Distributed Routing Algorithm for LEO Satellite Network Based on Deep Reinforcement Learning

Yudie Chen[1,2,3], Lan Wang[1], Houze Liang[2], Dong Lv[4], Weizhi Wu[4], Xiang Chen[2(✉)], and Terngyin Hsu[5]

[1] College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China
[2] School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China
chenxiang@mail.sysu.edu.cn
[3] Research Institute of Tsinghua University in Shenzhen (RITS), Shenzhen, China
[4] IPLOOK Technologies Co. Ltd., Guangzhou, China
[5] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

Abstract. Low earth orbit (LEO) satellites have the advantages of low transmission delay and wide coverage, which are widely used in current and future satellite communication systems. However, the large-scale and unevenly distributed traffic will easily cause severe inter-satellite-link (ISL) congestion. Therefore, the inter-satellite routing design of the LEO satellite networks is of vital importance. Traditional shortest path-based routing algorithms in LEO networks might cause overlapped loops that aggravate network delay. In this paper, by introducing the deep Q-network (DQN) into routing design, a distributed routing algorithm is proposed to reduce the delay and relieve congestion. Particularly, a reward function, considering routing loop as restriction to identify the best pathways, is designed to prevent the agent from getting stuck in a loop owing to duplicate paths. Meanwhile, instead of simply selecting a fixed shortest path, the proposed DQN model selects the next hop with a certain random probability, which helps to avoid overlapped paths and efficiently relieve congestion. A hybrid OPNET-STK simulation platform is built up for a typical LEO constellation with more than 200 satellites. Simulation results reveal that, compared with the traditional algorithms, the proposed approach can effectively relieve the ISL congestion and brings higher throughput for the LEO networks.

Keywords: low earth orbit satellite · deep Q networks · routing algorithm

This research is supported in part by the State's Key Project of Research and Development Plan under Grants (2019YFE0196400), and in part by the Shenzhen Natural Science Foundation under Grant JCYJ20200109143016563.

K. Rabie et al. (Eds.): IoTaaS 2022, LNICST 506, pp. 161–173, 2023.
https://doi.org/10.1007/978-3-031-37139-4_15

1 Introduction

Satellite communication technologies are developing continuously, and LEO satellite communication networks with the advantages of global coverage and on-demand services have been re-injected with new vitality [1]. However, with the rapid development of LEO satellite communication networks, the user traffic is also growing dramatically and the distribution is very uneven, making the ISLs prone to congestion. Then the Quality of Service (QoS) of satellite networks is greatly reduced [2]. Therefore, how to design a load balancing routing strategy with regional distribution is particularly imperative in the construction of LEO satellite communication networks [3].

A number of routing algorithms have been proposed for LEO satellite networks. Dijkstra Shortest Path (DSP) algorithm utilizes a greedy algorithm strategy to traverse all nodes until a shortest path is found between the source node and destination node. However, DSP algorithm cannot adjust routing strategy when the LEO satellite network topology changes dynamically, which will cause severe packet loss when the traffic is enormous and uneven. On this basis, several algorithms for dealing with network congestion are proposed. Tarik et al. [4] proposed an Explicit Load Balancing (ELB) algorithm that can exchange congestion information of neighboring satellites. When a node is congested, it requests to reduce the packet transmission rate by sending self-congestion notifications to its neighboring nodes, so that the neighboring nodes can bypass this congested node to choose a more suitable path. Besides, by differentiating the services and guaranteeing the service rates of different QoS, ELB can reduce the packet loss rate via better traffic scheduling. Nevertheless, ELB does not consider the next-hop delay and hence cannot ensure the efficiency and speed of routing. The Distributed Hierarchical Routing Protocol (DHRP) [5] assigns a speaker in each orbit. The speaker periodically collects queuing delay information and sends it to other satellites in the same orbit to assist in updating the whole network topology. However, it is a challenge for DHRP to determine a proper period of delay information broadcasting and then balance the real time of optimal routes and communication load. Song et al. [6] proposed a Traffic Light-based intelligent Routing (TLR) algorithm for satellite networks. A set of traffic light signals is used in this algorithm to indicate the congestion status of the current node and the next node. An approximately optimal transmission path for each packet can be obtained by combining initial planning and real-time adjustment. The public waiting queue scheme of TLR can fully use the free space in the cache queue and reduce the packet loss rate. However, the TLR algorithm defaults the transmission delay to a fixed value. It simplifies the time-varying transmission delay model to a static model, so the next hop of the TLR algorithm is not always the best choice. In summary, the above traditional algorithms all select a fixed shortest path, so that there will be a large quantity of repeated paths in different routes, which will aggravate congestion of ISLs and degrade the QoS of users.

