Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5124960, 12 pages
https://doi.org/10.1155/2022/5124960

Research Article

RLMR: Reinforcement Learning Based Multipath Routing for SDN

Chao Chen,[1] **Feifan Xue**,[1] **Zhengyong Lu**,[1] **Zhongyun Tang**,[1,2] **and Chuanhuang Li**[1]

[1]*School of Information and Electronic Engineering (Sussex Artificial Intelligence Institute), Zhejiang Gongshang University, Hangzhou, Zhejiang 310000, China*
[2]*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, Zhejiang 310000, China*

Correspondence should be addressed to Chuanhuang Li; chuanhuang_li@zjgsu.edu.cn

Received 17 September 2021; Revised 17 November 2021; Accepted 7 January 2022; Published 18 February 2022

Academic Editor: Xu Zheng

In recent years, as a new subject in the computer field, artificial intelligence has developed rapidly, especially in reinforcement learning (RL) and deep reinforcement learning. Combined with the characteristics of Software Defined Network (SDN) for centralized control and scheduling, resource scheduling based on artificial intelligence becomes possible. However, the current SDN routing algorithm has the problem of low link utilization and is unable to update and adjust according to the real-time network status. This paper aims to address these problems by proposing a reinforcement learning-based multipath routing for SDN (RLMR) scheme. RLMR uses Markov Decision Process (MDP) and Q-Learning for training. Based on the real-time information of network state and flow characteristics, RLMR performs routing for different flows. When there is no link that meets the bandwidth requirements, the remaining flows are redistributed according to the Quality of Service (QoS) priority to complete the multipath routing. In addition, this paper defines the forward efficiency (FE) to measure the link bandwidth utilization (LBU) under multipath routing. Simulation results show that compared with the current mainstream shortest path algorithm and ECMP algorithm, the routing algorithm in RLMR has advantages in FE, jitter, and packet loss rate. It can effectively improve the efficiency and quality of routing.

1. Introduction

As the rapid development of the Internet, people are increasingly dependent on the network, and the amount of data in the network is also exploding. However, the current routing strategy based on Dijkstra's Shortest Path (DjSP) is so fixed that the routing path has a high repetition rate, while the other slightly longer links cannot be fully utilized. To reduce the wasted resources, multipath routing is gradually becoming a hot spot for solving this problem.

In traditional network, existing multipath routing such as ECMP [1] and WCMP is less efficient in matching the current data flow requirements for QoS [2, 3] and collecting real-time operating status of the network. Google's B4 network [4] is currently the most famous and influential new generation network based on SDN. Its flow engineering system is the idea of multipath routing for each different flow. The maximum-minimum fair bandwidth allocation algorithm is provided to greatly improve the system LBU.

In recent years, Artificial Intelligence (AI) has developed rapidly as a new subject in computer science, especially in the field of RL [5] and Deep Reinforcement Learning (DRL) [6]. As a new type of network architecture, SDN [7] is widely concerned by academia and the business filed because of its separation of forwarding and control. AI's characteristics of high-efficiency and real-time are suitable for solving complicated network problems.

This paper based on the research of current SDN flow scheduling technology, combined with the advantages of RL in policy optimization and the characteristics of centralized control of SDN network resources. RLMR is proposed to perform multipath routing of network flow based on the current network status information and flow characteristics. Simulation shows that this scheme can find multiple

forwarding paths that match different flows characteristics and improve the LBU. At the same time, compared with the existing flow scheduling algorithm, the flow scheduling scheme proposed in this paper can effectively reduce the network delay and reduce the network packet loss rate.

The rest of this paper is structured as follows. Section 2 describes the related work. Section 3 presents the architecture of RLMR. Section 4 shows the simulation results. Conclusion and future works are shown in Section 5.

2. Related Work

SDN breaks the limitations of traditional networks. Its core idea is to separate the forward layer and control layer. By centralized management of network resources and real-time monitoring of network status, the controller provides necessary information for flow scheduling decisions [8]. Taking advantage of SDN and planning an efficient path for data flow by programming flow scheduling strategies are still a hot topic in current network research. Zhu et al. [9] proposed an SVC optimization scheme with the help of SDN. This scheme uses the great power of the cloud to collect the status information of the whole network to improve SVC flows. However, most of the current SDN controllers use the shortest path algorithm and focus on the shortcomings of the shortest path algorithm in SDN. Many works have been done to optimize routing algorithm. Lin et al. [10] propose a scheme to optimize the traditional shortest path algorithm by improving the data storage structure. Yan et al. [11] proposed an SDN-based multipath QoS system HiQoS, which provides guaranteed service for different flows through the queue scheduling on the SDN switch. However, this scheme is still based on the idea of the shortest path algorithm, so that when the network topology is busy, this scheme is getting less efficient.

Zhu et al. [12] made a comparison based on whether it is dynamic and whether it is multipath. The results show that dynamic routing is better than static routing, and multipath is better than single path. Those results point out two optimization directions. For multipath, the remaining link bandwidth can be used to improve resource utilization. Combined with a dynamic routing decision, further reasonable routing can be achieved.

The uneven distribution of network flows increases the possibility of network link congestion. To obtain better network performance and make full use of link bandwidth, a series of work studies the multipath routing algorithm in SDN, which can reasonably distribute flow to multipath. Yang et al. [13] proposed a dynamic load balancing routing algorithm Deamp based on the multipath transmission that redefines the key link and the key degree of the link and optimizes link weight. [14] proposed a load balancing multipath routing algorithm based on SDN, which obtains the load information of the global link through the controller and calculates all forwardable paths between the source and destination nodes, and then selects the path with the smallest link load and sends the flow table to switch. This scheme can give the routing decision which has optimal link load for each flow entering the network. However, in the real network environment, the network state is so complex that this scheme requires too much calculation.

With the development of machine learning algorithms, various schemes dedicated to solving routing problems in a computer network through machine learning algorithms have emerged [15–18]. Cheng [19] proposed a routing algorithm based on RL, which can effectively select the optimal path. But all flows are treated uniformly in routing, without distinguishing. Rischke et al. [20] proposed QR-SDN, which is a reinforcement learning-based scheme. QR-SDN uses an entire route path as action and all available paths as action space. In this way, flows can be maintained maximum integrity. The disadvantage is that any change of node will cause the change of entire action space. Besides, when nodes increase, the action space will get too large to converge. Another machine learning solution is to take the next hop as the action. Guo et al. [21] focused on the SDN-IoT field, took black hole attacks and DDoS attacks into account, proposed the QoS-Aware secure routing based deep reinforcement learning. The proposed scheme takes next-hop as an action and updates the model by maximizing the cumulative reward.

