Hindawi
Security and Communication Networks
Volume 2022, Article ID 4457645, 11 pages
https://doi.org/10.1155/2022/4457645

Research Article

DSOQR: Deep Reinforcement Learning for Online QoS Routing in SDN-Based Networks

Lianming Zhang, Yong Lu, Dian Zhang, Haoran Cheng, and Pingping Dong

College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China

Correspondence should be addressed to Pingping Dong; ppdong@hunnu.edu.cn

Received 27 May 2022; Accepted 16 November 2022; Published 29 November 2022

Academic Editor: Biao Han

With the rapid development of mobile communication technology, there are an increasing number of new network applications and services, and the existing best-effort routing algorithms cannot meet the quality-of-service (QoS) requirements of these applications and services. QoS-routing optimization solutions based on a software-defined network (SDN) are often targeted at specific network scenarios and difficult to adapt to changing business requirements. The current routing algorithms based on machine learning (ML) methods can generally only handle discrete, low-dimensional action spaces and use offline network data for training, which is not effective for dynamic network environments. In this study, we propose DSOQR, which is an online QoS-routing framework based on deep reinforcement learning (DRL) and SDN. DSOQR collects network status information in real time through the software-defined paradigm and carries out on-policy learning. Under this framework, we further propose SA3CR, which is a QoS-routing algorithm based on SDN and asynchronous advantage actor-critic (A3C). The SA3CR algorithm can dynamically switch routing paths that meet the conditions according to the current network status and the needs of different service types to ensure the QoS of target traffic. Experimental results show that DSOQR is effective and that the SA3CR algorithm has better performance in terms of delay, throughput, and packet loss rate.

1. Introduction

With the rapid development of cloud computing, artificial intelligence, the Internet of things, and 5G communication technologies, new services such as live streaming, online mobile games, and video conferencing have emerged in large numbers, and network data have increased exponentially [1, 2]. A report by Cisco [3] shows that, by 2023, global mobile application downloads will reach 299 billion. In response to the ever-increasing diversified needs in the network, best-effort routing algorithms have difficulty accurately perceiving the current network status to ensure the quality of service (QoS) of flow. The existing traditional best-effort routing algorithm is difficult to accurately sense the current network state to ensure the QoS of flows and has problems such as not being easy to deploy, poor scalability, only suitable for specific scenarios, and coarse control granularity [4].

The software-defined network (SDN), as a new network architecture, provides a solution to traditional QoS-routing problems [5, 6]. SDN separates the control plane from the data plane, allows network administrators to monitor the network status in real time and obtain a global view of the network, facilitates the rational utilization and deployment of network resources by administrators, and simplifies network management. SDN provides a programmable interface, which reduces the difficulty of providing routing optimization for flows. However, most of the current SDN-based routing schemes use the shortest path algorithm, and flows share the same path, which is likely to cause link congestion. Some SDN-based routing schemes that use heuristic algorithms can only be adapted to specific network scenarios, and algorithms have relatively high complexity [7].

In recent years, introducing intelligent algorithms in machine learning (ML) into SDN-based QoS-routing

optimization has provided convenient conditions for the realization of more intelligent dynamic routing [8]. The training methods of intelligent routing algorithms for ML are divided into online and offline. Intelligent routing models based on supervised learning (SL) adopt offline training methods, and models based on reinforcement learning (RL) can be trained both online and offline. When the model is trained offline, it is first necessary to collect the required data from the network environment and then use the processed data for offline training of the intelligent routing model. The final obtained training model is deployed in the network environment to make online routing decisions. However, offline training for intelligent routing algorithms often faces two challenges: (1) the collection of training data may require a relatively high cost and (2) the network state in the scene may be different from the training dataset, which causes the routing algorithm to fail to achieve expected results. Online training can ensure that the model adapts to changes in the network environment while avoiding difficulties and additional costs caused by the establishment of an offline environment. However, RL models may produce unpredictable behaviors during training. During routing, these behaviors may cause problems such as routing loops and link congestion [9].

In this study, we use an SDN as the basic framework, use asynchronous advantage actor-critic (A3C) to make intelligent routing decisions, and propose a new intelligent online QoS-routing optimization solution. The main contributions of this study are as follows:

We propose an intelligent QoS-routing optimization algorithm based on SDN and A3C (SA3CR). The SA3CR algorithm uses the QoS-A3C algorithm to dynamically update link weight and then generates the routing strategy according to the requirements of the QoS target flow and by considering multiple QoS metric parameters.

We propose DSOQR which is an online QoS-routing framework based on SDN and the on-policy property of A3C. The framework can realize the real-time collection of network information data and online strategy learning, thus realizing the optimization of QoS routing.

We realize an SDN-based intelligent QoS-routing optimization system. The system can collect network state information in real time, generate a dynamic intelligent-routing policy, and dispatch a routing policy in real time. We also test the SA3CR algorithm on OS3E and NSFNet network topologies. Experimental results show that the SA3CR algorithm obtains higher performance than ECMP, KSP, NAF_R, and DDPG_R.

The remainder of this study is organized as follows. We first introduce the related research work of intelligent QoS routing based on SDN in Section 2. Section 3 describes the framework of the proposed DSOQR. In Section 4, we describe the workflow of DSOQR and the online processing of routing. Extensive experiments are conducted in Section 5. Finally, Section 6 concludes the paper.

2. Related Works

In the SDN, the controller can update the flow table in the switch through routing policies to implement routing control. However, most of the current routing optimization methods are based on the heuristic algorithm, which has high complexity. However, ML can quickly calculate the routing scheme that is close to the optimal solution through training and does not require an accurate mathematical model underlying the network [8]. The current ML algorithms used to solve the routing optimization problem in the SDN can be summarized into the following two categories.

2.1. SDN Routing Based on SL. SL is a technique for learning through historical data tags. Azzouni et al. [10] proposed a dynamic routing framework of SDN based on deep neural network (DNN)-NeuRoute. NeuRoute uses a DNN to learn traffic characteristics and generate forwarding rules to optimize network throughput. Prasad et al. [11] proposed an application-aware multipath packet forwarding (AMPF) mechanism based on application recognition and path state awareness, which integrates ML and SDN. The AMPF mechanism divides the priority according to flow characteristics, finds k alternative paths, evaluates multiple QoS parameters of each possible path characteristic, and assigns paths to flows according to the priority of their class.

However, regardless of whether the input and output of the heuristic algorithm are used as the training set or the traffic is classified in advance by perceptual prediction, a large amount of label data needs to be obtained for training in the training process, which will lead to high complexity. Therefore, we need to make the network model have self-sensing characteristics for traffic, try not to use labels, and make the network realize intelligent routing through independent learning of historical data or online learning.