In recent years, the introduction of deep learning has brought new room for improvements in routing algorithms. Bomin Mao et al. [7] used the deep

belief network to extract the deep dimension features in the routing information. Taking the current routing information tensor as input, the deep belief network can automatically give the optimal next hop. However, the statistics of network link states are hysteretic, and the modeling and implementation of the tensor are relatively complex for all networks. Peiliang Zuo et al. [8] proposed an intelligent distributed routing algorithm in dynamic LEO satellite networks, which integrates the spatial location of nodes, mutual distance, queuing delay and available bandwidth to select routing paths. Despite the fact that the above routing algorithms have has good convergence and lower latency, these methods do not constrain the duplicate paths when training the model, resulting in relatively more loops.

In this paper, we propose a distributed routing algorithm based on deep reinforcement learning. Our contributions can be summarized as:

- The routing algorithm selects the next hop with a certain random probability instead of the minimum delay. It avoids the problem of path overlap in different routing paths, effectively alleviates the congestion caused by using a single optimal path and increases the network's overall link utilization as well.
- In the training of the Deep Q Network (DQN) model, the queuing delay and propagation delay between nodes are also embedded into the reward function to help to select the optimal path in the LEO satellite networks. Furthermore, the reward function constraint on the agent is used to prevent from getting into loops by choosing duplicate paths.

The rest of this article is outlined below. Section 2 briefly introduces the system model and optimization problem construction based on the DQN architecture. Section 3 presents the implementation of the proposed method in the OPNET platform. Simulation results and analysis are performed in Sect. 4. Finally, Sect. 5 draws some conclusions.

2 System Model

2.1 System Model

The inter-satellite link routing algorithm proposed in this paper is based on a LEO constellation composed of 228 satellites and 6 ground stations(GS), as shown in Fig. 1. The constellation has 12 evenly distributed inclined orbits and in each orbit there are 19 satellites, which are also evenly distributed. We utilize the STK software to generate the corresponding parameters such as orbit number, orbit inclination, geocentric coordinates. All of the these parameters are then imported to the OPNET platform to build a simulation environment of this constellation. Besides, each satellite has 4 inter-satellite links (2 inter-orbital links and 2 intra-orbital links) for communicating with neighboring satellites.Two adjacent satellites can only communicate when their transmitters and receivers oriented towards each other are using the same frequency band. The processing

capacity and load capacity of different satellite nodes in the constellation are significantly different. In addition, 6 ground stations are distributed in densely populated or vast cities.

Packets are generated in a uniform distribution at ground station and forwarded to the closest satellite that the source ground station can access. Then they are routed to the satellite covering the destination ground station. Finally, they are delivered to the destination ground station. When the networks are bearing great service traffic, the satellites in a specific region might be overloaded, while the ones in other regions can be idle. In this case, the queuing delay and available bandwidth of different nodes and even different directions of the same node can be completely different. To realize traffic scheduling, we suppose each LEO satellite are equipped with a performance monitor, which can record the packet sending rate and calculate the current waiting time of the queues in four directions.

