IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 19, 2020, accepted April 30, 2020, date of publication June 1, 2020, date of current version June 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2995511

Deep Q-Learning for Routing Schemes in SDN-Based Data Center Networks

QIONGXIAO FU[1], ENCHANG SUN[1,2], (Senior Member, IEEE), KANG MENG[1], MENG LI[1], AND YANHUA ZHANG[1]

[1]Faculty of Information Technology, Beijing University of Technology, Beijing, China
[2]Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004, USA

Corresponding authors: Enchang Sun and Yanhua Zhang (ecsun@bjut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61901011, Grant 61671029, Grant 61601330, and Grant 61571021, in part by the Foundation for University Key Teachers from the Ministry of Education of China under Grant 201806545031, and in part by the Foundation of Beijing Municipal Commission of Education under Grant KM201610005004 and Grant KM201610005007.

ABSTRACT In order to adapt to the rapid development of cloud computing, big data, and other technologies, the combination of data center networks and SDN is proposed to make network management more convenient and flexible. With this advantage, routing strategies have been extensively studied by researchers. However, the strategies in the controller mainly rely on manual design, the optimal solutions are difficult to be obtained in the dynamic network environment. So the strategies based on artificial intelligence (AI) are being considered. This paper proposes a novel routing strategy based on deep Q-learning (DQL) to generate optimal routing paths autonomously for SDN-based data center networks. To satisfy the different demands of mice-flows and elephant-flows in data center networks, deep Q networks are trained for them respectively to achieve low latency and low packet loss rate for mice-flows as well as high throughput and low packet loss rate for elephant-flows. Furthermore, with the consideration of the distribution of traffic and the limitated resources of data center networks and SDN, we choose port rate and flow table utilization to describe the network state. Simulation results show that compared with Equal-Cost Multipath (ECMP) routing and Selective Randomized Load Balancing (SRL)+FlowFit, the proposed routing scheme can reduce both the average delay of mice-flows and average packet loss rate, while increase the average throughput of elephant-flows.

INDEX TERMS Data center, SDN, flow types, deep Q-learning.

I. INTRODUCTION

With the rapid growth in cloud computing, big data, and other technologies, the scale of data center networks is expanding continuously [1]. The traditional networks cannot meet the requirements of the existing data center networks due to the difficulties in network management and deployment. The emergence of SDN indicates the way to solve the above problem. It's a novel networking architecture which separates the control plane from data plane of the forwarding device. Data center networks can benefit from the centralized control of SDN to make intelligent dynamic decisions. In data center networks, routing is an important research point and has been researched for a long time. With the advantage of SDN, routing strategies can be deployed conveniently and flexibly based on the global view of network. To further achieve effective routing in data center networks, flows are categorized into two types: elephant-flows and mice-flows. Elephant-flows carry large amounts of data and last for a long time, while mice-flows do the opposite. In the light of traffic characteristics, literature [2]–[11] are focus on the routing approaches in data center networks based on SDN. However, all of these routing strategies need to be designed manually. In the face of the changing network environment, they are difficult to achieve optimal solution. The rise of AI brings a new idea for us to dispose of routing problem. Reinforcement learning (RL) represented by Q-learning (QL) [12] is an important branch of AI. It seeks the optimal strategy through continuous "trial and error" interactions with the environment. Nevertheless, the diversity of network state may cause the storage space required for Q-table to be too large to use normally. DQL [13] which adopts neural network to fit Q-table is a good solution for the above problem. NetworkAI [14] proposes an intelligent architecture for

The associate editor coordinating the review of this article and approving it for publication was Dr. M. Sabu Thampi.

self-learning control strategies in SDN networks. It combines SDN with deep reinforcement learning (DRL) and presents a simple example that solving QoS routing problem with DQL. However, the paper focuses on the design of architecture and the specific design of routing scheme is not explained.

In this paper, we propose a routing strategy based on DQL for data center networks based on SDN. We build two DQNs to make routing decisions intelligently. One for elephant-flows to achieve low packet loss rate and high throughput, the other for mice-flows to achieve low packet loss rate and low latency. In this way, the approximate optimal routing strategy can be obtained. To summarize, we make the following contributions in this paper.