At the same time, as network services require higher quality, routing strategies that satisfy QoS constraints have always been a hotspot in academic and industrial research. A lot of research related to the QoS level has appeared in response to the demands of different flows with different service quality. Bueno et al. [22] proposed a QoS policy configuration framework that can provide end-to-end network services according to the specific needs of online interactive applications. Tomovic et al. [23] proposed an SDN-based QoS-aware algorithm, which uses different routing strategies for QoS flow and non-QoS flow to meet the requirements of QoS flow.

In summary, the existing SDN routing schemes still have the problems of low efficiency and high computational complexity. In response to the above problems, this paper proposes RLMR to optimize flow forwarding. Considering that different services have different requirements for QoS, the algorithm provides different routing priorities for different services. When the link bandwidth is not enough, RLMR divide a large flow into multiple small flows, thereby improving the LBU.

3. Schemes

3.1. System Framework. In a traditional routing strategy based on the shortest path algorithm, multiple flows occupy the same link so that congestions may occur. That is because the shortest path from the source node to the destination node in the network is too fixed. To avoid such situations, the routing scheme studied in this paper is based on the current network available bandwidth and the current characteristics of flow to be forwarded (including the flow size, QoS level, etc.). By setting different reward functions for flow with different QoS levels, multiple paths are planned to forward the flow to improve the LBU. The specific structure is shown in Figure 1.

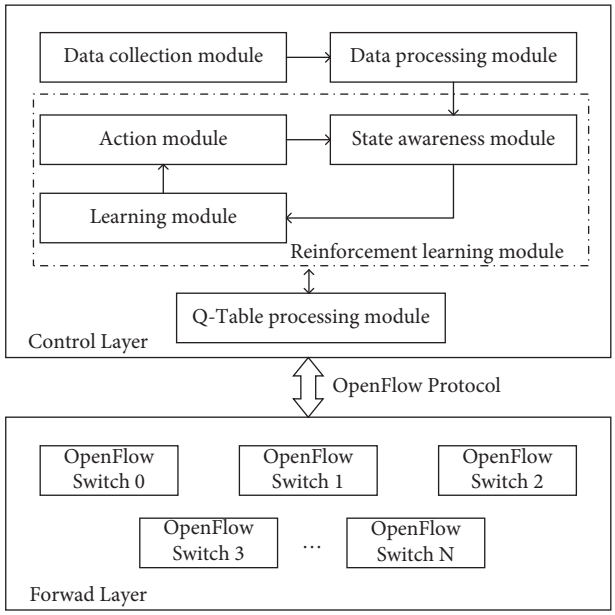

Figure 1: RLMR architecture.

As shown in Figure 1, there are 6 modules: Data collection module, Data processing module, State awareness module, Learning module, Action module, and Q-Table processing module. The function of six modules is listed as follows:

(i) Data collection module: the data collection module takes advantage of centralized control of SDN to obtain global network information. This module sends the obtained network information and flow information to the data processing module.

(ii) Data processing module: the processing result of the data processing module is the input of the whole RL module. This module processes the information obtained from the data collection module, extracts the available features, and sends them into the RL module. The results are the network topology matrix and flow feature matrix. The network topology matrix contains link available bandwidth information and total bandwidth information. The flow feature matrix contains the flow source address, a destination address, QoS level, and flow bandwidth demand.

(iii) State awareness module: the state awareness module has the following two functions. First, it maps the environment state to agent's internal perception and updates the mapping result when the current environment state changes after the agent selects an action. The input obtained from the data processing module can be regarded as the initial state of the environment. Second, at the end of the training of RL module, the current flow can be judged from two matrices to determine whether this chosen link's bandwidth meets the demand of the flow. If the resulting path does not meet the demand, the state awareness module continues to plan the path; otherwise, the results will be sent to the Q-table processing module.

(iv) Learning module: the learning module is the core of the RL module. It can update the strategy according to the reward value of the current environment state and the internal perception acquired from the State awareness module, and then, it will select actions that are conducive to the accumulation of total rewards. The selected action is sent to the Action module.

(v) Action module: the action module executes received actions and then updates the current environment state.

(vi) Q-table processing module: With the information of Q-table, this module generates flow tables and post flow tables to the OpenFlow switch in the Forward layer. After the Forwarding layer receives the flow table, flows will forward according to the policy.

QoS is mainly reflected in delay. A higher QoS level service has higher delay sensitivity, and a lower QoS level service has lower delay sensitivity. The delay in this paper mainly considers the number of routers that the flow passes through. A path with fewer router hops is preferentially assigned to a flow with a higher QoS level.

In this paper, the multipath scheme used to improve LBU is as follows: The Data collection module collects a flow that needs to route, and the Data processing module extracts the characteristics and network topology information. Then this module obtains network topology information matrix T and flow = $\{s, d, \beta, B_{\min}\}$, where s stands for the source node, d stands for destination node, β stands for QoS level, and Bmin stands for flow size. The Data processing module input the result to the RL Part for routing decision to obtain the optimal path R. If the result link bandwidth is bigger than the minimum demand for this flow, the flow decision is completed. Otherwise, the flow will be offloaded. When the flow is offloaded, the available bandwidth of the link and the QoS level of the flow are comprehensively considered. For a flow with a large QoS level, more link bandwidth is allocated on the path, and the RL module is continued to obtain a suboptimal path, repeat the above process until the flow is allocated or there is no path forwarding. Finally, the output of the RL module is input to the Q-table processing module, and the flow table is sent to the switch to complete the routing decision of this flow.

3.2. System Modelling. RLMR system uses MDP for modelling. The MDP quadruple proposed in this paper is defined as follows.