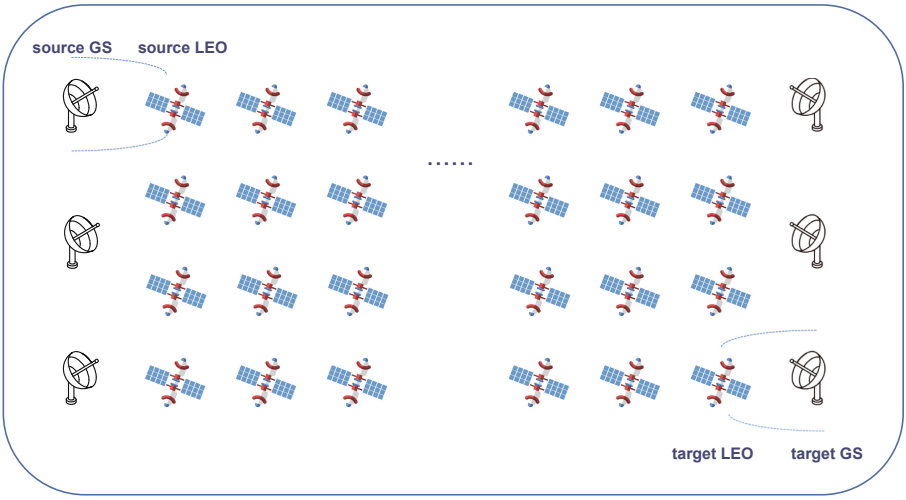

Fig. 1. Topology of LEO satellite networks.

2.2 Problem Formulation

The end-to-end(E2E) delay of ISLs routing is usually composed of propagation delay and queuing delay. The propagation delay is associated with the distance between two satellites. Additionally, the queuing delay is related to the congestion level of the satellite link. This paper defines the queuing delay of the links as:

$$D_q = l/r_q, \tag{1}$$

where l is the total length of the data packets in the queue and r_q is the data transmission rate of the queue. The propagation delay is determined by the distance s between satellite nodes and the light speed c, denoted as:

$$D_p = s/c. \tag{2}$$

Therefore, the E2E delay of the inter-satellite link can be expressed as:

$$D = D_p + D_q. \tag{3}$$

The packet loss rate in the LEO network is also an important concern of the routing algorithm. The packet loss rate is defined as:

$$P = N_r/N_s, \tag{4}$$

where N_s represents the number of packets forwarded from the satellite covering the source ground station, meanwhile, N_r represents the number of packets received by the satellite covering the target ground station.

In addition, loops in the network will performance of the routing algorithm, such as E2E delay and packet loss rate, etc. The loop rate is defined as following:

$$R = R_l/R_w. \tag{5}$$

In order to save computing resources, we only calculate the routing paths between the satellites visible to the ground station. We denote the number of routing paths between these visible satellites as R_w, and denote the routing paths with loops as R_l. R is used as a metric to analyze the impact of routing loops on delay and packet loss rate.

3 The Proposed DQN-Based Routing Algorithm

Agents in reinforcement learning (RL) can interact with the environment to search for the best path without knowing the environment. The value of a state and an action under a given policy is usually referred to as Q value. Traditional RL algorithms use a Q-table to record the Q values. However, when the combination of states and actions is complex, the Q-table is required to have a large storage space, so using traditional RL algorithms is not the best choice. In contrast, DQN uses a deep neural network as a function that takes the state and action as inputs and output the corresponding Q value. This avoids maintaining a large table. Therefore, this paper introduces a routing algorithm based on DQN, which is suitable for the above-mentioned LEO satellite network. The routing algorithm aims to find the optimal path under the constraint of packet loss rate and minimize the end-to-end delay.

3.1 Markov Decision Process Formulation

As depicted in Fig. 2, in reinforcement learning, the agent explores the environment in a trial-and-error manner, earning rewards through interacting with

the environment. The purpose of training the model is to maximize the reward obtained by the agent. In the initial stage of model training, the agent takes the current node as the input of DQN, and selects the next hop with random probability and gets feedback from the environment, that is, the reward of this hop. The four-tuple information, including state, action, next state and reward, generated by each interaction between the agent and the environment will be saved into the experience pool. When the training episodes reach a certain number, i.e., the *BatchSize*, the model will be updated. The proposed method combines the routing in the LEO constellations with RL, and the state, action, and reward are defined as follows:

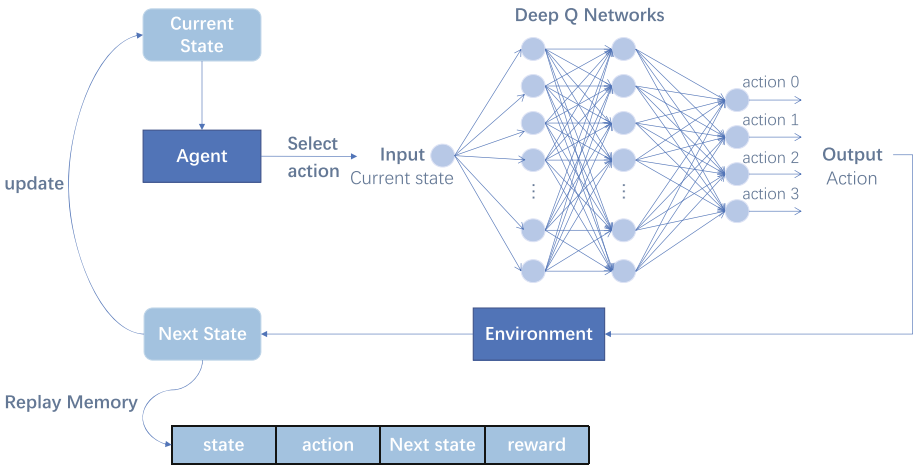

Fig. 2. DQN model of the proposed algorithm.

State: The state of satellite node i at moment t can be expressed as:

$$s_t^i = \{P_{i,j}, Q_{i,j}, D_{i,j}, T_i\}, j = 1, 2, 3 \cdots, \tag{6}$$

where $P_{i,j}$ denotes the propagation delay between nodes i and j, which is proportional to their distance. $Q_{i,j}$ is the queue delay between node i and node j. The queue delay grows as the packet generating rate and volume increase, and the queue delay is usually related to the packet loss rate. $D_{i,j}$ denotes the Euclidean distance between node i and node j. T_i is the identification of the target node, which equals 1 if the current node is the target node and equals 0 otherwise.

Action: The action of the satellite can be expressed as:

$$a \in \{node_{i1}, node_{i2}, \cdots, node_{ij}\}, \tag{7}$$

where a a represents the action taken at the current node, that is, selecting one of the four neighbor nodes, and $node_{ij}$ indicates that the next hop of node i is node j. The agent uses a greedy strategy based on exploration-exploitation for each action selection. At the early stage of training, the agent randomly selects the action with a higher probability and judges the correctness of the selection based on the long-term reward obtained. The agent records the training parameters periodically through replay memory. In the later stage of training, the agent selects the action with a higher probability of obtaining the largest long-term reward. The strategy is denoted as:

$$a = \begin{cases} random\ action\ with\ probability & \epsilon \\ \arg\max_{a \in A} Q^{\pi}(s,a)\ with\ probability & 1-\epsilon \end{cases} \tag{8}$$

Reward: The reward function is:

$$R = k_1(D_{i,target} - D_{i-1,target}) + k_2 H_i + k_3 Rep_i. \tag{9}$$

$D_{i,target}$ is the distance from node i to the target node, $D_{i-1,target}$ is the distance from the previous hop to the target node. If $D_{i,target} - D_{i-1,target}$ is negative, which indicates that the currently selected node is closer to the target node than the previous node, the coefficient k_1 is positive, and the reward value is positive. H_i indicates the number of steps from the source node to the current node. If the number of steps exceeds the given threshold, the exploration is terminated by k_2 to prevent the agent from making multiple repetitions of meaningless exploration in the environment and falling into a dead loop.Besides, Rep_i is the duplicate path flag. We set k_3 to be negative, when the next hop has been marked, $k_3 Rep_i$ will be negative to impose a penalty on this choice.

3.2 Distributed Satellite Routing Algorithm Design

In the network topology shown in Fig. 3, the routing process is as follows:

Step 1: The LEO satellite detects the number of the communicative adjacent nodes by sending hello packets to its four neighbors, and constructs a network topology map based on the spatial location information, queue delay, and propagation delay of the corresponding neighbor nodes.

Step 2: The ground station calculates the visibility with the LEO satellites at the current moment and fills in the visibility table. The source ground station randomly generates packets and requests transmission to the destination ground station at a certain time interval. It then sends the packets to the satellite (source satellite) that covers the ground station and has the shortest distance from the ground station.

Step 3: The network topology map is used as a environment to train the DQN model.

Step 4: The source satellite transmits the packets to the node selected based on the action given by the DQN. The selected node is then regarded as a new source satellite, and the above process repeats until the data is finally transmitted to the destination node.