- An intelligent network architecture is built for routing in data center. According to the traffic characteristics of data center network, it will dynamically generate optimal routing strategies for elephant-flows and mice-flows, respectively.
- Specific design of DQL algorithm is presented, including the design of state space, action space and reward function. To better describe the state of the network, we combine the port rate and flow table occupancy rate in switches for the purpose, which reflect the distribution of traffic in the network and the utilization of network resources, including link bandwidth resources and flow table resources.
- The effectiveness of the proposed routing algorithm is verified by simulations. It is illustrated that the performance of mice-flows can be improved in the aspects of packet loss rate and delay, while elephant-flows behave better in terms of packet loss rate and throughput.

II. RELATED WORKS

In data center networks, routing has been studied for a long time as an crucial research orientation. With the rise of SDN in recent years, lots of routing strategies based on SDN have been proposed, so that fine-grained flow control is achieved. Load balance routing is widely researched in data centers to ensure the sustainability of the network, it tries to guarantee the transmission quality of the flows while reserve some space for possible subsequent flows. Literature [2]–[4] focus on balancing load for elephant-flows. [2] selects the path that would accommodate the flow for routing and [3] chooses the least crowded path. [4] splits and sends elephant-flows through multiple paths based on the ratios which are dynamically computed. The researchers in [5]–[7] design rerouting schemes to further balance the network load. They periodically determine whether the network load is balanced by setting the parameter such as load balance degree. When the parameter exceeds the threshold, flow scheduling or flow splitting is triggered. The above shemes can effectively reduce the packet loss rate and increase the throughput of elephant-flows. It should be noted that the load balance mentioned above refers to link load balance, another called flow table load balance has been proposed recently with the consideration of the limited flow table capacity in SDNs. Research [8] introduces flow table load balance for mice-flows which account for the majority of traffic in data centers to prevent the packet loss caused by flow table overflow. In addition, considering the low latency characteristic of mice-flows, the following routing schemes are put forward. Literature [9] picks the path with the lowest delay for mice-flows and [10] assigns dedicated low-latency paths for them. [11] decreases the delay by reducing the number of flow rules installed to transmit mice-flows. However, all of these solutions are based on manual design which is non-intelligent, it implies that when similar traffic patterns happen, the same paths will be selected even the routing strategies have resulted in poor network performance. So, they lack the ability to learn from previous experiences[15]. Facing the problem, AI as a popular tool provides solution. Literature [15], [16] apply deep learning to avoid congestion in different network scenario. A Convolutional Neural Network (CNN) is trained for each path combination to generate the result that any chosen path is congested or not according to the input traffic pattern. [12] realizes QoS adaptive routing based on QL which is a typical reinforcement learning algorithm, QoS-aware reward function is put forward to direct the learning process of optimal routing. Nevertheless, because of the fine-grained flow control and the changing network environment, QL calls for huge storage space to maintain Q-table. To overcome the defect, DQL that combines deep learning with QL is proposed. In [14], DRL is applied to solve large scale network control problems, and a simple example that using DQL for QOS routing is present. It lays stress on the proposed architecture which introduces DRL to SDN, but the design of the routing sheme is not account for. DROM [17] and TIDE [18] are rencent DRL mechanisms for routing optimization. However, their actions depend on link weights modification, the optimal routing path can only be obtained indirectly by shortest path algorithms.

III. SYSTEM ARCHITECTURE

For the purpose of combining SDN-based data center networks with DQL, we introduce AI agent contained in AI plane to traditional SDN architecture, realizing intelligent routing decisions. We present the proposed system architecture in Figure 1. It contains three planes: data plane, control plane and AI plane. The functions of these three planes are described in detail below.

A. DATA PLANE

The data plane is mainly composed of switches which focus on packet forwarding, and all these switches support OpenFlow protocol. In particular, Fat-Tree [20] as a typical data center network topology is adopted to be data plane in this paper, which is shown in Figure 2. There are multiple paths between the source and destination nodes in Fat-Tree topology, providing data centers with high bandwidth and good fault tolerance.