(i) *State Set.* In the network topology, this paper defines state i as S_i as shown in (1). Here, p represents the switch where the agent is currently at. x represents how many hops agent passes. β and η represents current flow's QoS level and current LBU. N represents the total number of switches in the topology.

$$S_i = [p, x, \eta, \beta] p \in [1, 2, \ldots, N], \quad x \in N^*, \eta \in (0, 100), \beta \in [1, 2, \ldots, 100]. \tag{1}$$

(ii) Action Set. Agent can only be transmitted at the connected network node. Therefore, this article defines the network connection status as shown in the following equation:

$$T[p_l][p_m] = \begin{cases} 0, & p_l, p_m \text{ disconnected} \\ 1, & p_l, p_m \text{ connected} \end{cases} \quad (p_l, p_m \in 1, 2, \ldots, N). \tag{2}$$

Since the agent can only be transmitted between connected network nodes, this paper defines action a_i from the action set $A(S_i)$ as shown in the following equation:

$$a_i \in A(S_i) = \{p_m | T[p_l][p_m] = 1\} \quad (p_l, p_m \in 1, 2, \ldots, N). \tag{3}$$

(iii) State Transition. In each training round, the agent is in the state S_i after selecting an action in the action set. If the action is not the state that has been selected for the round, the agent will move to the next state.

(iv) *Reward Function.* In RL, the agent must find the optimal action that maximizes the overall reward. The reward represents the success of agent's action decision, and the reward function tells the agent about which actions will be selected to accumulate the maximum overall reward. The action of the agent not only affects the direct ward but also affects the subsequent reward. The reward function will be given in the following part.

The quality update function of RL proposed in this paper is as (4), in which learning rate α and discount rate γ will be determined in the next section.

$$Q(S_i, a_i) \leftarrow Q(S_i, a_i) + \alpha \left[R(S_i, a_i) + \gamma \max_{a_{i+1}} Q(s_{i+1}, a_{i+1}) - Q(S_i, a_i) \right]. \tag{4}$$

In reinforcement learning, the choice of action strategy and the design of reward function are related to the implementation effect of the whole algorithm. These two parts will be introduced in detail in the next two sections.

3.3. Action Strategy. In the RL, the task of the agent is to keep trying in the system and find a strategy. The quality of strategy depends on the cumulative rewards obtained after the execution of this strategy. Our best strategy is to choose the action with the largest Q in each state. In fact, due to a limited number of attempts, exploration and utilization are contradictory, and the problem to be solved by the strategy is to achieve a better balance between exploration and utilization. Exploration refers to the actions that the agent has not performed before. Utilization refers to the agent obtaining the current optimal action from the previously learned experience. In this paper, exploration means to select links that have not been selected before to find more possibilities. Utilization means selecting links that have been selected to improve the known route planning route.

Currently, the four action strategies widely used are the random strategy, the greedy strategy, the ε-greedy strategy, and the softmax strategy. $\pi(a|s)$ represents action strategy, and it means the probability that the agent chooses action a in state s. According to this strategy, the action to be selected in the state s ($a = \pi(s)$) can be known.

The random strategy refers that the decision-maker selects possible actions with equal probability at each step state s. When the network topology is large, using this strategy may not find the destination node and cannot send the packet to the destination node, so this strategy is not desirable.

The greedy strategy is a definite strategy. That is, the decision-maker chooses the action a with the largest quality value Q at each step state s. It only uses the quality value which is already known by the decision-maker and never explores other states. But in that way, a bigger quality value may exist. In routing, if an agent does not explore other unknown links, the already known link may not be the optimal path or even cause link congestion, so this strategy is not desirable.

The softmax strategy is a strategy based on probability. It calculates the probability of selecting an action based on the quality of the optional action *Q(s, a)* in the current state *s*. The basic idea is that the larger the *Q(s, a)* is, the greater the probability of selecting the action will be. It maps agent's optional action set to a probability distribution, and then the agent selects actions according to this probability distribution. The softmax strategy's formula is as follows:

$$\pi(a|s) = \frac{\exp(Q(s, a)/\tau)}{\sum_{i=1}^{n} \exp(Q(s, i)/\tau)}. \tag{5}$$

Here, n represents the size of the optional action set of the agent in the state *s*. *Q(s, a)* represents the quality function of selecting the action a in the state s, and τ is called the temperature parameter, which can change the weight of

exploration and utilization. In the process of planning the path, when τ tends to infinity and the agent chooses to forward the action, the probability of all actions in the action set would tend to be equal; that is, the agent would tend to only explore; when τ tends to 0, the agent would choose to forward action. The larger the *Q(s, a) is*, the greater the probability of being selected will be, and the agent will tend to only use it. Because the value range of τ is too large, the value of τ cannot be determined well, so this strategy is also unacceptable.

The ε-greedy strategy compromises exploration and utilization based on the probability ε. Specifically, the agent adopts a random strategy with the probability of ε at each step state s. When the route is planned, the agent has a certain probability to explore a new path, and a greedy strategy with a probability of $1-\varepsilon$ will be token, which means that the agent will select the forwarding action corresponding to the largest quality value. The ε-greedy strategy's formula is as follows:

$$\pi(a \mid s) = \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|A(s)|}, & \text{if } a = \arg\max_a Q(s,a), \\ \dfrac{\varepsilon}{|A(s)|}, & \text{if } a \neq \arg\max_a Q(s,a). \end{cases} \tag{6}$$

The ε-greedy strategy can balance the weight between choosing a known action and choosing an unknown action. Therefore, in this paper, we choose the ε-greedy strategy. In the simulation part, the value of ε will be determined.

3.4. Reward Function. A key issue of RL is the generation of reward values and the weight update function. While the state transition of agent occurs, the system will feedback a reward to the agent according to the reward function R.

The goal of RLMR is to plan reasonable multipath through training, so the setting of the reward value R is also crucial. This paper mainly considers LBU and QoS. The design of the reward function should meet the following:

(i) Consider both QoS level β and LBU η

(ii) Encourage paths with large β to allocate paths with few hops

In summary, reward function (6) designed in this paper is as follows:

$$R(S_i, a_i) = -\beta g(x) + \eta + \delta(a_i). \tag{7}$$

In (7), $R(S_i, a_i)$ represents the reward received by the agent when selecting action a_i from action set. It represents the reward generated when the agent in state S_i selects the next hop a_i. $\delta(a_i)$ is defined as impulse function. When the destination node is the chosen action, the function's value is 100. Otherwise, the function's value is 100.

$$g(x) = \frac{e^{4x/l_m} - e^{-4x/l_m}}{e^{4x/l_m} + e^{-4x/l_m}}. \tag{8}$$

As shown in (7) and (8), $g(x)$ is cost function. The cost function stands for the cost increases with the rising number of hops that agent passed x. When x is getting bigger the cost increases, and $g(x) \in (0, 1)$. l_m represents total number of nodes in the network topology. Considering that when the network topology is getting larger, the more agent passes and the higher the delay is, so the agent can only be encouraged to forward through part of the link, so the cost function should meet, and rapid growth in the early stage and gradually stabilized, if the number of hops passed by agent reaches l_m, then the cost function value will reach the maximum. The trend of function $g(x)$ is in Figure 2.