The workflow of the proposed DQN based routing algorithm is given in Algorithm 1.

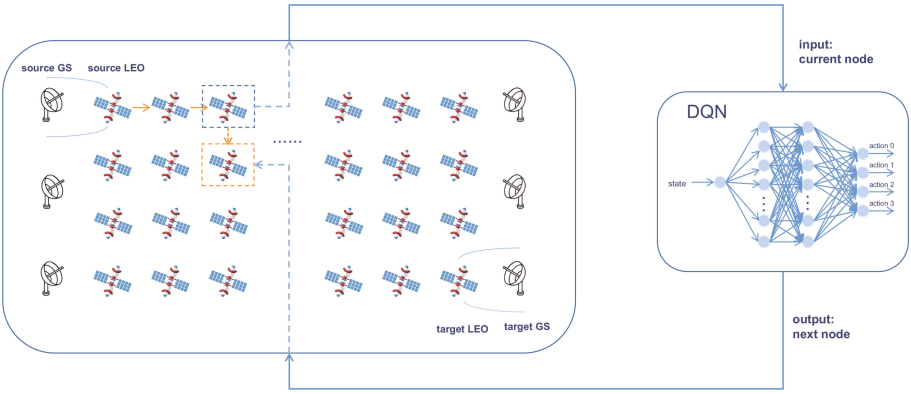

Fig. 3. Illustration of path selection based on DQN.

4 Simulation Results and Analysis

4.1 Parameters

In this paper, the network topology map generated on the OPNET platform is used as the environment for DQN training, and we verify the capability of the trained model in determining inter-satellite routing paths through three metrics, i.e., E2E delay, packet loss rate and throughput. The simulation parameters of the OPNET platform and the DQN model are summarized in Table 1.

4.2 Comparison of Experimental Results

Since this paper only studies the transmission characteristics of inter-satellite routing, the transmission time from the coverage satellite of the source earth station (source satellite node) to the coverage satellite of the target earth station (target satellite node) is referred to as the E2E delay, and the packet loss in this path is calculated by Eq. (4).

Algorithm 1. The proposed DQN-based routing algorithm

Input: State S, action A, discount factor γ, learning rate α, target network update frequency T, source satellite node and target satellite node.

Output: the next-hop node.

Initialize replay memory RM, prediction network $Q(s, a|\theta)$ with random weights θ, target network $Q(s, a|\theta')$ with weights $\theta' = \theta$, eps, $eps\_threshold$, R_s, R_i, $Path$
for each episode **do**
  Initialize the beginning state s_0
  for each decision step t **do**
    Generate a random number $samp$ from 0 to 1
    if $samp \leq eps\_threshold$ **then**
      select a random action a
    else
      $\arg\max_{a \in A} Q^{\pi}(s, a)$
    Save a into path; take action a and make state transition $s \rightarrow s'$; and calculate R
    Store transition(s_t, a_t, r_t, S_{t+1}) to RM
    Randomly extract mine-batch of transitions(s_j, a_j, r_j, S_{j+1}) from RM
    Update target network $Q(s, a|\theta')$
    Set $s_t = s_{t+1}$
    Update the weights of Target Net periodically with $\theta'_i = \theta_i$
  end for
end for
return $Path$

E2E Delay. First, as shown in Fig. 4, we compare the delay performance of the algorithm proposed in this paper with the traditional DSP routing algorithm, the Speaker routing algorithm and the traditional DQN routing algorithm in the case of different traffic volume. When the packet rate is below 125 pkts/sec, the end-to-end delay of the three routing algorithms is lower because the queue's processing rate is relatively high, and the packets in the queue can be sent through the transmitter in time for the next hop. The delay of the DSP routing algorithm and the Speaker routing algorithm gradually increases as the packet sending rate increases above 125 pkt/sec. And the end-to-end delay of the routing algorithm proposed in this paper decreases significantly compared with the former because the path selected by Dijkstra has a large number of duplicate paths locally such that the queuing delay between queues increases. And the proposed algorithm in this paper has randomness in selecting the next hop, which can alleviate the packet queuing wait due to the same path. The traditional DQN algorithm has a higher end-to-end delay because it does not restrict the agent from choosing repeated paths and makes the agent fall into loops. These loops also raise the queue delay of associated links, lowering the overall network's end-to-end latency performance. Through a well-designed reward function, the distributed