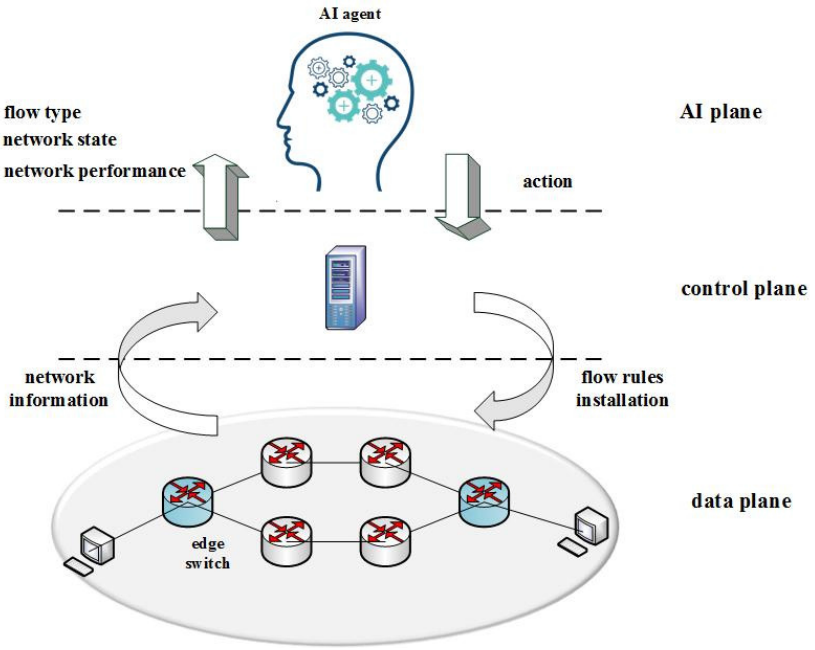

FIGURE 1. System architecture.

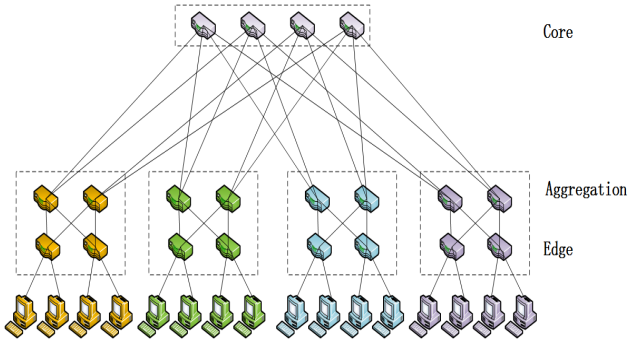

FIGURE 2. Fat-Tree topology.

B. CONTROL PLANE

The control plane interacts with the data plane through the south interface protocol(OpenFlow). The controller obtains network topology by using Link Layer Discovery Protocol, and it periodically sends state query messages to each switch to acquire the state information of switches, such as flow table status and port status. When a new flow arrives at the network, the controller first calculates the flow rate and estimates the flow type based on the flow statistics. If the flow rate exceeds thethreshold(According to previous studies, this value was often set to 5% of the link capacity), we regard the flow as elephant-flow. Otherwise, we regard the flow as mice-flow.

In addition, the network performance evaluation(e.g., packet loss rate, delay, throughput) is also collected. All of these information is prepared for routing strategies formulation. The controller converts the strategies to flow rules and installs them on the corresponding switches.

C. AI PLANE

The AI plane is the core part of the system. AI agent can learn the optimal routing scheme autonomously from previous experience. Intelligent routing policies are dynamically generated to improve the performance of mice-flows and elephant-flows. The plane gains the flow type, network state information and network performance evaluation from the control plane through the north interface. For different flow types, the AI agent learns the different optimal routing strategy with network state information and network performance evaluation. In detail, two DQNs are designed for mice-flows and elephant-flows, respectively. Our goal is to achieve low latency and low packet loss for mice-flows, while low packet loss and high throughput for elephant-flows. Through QL, the agent obtains the optimal routing strategy by interacting with the environment. Furthermore, deep neural network is used to approximate the huge policy table, thus we can obtain the optimal routing path according to the input network state and flow type quickly.

IV. PROBLEM FORMULATION

We model the data center network as a directed graph $G = \{V, E\}$, where V represents the set of all switch nodes and E denotes the set of links between switches. The flow table capacity of each switch is R_m, and the capacity of each link is C_m. During the period from t_1 to t_n, $n \to +\infty$, we assume that the set of all mice-flows and all elephant-flows are $F_{mice} = \{f_w \,|w \in [1, p]\}$ and $F_{elephant} = \{f_v \,|v \in [1, q]\}$, where p and q are the number of mice-flows and elephant-flows, $p \to +\infty$, $q \to +\infty$. Furthermore, we set the existing flows in the network at time t_i as

