestion, and realizes the path routing under load balance while ensuring that the delay cost increases only slightly and in an acceptable fashion.

The paper is organized as follows. Section 2 provides an overview of the current research on LEO satellite routing, including its load-balancing challenges and related intelligent algorithms. Section 3 presents the system model and algorithm design of the proposed DQN-based load-balancing routing algorithm for LEO satellites. Section 4 describes the experimental simulations and resulting evaluations carried out to validate the proposed algorithm. Finally, in Section 5, the paper concludes with a summary of the contributions and directions for future research.

2. Related Work

There are no agreed-upon values for the orbit altitude and orbit period of LEO satellites. However, many sources claim that the orbit altitude of these satellites changes from 160 km to 2000 km, while their orbit period, mainly depending on the orbit altitude, ranges between 90 min and 2 h [8]. The configuration of the inter-satellite links topology is closely related to the constellation design. Therefore, in this section, we provide a comprehensive review of the current state of research on LEO satellite routing technology, load-balancing technology, and machine-learning-based satellite routing technology. Specifically, we discuss the latest developments in these areas and present the relevant literature to help readers better understand the challenges and opportunities facing LEO satellite networks. By doing so, we aim to lay the foundation for the proposed DQN-based load-balancing routing algorithm for LEO satellites in the following sections.

2.1. LEO Satellite Routing

The LEO satellite constellation can be divided into two categories, namely the inclined-orbit-based Walker Delta constellation and the polar-orbit-based Walker Star constellation. As shown in Figure 1, the Iridium constellation [9], a typical polar-orbit constellation, has been studied by many scholars as a low-orbit constellation because of its representative constellation configuration and easy mathematical model construction.

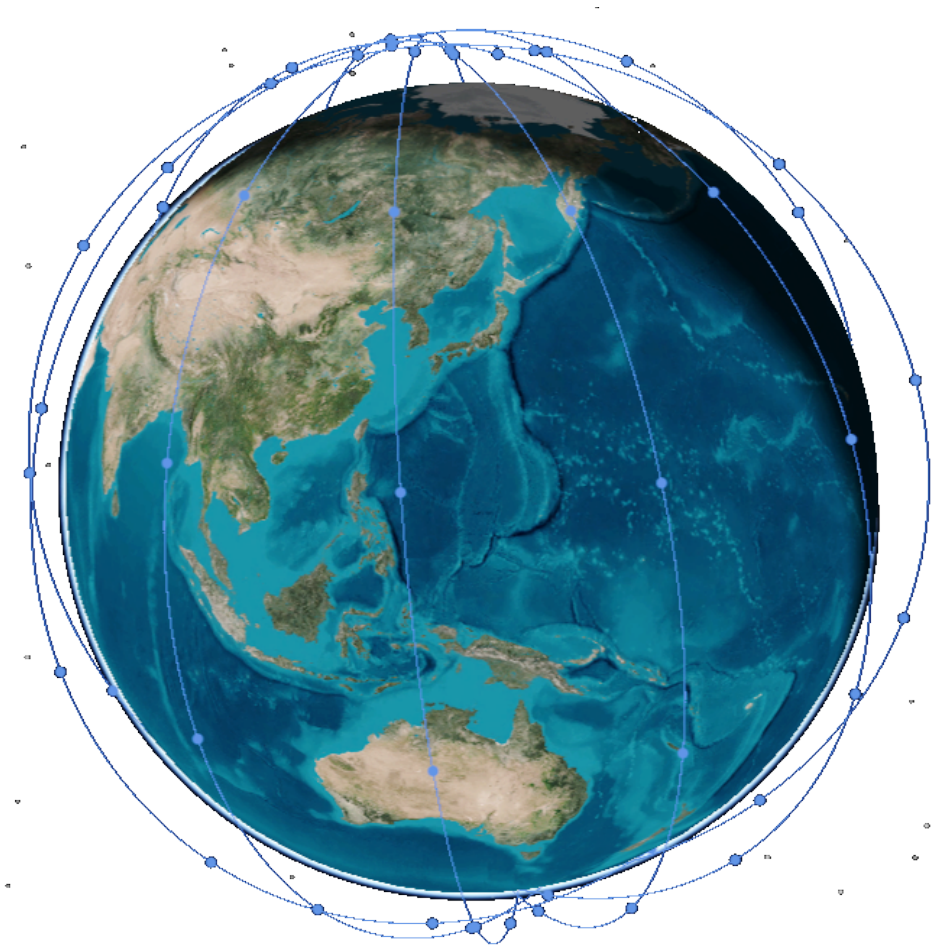

Figure 1. Illustration of the Iridium constellation.

The orbit and topology of the Iridium constellation are shown in Figure 2. The topology changes rapidly due to the mobility of the satellites and the changing connections. The satellites running in the reverse orbit cannot establish communication links. In addition, the communication links change when the satellites travel across the pole. Due to the

challenges introduced by the dynamic topology, the routing algorithm for satellite networks has attracted a lot of research interest. This field of work mainly falls into the following two kinds, the centralized routing algorithm and the distributed routing algorithm.

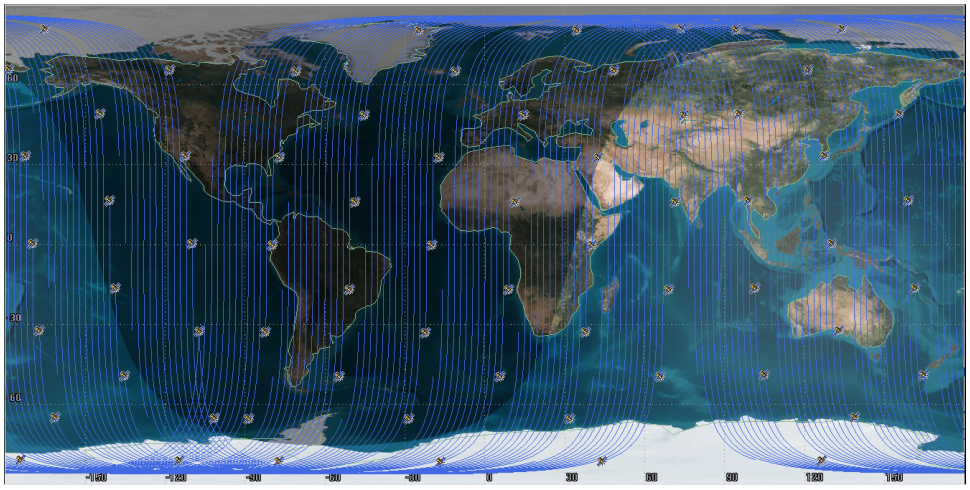

Figure 2. Iridium satellite trajectory diagram.

The distributed routing algorithm can adapt to the dynamic scenario of the satellite network. This is because the algorithm determines the next hop based on the status of the neighbor satellites, e.g., the remaining bandwidth and the queue utilization rate. Thus, when the status of the neighbor satellites changes, the algorithm can become aware of it rapidly and make a quick decision on the routing strategy according to the dynamic environment. By taking the design idea of the existing ground-distributed routing algorithm into account, the authors in [10] investigated the routing methods for the satellite network. The onboard buffer space was improved by fully considering the features of the LEO satellites. The data packets were classified, and the corresponding routing method was designed in [11].

Different from the distributed routing algorithm, the centralized one needs to derive the global information of the satellite network [12]. The master control nodes first collect the global information and then perform the routing path computation. After obtaining the routing results, they transmit the whole routing strategy to the other nodes. The authors in [13] designed an improved Distributed Hierarchical Routing Protocol (DHRP) for the satellite network. This protocol set up master and candidate nodes and therefore showed superior routing performance compared with the traditional Discrete Relaxation Algorithm (DRA). The authors in [14] proposed a Hybrid Global–Local Load Balancing Routing (HGL) algorithm. However, it was ineffective when the large-scale traffic flows changed suddenly. The authors in [15] proposed a probability ISL Routing (PIR) algorithm in which the communication delay was leveraged to evaluate the path selection performance. This algorithm also considered the cost of the inter-satellite links.

Although the aforementioned algorithms have made great progress in the adaptability to the dynamics of the LEO satellites, their failure to consider the satellite load remains a significant drawback.

2.2. Load Balancing of LEO Satellites

The inter-satellite link length of LEO satellites changes with the latitude of the satellites. The conventional shortest-path algorithm design for the routing paths only relies on the length of the path, which results in traffic aggregation when the latitude is high [1,16]. Figure 3 depicts a 3D schematic of the distribution of satellite traffic, created using NS3 network simulation software. The black dots represent LEO satellites, while the line segments denote inter-satellite links carrying traffic. The thickness and color of each segment correspond to its bandwidth utilization and traffic, respectively. The thicker lines indicate

higher traffic, while the darker colors show greater bandwidth utilization. Notably, the LEO satellite network experiences congestion primarily in high-latitude and densely populated regions. In addition, the uneven distribution of the gateway station on the ground gives rise to the load imbalance of the satellite networks. User mobility and the global population distribution are also key factors affecting the distribution of traffic flows [17]. The high mobility of the satellites leads to rapid changes in the high-load coverage area between satellites [18,19].