2.2. SDN Routing Based on RL. RL is a self-learning technology and can carry out dynamic decision management. Sendra et al. [12] proposed a learning process of replacing the Q table with a DNN in an SDN topology, using the experience replay mechanism for intensive training and randomly extracting data from the experience pool to learn previous experience information, reducing the packet loss rate and delay. However, a deep Q network (DQN) is only suitable for the control and optimization of low-dimensional discrete spaces and cannot converge in real time. Sun et al. [13] proposed an SDN framework based on ML, which mainly uses a deep deterministic policy gradient (DDPG) to optimize SDN routing. At the same time, a DDPG routing optimization mechanism is proposed, which analyzes and measures the network by the controller to obtain the global network state and determines an optimal behavior, that is, a set of link weights. Link weights are updated by maximizing rewards, and the controller constantly generates new rules to establish a new path. By analogy, iterative optimization continues until the optimal solution is obtained.

To alleviate the instability that occurs when the traditional strategy gradient method combines with the neural network, the above RL model adopts an empirical playback mechanism to eliminate the correlation between training data. However, there are two problems with the experience playback mechanism: (1) each real-time interaction between the agent and the environment requires considerable memory and computing power and (2) the experience replay mechanism requires agents to adopt an off-policy method to learn, which can update based on the data generated by the old policy. The A3C algorithm [14] utilizes a multithreaded approach to execute multiple agents asynchronously. Through different states experienced by the parallel agent, the correlation between state transfer samples generated in the training process is removed, thus overcoming the above two problems. In this paper, we propose DSOQR, an online QoS-routing framework based on SDN and DRL. It can perform online strategy learning while collecting real-time data for the effective use of network resources.

3. Framework of DSOQR

In this section, we introduce the framework of the proposed DSOQR, as shown in Figure 1. It consists of the data plane, control plane, and DRL plane running on the control plane. The data plane is composed of software-defined forwarding equipment, including OpenFlow switches. The software-defined capability of the data plane can facilitate the controller in the control plane to detect the network status in real time, flexibly collect the required information, and flexibly change forwarding rules according to requirements. The control plane is composed of a controller, including a QoS parameter measurement module, a network-monitoring module, and a DSOQR routing module. The controller collects and organizes the required network status information, transmits the global view to the DRL plane, and dynamically updates the routing rules of the data plane according to the routing update strategy issued by the DSOQR routing module. The agent of the DRL plane is implemented based on a DRL algorithm, which mainly provides decision-making for dynamic routing rules. The specific implementation is detailed in Section 4.

3.1. DRL Plane of DSOQR. The DRL plane is an agent responsible for DRL for network QoS optimization. The DSOQR can use the global view of the network provided by the control plane and generate dynamic routing rules online based on the network resource information of the data plane obtained in real time to adapt to constantly changing traffic distribution characteristics in the network. In this study, we use the A3C model as the agent of the DRL plane and define four key elements.

3.1.1. Normalization. To facilitate data processing and speed up learning, we normalize QoS parameters using the Z-score method, which uses the ratio of the mean and standard deviation of the original data to standardize the data.

3.1.2. Input State. To obtain the input state vector s, the QoS parameter information collected by QoS measurement and network monitoring modules should be first obtained: throughput $B = [B(1), B(2), \ldots, B(n)]$, packet loss rate $L = [L(1), L(2), \ldots, L(n)]$, and delay $D = [D(1), D(2), \ldots, D(n)]$, where n is the number of links. Once the SDN controller receives multiple service requests at the same time, it may interfere with the routing path selection. Therefore, before splicing the state vector s, DSOQR will increase the throughput B by a flow threshold $b(i)$, which is determined according to the current throughput of each link, to increase the link weight to reduce the possibility of link congestion. Then, we can obtain B', L', and D', respectively, based on the normalization of corresponding parameters with the Z-score standard function. Finally, after stitching and normalization, we can obtain the input state vector $s = [B', L', D']$.

3.1.3. Output Action. The output action vector a is generated by the DRL agent based on the QoS strategy function and the input state s. It consists of n elements. Each element represents the distribution proportion of the available forwarding path between each node pair; that is, if the link weight is w, then $a = [w_1, w_2, \ldots, w_n]$.

3.1.4. Reward Value. The reward value R is the QoS evaluation of the income obtained from the environment after the action is performed. To take into account multiple QoS standards, we use the weighted sum of the three QoS parameters after normalization and the corresponding weight coefficients to measure the quality of the link. The weight coefficient of throughput B′ is $\alpha \in [0, 1]$, the weight coefficient of packet loss rate L' is $\beta \in [0, 1]$, and that of delay D′ is $\lambda \in [0, 1]$. If a network wants better QoS, the network link must have a higher throughput, lower delay, and lower packet loss rate. According to the experimental scenario in this paper, the relationship between the weight coefficients of the three parameters follows the work described in [13]. In the experiment, the values of these three parameters are set as follows: $\alpha = 0.7$, $\beta = 0.2$, and $\lambda = 0.1$. The reward value R could be elastically calculated with the QoS parameter, and their weight coefficients under different QoS strategies are as follows:

$$R = -(\alpha \cdot B' + \beta \cdot L' + \lambda \cdot D'). \quad (1)$$

3.2. Control Plane of DSOQR. The control plane of DSOQR is composed of one or more controllers and is the core part of the network. It can provide the status information of the underlying network and QoS parameter measurement information for the DRL plane and dynamically update the routing rules of the data plane according to the routing policy issued by the DRL plane. The control plane of the proposed DSOQR is composed of three modules: QoS parameter measurement, network monitoring, and DSOQR routing.