$F_{t_i} = \left\{ f_{t_i}^j = (s_{t_i}^j, d_{t_i}^j, r_{t_i}^j) \,|t_i \in [t_1, t_n), j \in [1, m] \right\}$. $f_{t_i}^j$ is the j-th flow at t_i, while $(s_{t_i}^j, d_{t_i}^j, r_{t_i}^j)$ stands for the source switch, destination switch and bandwidth requirement of this flow, m is the total number of flows in the network at the moment. $f_{t_i}^j(u, v)$ describes the relationship between $f_{t_i}^j$ and link (u, v), $u \in V$ and $v \in V_{N(u)}$. $V_{N(u)}$ is the set of neighbor switches of u. If $f_{t_i}^j$ is on link (u, v), $f_{t_i}^j(u, v) = 1$. If not, $f_{t_i}^j(u, v) = 0$. We use $d(x)$, $p(x)$, $t(x)$ respectively to represent the average delay, average packet loss rate and average throughput of the flow x. So, we can calculate the following indicators:

$$D_{mice} = \lim_{p \to +\infty} \frac{\sum_{w \in [1,p]} d(f_w)}{p} \tag{1}$$

$$P_{mice} = \lim_{p \to +\infty} \frac{\sum_{w \in [1,p]} p(f_w)}{p} \tag{2}$$

$$P_{elephant} = \lim_{q \to +\infty} \frac{\sum_{v \in [1,q]} p(f'_v)}{q} \tag{3}$$

$$T_{elephant} = \lim_{q \to +\infty} \frac{\sum_{v \in [1,q]} t(f'_v)}{q} \tag{4}$$

D_{mice}, P_{mice} are the average delay and average packet loss rate of mice-flows during t_1 to t_n, while $P_{elephant}$, $T_{elephant}$ are the average packet loss rate and average throughput of elephant-flows during t_1 to t_n.

According to traffic characteristics of data center networks, we set our target of the routing problem as follows:

$$\min D_{mice} \tag{5}$$

$$\min P_{mice} \tag{6}$$

$$\min P_{elephant} \tag{7}$$

$$\max T_{elephant} \tag{8}$$

$$\text{subject to: } \sum_{v \in V_N(u)} [f_{t_i}^j(u, v) - f_{t_i}^j(v, u)] = \begin{cases} 1, & u = s_{t_i}^j \\ -1, & u = d_{t_i}^j, \\ 0, & u \neq s_{t_i}^j, d_{t_i}^j \end{cases} \quad \forall u \in V \tag{9}$$

$$\sum_{j \in [1,m]} r_{t_i}^j f_{t_i}^j(u, v) \leq C_m, \quad \forall (u, v) \in E \tag{10}$$

$$\sum_{v \in V_N(u)} f_{t_i}^j(u, v) \leq R_m, \quad \forall u \in V \tag{11}$$

Constraint(9) is the classical flow conservation constraint, it ensures that the ingress flows are equal to the egress flows for each switch. Constraint(10) and constraint(11) represent capacity limits for links and flow tables, respectively. Furthermore, the link capacity C_m can be also expressed as the minimum of the allowable port rates at both ends of the link. In this paper, the rate of each port is limited to a same value, so the link capacity is equal to port capacity here.

The routing process is actually a network state transition process. Here, in order to better express the network state, for the network state at a certain moment, we select the instantaneous state of this moment and recent n-1 moments to represent it. Based on this setting, we approximate the routing process as a Markov Decision Process (MDP), and the relevant parameters are designed as follows:

A. STATE SPACE

Two state objects are considered here: flow table state and port state. We call these two states as network state collectively. As shown in Figure 3, we treat the state of the network as an image, and treat different network features as different pixel channels. Here, channels respectively represent the flow table utilization rate and the port rate of each switch at current and previous moments. Therefore, the state space can be expressed as follow:

$$State = \{s = [FT_{sw_i,t_j}, PS_{p_k,sw_i,t_j}] | i \in [1, n], j \in [1, m], k \in [1, z]\}$$

where n, m and z are respectively the number of switches, moments and ports of a single switch. FT_{sw_i,t_j} represents the flow table utilization rate of switch i at the moment t_j, it is in the range of 0 to 1. Meanwhile, PS_{p_k,sw_i,t_j} represents the port rate of port k in switch i at the moment t_j, it doesn't exceed $ps_{\max}$. $ps_{\max}$ is a fixed value, it's the maximum amount of traffic that a port can pass per second.

In particular, FT_{sw_i,t_j} is a finite set with v elements (v is the flow table capacity), while PS_{p_k,sw_i,t_j} is continuous. To reduce computational complexity, we divide it into several levels to achieve the purpose of decentralization. The number of levels should not be too much or too little. It is supposed to ensure state differentiation while reducing computational complexity as far as possible. Here, we select ten levels in our scheme. Then PS_{p_k,sw_i,t_j} turns into a finite set with ten elements.