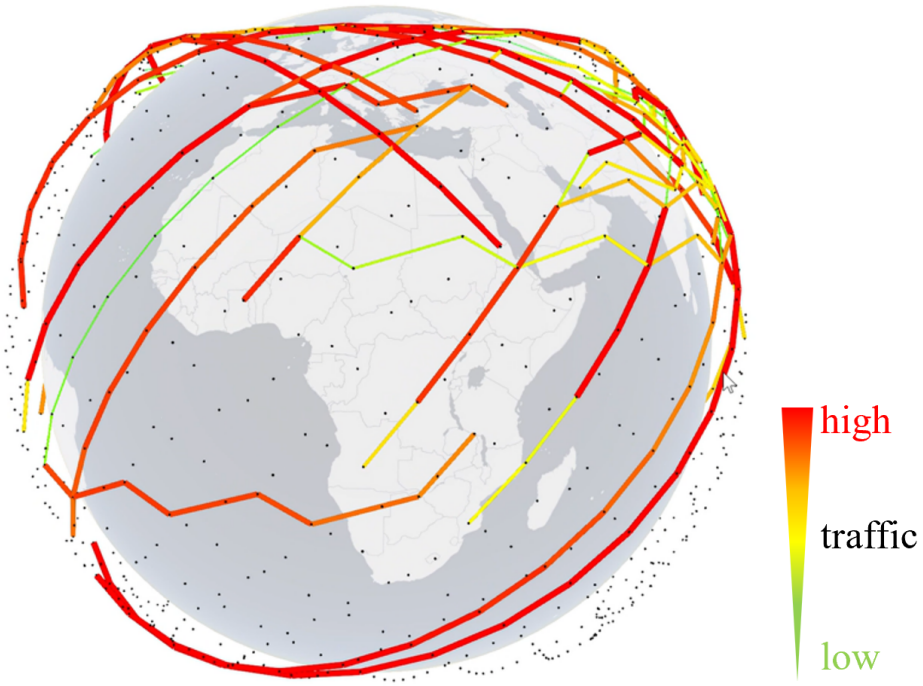

Figure 3. Traffic distribution diagram.

A path-based load-balancing satellite routing algorithm with the objective of minimizing the maximum network flow was proposed in [17,20]. This algorithm avoided traffic-flow aggregation in high-latitude areas by setting all inter-satellite links with the same path length and giving all paths the same priority. The authors in [21] divided the transmission areas into the heavy-load ranges and the light-load ranges considering the relationship between the reverse slots and the gateway stations. The congestion indicator was utilized for the heavy-load range, and the path with the minimum weight was exploited to deal with the uneven traffic-flow distribution. Nevertheless, this method required the link status information of the whole network and could not make decisions in real time [22]. The Elastic Load-Balancing (ELB) algorithm was proposed in [23,24], which realized the exchange of congestion information between satellite nodes. Thus, the ELB achieved the load-balancing goal and avoided traffic congestion. The occupancy of the queue was leveraged to decide whether the satellite node was idle or busy. When a node was marked as a busy one, it emitted a message to its neighbor nodes to decrease its transmission rate. The TLR algorithm proposed in [25] took both the current status of the congestion and the possible status of the next hop congestion into account. The authors in [26] proposed an iterative Dijkstra mechanism to select the optimal transmission paths for load-balancing routing. The authors in [27] considered the link delay to further promote the routing performance. Cooperative game theory was exploited in [28] to bring to equilibrium the tradeoff between the load and the transmission delay when exploring the LEO routing algorithm. Fuzzy theory was utilized to achieve the requirements of various users in [29]. The transmission overhead and the routing convergence were evaluated as the key performance metrics in [30]. The on-demand dynamic routing algorithm was proposed

with the track prediction. Energy consumption was also considered in [31] to promote the service quality for the users.

The existing literature highlights the benefits of LEO routing in terms of load performance [32,33]. However, challenges related to the insufficient local optimization and weak dynamic adaptability still remain unsolved, which could potentially impede the development of the LEO satellite network.

2.3. Machine-Learning-Based Satellite Routing

The complex satellite network environment and dynamic inter-satellite links make the satellite routing algorithm difficult to calculate. Reinforcement learning has been widely used in various new industries because of its unique ability to deal with sequential decision-making problems [34]. The content caching problem was investigated in [35]. The Q-learning algorithm was utilized in the cloud content distribution systems. In [36], congested links in the Internet of Things (IoT) were identified using Q-learning to increase the fault tolerance rate. Similarly, Q-learning was used in [37] to improve the throughput of wireless sensor networks (WSNs) and solve the problem of the energy consumption of devices. In [38], to solve the optimal allocation problem of cache, bandwidth, and other resources in the Internet of Vehicles, they used deep reinforcement learning to solve the model. The authors of [39] designed a centralized satellite routing algorithm for the space–ground integrated network using the Deep Deterministic Policy Gradient (DDPG) [40] in machine learning [41,42]. The decision center of the proposed strategy was set on the ground. The decision center obtained the traffic information of the whole network in real time and sent the routing information to the relevant satellites after making a decision. However, the drawback of this strategy was that the delay of the satellite communication was very large, and the transmission routing decision could not be made in time. The network burden was increased; so, it was not suitable for large-scale use. A routing algorithm based on Multiagent Deep Deterministic Policy Gradient (MADDPG) [43] was proposed in [44], which was a routing strategy that was deployed on each satellite after centralized training and solved part of the problems of the above centralized routing algorithm. The centralized training method had limitations in acquiring sufficient data, which became increasingly difficult as the network size and deep reinforcement learning algorithm training complexity increased. To address the dynamic load-balancing challenges in the LEO satellite network, this paper introduces a DQN-based load-balancing routing algorithm for LEO satellites.

3. System Model

This section explains the research scenario of the proposed DQN-LLRA algorithm, the basic framework of the DQN algorithm, and the framework and mechanism of the DQN-LLRA algorithm.

In this paper, the Iridium satellite, a typical polar LEO constellation, is taken as the research scene. The Iridium constellation has six polar orbits, and 12 LEO satellites are distributed on each orbit, including 11 communication satellites and one standby satellite. In total, there are $6 \times (11 + 1) = 72$ satellites in the whole constellation. The interorbit inter-satellite links change periodically with the movement of satellites. The connectivity of the inter-satellite link changes periodically with the motion of the satellite. The satellites in the first orbit move in opposite directions to the satellites in the sixth orbit; so, there is no direct communication between the satellites in the two orbits. As shown in Figure 4, in a typical constellation, there are up to four inter-satellite links around a satellite, connecting it to four satellites, two of which are in the same orbit and two in adjacent orbits. Communication between satellites with an inter-satellite link is possible.

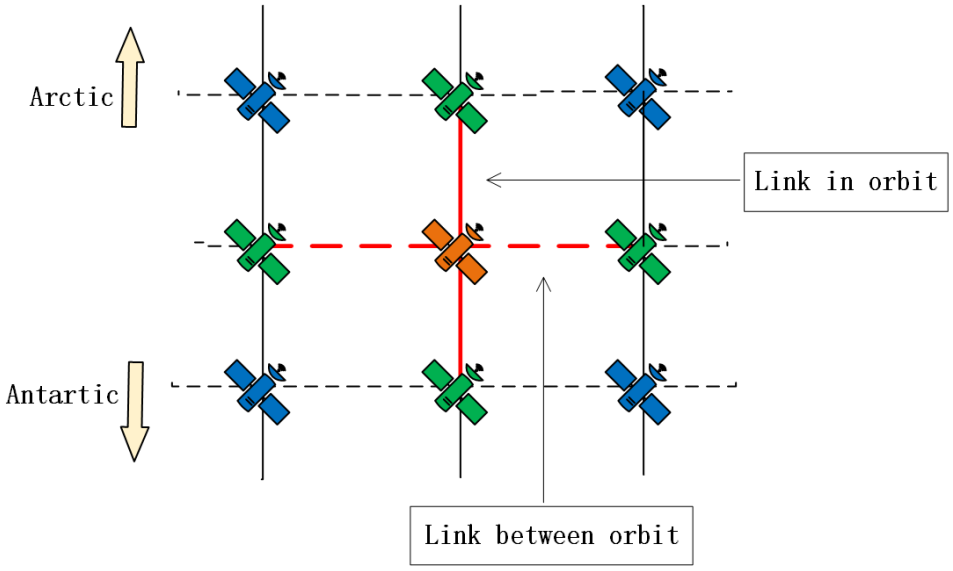

Figure 4. Schematic diagram of the inter-satellite links.

In order to represent the relevant parameters of the satellite topology, we constructed this topology as an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ represents the set of satellite nodes, and n is the total number of satellite nodes. We denoted the queue utilization of each node v_i by QU_i, which represents the ratio of the number of packets currently existing in the queue of satellite node v_i to the queue capacity of the node v_i, and can be expressed mathematically as:

$$QU_i = \frac{\text{Number of packets in queue of node } i}{\text{Total queue size of node } i}. \quad (1)$$

As stated earlier $E = \{e_{(v_1,v_2)}, e_{(v_2,v_3)}, \cdots, e_{(v_i,v_j)}\}$ represents the set of inter-satellite links, and $e_{(v_i,v_j)}$ represents the inter-satellite link between satellite v_i and satellite v_j. We denoted the time delay and bandwidth of each inter-satellite link by $C_{(v_i,v_j)}$ and $B_{(v_i,v_j)}$, respectively. Our DQN-LLRA algorithm was implemented on the basic model described in the next section.