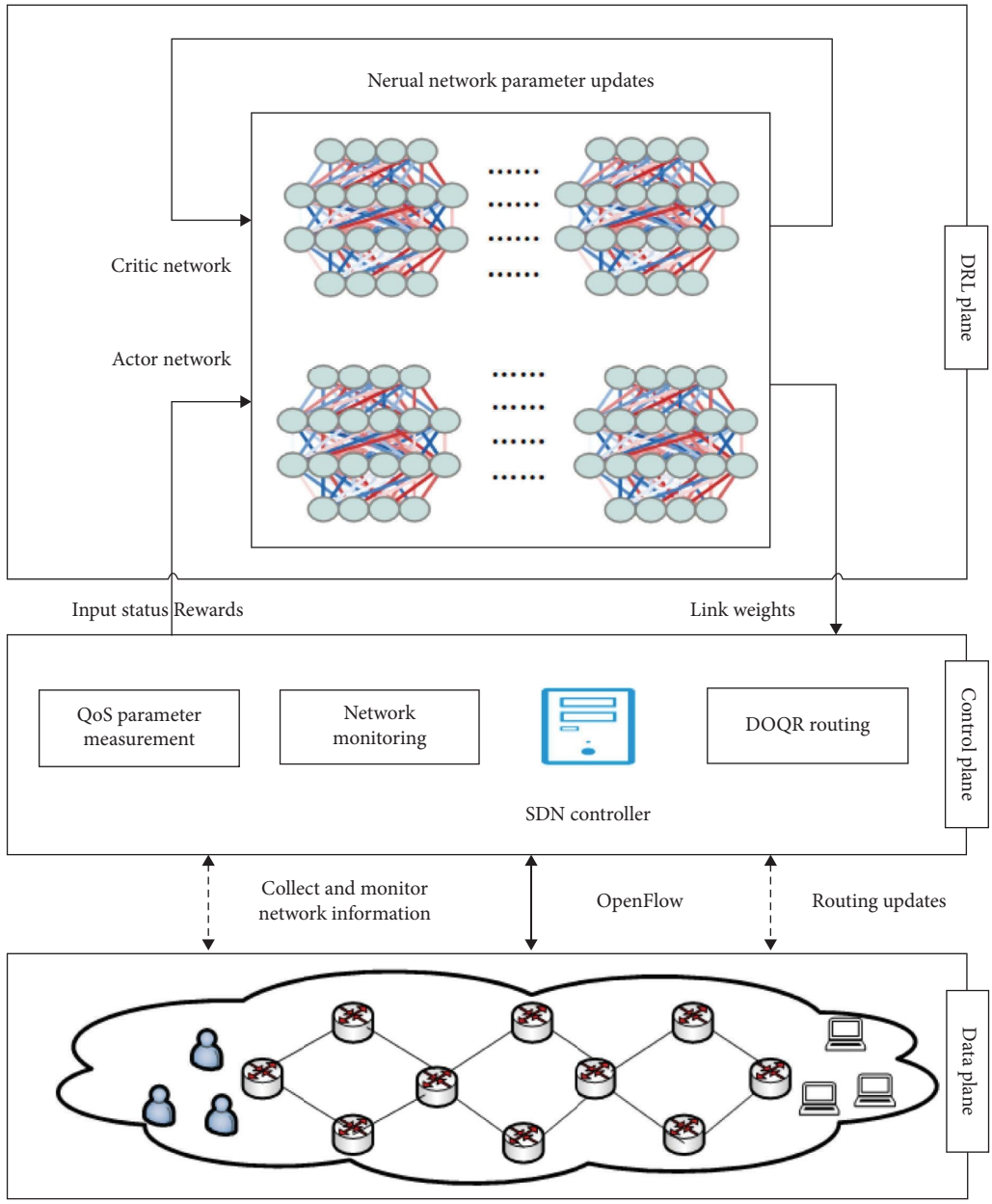

Figure 1: Framework of DSOQR.

3.2.1. QoS Parameter Measurement. The module mainly collects a variety of link status information and plays an important role in the system. We use the SDN controller to send the request message to a switch in the data plane and obtain the real-time network status information through the interface of the data plane. This information is processed and finally used as the input of DSOQR routing. QoS parameters (such as delay, throughput, and packet loss rate) are collected at fixed intervals (0.5 ms) and obtained in the following methods.

For the delay, we can obtain the time when the SDN controller sends and receives link layer discovery protocol (LLDP) data packets to switches S_1 and S_2, which are T_1 and T_2, respectively. Meanwhile, we use ECHO messages to obtain round-trip times T_{C-S_1} and T_{S_2-C} from the controller to the switch. Therefore, we can obtain the delay $D_{S_1-S_2}$ between switches S_1 and S_2 by calculating the departure time and arrival time of the LLDP data packet and the difference between the time from the switch to the controller and the time from the controller to the switch, and we have

$$D_{S_1-S_2} = T_2 - T_1 - T_{C-S_1} - T_{S_2-C}. \quad (2)$$

For the throughput, the SDN controller sends an OpenFlow port status request message to switches S_1 and S_2 at times T_1 and T_2, respectively. When the status message is returned by the switch, they can be used to obtain the number of packets received and forwarded by the port P_1 of the switch S_1 and the port P_2 of the switch S_2, respectively. Packet sizes received and forwarded by the switch port P_1 are RX_1 and TX_2 and those by the switch port P_2 are RX_1 and

TX_2. Also, we take the input and output throughput of the switch S_1 of the port P_1 and the switch S_2 of the port P_2 as I_{S_1,P_1}, O_{S_1,P_1}, I_{S_2,P_2}, and O_{S_2,P_2}, respectively. Therefore, from T_1 to T_2, the throughput B_{S_1,S_2} of the link is equal to the smaller value of the throughput B_{S_1,P_1} on P_1 of S_1 and the throughput B_{S_2,P_2} on P_2 of S_2 and can be calculated as follows:

$$\begin{aligned} B_{S_1,S_2} &= \min\left\{B_{S_1,P_1}, B_{S_2,P_2}\right\} \\ &= \min\left\{I_{S_1,P_1} + O_{S_1,P_1}, I_{S_2,P_2} + O_{S_2,P_2}\right\} \\ &= \min\left\{\frac{\left(\mathrm{RX}_{1,T_2} - \mathrm{RX}_{1,T_1}\right)\left(\mathrm{TX}_{1,T_2} - \mathrm{TX}_{1,T_1}\right)}{T_2 - T_1}, \frac{\left(\mathrm{RX}_{2,T_2} - \mathrm{RX}_{2,T_1}\right)\left(\mathrm{TX}_{2,T_2} - \mathrm{TX}_{2,T_1}\right)}{T_2 - T_1}\right\}. \end{aligned} \tag{3}$$

For the packet loss rate, we can obtain the difference $\Delta\mathrm{TX}_{1,S_1 \longrightarrow S_2} = \mathrm{TX}_{1,T_2} - \mathrm{TX}_{1,T_1}$ between the number of packets sent and the number of received packets on the port P_1 of the switch S_1 at times T_1 and T_2 and the difference $\Delta\mathrm{RX}_{2,S_1 \longrightarrow S_2} = \mathrm{RX}_{2,T_2} - \mathrm{RX}_{2,T_1}$ between the number of packets sent and the number of received packets on the port P_2 of the switch S_2. Therefore, the packet loss rate $L_{S_1 \longrightarrow S_2}$ of a link from S_1 to S_2 is calculated as follows:

$$L_{S_1 \longrightarrow S_2} = 1 - \frac{\Delta\mathrm{RX}_{2,S_1 \longrightarrow S_2}}{\Delta\mathrm{TX}_{1,S_1 \longrightarrow S_2}}. \tag{4}$$

3.2.2. Network Monitoring. The module is mainly used to monitor the QoS parameter status and network flow information and obtain various QoS requirements of the service by monitoring flows. If the current routing decision fails to meet QoS requirements, rerouting is performed. Specifically, it monitors the remaining bandwidth of the link, the throughput of the port during the routing of the current flow, and the delay spent in the routing process that does not meet QoS requirements of the service. If so, the DSOQR routing module needs to recalculate the strategy that meets QoS requirements, and the controller reissues the flow table to the switch.