Table 1. Simulation Parameters

Hyper-parameter	Value
Packet capacity(pks)	50
Bit capacity(bits)	102400
Queue service rate(Kbps)	200
Packet Size(bits)	1024
γ, lr	0.999, 0.01
ϵ_{start}, ϵ_{end}, ϵ_{decay}	1, 0, 1000
Experience-replay memory capacity	4000
Target network update frequency C	100
Experience-replay batch size	640

DQN routing algorithm suggested in this research prevents the agent from falling into the loop, resulting in a lower delay.

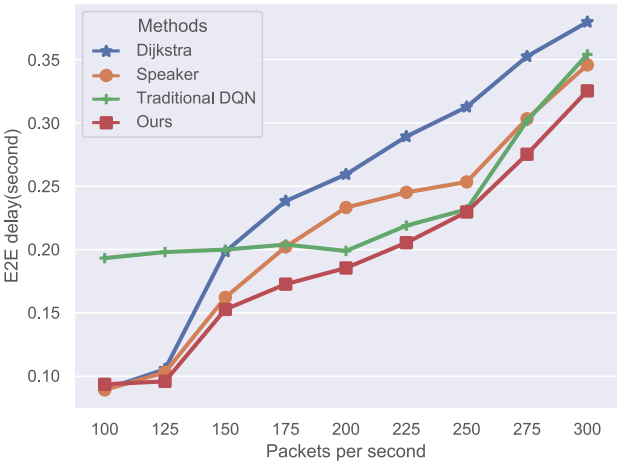

Fig. 4. Comparison of average end-to-end latency between different algorithms.

Packets Loss Rate. As illustrated in Fig. 5, the packet loss rates of the four routing strategies are compared. When the packet rate is low, the queue is less congested, the traditional DQN algorithm cannot reach the target satellite due to the influence of the loops, causing a certain amount of packet loss. The other three algorithms do not lose data packets. As the packet rate increases, the queuing delay of packets increases, and the traditional DSP routing algorithm ignores this dynamic change and still transmits packets in the original path,

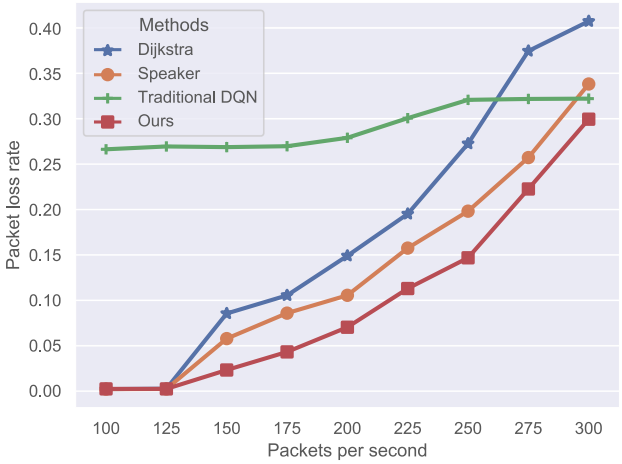

Fig. 5. Comparison of packet loss rates between different algorithms.

resulting in a large number of packet losses. Although the Speaker algorithm periodically updates the queuing delay information in the network, there is a lag, which still causes certain packet losses. However, the algorithm proposed in this paper randomly selects the next hop based on a greedy strategy, which avoids the duplication of local paths and the queuing delay is lower than the other two algorithms. Hence the packet loss rate is generally reduced by 5% - 10%.

Table 2. Comparison of hops between different algorithms.

Algorithms \ Packets/sec	100	125	175	200	250	300
Dijkstra	6.350059	6.350853	6.307295	6.172909	5.840352	5.539334
Speaker	6.350059	6.351141	6.235041	6.238398	6.191605	6.215915
traditional DQN	6.305049	6.268570	6.318532	6.217788	6.261644	6.260731
Ours	6.681810	6.680046	6.704537	6.555520	6.448777	6.496003

Throughput. Figure 6 shows a comparison of the throughput of the four routing strategies. The throughput of the traditional DQN algorithm is the lowest due to packet loss caused by the loops. The traditional DSP algorithm's throughput grows slowly as the number of packets sent increases. In contrast, the Speaker routing algorithm's throughput fluctuates more, owing to the Speaker mechanism's need to update the network topology on a regular basis by sending information packets, which add to the network's load. The algorithm suggested in this paper's throughput grows quicker than the previous algorithm' throughput.