B. ACTION SPACE

The action space can be described as follows:

$$Action = \{a_{p_1}, a_{p_2}, \ldots a_{p_k}, \ldots a_{p_N}\}$$

where p_1 to p_N are all paths in the network, $a_{p_k} \in \{0, 1\}$.

If $a_{p_k} = 1$, the current flow is assigned to path k. If $a_{p_k} = 0$, the result is the opposite. In particular, we consider that flows are indivisible, so each flow can only be assigned to one path. $Action$ satisfies the following equation: $Action * One = 1$. Where, One is an N-dimensional column vector with all 1's in it.

Especially, for a certain flow (ip_src, ip_dst) under a state, the executable actions are in the alternative path set between the source and destination servers of the flow. Taking the Fat-Tree topology with k parameter as an example, the maximum number of actions is $(k/2)^2$.

C. REWARD FUNCTION

Considering the characters of elephant-flow and mice-flow, we formulate different reward functions for these two types of flow. For elephant-flow, the goal is to minimize packet

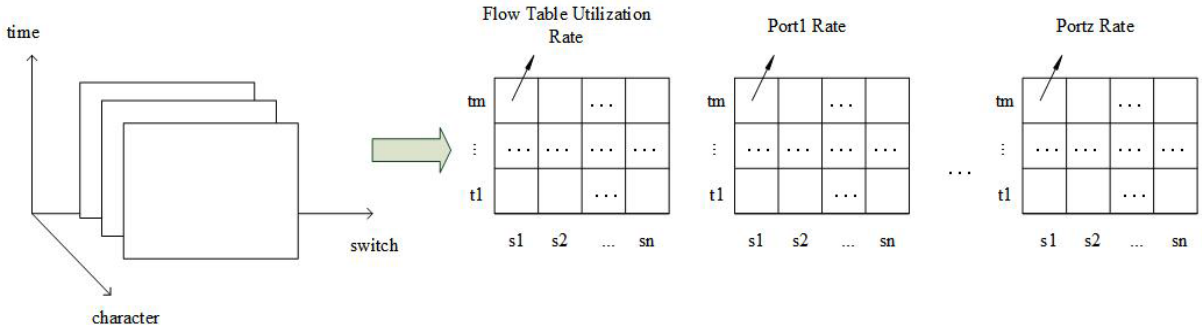

FIGURE 3. Network state.

loss rate and maximize throughput. And for mice-flow, the goal is to minimize packet loss rate and latency. Therefore, the reward functions are set up as follows:

For elephant-flows,

$$R_{elephant} = \alpha * (1 - PLR) + \beta * TP \tag{12}$$

where PLR represents the average packet loss rate of elephant-flows in the network, TP is the average throughput of elephant-flows after processing(Average throughput divided by the maximum receiving rate at the receiving end). This is done for bringing the two indicators into the same order of magnitude(0-1) to facilitate comprehensive evaluation. α and β are the weights of the two indicators, respectively, indicating the importance of the indicators. They satisfy that $\alpha + \beta = 1$.

For mice-flows,

$$R_{mice} = \lambda * (1 - PLR2) + \mu * (1 - DL) \tag{13}$$

where $PLR2$ represents the average packet loss rate of mice-flows, and DL is the normalized average delay of mice-flows. Both of these indicators are between 0 and 1. λ and μ are the weights of the two indicators respectively and $\lambda + \mu = 1$.

V. ALGORITHM DESIGN

RL is a tool to solve the MDP problem. QL is a classical RL algorithm, which is based on value. It sacrifices some of its current earnings for its long-term earnings. Q stands for $Q(s, a)$, it is the expected benefit of taking action $a(a \in A)$ at a certain state $s(s \in S)$. The main idea of the algorithm is to build a Q-table to store Q, and then select the action that can obtain a large profit according to the Q value. However, the state space is too large to build a Q-table in finite memory in our scenario. To address the problem, DQL is adopted here.

A. DQL ALGORITHM

In this section, we introduce DQN in detail and show our improvement to DQN for the routing problem.

DQL is an algorithm that combines deep neural network and QL. Deep neural network has good generalization ability and can approximate almost any nonlinear function. Therefore, on the basis of QL algorithm, the deep neural network is used to establish the mapping relationship between state and action, so as to realize the accelerated solution of the problem and solve the dimension disaster problem caused by the large scale of system state.