3.2.3. DSOQR Routing. DSOQR routing consists of two algorithms: the QoS-A3C algorithm and the SA3CR algorithm. The intelligent QoS-routing algorithm SA3CR, based on the network state information collected by the software-defined paradigm in real time and trained online by the QoS-A3C algorithm, is able to dynamically switch the routing path that satisfies its conditions according to the current network state and the demands of flows of different service types, thus ensuring the QoS of target traffic.

The A3C algorithm adopts the actor-critic framework. The actor-critic algorithm consists of an actor and a critic. The actor makes corresponding actions based on-policy iteration, and the critic is used to evaluate the quality of actions based on-policy value functions. The agent obtains the action through the actor network. After executing the action, the environment will feed back a new state and a reward value R to the agent. The critic network evaluates the reward value R and transmits the result to the actor network for parameter update. The critic's policy value function is

$$V_\pi = \mathbb{E}_\pi\left(R + \gamma V_\pi\left(s'\right)\right), \tag{5}$$

where $\gamma \in [0, 1]$ is the discount factor, implying the impact of the future on the present calculation, and s' is the state at the next moment. The action value function of the strategy is

$$Q_\pi(s, a) = R + \gamma V_\pi\left(s'\right). \tag{6}$$

The A3C algorithm proposes an advantage function A in Actor networks, using n step sampling, to improve training efficiency, and its advantage function can be defined as

$$A(s, a) = R + \gamma R_{t+1} + \ldots \gamma^{n-1} R_{t+n-1} + \gamma^n V_\pi\left(s'\right) - V_\pi(s). \tag{7}$$

Compared with the actor-critic algorithm, the entropy term coefficient e is added to the loss function of the policy network in the A3C algorithm, so the gradient of the actor in the A3C algorithm is updated as follows:

$$\nabla_\theta J(\pi) = E_{p^s}\left[A(s, a) * \nabla_\theta \log\, \pi(a|s)\right] + e\nabla_\theta H\left(\pi\left(s_i; \theta\right)\right). \tag{8}$$

The gradient of the critic network part of the A3C algorithm is updated as

$$dw \leftarrow dw + \frac{\partial\left(R - V\left(s_i, \omega'\right)\right)^2}{\partial\omega'}. \tag{9}$$

The biggest advantage of A3C compared to other RL algorithms is the use of an asynchronous training framework, where multiple agents are executed to interact with the environment, the empirical values obtained during the interaction are sampled, the whole network is updated, and the optimal network policy is obtained.

The QoS-A3C algorithm is different from other DRL learning algorithms that use off-policy learning, and it uses on-policy learning. It also uses an asynchronous training mechanism to execute multiple agents, which

improves the training speed of the network. First, QoS-A3C is based on the actor-critic framework and normalizes the parameters collected by the QoS parameter measurement module and the network monitoring module and then conducts online training to make corresponding actions, i.e., the weights of each link $[w_1, w_2, \ldots, w_n]$. Then, next training is performed based on the reward value fed by the SA3CR algorithm and the network environment, and if the reward value is positive, the probability of selecting that action increases, and vice versa decreases. After repeated training and continuous learning of the DNN, the optimal action can be obtained. The QoS-A3C algorithm is shown in Algorithm 1.

The idea of the SA3CR algorithm is as follows: for data flows without QoS requirements, we use Dijkstra's algorithm and give negative feedback to the A3C model. For service flows with QoS demands, the link weight values $[w_1, w_2, \ldots, w_n]$ obtained from the QoS-A3C algorithm are weighted, as well as the priority of the flow is considered for outputting the best path, so as to achieve dynamic routing switching from the source to the destination and network flow online routing function. The SA3CR algorithm is shown in Algorithm 2.

To adapt to the dynamically changing network topology and traffic characteristics, the above model adopts a closed-loop learning method to regularly train the intelligent routing model according to latest network traffic characteristics. The worst-case complexity of the algorithm SA3CR is derived as follows: The DRL agent accesses all states in the state and action space, which implies the complexity that depends on the size of the space. In SA3CR, the size of the state space is $O(3 * k)$, where k is the number of links. The size of the action space is limited to k links associated with each state. Therefore, its complexity is $O(kN^2)$. Since k is a constant, the worst-case complexity is $O(N^2)$.

3.3. Data Plane of DSOQR. The data plane of DSOQR is composed of several network elements, each of which can contain one or more switches. Each switch can detect the network status, flexibly collect the required information, and use it to forward and process data according to the control logic of the control plane of DSOQR.

4. Management of DSOQR

In this section, we first describe the basic workflow of our proposed DSOQR, which operates in three steps, i.e., collection, training and inference, and routing. These three steps are all performed online. Then, we focus on the online processing of routing.

4.1. Workflow of DSOQR. The framework of DSOQR uses SDN to collect network status information in real time and learns through on-policy. It can dynamically switch routing paths that meet its conditions according to the current status and the needs of different service types to ensure the QoS of target traffic. The basic workflow of DSOQR is as follows: First, on the control plane, the network-monitoring module regularly monitors traffic, and the QoS parameter measurement module counts the required QoS parameter information, including the throughput matrix, delay matrix, and packet loss rate matrix. Then, the SDN controller transmits this matrix information to the DRL plane, which is used to train the neural network to generate flow entries and maximize the reward to meet the requirements of the QoS flow. Since the DRL plane uses the A3C model, multiple subagents can be used to randomly explore the current network environment. The central network of the A3C model selectively performs gradient update and network learning according to the exploration situation of each agent to better adapt to the current network environment, as shown in Figure 2. Therefore, we can obtain a globally optimal path weight as the output according to the locally optimal path weight obtained by each subagent in the A3C model. Finally, the DSOQR routing module of the control plane will select the best path based on the weight of the link and convert the routing information into flow entries. The SDN controller sends the flow table entry to the corresponding switch to complete the flow table update. The dynamic routing switch from the source to the destination and the online routing of network flows are realized.