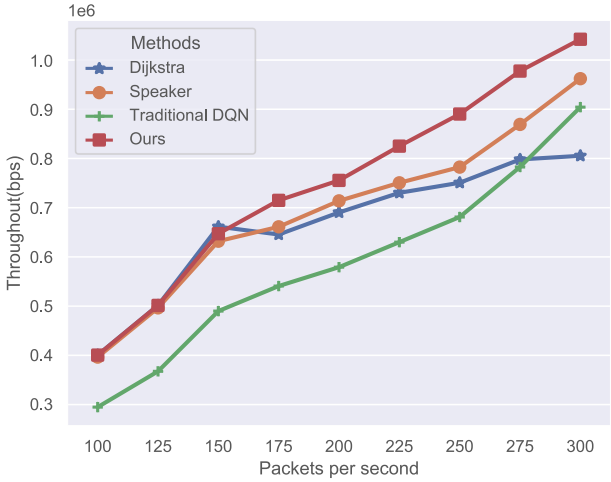

Fig. 6. Comparison of network throughput between different algorithms.

Compared with other algorithms, the algorithm proposed in this paper significantly improves the routing performance in delay, packet loss rate, and throughput. However, it is accompanied by a slight increase in the number of hops, as shown in Table 2. This is because the algorithm proposed in this paper does not choose a fixed shortest path when selecting a path but randomly selects the next hop with a certain probability, so it does not make the best decision regarding the number of hops.

5 Conclusion

In order to alleviate the problem of congestion in inter-satellite links caused by the growth of user traffic and uneven distribution, this paper proposes a distributed routing algorithm based on deep reinforcement learning. The randomness of selecting the next hop avoids local path overlap in different routing paths and effectively relieves the congestion caused by the fixed shortest path. The DQN mdoel is trained with a reward function that constrains the occur of path loops. Then, the optimal path is selected in the LEO satellite networks based on the queue delay and propagation delay between nodes. The simulation results of OPNET in this scenario show that the algorithm can effectively alleviate congestion and has lower packet loss rate, lower latency and higher throughput compared with existing algorithms, improving network throughput while ensuring low latency.

References

1. Liu, C., Xu, M., Geng, N., Zhang, X.: A survey on machine learning based routing algorithms. J. Comput. Res. Develop. **57**(4), 671–687 (2020)

2. Wang, F., Jiang, D., Qi, S.: An adaptive routing algorithm for integrated information networks. China Commun. **16**(7), 195–206 (2019)
3. Zhou, W., Zhu, Y.F., Li, Y.Y., Li, Q., Yu, Q.Z.: Research on hierarchical architecture and routing of satellite constellation with IGSO-GEO-MEO network. Int. J. Satellite Commun. Netw. **38**(2), 162–176 (2020)
4. 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. Networking **17**(1), 281–293 (2008)
5. Bai, J., Lu, X., Lu, Z., Peng, W.: A distributed hierarchical routing protocol for non-GEO satellite networks. In: Workshops on Mobile and Wireless Networking/High Performance Scientific, Engineering Computing/Network Design and Architecture/Optical Networks Control and Management/Ad Hoc and Sensor Networks/Compil, pp. 148–154. IEEE (2004)
6. Song, G., Chao, M., Yang, B., Zheng, Y.: TLR: a traffic-light-based intelligent routing strategy for NGEO satellite IP networks. IEEE Trans. Wireless Commun. **13**(6), 3380–3393 (2014)
7. Mao, B., et al.: A tensor based deep learning technique for intelligent packet routing. In: GLOBECOM 2017–2017 IEEE Global Communications Conference, pp. 1–6. IEEE (2017)
8. Zuo, P., Wang, C., Yao, Z., Hou, S., Jiang, H.: An intelligent routing algorithm for LEO satellites based on deep reinforcement learning. In: 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), pp. 1–5. IEEE (2021)