The second requirement is to encourage flow with large-β to allocate paths with few hops. Therefore, the cost function $g(x)$ in the reward function is multiplied by the flow QoS level -β. In this way, under the same circumstances, the more path hops agent forwards, the greater the cost of the flow with a large QoS level will be, then the routing strategy will choose paths with few hops for a large QoS level flow.

Based on the above definition, the model of RLMR in routing decisions is as follows: Each node in the network maintains a Q-Table as the strategy of the next hop is chosen. For each Q-Table, the columns represent the action set, which is the neighbor nodes, and the rows represent the destination node. Figure 3 is Q-table of node S1, in which Q (S2, S3) represents that For traffic that destination node is S2, then agent takes an action from columns S2 in Q-Table S1 by ε-greedy action strategy. According to the results of this decision, the agent gets a reward by reward function (7), and update Q-Table by quality update function (4).

3.5. Multipath Routing Algorithm. According to the principles of RL and the environment of SDN routing, the multipath routing algorithm in RLMR is shown in Algorithm 1.

Pseudocode shows that the algorithm performs multipath routing for flow to be distributed. The T in the algorithm is a two-dimensional matrix that represents the current link state. If the switch is connected, its value will be set to 1. Otherwise, its value will be set to 0. When performing multipath division, it is necessary to cut off the congested link, as shown in the algorithm $T[s_i][s_j] = 0$. For the agent, the worst case is that N nodes in the network are neighbor nodes (except the destination node). In this case, the time complexity and space complexity of this algorithm are $O(n^2)$.

4. Results and Discussion

4.1. Simulation Environment. Before the simulation, we need to prepare some necessary environment. Mininet is a network simulator connected by some virtual nodes, routers, and switches. It adopts lightweight virtualization technology and supports OpenFlow protocol. Ryu is an open-source SDN controller implemented by Python. It supports all OpenFlow protocols. Users can write their own applications in Python.

The simulation system environment is Ubuntu 14.04, using Ryu controller and Mininet to test the effectiveness of

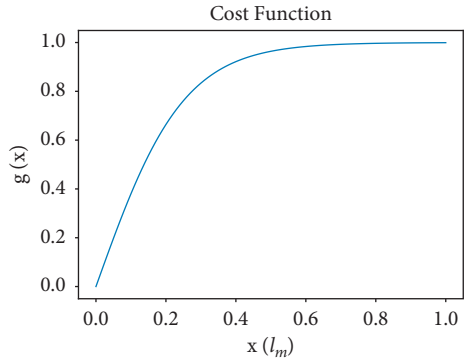

Figure 2: The trend of function $g(x)$.

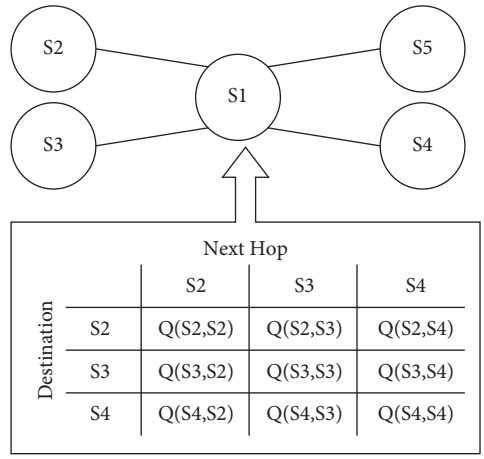

Figure 3: Q-table of node S1.

(1) $s \longleftarrow$ Source, $d \longleftarrow$ Destination, $\beta \longleftarrow$ QoS level, $B \longleftarrow$ The minimum available bandwidth of the current path, $B_{\min} \longleftarrow$ Flow size, $Q(s, a) \longleftarrow 0$, $\pi(s) \longleftarrow$ Action strategy, $R \longleftarrow$ Reward function, Routing $(S, D) \longleftarrow$ Routing set, *iterations* $\longleftarrow$ Total training times, *steps* $\longleftarrow$ a single training step
(2) for i in range (*iterations*)
(3) while $B_{\min} \neq 0$
(4) for j in range (steps)
(5) $a \longleftarrow \pi(s)$ choose action
(6) $s' = a$
(7) $r \longleftarrow R$ (calculate reward value)
(8) use quality update function $Q(s, a)$
(9) $s = s'$
(10) if $s == d$
(11) Get Routing from $Q(s, a)$
(12) Get two nodes with the smallest available bandwidth s_i, s_j
(13) Routing $(S, D) \longleftarrow$ Routing
(14) Break
(15) end if
(16) end for
(17) if $B_{\min} > B$
(18) $T[s_i][s_j] = 0$
(19) update T
(20) $B_{\min} = B_{\min} - B\beta / \sum \beta$
(21) else

Algorithm 1: Continued.

```
(22)            B_min = 0
(23)          if B = = 0
(24)            break
(25)          end if
(26) end for
(27) return Routing (S, D)
```

Algorithm 1: Multipath routing algorithm in RLMR.

the routing scheme. Topology built by Mininet is shown in Figure 4, which includes 9 OpenFlow switches and 5 hosts.

4.2. Model Parameter. In RLMR, the convergence rate of model results is affected by the following parameters:

(i) Learning rate α: α is between (0, 1). If α is too big, it may not converge to the optimal solution, and the model is prone to oscillation. On the contrary, if α is too small, the convergence speed will be slow; the model is easy to fall into local optimality and overfitting. This paper will determine the value of the learning rate in the simulation.

(ii) Discount rate γ: γ is between (0, 1), γ determines which agent takes between the current reward and the long-term reward. If γ is 0, the model will not learn any future reward information but only focus on the reward of the current state. If γ is greater than or equal to 1, the reward will not converge because it keeps accumulating and does not decay. This paper will determine the value of the discount rate in the simulation.

(iii) Steps of exploration in each round: This paper defines that the maximum number of exploration steps in each round is 20. When the exploration steps exceed 20, the agent still fails to reach the destination node, which means that the current training strategy is inappropriate and cannot reach the destination node.

(iv) Greedy strategy ε: In this paper, the agent need to choose known links and explore unknown links, so it is important to select an appropriate ε. The initial value of the exploration factor ε was set as 0.1, which will be determined in the simulation.