4.2. Online Processing of Routing. We use the on-policy of the A3C model, and a QoS flow guarantee mechanism is added to the DSOQR routing module, which is composed of a QoS parameter measurement, and it ensures that the intelligent routing model based on DRL can perform reliable online learning. During training, it is judged whether the path weight decision made by the A3C model meets the requirements of flows. The DSOQR routing module directly adopts a simple and reliable algorithm for routing deployment, such as shortest path routing for the routing decision that does not meet the QoS flow. At the same time, a negative reward value is applied to the A3C model to reduce the possibility that the A3C model will generate similar path weight decisions again.

5. Performance Evaluation

In the following sections, we evaluate the performance of our proposed DSOQR, especially the performance of the SA3CR algorithm in terms of delay, throughput, and packet loss rate.

5.1. Experimental Setup. In the experiments, we use the delay, loss rate, and load of the system to evaluate the resulting performance. We compare the performance achieved by our method with the results obtained by the other two DRL-based methods (NAF_R [12] and DDPG_R [13]) and two traditional methods (KSP [15] and ECMP [16]). DRL-based methods are implemented based on Python and TensorFlow. Traditional methods are available in the Linux kernel, and we test them by issuing corresponding commands. All evaluations in this section are performed in

Input: Network status information.
Output: Link weight.
(1) Initialize parameters θ and ω of the public A3C, corresponding parameters θ' andω' of the current thread A3C, the global sharing counter cnt, and the global maximum sharing counter $cnt_{\max}$.
(2) **repeat**
(3) Update step counter $t = 1$.
(4) Reset the amount of gradient updates for the actor and critic: $d_\theta \leftarrow 0, d_\omega \leftarrow 0$.
(5) Current thread A3C synchronization public A3C parameters: $\theta' \leftarrow \theta$, $\omega' \leftarrow \omega$.
(6) $t_{\text{start}} = t$, get the current network status s_t.
(7) **repeat**
(8) Choose the routing weight action a_t based on-policy $\pi(a_t|s_t;\theta')$.
(9) Execute the routing weight action a_t to get the path reward r_t and the new state s_{t+1}.
(10) $t \leftarrow t+1, cnt \leftarrow cnt+1$.
(11) **until** s_t is the stream termination state.
(12) Path discount reward value at flow termination $R = V(s_t, \theta'_v)$.
(13) **for** $i \in (t-1, \ldots, t_{\text{start}})$**do**
(14) Calculate the path discount bonus value for each moment $R \leftarrow r_t + \gamma R$.
(15) Cumulative gradient update θ': $d_\theta \leftarrow d_\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\omega')) + e\nabla_{\theta'} H(\pi(s_i;\theta'))$.
(16) Cumulative gradient update ω': $d_\omega \leftarrow d_\omega + \partial(R - V(s_i,\omega'))^2/\partial\omega'$.
(17) **end for**
(18) Update global A3C parameters: $\theta = \theta - \alpha d_\theta, \omega = \omega - \beta d_\omega$.
(19) **until** $cnt > cnt_{\max}$.

Algorithm 1: QoS-A3C algorithm.

Input: Graph
Output: Best_path.
(1) Cpath = getPaths (graph, src, dst);
(2) Path = getDijkstraPaths (graph, src, dst);
(3) **if** Type_flow = = QoSflow **then**
(4) PathLabel = setAslabel (Cpath)
(5) NetworkState = getNetworkState (Portlist, Switchlist, and PathLabel)
(6) LinkWeight = QoS-A3C (NetworkState)
(7) Best_path = Min (Sum_Cpath (linkweight))
(8) SetAsRoute (Best path, Src, Dst, FlowID, and priority)
(9) **else**
(10) SetAsRoute (Path1, Src, Dst, FlowID, and priority)
(11) Best_path = Path
(12) **end if**
(13) **return** Best_path

Algorithm 2: SA3CR algorithm.

network scenarios of 14-node and 21-edge OS3E [17] and 14-node and 19-edge NSFNet [18], which are built in the data plane through Mininet [19]. We set the delay, bandwidth, packet loss rate, and other parameters through TC Link, which are set to 1 ms, 50 Mbps, and 1%, respectively, and we use Python 3.5 to implement the three modules of QoS measurement, network monitoring, and DSOQR routing in the Ryu controller [20], as well as the DRL agent based on TensorFlow 2.0. They all run on workstations, which use an Intel i7-5500U CPU, 8 GB memory, and Ubuntu 16.04 64 bit operating system, and a grid search is used to properly fine-tune hyperparameter values in DSOQR, as shown in Table 1.

To simulate the network traffic in real network scenarios, we use the traffic matrix (TM) generation tool [21] to generate 20 TMs among 14 nodes, where the size of traffic sent from the source node i to the destination node j is the element in TM. The average traffic size sent by each node ranges from 0 to 80 Mbps and obeys a uniform distribution. In Mininet, any two different nodes can communicate with each other via the Iperf3 [22] tool, which is used to send UDP packets based on generated TM. Therefore, in the network environment, we can simulate network traffic and implement its dynamic changes. We use the Ryu controller to measure QoS parameters and store measurement results.

5.2. Delay. The impact of the request rate of the data on the average delay is shown in Figure 3. With the continuous growth of traffic, the average delay of the SA3CR algorithm is