4. Dqn-Based Load-Balancing Routing Algorithm

4.1. DQN-LLRA Framework and Mechanisms

Our paper presents a load-balancing routing algorithm for LEO satellites that leverages deep reinforcement learning. We modeled the satellite-routing decision-making process as a Markov decision process. The Markov decision process is a mathematical framework used to model decision-making processes, where only the immediate state affects the outcome of a current decision, rather than previous states. In our approach, each decision-making satellite was treated as an agent and interacted with the surrounding nodes to gather information about the current state in order to make informed routing decisions.

The proposed algorithm used DQN to train the agents and make optimal routing decisions. At each routing node, the agent made a decision and selected the next route based on the information obtained from the surrounding nodes. The reward function was used to evaluate the quality of the decision made by the agent, and the agent moved on to the next routing node to continue making decisions.

The DQN-LLRA algorithm architecture is shown in Figure 5. The proposed algorithm aims to optimize the routing decisions of LEO satellites by balancing the workload among the satellites effectively. By modeling the satellite-routing decision-making process as a Markov decision process and using deep reinforcement learning, the proposed algorithm can continuously learn and adapt to changing network conditions, leading to more efficient routing decisions.

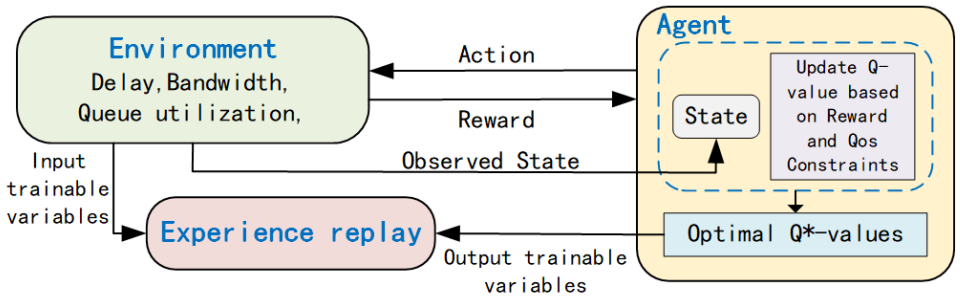

Figure 5. Architecture diagram of DQN-LLRA.

The following section provides details on the DQN-LLRA algorithm, including the state space, action space, and the reward function settings.

(1) The state space: The state of satellite node i at time t is represented as S_t^i, where $S_t^i = \{D_{ij}, B_{ij}, C_i, C_j, QU_j, done\}$, with $j = 1, 2, 3, 4$. D_{ij} denotes the delay of the inter-satellite link between node i and neighboring node j, B_{ij} represents the bandwidth of the inter-satellite link between node i and neighboring node j, C_i represents the shortest hop count from the current node i to the destination node, and C_j denotes the shortest hop count from neighbor node j to the destination node. The pair of numbers C_i, C_j was introduced to enhance the algorithm's convergence rate and avoid undesired ping-pong routing. C_i and C_j are calculated by the shortest path algorithm, i.e., the Dijkstra algorithm. The agent leveraged the Dijkstra algorithm to calculate C_i and C_j when acquiring state information. When $C_i - C_j > 0$, the reward function gave positive feedback. When $C_i - C_j \leq 0$, the reward function gave negative feedback. After multiple training rounds, the agent chose a node as close as possible to the destination node when making a decision. This avoided the occurrence of ping-pong routing and, at the same time, accelerated the agent to find the optimal path, thus speeding up the convergence speed of the algorithm. The symbol QU_j denotes the queue utilization of neighboring node j, which indicates a load of node j. The symbol *Done* denotes a two-valued Boolean number that indicates whether the current node is the destination node or not. *Done* is true if the current node is the target node and false otherwise.

(2) The action: The action of the current satellite node i is expressed as $a_i = \{\text{neighbor}_j\}$, $j = 1, 2, 3, 4$, where neighbor_j represents that each decision agent selected a node j among the neighboring nodes according to the decision function.

(3) The reward function: In order to meet the load balancing requirements in the LEO satellite scenario, our reward function introduced the queue utilization on the satellite node to represent the load situation of the node. The reward value was calculated by the reward function when the current agent made a decision, and the state transition occurred. Our reward function is as follows in Equation (5).

$$r_i = \begin{cases} \alpha_1 \frac{C_i - C_j}{p} + \beta_1 (1 - QU_j) + \gamma_1 \frac{D_{ij}}{q} + \omega_1 \frac{B_{ij}}{o}, C_i \leq C_j, QU_j \leq 0.5 \\ \alpha_2 \frac{C_i - C_j}{p} + \beta_2 (1 - QU_j) + \gamma_2 \frac{D_{ij}}{q} + \omega_2 \frac{B_{ij}}{o}, C_i \leq C_j, QU_j > 0.5 \\ \alpha_3 \frac{C_i - C_j}{p} + \beta_3 (1 - QU_j) + \gamma_3 \frac{D_{ij}}{q} + \omega_3 \frac{B_{ij}}{o}, C_i > C_j, QU_j \leq 0.5 \\ \alpha_4 \frac{C_i - C_j}{p} + \beta_4 (1 - QU_j) + \gamma_4 \frac{D_{ij}}{q} + \omega_4 \frac{B_{ij}}{o}, C_i > C_j, QU_j > 0.5 \end{cases}, \quad (2)$$

where D_{ij}, B_{ij}, C_i, C_j, and QU_j, respectively, represent the delay and bandwidth of the inter-satellite link $e_{(v_i, v_j)}$, the shortest hops from node i to the destination node, the shortest hops from node j to the destination node, and the queue utilization rate of node j. The symbols $\alpha_i, \beta_i, \gamma_i, \omega_i, i = 1, 2, 3, 4$ denote the parameter weights where $\alpha_i + \beta_i + \gamma_i + \omega_i = 1$. o, p, and q are adjustment factors, which keep each state value at the same order of magnitude. Here, C_i, C_j, and QU_i were used for segmental processing of the reward function, and different parameter weights were set for each segment function, in order to increase the penalty value of a poor decision and accelerate the convergence speed of the training

and the accuracy of the decision. We used four parameter weights to adjust the final result of the reward function. We can adjust the ratio of the parameter weights according to different optimization goals. The ratio of the parameter weights directly affects the relative performance of the final training model. The main optimization objectives of this paper are the hop count and queue utilization of the path. Therefore, in the simulation part, we set the parameter weights of the first two items to be larger and the parameter weights of the latter two items to be smaller. In addition, we imposed the following constraints on the reward function as follows in Formula (6).

$$r_t = \begin{cases} 1, done = True \\ r_t, done = False \end{cases}. \tag{3}$$

When the current node was the destination node, the decision was ended, the reward value was 1, and the cumulative reward was calculated as follows:

$$R_t = \sum_{t=0}^{\infty} \gamma^t r_{t+1}, \tag{4}$$

where, γ^t represents the decay value of the future reward at time t, while r_{t+1} denotes the instantaneous reward at time $t+1$.

In summary, the training flowchart of the DQN-LLRA algorithm is as follows in Figure 6.