QL updates the value function as follow:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \tag{14}$$

where $\alpha \in (0, 1]$ is the learning rate. $Q(s, a)$ and $Q(s', a')$ are the Q values of current moment and next moment, respectively.

Instead of searching Q values in Q-table, DQL uses deep neural network such as CNN to estimate $Q(s, a)$, i.e., $Q(s, a; \theta) \approx Q(s, a)$, where θ represents the set of weights and biases which are the parameters of neural network. The network is trained by minimizing the loss, the loss function can be expressed as follow:

$$L(\theta) = E[(r + \gamma \max_{a'} Q(s', a'; \theta') - Q(s, a; \theta))^2] \tag{15}$$

$r + \gamma \max_{a'} Q(s', a'; \theta')$ is the target Q value calculated by QL. While, $Q(s, a; \theta)$ is the Q value estimated. Our goal is to get the estimate Q close to the target Q.

To obtain the two types of Q value, we adopt two independent neural networks with the same structure: evaluated Q-network and target Q-network. The former generates the estimate Q according to the current state. It changes parameters in each episode to decrease the loss. While, the latter outputs Q corresponded to the next state, preparing for the calculation of the target Q. It updates parameters with evaluated Q-network every some steps.

To provide training samples, DQN has a reply memory which stores historical experiences. Experiences are selected randomly from the reply memory to train the neural network. In this way, the problem of time-correlation of samples is solved and the stability of training is improved.

We summarize the workflow of DQL in Figure 4.

In particular, in the routing scenario, the information about the arrival flow of next moment is unknown, including the flow type as well as the source and destination IP address of the flow. And the available paths for the flow are uncertain. We let $Q1$ and $Q2$ be the action value function for mice-flows and elephant-flows, respectively. We set the alternative path set as $A\_set = \{a\_set_{i,j} | i \in [1, m], j \in [1, m], i \neq j\}$,

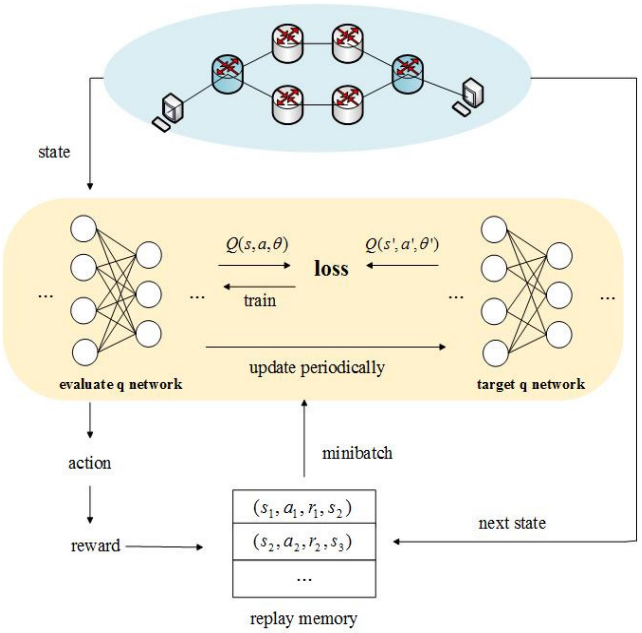

FIGURE 4. The workflow of DQL.

where m is the number of edge switches, $a\_set_{i,j}$ represent the available paths between the i-th and j-th edge switches. A_set is made up of $m(m-1)$ sets ($a\_set_{i,j}$), and the optional path set for the flow may be any $a\_set_{i,j}$ from A_set. Based on the above settings and considerations, we improve the target Q value to fit our scenario. The new target Q value can be expressed as $r + \gamma[\frac{1}{m(m-1)}(p\sum_{i,j}\max_{a'\in a\_set_{i,j}} Q_1(s', a'; \theta_1') + q\sum_{i,j}\max_{a'\in a\_set_{i,j}} Q_2(s', a'; \theta_2'))]$, where p, q are weight factors, they are configured to the proportion of mice-flows and elephant-flows in the network, respectively. In this way, we can solve the problem that the information about the arrival flow of next moment is unknown, so that we can better assess the next state.

B. ROUTING ALGORITHM BASED ON DQL

We train different DQNs for elephant-flow and mice-flow. With the advantage of QL, the agent can learn the optimal routing path of each state by trial-and-error while guarantee the long-term benefits, the result will be more accurate than the manually designed routing scheme. CNN is used to fit the policy table–Q table, so that the agent can quickly obtain the optimal path, saving the memory space and lookup time of policy table.