According to Section 3.3, this paper uses ε-greedy strategy (ε was taken as 0.1 temporarily, and then determined by simulation). We compare the convergence effects to find optimal α and γ.

In Figure 5, the y-axis represents the number of hops that the agent passes to reach the destination node in the network. It shows that the value of the y-axis is larger at the beginning because the model needs to explore links that have not performed actions at the beginning of training. As the number of training increases, the actions that have been selected become more and more so that it gradually converges. When the y-axis converges, it means that after the model is trained, the agent can pass through a specific node to reach the destination node. The smaller the y-axis value, the fewer hops agent pass through when it reaches the destination node.

It can be seen from Figure 5 that the learning rate α has a greater impact on the model convergence than the discount rate γ, and as the model learning rate α increases, the model can converge faster. When the discount rate $\gamma = 0.7$, the model converges fastest, and the y-axis value is small under the same conditions. Therefore, the learning rate is set to $\alpha \in (0.5, 0.9)$ and the discount rate is set to $\gamma \in (0.5, 0.9)$.

To obtain the accurate values of the learning rate α and the discount rate γ, the required learning rate and discount rate of the model are determined by comparing the model errors at different α and γ. The simulation results are shown in Figure 6.

It can be seen from Figure 6 that the model's error gradually decreases with the increase of training episodes, and the model error can gradually approach zero after 1000 episodes. When α is the same, the bigger the γ, the bigger the model error. Comparing different learning rate α and discount rate γ models, it can be observed that when $\alpha = 0.8$ and $\gamma = 0.6$, the model performs better in the simulation and can quickly reduce the simulation error to the minimum.

According to the performance of the different learning rates and discount rate in simulation, this paper sets the learning rate α of the quality update function in the RL to 0.8 and the discount rate γ to 0.6.

To make a better compromise between exploration and utilization, the action strategy adopted in this paper is the ε-greedy strategy. The agent adopts a random strategy with a probability ε. The bigger the ε, the greater the probability that the strategy will be used to select the action, the worse the model converges. When ε is too large, the model may not converge, and the final path planned by the model may not be the optimal path. Therefore, in the action strategy of this paper, the probability of utilization should be greater than that of exploration, that is, the value of e-greedy strategy $\varepsilon \in (0, 0.5)$. This paper determines ε by comparing the accuracy of different ε models. The simulation results are shown in Figure 7.

It can be seen from Figure 7 that under the same conditions, as the number of iterations increases, the model accuracy rate shows an upward trend. And when the iterations are large enough, the model accuracy rate reaches the maximum when $\varepsilon = 0.1$. The value of the ε-greedy strategy ε is set to 0.1.

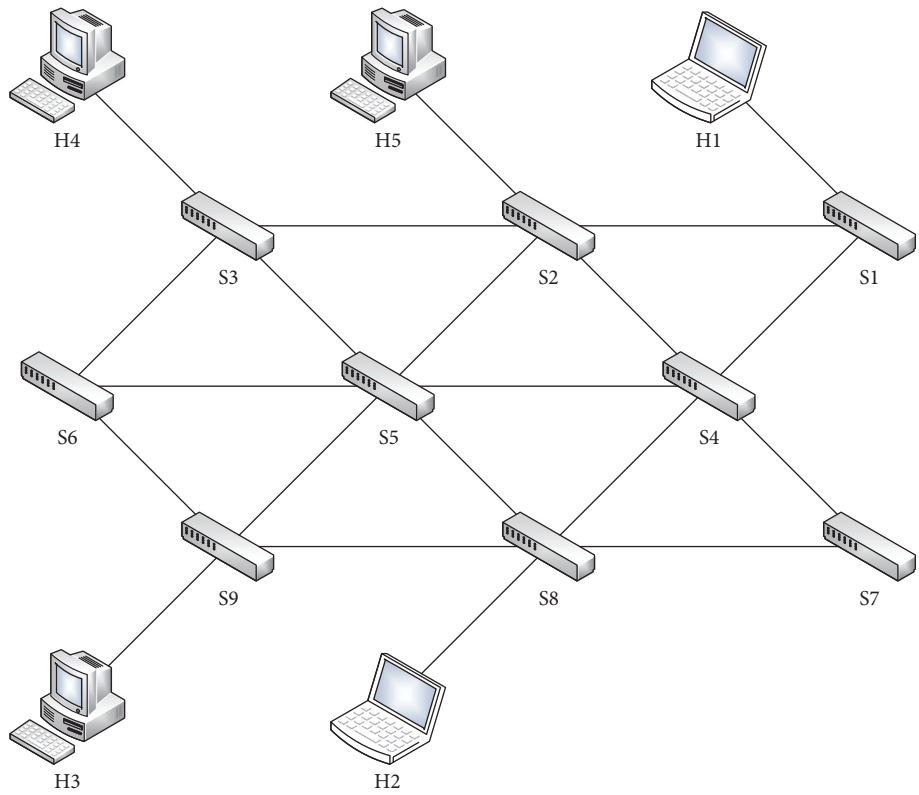

Figure 4: SDN network topology.

4.3. LBU Verification Simulation. DjSP is a typical shortest path algorithm. The advantage of this algorithm is that it can calculate the shortest path from any node to all other nodes. However, the disadvantage of this algorithm is also obvious. If network packet forwarding only depends on the shortest path, it will easily lead to link congestion.

The purpose of this algorithm is to improve the LBU. To verify the feasibility of this algorithm and compare it with DjSP, this paper defines FE as e to reflect the quality of LBU. The higher the value of e, the higher the value of LBU, the lower the value of e, the lower the value of LBU. FE's specific formula is as follows:

$$e = \frac{\sum_i B^i_{\text{send}}}{\sum_i B^i}. \tag{9}$$

In (9), B^i represents the bandwidth requirement of the ith flow, B^i_{send} represents the size of the flow that is successfully forwarded by the ith flow, B^i_{send} is definite as follows:

$$B^i_{\text{send}} = \min_i \left(B^i, \min_B R^i_B \right). \tag{10}$$

In (10), R^i_B represents the bandwidth of all links that are selected by the ith flow forwarding, $\min_B R^i_B$ is the smallest link bandwidth in this route.

To compare with RLMR, the shortest path algorithm DjSP is used in the simulation. In addition, considering that different congestion levels of the current network link bandwidth will affect the evaluation of the algorithm, this paper will also compare FE under different congestion conditions.