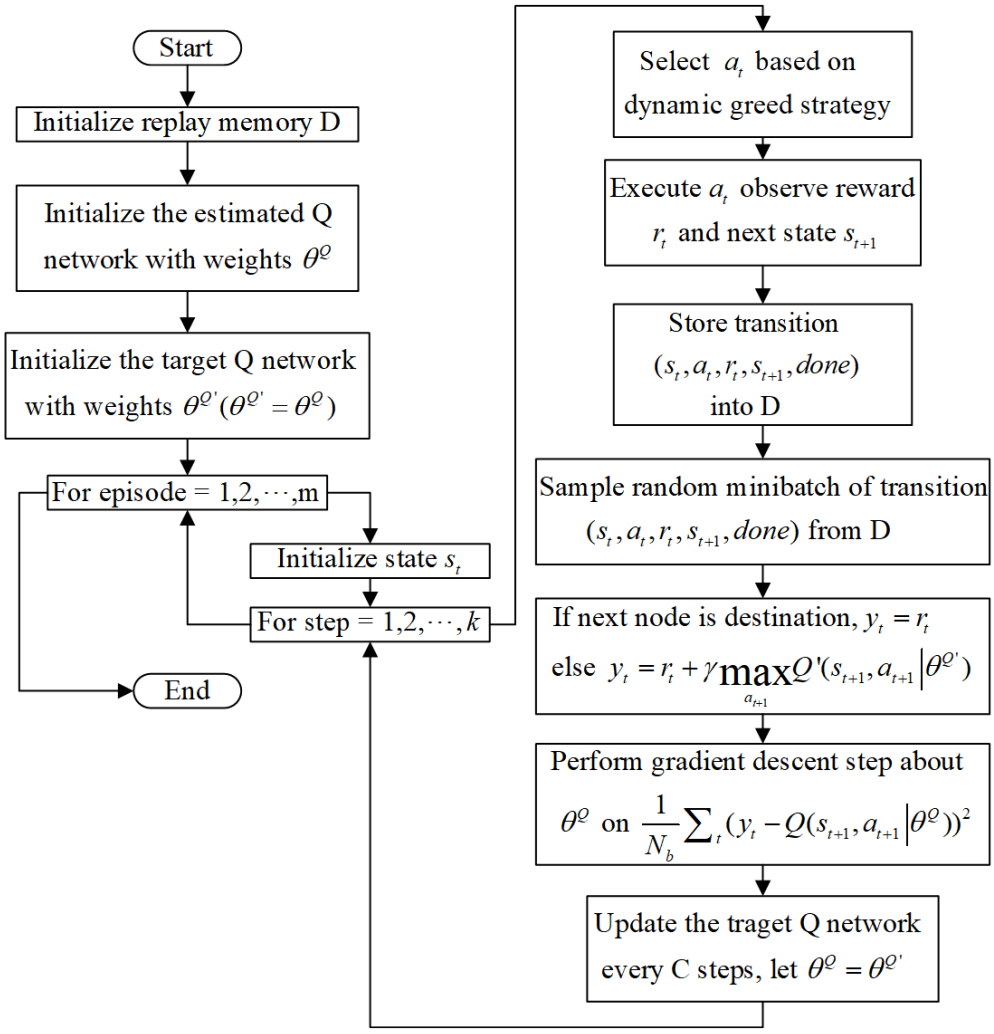

Figure 6. The training flowchart of the DQN-LLRA algorithm.

Initially, the experience replay pool, estimation network, and target network were initialized for training. The Iridium satellite topology constellation randomly generated source and destination node pairs, and the routing decision was made by the source node acting as the agent with an initialized state. In the decision-making process, the agent made decisions according to the following dynamic greedy strategy in Equation (8).

$$\pi_i = \begin{cases} \text{random action with probability } e^{-steps \cdot 0.4} \\ \arg\max\limits_{a_{t+1}} Q'(s_{t+1}, a_{t+1} \big| \theta^{Q'}) \text{ with probability } 1 - e^{-steps \cdot 0.4} \end{cases}, \quad (5)$$

where steps is the number of training steps. With the progress of training, the decision tended to choose the next hop decision with the largest Q value. After each decision, the reward value was calculated according to the reward function in (5). After the agent made a decision, $(s_t, a_t, s_{t+1}, done)$ was stored in the empirical replay pool for reserve. The estimated network updated the network weights according to the gradient descent method, and the target network updated the network weights from the estimated network every C steps. Upon reaching the destination node, the route ended, and the cumulative reward value was computed using the cumulative reward function. Repeated random generation of source and destination node pairs was used for training.

The pseudo-code for the DQN-LLRA algorithm is as follows in Algorithm 1.

Algorithm 1 DQN-based load-balancing routing algorithm for LEO satellite networks

Input: Source node and destination node, satellite topology
Output: Route path

for episode = 1 to M **do**
  Initialize state s_t;
  Move the agent to the source node;
  for step = 1 to k **do**
    Select a_t based on dynamic greed strategy;
    Add the current node to the path;
    Execute a_t, observe reward r_t and next state s_{t+1};
    Store transition $(s_t, a_t, r_t, s_{t+1}, done)$ into D;
    if next node is destination **then**
      $y_t = r_t$
    else
      $y_t = r_t + \gamma \max\limits_{a_{t+1}} Q'(s_{t+1}, a_{t+1} \big| \theta^{Q'})$
    end if
    if D. size $>$ R **then**
      Sample random minibatch of transition $(s_t, a_t, r_t, s_{t+1}, done)$ from D;
      Perform gradient descent step about θ^Q on $\frac{1}{N_b} \sum_t \left(y_t - Q(s_{t+1}, a_{t+1} \big| \theta^Q)\right)^2$;
      Update the target Q network every C steps, let $\theta^Q = \theta^{Q'}$
    end if
  end for
end for

In the decision stage of the DQN-LLRA algorithm, on the basis of the above-trained decision model, firstly, the source node, destination node, and network topology were input to initialize the routing path. The agent selected the next hop node according to its surrounding environment. Then, the selected node was added to the routing path. The agent moved to the next node. The algorithm checked whether the next node was the destination node. If it was, the decision-making process was completed, and the path is output. Otherwise, the routing decision continued.

Figure 7 illustrates the decision-process flowchart of the DQN-based load-balancing routing algorithm.

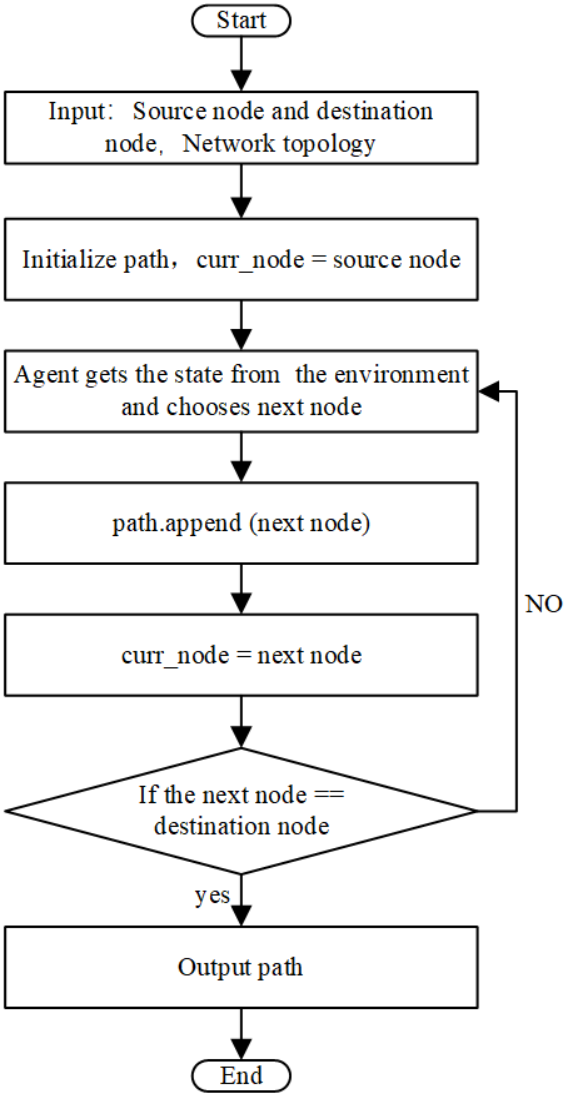

Figure 7. The decision flowchart of the DQN-LLRA algorithm.

Algorithm 2 presents the pseudo-code for the decision process of the DQN-based load-balancing routing algorithm.

Algorithm 2 The decision process of the DQN-based load-balancing routing algorithm

Input: Source node and destination node, satellite topology
Output: Route path
Initialize path
current node = source node
next node = None
while current node is not destination **do**
    Agent obtains the state from the environment
    Agent chooses the next node
    Add the next node to the path
    current node = next node

4.2. Basic Model of the DQN

In the abovementioned Iridium satellite model, each node needed to send the current data packet to a suitable node among the four surrounding nodes according to the current service requirement. In such a decision-making scenario, the current satellite could only

receive the status of the four surrounding nodes and links. In addition, this state was a changing continuous state. The decision to be made was to choose one of the four surrounding nodes, and the decision was a discrete decision. The model of the DQN algorithm continuous state and the discrete decision have good adaptability. Therefore, this work used the DQN algorithm to train the model.

On the basis of Q-learning, the DQN algorithm combines deep learning, which can deal well with the problems of a continuous state and a discrete action space. In the continuous state space, the algorithm does not need to maintain a large number of Q tables. The DQN algorithm can adapt to the continuous state space. Due to the large state space, a deep neural network can be used to perform nonlinear fitting of the action value function. The input of the neural network was the current state s and the action a, and the output was the action value function $Q(s, a)$, as shown in Figure 8. The DQN algorithm reduced the correlation between the training data through experience replay technology so as to make the neural network training more stable. At the same time, the DQN also used the target network to update the temporal difference (TD) bias, which further reduced the temporal correlation between pieces of the training data. The experience replay technology stored the state transition of the interaction with the environment in the experience replay pool. The DQN algorithm used a target network with the same structure as the online network. This target network was periodically updated by copying the parameters from the online network. By using the target network to generate target Q-values, the temporal correlation between the fragmented sample data was reduced, leading to a more stable training process.

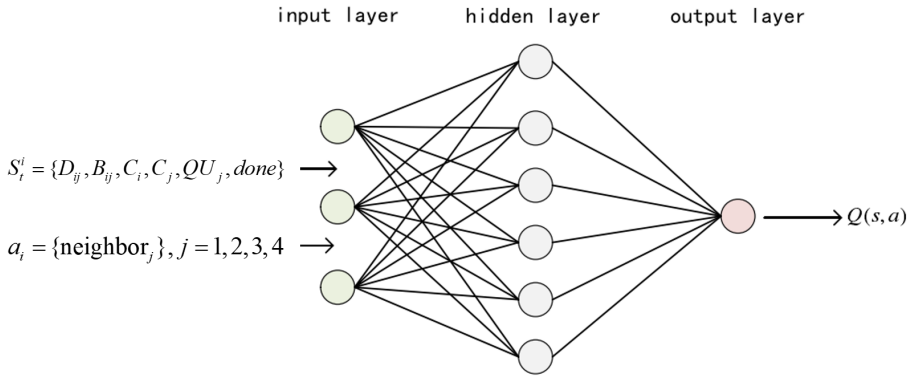

Figure 8. The DQN neural network.