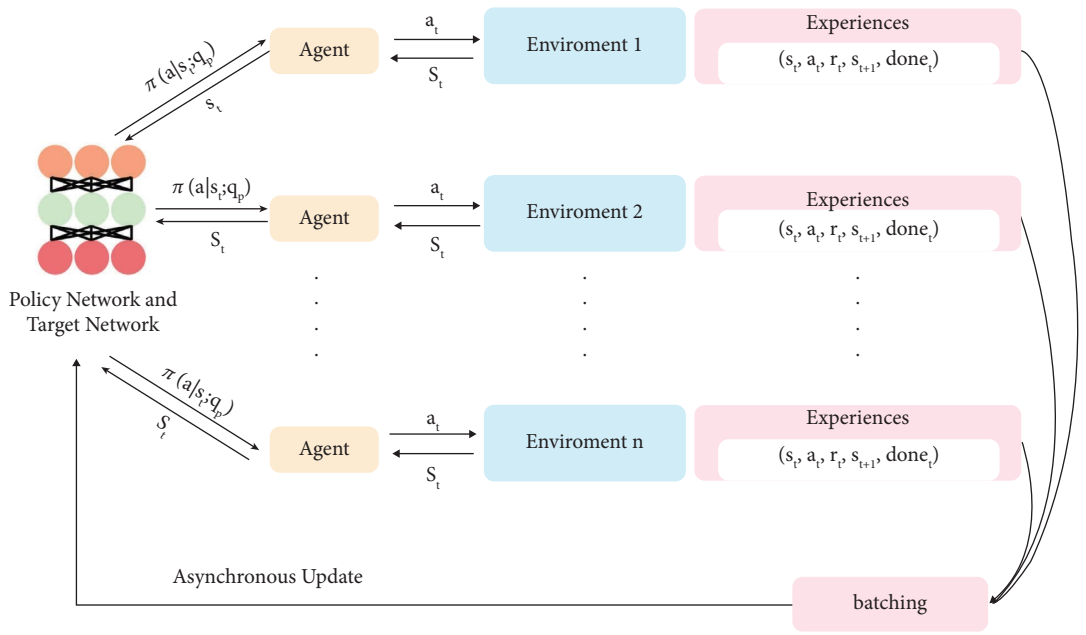

Figure 2: Workflow of DSOQR.

Table 1: DSOQR hyperparameter configuration.

Hyperparameter	Value
Training episode	2000
Batch size	50
Actor learning rate	0.0001
Critic learning rate	0.001
Gamma	0.95
Entropy beta	0.01

less than the average delay of the other four algorithms, including DDPG_R, NAF_R, ECMP, and KSP. For example, when the data rate ranges from 0 to 50 Mbps, the average delay of each algorithm rises steadily. When the data rate reaches 50 to 80 Mbps, the average delay increases rapidly because the data rate exceeds the maximum bandwidth of the link.

Figure 3(a) presents the average delay produced by SA3CR, DDPG_R, NAF_R, ECMP, and KSP in the OS3E network. If the data rate reaches 80 Mbps, the average delay of the SA3CR algorithm is 41 ms. The average delays of the DDPG_R algorithm, NAF_R algorithm, ECMP algorithm, and KSP algorithm are 45 ms, 48ms, 57 ms, and 60 ms, respectively, which are higher than that of the SA3CR algorithm. Compared with the other four algorithms, the average delay of the SA3CR algorithm has reduced by 8%, 15%, 28%, and 32%, respectively. Figure 3(b) presents the average delay produced by SA3CR, DDPG_R, NAF_R, ECMP, and KSP in the NSFNet network. The mean delay produced by the SA3CR algorithm are, on average, 6%, 10%, 19%, and 21% lower than those given by DDPG_R, NAF_R, ECMP, and KSP algorithms, respectively.

The KSP algorithm selects the first k paths for routing, and there is data sharing of one or more links during the routing process. Because the link bandwidth is limited, it will cause link congestion, leading to increased delay. NAF_R is a continuous DQN algorithm and has overevaluation and slow learning and convergence speed. As a result, the delay of the NAF_R algorithm is lower than that of the DDPG_R algorithm. Although the DDPG_R algorithm has excellent network exploratory properties, it easily falls into suboptimal solutions. Meanwhile, the SA3CR algorithm relies on multithreading to expand the search space. Thus, it can achieve very good results and has the best performance.

5.3. Throughput. Figure 4 shows the effect of the link load on the network throughput. When the link load starts to increase, the network throughput has rapid growth, and the throughput obtained using the SA3CR algorithm has the fastest growth rate and is also the largest. For example, when the link load ranges from 0.8 to 1.6, the growth rate of the throughput obtained using each algorithm slows down, which is mainly due to the limited link capacity, which has begun to become saturated.

Figure 4(a) presents the values of the average throughput produced by SA3CR, DDPG_R, NAF_R, ECMP, and KSP in the OS3E network. When the link load is 1.6, the throughput obtained using the SA3CR algorithm is close to 53 Mbps, which is higher than the throughput obtained using the DDPG_R algorithm by 51 Mbps. The NAF_R algorithm throughput is close to 49 Mbps. The throughput obtained using the KSP and ECMP algorithms is relatively close, approximately 44 Mbps. Compared with the four algorithms, the network throughput obtained using the SA3CR algorithm has increased by 4%, 9%, 17%, and 18%, respectively. Figure 4(b) presents the values of the average throughput produced by SA3CR, DDPG_R, NAF_R, ECMP, and KSP in the NSFNet network. The average throughput

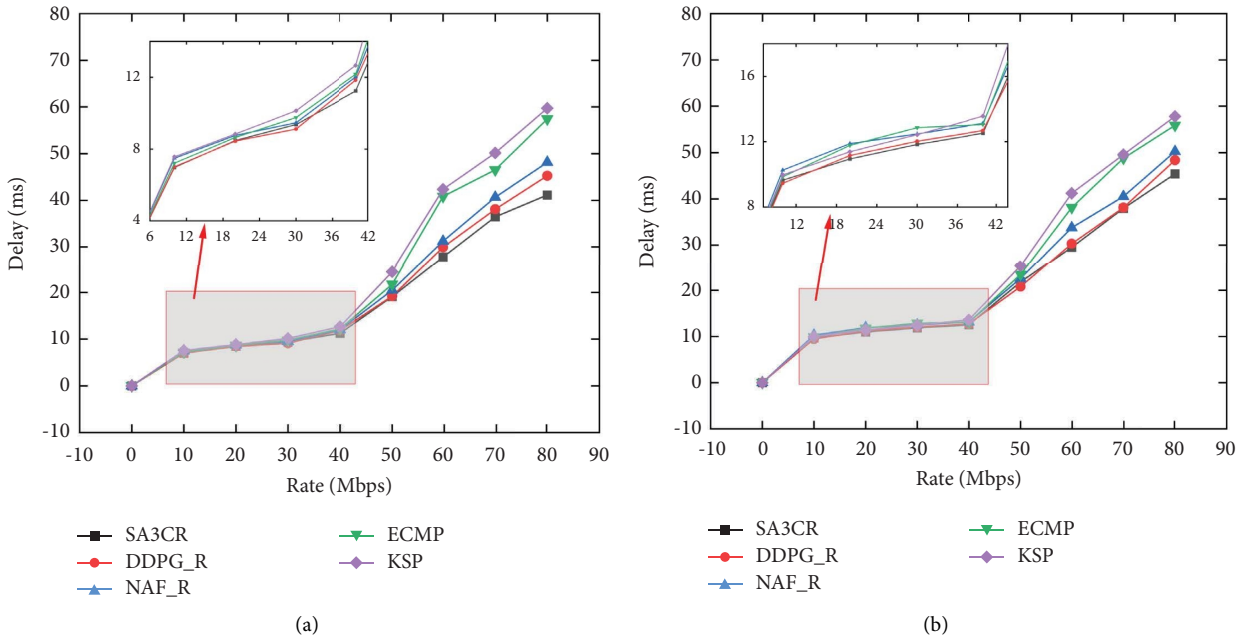

Figure 3: Delay vs. rate: (a) the OS3E network and (b) the NSFNet network.