According to the RLMR and SDN network topology, the network link bandwidth is set to 200. The sending end is set to h1~h5, and the receiving end is set to h1~h5. Five sending ends randomly send data to other receiving ends with a probability of 20%. All hosts totally send 30 static flows. Static flow refers to the flow that, once injected into the network, it will occupy the link bandwidth until the end of the simulation.

According to the scheme in this paper, the algorithm will distribute flow based on the current network bandwidth status. If the available bandwidth of the currently planned path is greater than the size of the flow to be distributed, the flow is forwarded. Otherwise, the flow is divided into multiple small flows, the small bandwidth is reasonably used, and the path is planned until all the flows are sent. Flow FE is shown in Figure 8 and Figure 9.

In Figures 8 and 9, the x-axis represents network topology link bandwidth congestion. The y-axis represents flow FE. The triangle in the figure represents the average flow FE for the 100 rounds of simulation. Under the same circumstances, the greater the FE, the greater the network LBU. Combined with Figure 10, it is shown that under the same conditions, the flow FE of RLMR is greater than that of DjSP. It shows that the algorithm can divide a large flow into multiple small flows, reasonably use links with small bandwidth, and find different forwarding paths for small flows, thereby improving network LBU.

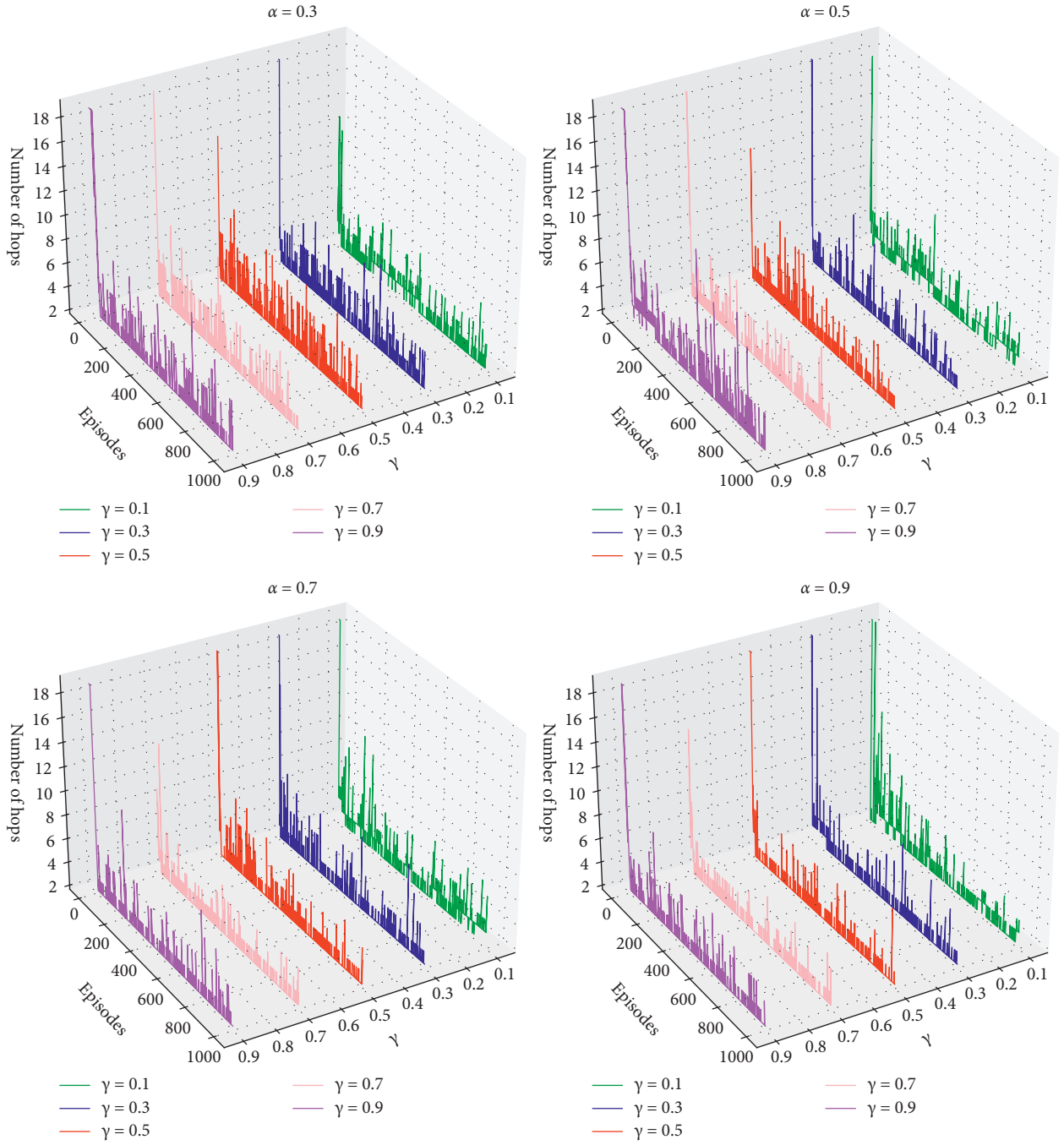

Figure 5: Convergence contrast diagram of different α and γ.

4.4. QoS Verification Simulation. ECMP algorithm can evenly distribute traffic over multiple equivalent links. Compared with DjSP, this algorithm can greatly improve network throughput and LBU. The disadvantage of this algorithm has a bad LBU when the difference between network paths is large. To compare the three schemes for flow's QoS guarantee, this paper expects to use packet loss rate and jitter as descriptors to measure QoS where jitter represents the difference in delay between packets.

The simulation topology is shown in Figure 4. Each link bandwidth is set to 100 Mb/s, and the delay is 2 ms. Data source is h3, and data destination is h1. Flow size increases from 10 Mb/s to 60 Mb/s at intervals of 5 Mb/s.

The result of packet loss rate is shown in Figure 11. When the data source sending rate is lower than 30 Mb/s, no packet loss in all three policies. Dijkstra algorithm begins to lose when the sending rate is greater than 30 Mb/s, and the ECMP algorithm begins to lose when the rate is greater than 45 Mb/s. The algorithm in this paper has no packet loss. Additional simulation proof that this algorithm had a slight packet loss until the packet sending rate was 80 Mb/s.