When a satellite service request was received by a certain satellite, the state (input) of the current satellite (agent) was determined by the status information of the surrounding satellites, and the action (output) was to select one of these satellites as the next hop in routing. At this time, in the DQN network, $Q(s, a)$ was calculated by the estimation Q network according to the current satellite state s and the current routing decision a. After $Q(s, a)$ was calculated, the agent took an action that maximized this value and selected the satellite with the highest Q-value as the next hop for routing the service request. The selected satellite then updated its status information and sent the service request to the next hop in the route until it reached its destination.

During the process of selecting the next hop, the DQN network learned from the experiences gained through each interaction with the environment. Specifically, the agent stored these experiences in a replay memory buffer and used them for training the Q network. The experiences included the current state s, the chosen action a, the reward r received from the environment, and the resulting next state s'.

The Q network was trained by minimizing the difference between the predicted Q-value and the target Q-value. The target Q-value was calculated based on the Bellman equation, which considers the discounted future rewards of taking an action from the current state.

Through this reinforcement learning process, the DQN network could gradually improve the quality of its routing decisions and adapt to the changing environment of the satellite network.

The flow chart of the DQN algorithm is shown in Figure 9. During training, an experience replay pool was used to store information about each interaction with the environment, including the states, actions, and immediate rewards from environmental feedback. For each decision, the generated samples (s, a, s', r) were stored in the experience replay pool, and a certain number of samples was taken out at intervals to train the network so as to reduce the correlation of the training data and prevent overfitting. From the data extracted from the experience pool, the value of (s, a) was input into the online Q network for computing $Q(s, a)$, and the state s' was input into the target Q network for computing the target Q value. The online Q network parameters were updated by the gradient descent method based on the error function gradient every C steps. The target Q network periodically updated its own parameters by replicating the parameters of the online Q network.

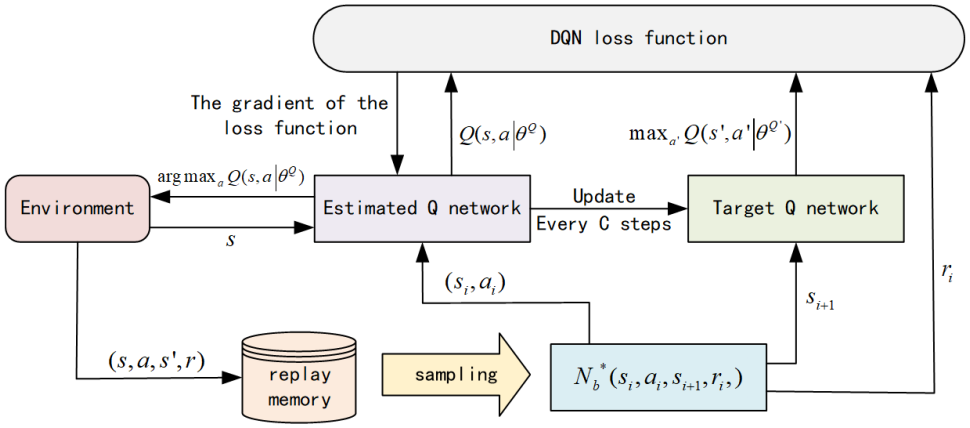

Figure 9. The flowchart of the DQN algorithm.

In the DQN algorithm, the Q-network output the action-value function, which was fitted using a neural network due to the continuous state space. The Q-network was updated using the mean square error, which was calculated through the TD bias. The TD deviation was computed using both the target Q-network and the online Q-network. The TD deviation was calculated as follows.

$$L = \frac{1}{N_b} \sum_t \left(y_t - Q(s_t, a_t | \theta^Q)\right)^2. \tag{6}$$

Here, N_b is the random sampling capacity, θ^Q is the parameter of the estimated Q network, and y_t is the target Q value, which represents the Q value obtained by calculating the action a_{t+1} at the next state s_{t+1} in the target network, and the action that produces the maximum target Q value at the next state s_{t+1} is taken as the next action a_{t+1}. y_t is computed as follows:

$$y_t = r_t + \gamma \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1} | \theta^{Q'}), \tag{7}$$

where r_t represents the reward value obtained by the agent for taking action a_t under status s_t. The parameter of the target Q network was $\theta^{Q'}$, while $\gamma \in [0, 1]$ denoted the discount factor.

Thus, the update mode of the Q network parameters was obtained as follows in Equation (4).

$$\theta_{t+1}^Q = \theta^Q + \alpha[r_i + \gamma \max_{a_{i+1}} Q'(s_{i+1}, a_{i+1} | \theta^{Q'}) - Q(s_i, a_i | \theta^Q)] \nabla Q(s_i, a_i | \theta^Q) \ . \tag{8}$$

5. Simulation and Performance Analysis

In this work, we employed the Win11 64-bit system, used the Python torch deep learning framework, and utilized the programming language Python3.9 to carry out the simulation experiments and analyze the results. The experiment was carried out under the constructed Iridium satellite model, and the Iridium satellite topology was generated by using the official satellite data files of the Iridium constellation and Python networkX. As shown in Figure 10, the generated Iridium constellation contained six orbits, and each orbit contained 12 satellites. At the same time, the delay and bandwidth were randomly generated for each inter-satellite link. In order to simulate the dynamic network environment in the topology below, 70 streams of data flow were generated in a loop at the same time to propagate in the topology, so that the agent could obtain dynamic queue utilization and link bandwidth.

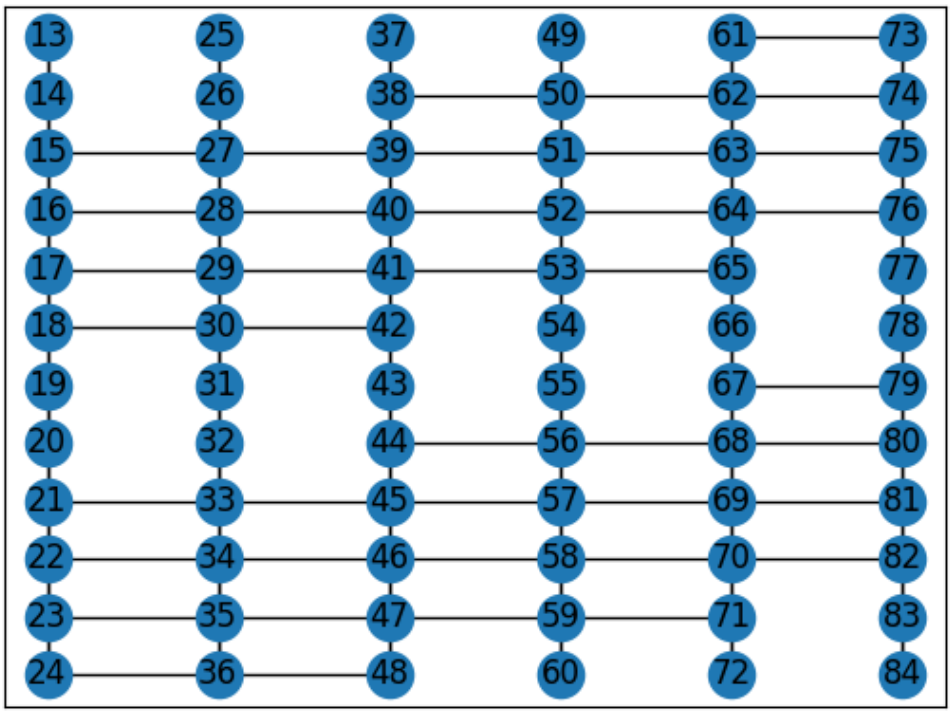

Figure 10. Iridium satellite topology.

The DQN algorithm in this paper was configured with a neural network that consisted of an input layer, an output layer, and two hidden layers. The discount factor was set to 0.99, and the experience replay pool size was 4000. The DQN-LLRA algorithm trained the network with 10,000 randomly generated data streams and produced a routing decision model after convergence.

The relevant simulation parameters are shown in Table 1.

Table 1. Parameter settings for evaluation.

Parameter	Symbol	Value
link delay	D_{ij}	random value (10–20 ms)
link bandwidth	B_{ij}	10 Mbps
discount factor	γ	0.99
learning rate	α	0.005
reward parameter weights	$\alpha_i, \beta_i, \gamma_i, \omega_i,$ $i = 1, 2, 3, 4$	0.45, 0.35, 0.1, 0.1 0.50, 0.30, 0.1, 0.1 0.40, 0.40, 0.1, 0.1 0.50, 0.30, 0.1, 0.1
reward adjustment factors	o, p, q	10, 20, 20

Figure 11 shows the variation trend of the reward with the training data stream.