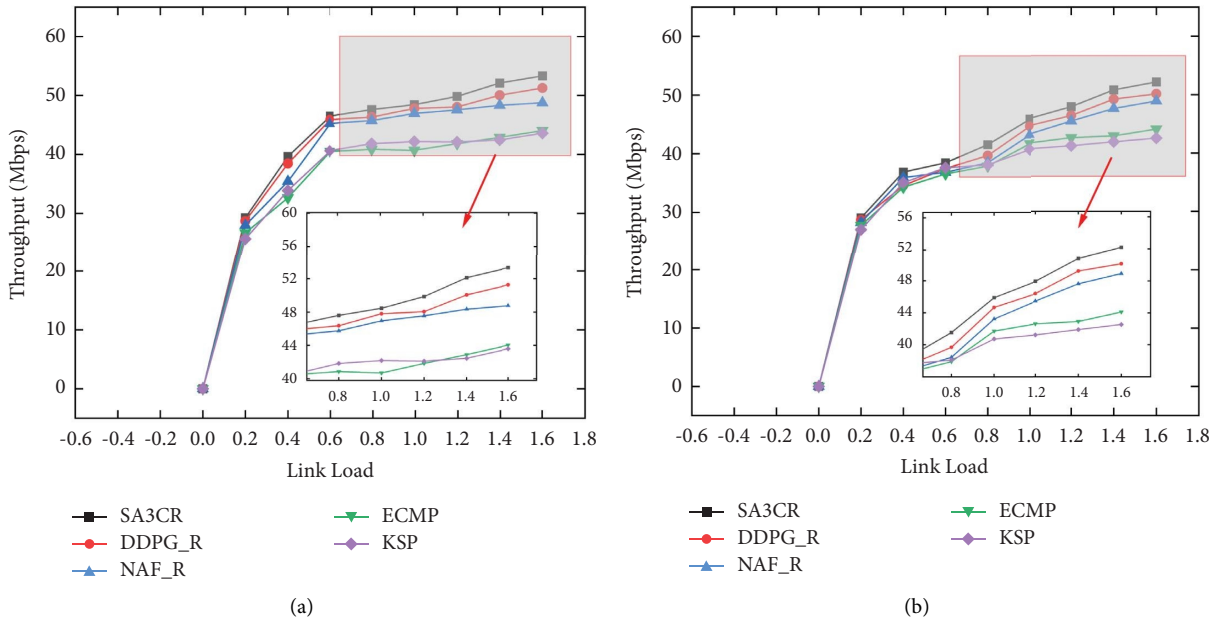

Figure 4: Throughput vs. link load: (a) the OS3E network and (b) the NSFNet network.

produced by the SA3CR algorithm are, on average, 4%, 6%, 15%, and 18% better than those given by DDPG_R, NAF_R, ECMP, and KSP algorithms, respectively.

For the KSP algorithm, there are k candidate paths that may share the same link. The link will be congested, and the transmission capacity of the network will decrease. Without comprehensively considering the bandwidth, delay, and reliability of each path in the current network, the ECMP algorithm adopts an equivalent multipath transmission. When the network state of the link is very different, the bandwidth cannot be used well, resulting in low throughput. The NAF_R algorithm converges slower than the DDPG_R algorithm in the continuous action space, so the throughput is slightly lower. However, the SA3CR algorithm considers the requirements of multiple QoS factors and uses state information of the link to obtain the weight for routing after multithread training to achieve the optimal throughput.

5.4. Packet Loss Rate. The effect of the bandwidth on the packet loss rate is plotted in Figure 5. We can see that when the bandwidth ranges from 0 to 50 Mbps, each algorithm has a small amount of packet loss, and the packet loss rate is relatively close, which is mainly due to the initial packet loss

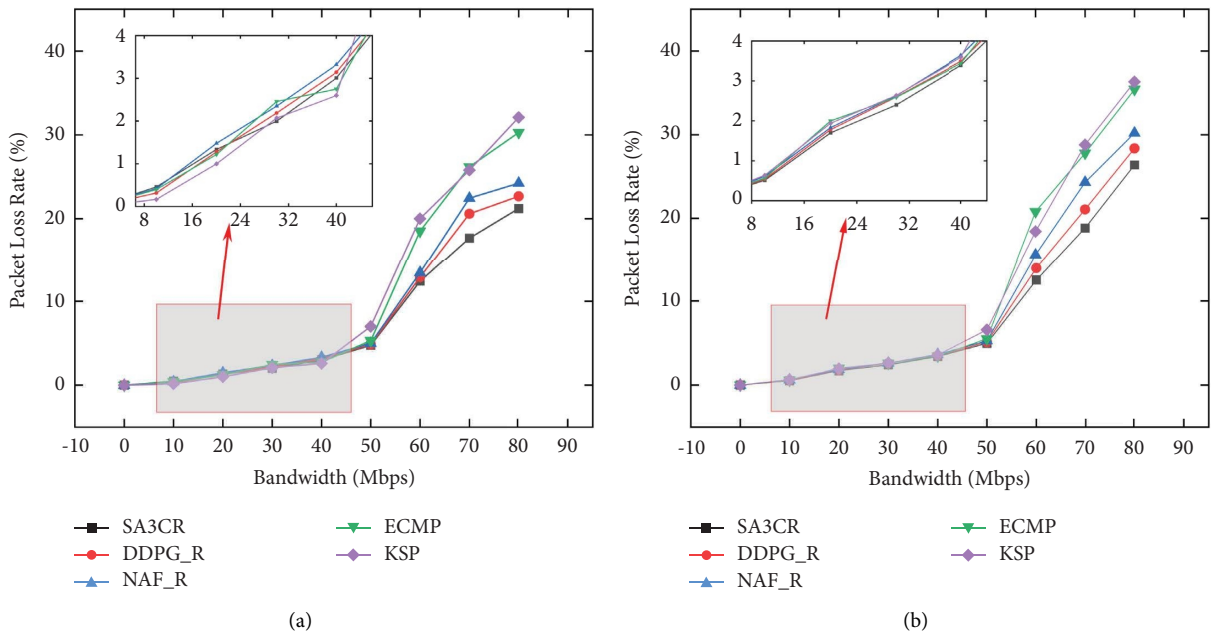

Figure 5: Packet loss rate vs. bandwidth: (a) the OS3E network and (b) the NSFNet network.

rate being set on each link. With the continuous increase in the bandwidth, and when the bandwidth reaches 50 to 80 Mbps and exceeds the maximum bandwidth that the link can withstand, the link congestion is relatively high, causing the packet loss rate to be relatively large.