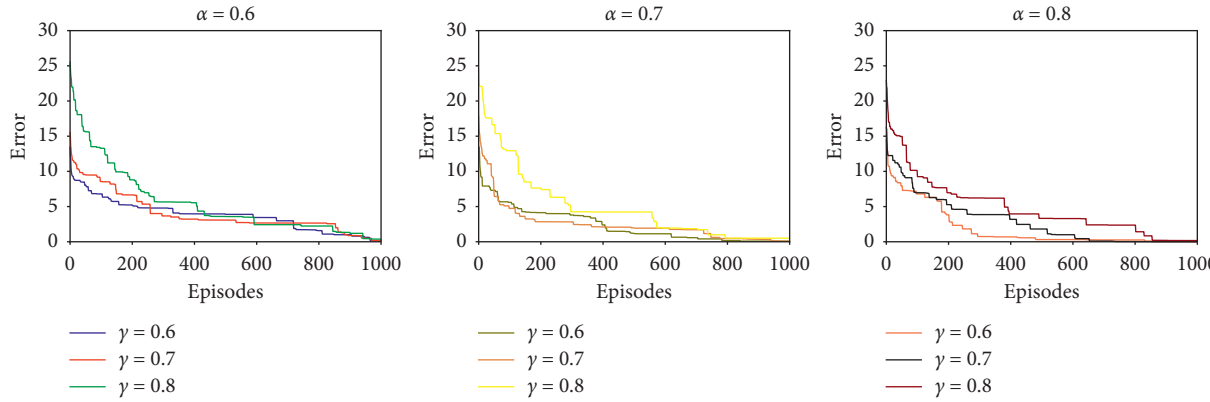

Figure 6: Error comparison of different α and γ.

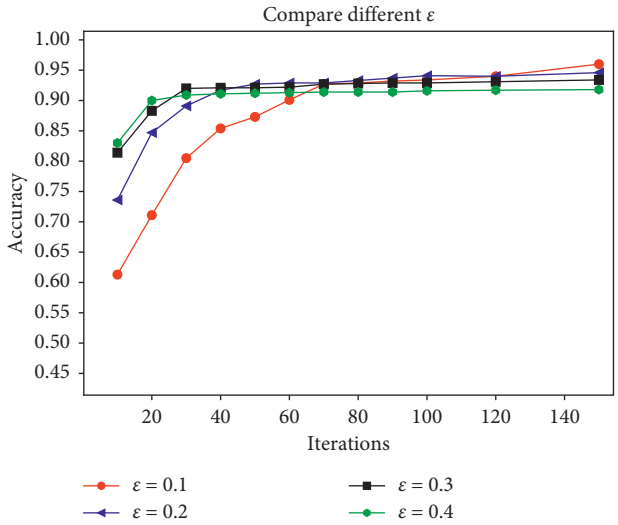

Figure 7: Comparison of different ε accuracy rates.

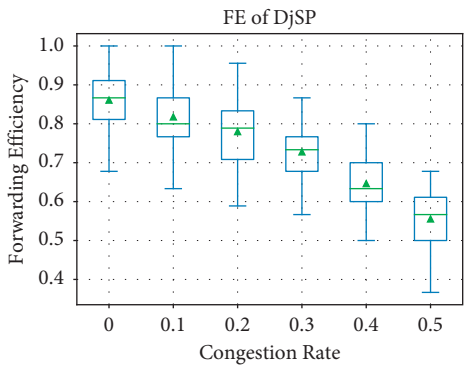

Figure 8: FE of DjSP.

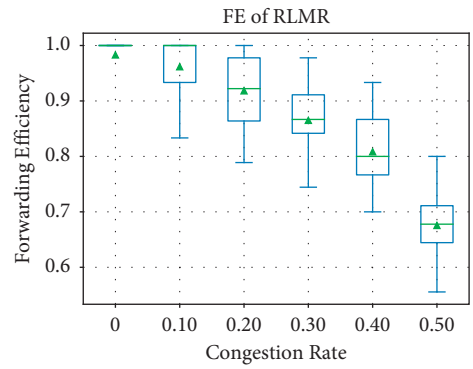

Figure 9: FE of RLMR.

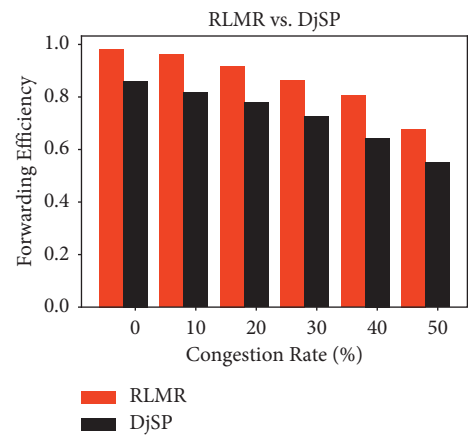

Figure 10: Comparison of RLMR and DjSP.

Figure 12 shows the comparison of three routing algorithms' jitter (inconsistent latency between packets). According to the results, Dijkstra's jitter is the largest overall, followed by ECMP's. In contrast, the algorithm in this paper can control the delay within a certain range and has a lower jitter.

As shown above, RLMR can indeed divide flow into multiple small flows when the link bandwidth is not enough, use the small bandwidth to forward flow. The simulation results show that this scheme can effectively improve the link utilization LBU. Compared with Dijkstra algorithm and ECMP algorithm, this algorithm has some advantages in terms of jitter and packet loss rate.

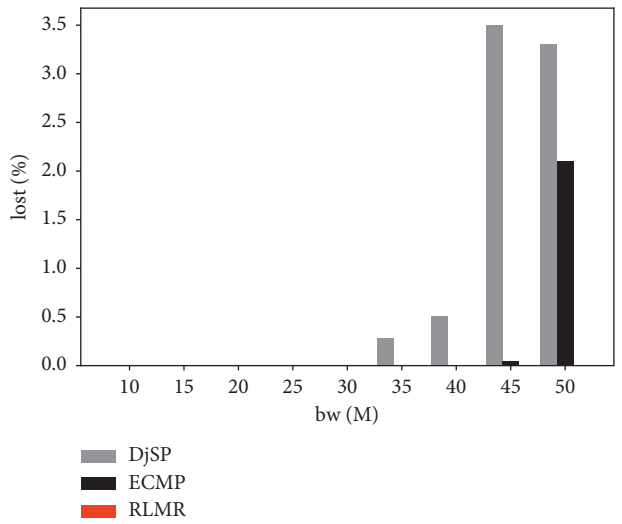

Figure 11: Packet loss rate comparison of RLMR, DjSP, and ECMP algorithm.