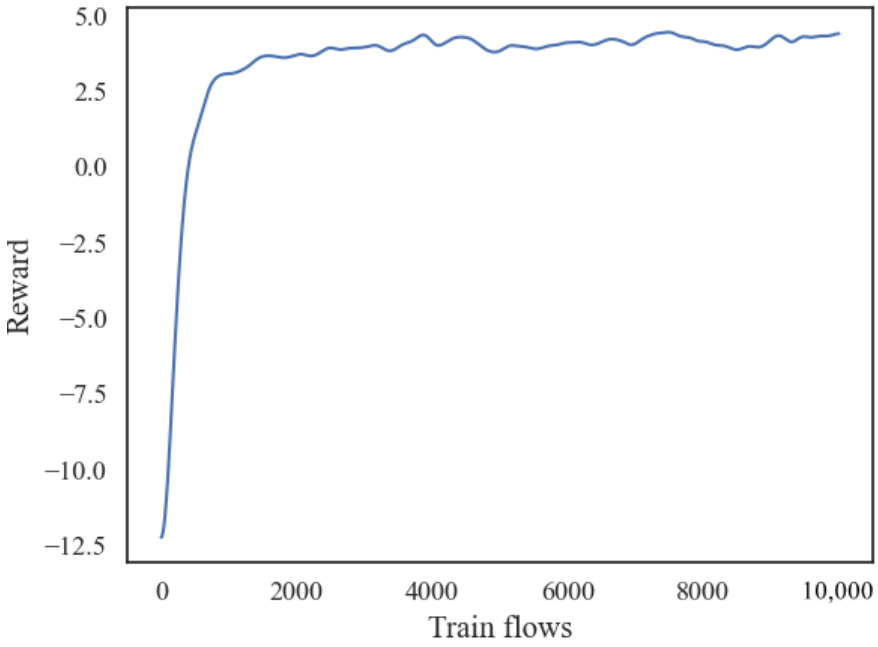

Figure 11. Reward as a function of the number of training streams.

The Dijkstra-QU algorithm was used to find the shortest path after transforming the queue utilization into the link weight. The algorithm aimed at the lowest queue utilization but did not consider the path hop count and path delay. We compared our proposed scheme with the routing scheme based on reinforcement learning in [45]. In this benchmark scheme, Q-learning was used as the solution method for the satellite-routing decision process. At the same time, the packets in the routing process were used as agents. Each packet made a routing decision for the next hop based on the surrounding environment of the current satellite and destination node. The following performance evaluation compared the traditional Dijkstra algorithm, the Dijkstra-QU algorithm, the Q-learning-based scheme, and the proposed DQN-LLRA algorithm.

To simulate the overhead resulting from the agent–environment interaction in the machine learning algorithms, we decreased the bandwidth around the decision node in both the Q-Learning-based intelligent routing scheme and the proposed routing scheme. This promoted the fairness of the comparison with other benchmark algorithms.

In this work, 100 pairs of the source and destination nodes were generated to evaluate the performance of the four different routing algorithms. For each algorithm, 100 routing paths were calculated, resulting in a total of 400 paths. A mathematical analysis was conducted to evaluate the delay performance of the algorithms.

From Figure 12, it can be seen that in the test with 100 pairs of source–destination nodes, the Dijkstra algorithm had the lowest latency among the four algorithms. In contrast, the Dijkstra-QU algorithm with queue utilization as the optimization objective had the highest latency. The DQN-LLRA algorithm proposed in this paper and the Q-learning-based routing algorithm both considered the impact of queue utilization, and their latency performance was between the Dijkstra algorithm and the Dijkstra-QU scheme. Obviously, the proposed algorithms had better latency performance than the Q-learning-based algorithm.

Figures 13 and 14 present an analysis of the load-balancing parameters for the three routing algorithms.

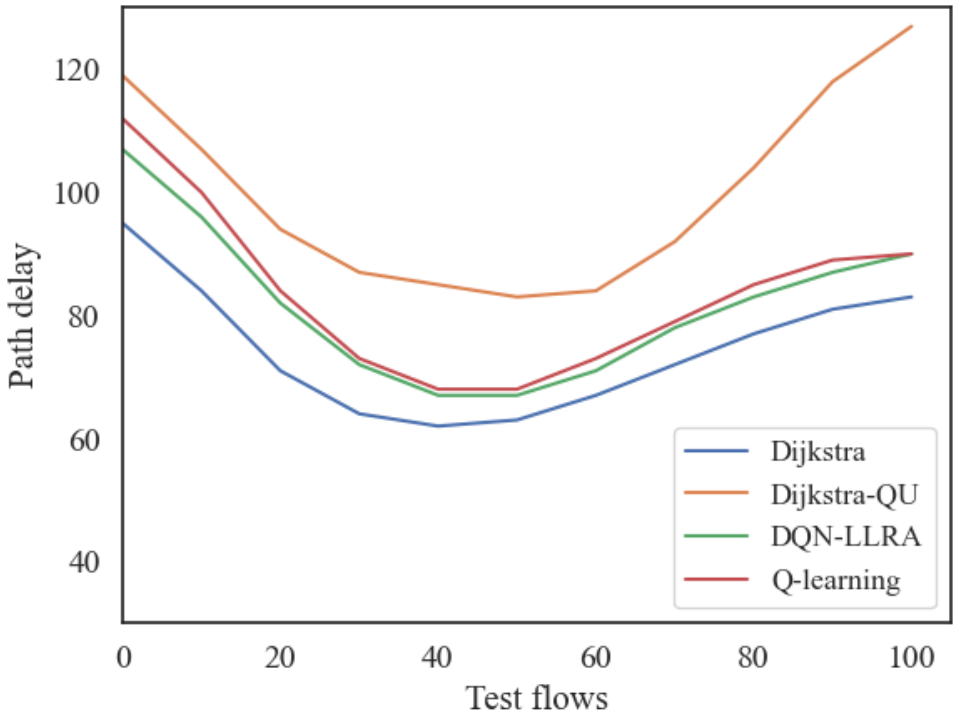

Figure 12. Path delay of different algorithms under various test flows.

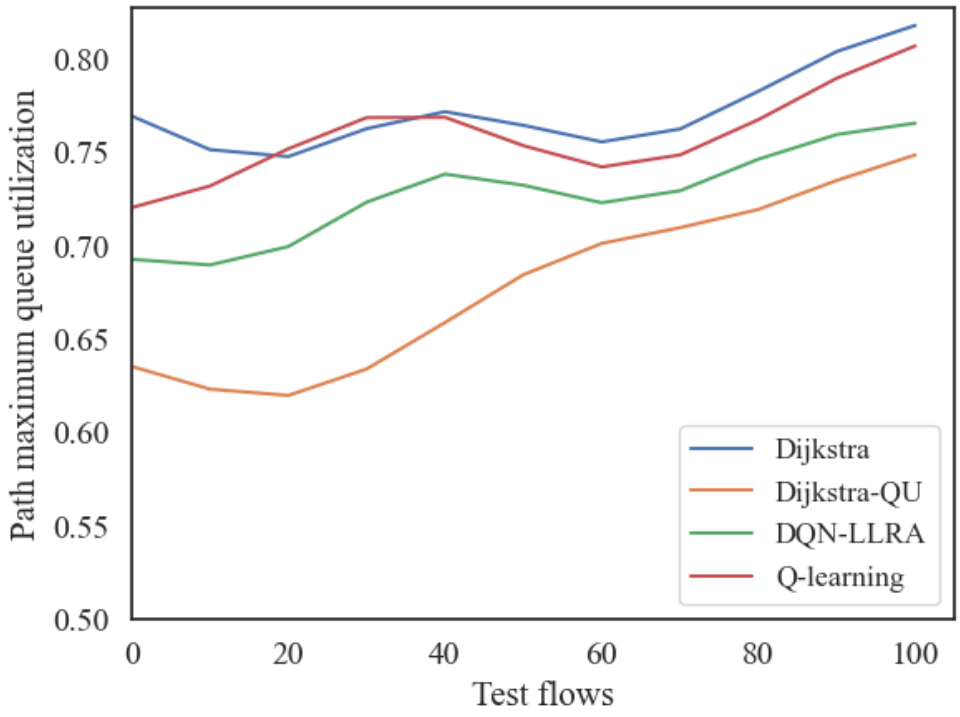

Figure 13. Path maximum queue utilization rates of different algorithms under various test flows.

From Figure 13, it can be seen that in the test with 100 pairs of source–destination nodes, the maximum queue utilization rate of the Dijkstra-QU algorithm was the lowest among the four algorithms, while the maximum queue utilization rate of the Dijkstra algorithm was the highest. The maximum queue utilization rate of the proposed DQN-LLRA algorithm and the Q-learning-based routing algorithm was between the Dijkstra algorithm and the Dijkstra-QU scheme.