Figure 5(a) presents the mean packet loss rate given by SA3CR, DDPG_R, NAF_R, ECMP, and KSP in the OS3E network. When the bandwidth reaches 80 Mbps, compared with the other algorithms, the average packet loss rate of the SA3CR algorithm is reduced by 6%, 12%, 30%, and 34%, respectively. The KSP algorithm has the highest packet loss rate, reaching 32%. The packet loss rate of the ECMP algorithm is slightly lower than that of the KSP algorithm, with a packet loss rate of 30%. Figure 5(b) presents the values of the average throughput produced by SA3CR, DDPG_R, NAF_R, ECMP, and KSP in the NSFNet network. The average throughput produced by the SA3CR algorithm are, on average, 7%, 13%, 25%, and 27% lower than those given by DDPG_R, NAF_R, ECMP, and KSP algorithms, respectively.

The KSP algorithm has multiple alternative paths for routing, but it has a higher packet loss rate than RL algorithms such as SA3CR, DDPG_R, and NAF_R. The intelligent optimization algorithm will try to reduce the occurrence of each path sharing a path when generating a path and avoid excessive congestion of the link. The SA3CR algorithm takes the requirements of QoS flows into comprehensive consideration and makes reasonable use of network resources according to the state of the network. It does not need to store sample data of network history in the way of experience pool as the NAF_R algorithm does but randomly extracts data for training to disrupt data correlation. Compared with the DDPG_R algorithm, the biggest advantage of the SA3CR algorithm is that it adopts an asynchronous training method, which greatly improves the convergence ability and training speed of the network, and the network model is superior.

6. Conclusions

In this study, we propose an online QoS-routing framework, namely, DSOQR, and an intelligent QoS-routing algorithm, namely, SA3CR, to meet network QoS requirements of various new applications and services. The framework of DSOQR uses a software-defined paradigm to collect network information in real time, and the SA3CR algorithm uses the DRL model to learn online strategies from the network information collected in real time, generate routing strategies quickly, and achieve fast routing. We also implement the framework of DSOQR and the SA3CR algorithm and verify the performance of the SA3CR algorithm in terms of delay, bandwidth, and packet loss rate. Future research will focus on the problem of controllers receiving multiple requests interfering with each other and how to improve the interpretability of intelligent routing algorithms.

Data Availability

The data used to support the findings of the study can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (nos. 61572191 and 61602171) and the Natural Science Foundation of Hunan Province (nos. 2022JJ30398 and 2022JJ40277).

References

[1] L. Zhang, H. Zhang, Q. Tang et al., "LNTP: LNTP: An End-to-End Online Prediction Model for Network Trafficn end-to-end online prediction model for network traffic," *IEEE Network*, vol. 35, no. 1, pp. 226–233, 2021.

[2] L. Zhang, K. Wang, D. Xuan, and K. Yang, "Optimal task allocation in near-far computing enhanced c-ran for wireless big data processing," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 50–55, 2018.

[3] CAIR, "Cisco Annual Internet Report (2018-2023)," 2022, https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf.

[4] B. Rathi and E. F. Singh, "Performance analysis of distance vector and link state routing protocols," *International Journal of Computer Science Trends and Technology*, vol. 3, no. 4, pp. 23–32, 2015.

[5] K. Wang, K. Yang, H.-H. Chen, and L. Zhang, "Computation diversity in emerging networking paradigms," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 88–94, 2017.

[6] S. Zhu, Z. Sun, Y. Lu, L. Zhang, Y. Wei, and G. Min, "Centralized qos routing using network calculus for sdn-based streaming media networks," *IEEE Access*, vol. 7, pp. 146566–146576, 2019.

[7] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200–218, 2017.

[8] J. Xie, F. R. Yu, T. Huang et al., "Asurvey of machine learning techniques applied to software defined networking (sdn): a survey of machine learning techniques applied to software defined networking (SDN): research issues and challenge ssearch issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2019.

[9] H. Mao, M. Schwarzkopf, H. He, and M. Alizadeh, "Towards safe online reinforcement learning in computer systems," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, MIT Press, Vancouver, Canada, December 2019.

[10] A. Azzouni, R. Boutaba, and G. Pujolle, "Neuroute: predictive dynamic routing for software-defined networks," in *Proceedings of the 2017 13th International Conference on Network and Service Management*, pp. 1–6, IEEE, Tokyo, Japan, January 2017.

[11] S. T. V. Pasca, S. S. Prasad, and K. Kataoka, "AMPF: Application-Aware Multipath Packet Forwarding Using Machine Learning and Sdn," 2016, https://arxiv.org/abs/1606.05743.

[12] S. Sendra, A. Rego, J. Lloret, J. M. Jiménez, and Ó. Romero, "Including artificial intelligence in a routing protocol using software defined networks," in *Proceedings of the 2017 IEEE International Conference on Communications Workshops*, pp. 670–674, IEEE, Paris, France, July 2017.

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

[14] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pp. 1928–1937, PMLR, New York, NY, USA, June 2016.

[15] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 652–673, 1998.

[16] S. S. Tomar, A. Rawat, S. Tokekar, and P. D. Vyavahare, "Investigations on equal cost multi-path feature in dynamic routing protocols in ipv6 networks," in *Proceedings of the 2019 IEEE Conference on Information and Communication Technology*, pp. 1–6, IEEE, Allahabad, India, December 2019.

[17] A. Kumar Singh, N. Kumar, and S Srivastava, "Pso and tlbo based reliable placement of controllers in sdn," *International Journal of Computer Network and Information Security*, vol. 11, no. 2, pp. 36–42, 2019.

[18] X. Hei, J. Zhang, B. Bensaou, and C.-C. Cheung, "Wavelength converter placement in least-load-routing-based optical networks using genetic algorithms," *Journal of Optical Networking*, vol. 3, no. 5, pp. 363–378, 2004.

[19] Mininet, "Mininet," 2022, http://mininet.org/.

[20] Ryu, "Component-based software defined networking framework," 2022, https://ryu-sdn.org/.

[21] P. Tune and M. Roughan, "Spatiotemporal traffic matrix synthesis," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 579–592, Washington, DC, USA, April 2015.

[22] Iperf, "The ultimate speed test tool for TCP, UDP and SCTP," 2022, https://iperf.fr/.