When a new flow arrives at the network, the controller first determines the type of the flow and obtains all paths between the source and destination servers of the flow from the precomputed alternative path set A_set. The selected paths constitutes the optional action set a_set for the flow. Get the current network status s and put it into CNN which is the evaluated q-network with action value function Q and parameter θ (CNN_1, Q_1, θ_1 for mice-flows and CNN_2, Q_2, θ_2 for elephant-flows), Q values corresponding to all paths in A_set in the current state can be obtained. Then, select a path randomly or select the path with the maximum Q value from a_set of the flow as action a, and install flow rules for switches on the path. Finally, calculate reward r and update the network state s_{t+1}, store (s, a, r, s_{t+1}) for CNNs training. In training, the target q-network CNN' with action value function $\overline{Q}$ and parameter $\overline{\theta}$ is adopted to work with CNN (CNN_1', $\overline{Q}_1$, $\overline{\theta}_1$ for mice-flows and CNN_2', $\overline{Q}_2$, $\overline{\theta}_2$ for elephant-flows). When the CNNs have been trained well, we can generate optimal routing strategies for flows from the flow type and current network state.

We present the learning phase and routing phase of routing algorithm based on DQL in Algorithm 1 and Algorithm 2. Among them, Algorithm 1 is executed at the initial stage of network operation. It constantly updates routing policies through learning from historical experience, and it has high computational complexity due to the large state space and action space. When the learning phase is complete, the trained CNNs are adopted in Algorithm 2. Routing strategies have been obtained in advance through Algorithm 1, they can be generated directly from CNNs without extra computation, so that the intelligent routing is realized. In particular, because elephant-flows and mice-flows have similar learning processes, Algorithm 1 is the general algorithm of the two types of flows.

Algorithm 1 Learning Phase

1: Obtain the alternative path set of the flow (a_set) from A_set
2: **while** new flow **do**
3: Select a_t randomly
4: Otherwise input s_t into CNN to gain the $Q(s_t, a; \theta)$, select $a_t = \arg\max_a Q(s_t, a; \theta)$
5: Calculate the reward r_t and monitor the next network state s_{t+1}
6: Store (s_t, a_t, r_t, s_{t+1}) in replay memory D
7: Select random m minibatch from D
8: Minimize the loss function $[r_i + \gamma[\frac{1}{m(m-1)}(p\sum_{i,j}\max_{a'\in a\_set_{i,j}} \overline{Q}_1(s_{i+1}, a'; \overline{\theta_1}) + q\sum_{i,j}\max_{a'\in a\_set_{i,j}} \overline{Q}_2(s_{i+1}, a'; \overline{\theta_2})) - Q(s_i, a_i; \theta)]^2$ and update the parameter θ
9: Set $\overline{\theta}_1(\overline{\theta}_2) = \theta$ every L steps
10: **end while**

VI. SIMULATION

A. EXPERIMENT SETUP

We establish the network topology using Mininet [19] emulator, which provides virtual network elements to easily create a network that supports OpenFlow protocol. Ryu [20] which is an open source SDN controller is chosen to operate the network. Iperf is selected to be responsible for generating the traffic. In addition, we select Fat-Tree, a typical data center network topology to be our topology. For the convenience of

Algorithm 2 Routing Phase

while new flow **do**
  Determine the type of flow
  Obtain the alternative path set of the flow (*a_set*) from *A_set*
  if mice-flow **then**
    Input s_t into CNN_1 to gain the $Q(s_t, a; \theta)$, select $a_t = \arg\max_a Q(s_t, a; \theta)$
    Install the flow rules according to a_t
  end if
  if elephant flow **then**
    Similar to the process above, change CNN_1 into CNN_2
  end if
end while

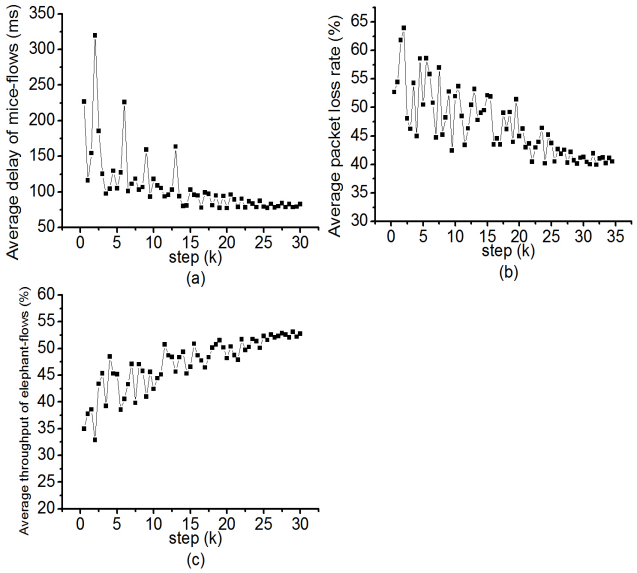

FIGURE 5. The changes of the three indicators during the training process.