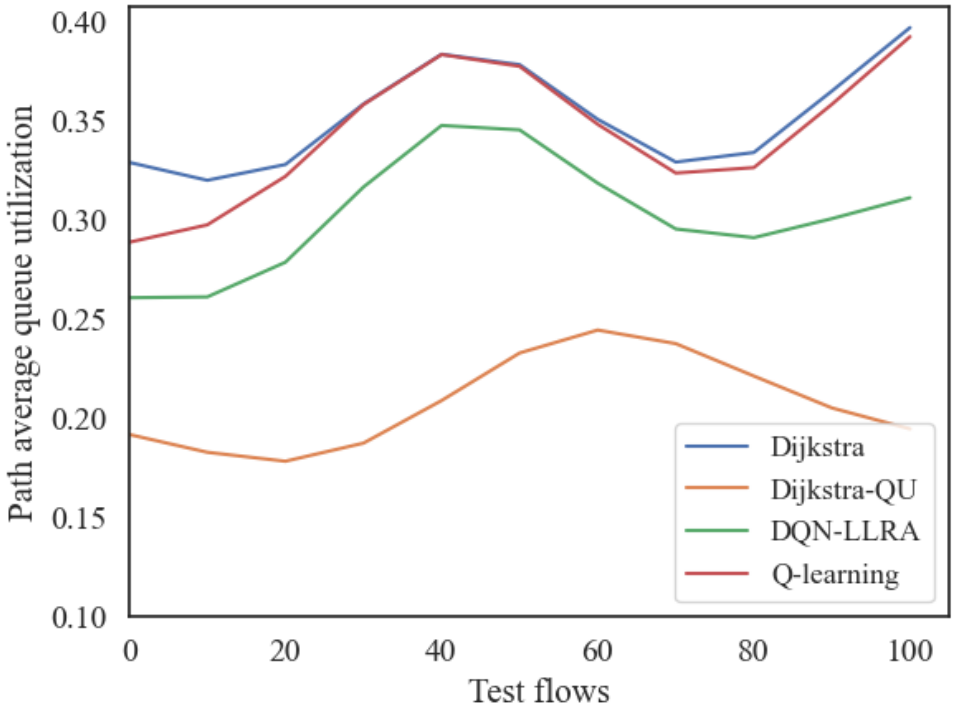

Figure 14. Path average queue utilization rates of different algorithms under various test flows.

From Figure 14, it can be seen that in the test with 100 source–destination node pairs, the average queue utilization rate of the Dijkstra-QU algorithm was the lowest among the four algorithms, while the average queue utilization rate of the Dijkstra algorithm was the highest. The average queue utilization rates of the proposed DQN-LLRA algorithm and the Q-learning-based routing algorithm were between the Dijkstra algorithm and the Dijkstra-QU scheme. Moreover, the proposed DQN-LLRA algorithm showed a lower queue utilization rate than the Q-learning-based algorithm.

Thus, the DQN-LLRA algorithm proposed in this paper can be considered superior to the traditional shortest-path algorithm and the Q-learning-based algorithm in terms of load-balancing performance.

According to the experimental results, the Dijkstra algorithm calculated the shortest path in the experiment, but the maximum average queue utilization and the link utilization rate for it were the highest, and as a load-balancing algorithm for satellite node processing, the power was poorer. By contrast, the Dijkstra-QU algorithm that set queue utilization as the optimization goal did better in determining the path to load, but its path delay was larger, and it also could not meet the requirements of the low delay of the satellite network. The path obtained by the DQN-LLRA algorithm in this paper not only ensured a low delay but also responded effectively to the load situation in the network, reducing the maximum and average queue utilization of the link. Compared with the Dijkstra algorithm, the proposed algorithm reduced the maximum link queue utilization by 8%, reduced the average queue utilization by 15%, and had a robust load-balancing ability. Compared with the Q-learning-based algorithm, the proposed algorithm reduced the maximum link queue utilization by 5%, reduced the average queue utilization by 13%, and had a robust load-balancing ability.

6. Conclusions

We proposed a Markov decision-process model for route generation and introduced a deep-reinforcement-learning-based LEO satellite-routing algorithm called DQN-LLRA in this paper. In the DQN-LLRA scheme, each routing node is controlled by a DQN agent, which can quickly select the best next hop for the current satellite node. The algorithm only relies on the state information of the surrounding nodes of the current satellite node

and obtains the optimal decision model through iterative training. Compared with the traditional routing algorithm through numerical simulation, our algorithm can reduce the maximum queue utilization and the average queue utilization of the routing path, reduce the occurrence of congestion to a certain extent, and realize the satellite network routing under load balancing subject to the condition of ensuring that the delay cost increases only a little. This paper also found that when the number of satellite nodes increased to thousands, the routing path exceeded 20 hops, and then the convergence speed of the DQN-LLRA algorithm decreased, and the accuracy of the decision model also decreased. Therefore, using more effective neural network structures such as Graph Neural Networks (GNN) for network feature extraction is an important direction for future research.

Although the proposed load-balancing routing algorithm for LEO satellites based on Deep Q-Network shows promising performance improvements compared to traditional routing algorithms, there are still some limitations and areas for further research. Firstly, the proposed algorithm is evaluated through simulations, and it remains unclear how well it would perform in real-world scenarios. Further experiments and measurements in actual LEO satellite networks are necessary to validate its effectiveness. Secondly, the proposed algorithm considers delay, bandwidth, and queue utilization of the surrounding satellite nodes as factors for selecting the best routing results. Other factors, such as link reliability, congestion level, and energy efficiency, may also be considered in future work. Lastly, while deep reinforcement learning has shown great potential in improving network routing, it is a computationally expensive approach that requires significant training time and resources. Hence, developing more efficient and scalable algorithms that can maintain good performance with a lower computational overhead is also an important direction for future research. The proposed DQN-LLRA algorithm is a novel approach for load-balancing routing in LEO satellite networks. However, further research is needed to address the abovementioned limitations and achieve more robust and efficient routing.

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

Funding: This paper was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1807500 and in part by the 54th Research Institute of China Electronics Technology Group Corporation Cooperative Project under Grant SKX212010039.

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

Acknowledgments: These authors would like to thank the China Electronics Technology Group for providing some orbital statistics.

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

Abbreviations

The following abbreviations are used in this manuscript:

SAGIN	Space–Air–Ground Integrated Network
GEO	Geosynchronous Earth Orbit
MEO	Medium Earth Orbit
LEO	Low Earth Orbit
DQN	Deep Q-Network
ISLs	inter-satellite links
DQN-LLRA	DQN-based load-balancing routing algorithm for LEO satellites
DRA	Discrete Relaxation Algorithm

DHRP	Distributed Hierarchical Routing Protocol
HGL	Hybrid Global–Local Load Balancing Routing
PIR	Probability ISL Routing
ELB	Elastic Load Balancing
IoT	Internet of Things
WSNs	Wireless Sensor Networks
DDPG	Deep Deterministic Policy Gradient
MADDPG	Multiagent Deep Deterministic Policy Gradient
MDP	Markov Decision Process
TD	Temporal Difference
GNN	Graph Neural Networks