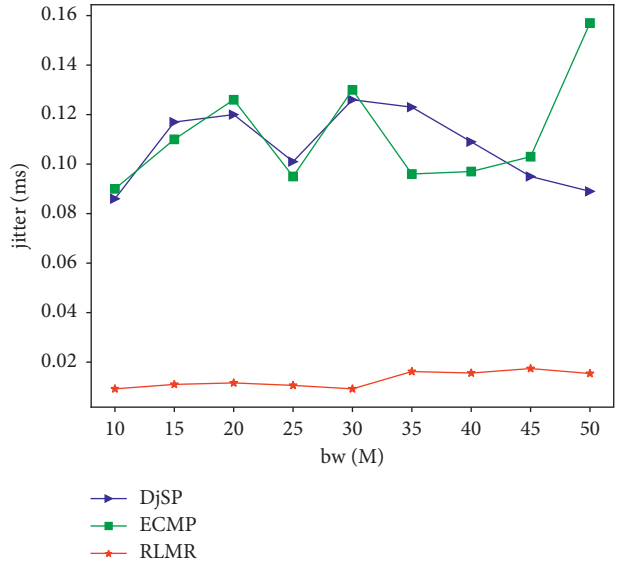

Figure 12: Jitter comparison of RLMR, DjSP, and ECMP algorithm.

5. Conclusion

To improve LBU, RLMR applies RL to SDN network routing planning and focuses on the analysis of the construction of the routing planning and the design of reward functions. The result of the simulation shows that after training the model, this algorithm can provide different QoS routes for different services, and when the link bandwidth is not enough, it can divide a large flow into multiple small flows, thereby improving the LBU.

It is necessary to verify the performance of this algorithm in a larger and more complex network in the future. In the simulation, this paper uses a simple network topology to verify the effectiveness and superiority of this algorithm. However, in the actual network, the network state is complicated and changeable. Simulation verification in the actual network can better illustrate the effectiveness and rationality of this algorithm.

It would be interesting to compare our scheme with these existing ones [9, 21] via simulation or in the testbed with realistic traffic traces. However, due to the time limitation, we leave it to future work.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grants 61871468, 61801427, and 62111540270, Zhejiang Provincial Key Laboratory of New Network Standards and Technologies (No. 2013E10012), and the Key Research and Development Program of Zhejiang under Grant No. 2020C01079.

References

[1] H. Hailong Zhang, X. Xiao Guo, J. Jinyao Yan, Bo Qianjun Shuai, and Q. Shuai, "SDN-based ECMP algorithm for data center networks," in *Proceedings of the 2014 IEEE Computers, Communications and IT Applications Conference*, pp. 13–18, Beijing, China, 2014.

[2] J. W. Guck, A. V. Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: a comprehensive survey and performance evaluation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 388–415, 2017.

[3] C. Lin, K. Wang, and G. Deng, "A QoS-aware routing in SDN hybrid networks," *Procedia Computer Science*, vol. 110, pp. 242–249, 2017.

[4] C.-Y. Hong, S. Mandal, M. Al-Fares et al., "B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, Budapest, Hungary, 2018.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*MIT Press, Cambridge, MA, USA, 2018.

[6] Z. Tang, H. Hu, C. Xu, and K. Zhao, "Exploring an efficient remote biomedical signal monitoring framework for personal health in the COVID-19 pandemic," *International Journal of Environmental Research and Public Health*, vol. 18, no. 17, p. 9037, 2021.

[7] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: from concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[8] H. Ren, X. Li, J. Geng, and J. Yan, "A Sdn-based dynamic flow scheduling algorithm," in *Proceedings of the 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, IEEE, Chengdu, China, 2016.

[9] Z. Zhu, S. Li, and X. Chen, "Design QoS-aware multipath provisioning strategies for efficient cloud-assisted SVC video

streaming to heterogeneous clients," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 758–768, 2013.

[10] Y. Lin and H. Liu, "Openflow-based routing algorithm design," *Computer and Telecom*, vol. 9, pp. 30-31, 2015.

[11] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "HiQoS: an SDN-based multipath QoS solution," *China Communications*, vol. 12, no. 5, pp. 123–133, 2015.

[12] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *Journal of Lightwave Technology*, vol. 31, no. 1, pp. 15–22, 2013.

[13] Y. Yang, J. Yang, and D. Qin, "Multipath routing algorithm for data center networks," *Journal of Tsinghua University*, vol. 3, 2016.

[14] N. Huangwu, C. Huang, and X. Huang, "Multipath routing algorithm for SDN-based fat tree data center network," *Computer Science*, vol. 43, no. 6, pp. 32–34, 2016.

[15] C. Yu, J. Lan, Z. Guo, and Y. Hu, "DROM: optimizing the routing in software-defined networks with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 64533–64539, 2018.

[16] P. Sun, Y. Hu, J. Lan, L. Tian, and M. Chen, "TIDE: time-relevant deep reinforcement learning for routing optimization," *Future Generation Computer Systems*, vol. 99, pp. 401–409, 2019.

[17] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntes-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," 2017, https://arxiv.org/abs/1709.07080.

[18] T. A. Q. Pham, Y. Hadjadj-Aoul, and A. Outtagarts, "Deep reinforcement learning based QoS-aware routing in knowledge-defined networking," in *Proceedings of the International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Springer, Cham, Germany, 2018.

[19] C. Cheng, *Research on Reinforcement Learning Based Routing Planning Algorithm in SDN*Zhejiang University of Technology, Hangzhou, China, 2018.

[20] J. Rischke, P. Sossalla, H. Salah, F. H. P. Fitzek, and M. Reisslein, "QR-SDN: towards reinforcement learning states, actions, and rewards for direct flow routing in software-defined networks," *IEEE Access*, vol. 8, pp. 174773–174791, 2020.

[21] X. Guo, H. Lin, Z. Li, and M. Peng, "Deep-reinforcement-learning-based QoS-aware secure routing for SDN-IoT," *IEEE Internet of Things Journal*, vol. 7, pp. 6242–6251, 2019.

[22] I. Bueno, J. I. Aznar, E. Escalona, J. Ferrer, and J. A. Garcia-Espin, "An OpenNaaS based SDN framework for dynamic QoS control," in *Proceedings of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, IEEE, Trento, Italy, 2013.

[23] S. Tomovic, N. Prasad, and I. Radusinović, "SDN control framework for QoS provisioning," in *Proceedings of the 2014 22nd Telecommunications Forum Telfor (TELFOR)*, IEEE, Belgrade, Serbia, 2014.