References

1. Mohori, M.; vigelj, A.; Kandus, G.; Werner, M. Performance evaluation of adaptive routing algorithms in packet-switched intersatellite link networks. *Int. J. Satell. Commun. Netw.* **2002**, *20*, 97–120. [CrossRef]
2. Liu, L.; Feng, J.; Pei, Q.; Chen, C.; Ming, Y.; Shang, B.; Dong, M. Blockchain-Enabled Secure Data Sharing Scheme in Mobile-Edge Computing: An Asynchronous Advantage Actor–Critic Learning Approach. *IEEE Internet Things J.* **2021**, *8*, 2342–2353. [CrossRef]
3. Zhang, Y.; Chen, C.; Liu, L.; Lan, D.; Jiang, H.; Wan, S. Aerial Edge Computing on Orbit: A Task Offloading and Allocation Scheme. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 275–285. [CrossRef]
4. Chen, C.; Wang, C.; Liu, B.; He, C.; Cong, L.; Wan, S. Edge Intelligence Empowered Vehicle Detection and Image Segmentation for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2023**, 1–12. [CrossRef]
5. Chen, C.; Yao, G.; Liu, L.; Pei, Q.; Song, H.; Dustdar, S. A Cooperative Vehicle-Infrastructure System for Road Hazards Detection With Edge Intelligence. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 5186–5198. [CrossRef]
6. Alagoz, O.; Hsu, H.E.; Schaefer, A.J.; Roberts, M.S. Markov Decision Processes: A Tool for Sequential Decision Making under Uncertainty. *Med. Decis. Mak.* **2010**, *30*, 474–483. [CrossRef]
7. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.A.; Fidjeland, A.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
8. Sağ, E.; Kavas, A. Modelling and Performance Analysis of 2.5 Gbps Inter-satellite Optical Wireless Communication (IsOWC) System in LEO Constellation. *J. Commun.* **2018**, *13*, 553–558. [CrossRef]
9. Pizzicaroli, J.C. Launching and Building the IRIDIUM® Constellation. In *Proceedings of the Mission Design & Implementation of Satellite Constellations*; van der Ha, J.C., Ed.; Springer: Dordrecht, The Netherlands, 1998; pp. 113–121.
10. Henderson, T.; Katz, R. On distributed, geographic-based packet routing for LEO satellite networks. In Proceedings of the Globecom'00-IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137), San Francisco, CA, USA, 27 November–1 December 2000; Volume 2, pp. 1119–1123. [CrossRef]
11. Svigelj, A.; Mohorcic, M.; Kandus, G.; Kos, A.; Pustisek, M.; Bester, J. Routing in ISL networks considering empirical IP traffic. *IEEE J. Sel. Areas Commun.* **2004**, *22*, 261–272. [CrossRef]
12. Liu, L.; Zhao, M.; Yu, M.; Jan, M.A.; Lan, D.; Taherkordi, A. Mobility-Aware Multi-Hop Task Offloading for Autonomous Driving in Vehicular Edge Computing and Networks. *IEEE Trans. Intell. Transp. Syst.* **2022**, 24, 2169–2182. [CrossRef]
13. Gounder, V.; Prakash, R.; Abu-Amara, H. Routing in LEO-based satellite networks. In Proceedings of the 1999 IEEE Emerging Technologies Symposium. Wireless Communications and Systems (IEEE Cat. No.99EX297), Richardson, TX, USA, 12–13 April 1999; pp. 22.1–22.6. [CrossRef]
14. Long, H.; Shen, Y.; Guo, M.; Tang, F. LABERIO: Dynamic load-balanced Routing in OpenFlow-enabled Networks. In Proceedings of the 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), Barcelona, Spain, 25–28 March 2013; pp. 290–297. [CrossRef]
15. Franck, L.; Maral, G. Routing in networks of intersatellite links. *IEEE Trans. Aerosp. Electron. Syst.* **2002**, *38*, 902–917. [CrossRef]
16. Mohorcic, M.; Werner, M.; Svigelj, A.; Kandus, G. Adaptive routing for packet-oriented intersatellite link networks: Performance in various traffic scenarios. *IEEE Trans. Wirel. Commun.* **2002**, *1*, 808–818. [CrossRef]
17. Kucukates, R.; Ersoy, C. Minimum flow maximum residual routing in LEO satellite networks using routing set. *Wirel. Netw.* **2008**, *14*, 501–517. [CrossRef]
18. Cigliano, A.; Zampognaro, F. A Machine Learning approach for routing in satellite Mega-Constellations. In Proceedings of the 2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Virtual, 25–27 November 2020; pp. 1–6. [CrossRef]
19. Wang, H.; Ran, Y.; Zhao, L.; Wang, J.; Luo, J.; Zhang, T. GRouting: Dynamic Routing for LEO Satellite Networks with Graph-based Deep Reinforcement Learning. In Proceedings of the 2021 4th International Conference on Hot Information-Centric Networking (HotICN), Nanjing, China, 25–27 November 2021; pp. 123–128. [CrossRef]
20. Kucukates, R.; Ersoy, C. High performance routing in a LEO satellite network. In Proceedings of the Eighth IEEE Symposium on Computers and Communications, Kemer-Antalya, Turkey, 30 June–3 July 2003; Volume 2, pp. 1403–1408. [CrossRef]
21. Liu, W.; Tao, Y.; Liu, L. Load-Balancing Routing Algorithm Based on Segment Routing for Traffic Return in LEO Satellite Networks. *IEEE Access* **2019**, *7*, 112044–112053. [CrossRef]

22. Ju, Y.; Zou, G.; Bai, H.; Liu, L.; Pei, Q.; Wu, C.; Otaibi, S.A. Random Beam Switching: A Physical Layer Key Generation Approach to Safeguard mmWave Electronic Devices. *IEEE Trans. Consum. Electron.* **2023**, 1. [CrossRef]
23. Taleb, T.; Mashimo, D.; Jamalipour, A.; Hashimoto, K.; Nemoto, Y.; Kato, N. SAT04-3: ELB: An Explicit Load Balancing Routing Protocol for Multi-Hop NGEO Satellite Constellations. In Proceedings of the IEEE Globecom 2006, San Francisco, CA, USA, 27 November–1 December 2006; pp. 1–5. [CrossRef]
24. Taleb, T.; Mashimo, D.; Jamalipour, A.; Kato, N.; Nemoto, Y. Explicit Load Balancing Technique for NGEO Satellite IP Networks With On-Board Processing Capabilities. *IEEE/ACM Trans. Netw.* **2009**, *17*, 281–293. [CrossRef]
25. Song, G.; Chao, M.; Yang, B.; Zheng, Y. TLR: A Traffic-Light-Based Intelligent Routing Strategy for NGEO Satellite IP Networks. *IEEE Trans. Wirel. Commun.* **2014**, *13*, 3380–3393. [CrossRef]
26. Liu, J.; Luo, R.; Huang, T.; Meng, C. A Load Balancing Routing Strategy for LEO Satellite Network. *IEEE Access* **2020**, *8*, 155136–155144. [CrossRef]
27. Geng, S.; Liu, S.; Fang, Z.; Gao, S. An optimal delay routing algorithm considering delay variation in the LEO satellite communication network. *Comput. Netw.* **2020**, *173*, 107166. [CrossRef]
28. Wei, S.; Cheng, H.; Liu, M.; Ren, M. Optimal Strategy Routing in LEO Satellite Network Based on Cooperative Game Theory. In Proceedings of the Space Information Networks: Second International Conference, SINC 2017, Yinchuan, China, 10–11 August 2017.
29. Jiang, Z.; Liu, C.; He, S.; Li, C.; Lu, Q. A QoS routing strategy using fuzzy logic for NGEO satellite IP networks. *Wirel. Netw.* **2018**, *24*, 295–307. [CrossRef]
30. Pan, T.; Huang, T.; Li, X.; Chen, Y.; Xue, W.; Liu, Y. OPSPF: Orbit Prediction Shortest Path First Routing for Resilient LEO Satellite Networks. In Proceedings of the ICC 2019–2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6. [CrossRef]
31. Hao, L.; Ren, P.; Du, Q. Satellite QoS Routing Algorithm Based on Energy Aware and Load Balancing. In Proceedings of the 2020 International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing, China, 21–23 October 2020; pp. 685–690. . [CrossRef]
32. Zuo, P.; Wang, C.; Wei, Z.; Li, Z.; Zhao, H.; Jiang, H. Deep Reinforcement Learning Based Load Balancing Routing for LEO Satellite Network. In Proceedings of the 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), Helsinki, Finland, 19–22 June 2022; pp. 1–6. [CrossRef]
33. Xu, Q.; Zhang, Y.; Wu, K.; Wang, J.; Lu, K. Evaluating and Boosting Reinforcement Learning for Intra-Domain Routing. In Proceedings of the 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Monterey, CA, USA, 4–7 November 2019; pp. 265–273. [CrossRef]
34. Ju, Y.; Chen, Y.; Cao, Z.; Liu, L.; Pei, Q.; Xiao, M.; Ota, K.; Dong, M.; Leung, V.C.M. Joint Secure Offloading and Resource Allocation for Vehicular Edge Computing Network: A Multi-Agent Deep Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 5555–5569. [CrossRef]
35. Liu, Y.; Lu, D.; Zhang, G.; Tian, J.; Xu, W. Q-Learning Based Content Placement Method for Dynamic Cloud Content Delivery Networks. *IEEE Access* **2019**, *7*, 66384–66394. [CrossRef]
36. Pan, S.; Li, P.; Zeng, D.; Guo, S.; Hu, G. A *Q*-Learning Based Framework for Congested Link Identification. *IEEE Internet Things J.* **2019**, *6*, 9668–9678. [CrossRef]
37. Wei, Z.; Liu, F.; Zhang, Y.; Xu, J.; Ji, J.; Lyu, Z. A Q-learning algorithm for task scheduling based on improved SVM in wireless sensor networks. *Comput. Netw.* **2019**, *161*, 138–149. [CrossRef]
38. Qiao, G.; Leng, S.; Maharjan, S.; Zhang, Y.; Ansari, N. Deep Reinforcement Learning for Cooperative Content Caching in Vehicular Edge Computing and Networks. *IEEE Internet Things J.* **2020**, *7*, 247–257. [CrossRef]
39. Tu, Z.; Zhou, H.; Li, K.; Li, G.; Shen, Q. A Routing Optimization Method for Software-Defined SGIN Based on Deep Reinforcement Learning. In Proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps), Big Island, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]
40. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.M.O.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
41. Ju, Y.; Yang, M.; Chakraborty, C.; Liu, L.; Pei, Q.; Xiao, M.; Yu, K. Reliability-Security Tradeoff Analysis in mmWave Ad Hoc Based CPS. *ACM Trans. Sens. Netw.* **2023**. [CrossRef]
42. Ju, Y.; Wang, H.; Chen, Y.; Zheng, T.X.; Pei, Q.; Yuan, J.; Al-Dhahir, N. Deep Reinforcement Learning Based Joint Beam Allocation and Relay Selection in mmWave Vehicular Networks. *IEEE Trans. Commun.* **2023**, *71*, 1997–2012. [CrossRef]
43. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6382–6393.
44. Qin, Z.; Yao, H.; Mai, T. Traffic Optimization in Satellites Communications: A Multi-agent Reinforcement Learning Approach. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 269–273. [CrossRef]

45. Yin, Y.; Huang, C.; Wu, D.; Huang, S.C.; Ashraf, M.W.A.; Guo, Q. Reinforcement Learning-Based Routing Algorithm in Satellite-Terrestrial Integrated Networks. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 3759631. [CrossRef]