simulation, we use a Fat-Tree [21] topology with a parameter of 4 as an example, which contains 20 switches and 16 servers. The link capacity and flow table capacity are set to 100 and 20, respectively. To emulate the traffic in data center networks, we set 20 percent of the flows to be mice-flows, and the others to be elephant-flows. The threshold of the two types of flows is 0.5 percent of the link bandwidth. And the duration is 5s for mice-flows, while 30s for elephant-flows. The related parameters of reward function are set as follows: $\alpha = \beta = \lambda = \mu = 0.5$.

B. PERFORMANCE AND RESULTS

In order to demonstrate the effectiveness of the proposed scheme, experiments were performed under different network loads (0.1, 0.5, 0.9). We take 0.9 for example, Figure 5 shows the training process of the AI agent under this load. From 5(a) and 5(b), we can find that the average delay of mice-flows and average packet loss rate display a decreasing trend with the increase of training steps. While the average throughput of elephant-flows shows an upward trend in this interval in 5(c). All indicators will level off after a certain number of steps. We record the convergence steps of each standard under each network load in Table 1. In particular, when the load increases, the indicators tend to stabilize in a shorter period of time. It should be noted that the average packet loss rate we measure in this paper is for all traffic in the network, including mice-flows and elephant-flows.

TABLE 1. Convergence steps (k).

Load \ Standard	Delay	Packet loss rate	Throughput
0.1	140	170	150
0.5	65	75	65
0.9	25	30	25

TABLE 2. Average delay comparison of mice-flows (ms).

Load \ Scheme	ECMP	SRL+FlowFit	Our Scheme
0.1	8.646	7.825	3.791
0.5	33.107	14.929	11.445
0.9	146.213	90.479	82.114

TABLE 3. Average packet loss rate comparison of all flows (%).

Load \ Scheme	ECMP	SRL+FlowFit	Our Scheme
0.1	27.756	28.421	18.454
0.5	44.873	36.841	28.354
0.9	56.840	49.344	40.234

TABLE 4. Average throughout comparison of elephant-flows (%).

Load \ Scheme	ECMP	SRL+FlowFit	Our Scheme
0.1	67.28	69.51	71.12
0.5	41.16	48.56	67.65
0.9	38.38	41.60	52.72

We compare our scheme with two methods. One is the classic data center network routing algorithm – ECMP [22], which adopts polling to allocate flows, not considering the network status. The other is SRL+FlowFit [23]. As a routing initialization algorithm, SRL randomly selects two equivalent shortest paths, and the path with the least load will be the inital path. Furthermore, FlowFit periodically monitors the state of the network and reassigns flows to optimal links. As shown in Table 2 to Table 4, the proposed scheme reduces the delay of mice-flows by an average of 55.08% and 28.16% under the network load of 0.1 to 0.9, compared with ECMP and SRL+FlowFit. The average packet loss rate is 33.17% and 25.5% lower than that of ECMP and SRL+FlowFit. Meanwhile, the average throughput of elephant-flows is 35.8% and 22.68% higher than that of the other methods. For the sake of clarity, we explain the calculation method of the above results here. Taking Table 2 as an example, we calculate the delay reduction of mice-flows of the proposed scheme compared with other schemes under each load, and then calculate the average value.

In addition, we obtain the path computation time of the three methods on our experimental platform, they are 2.27e-4s, 4.14e-4s, 2.53e-3s for ECMP, SRL+FlowFit and our scheme. The computation time of our proposed scheme is higher than that of the others because of the dot product operations in neural networks. We can further reduce the time by improving the neural network structure in the future work.

Based on the above results, we can infer that the scheme we proposed is able to learn the routing strategy through training, and the trained network could provide optimized routing strategy to achieve network performance improvement. With the help of the two DQNs, low delay and low packet loss of mice-flows are realized, while the high throughput and low packet loss of elephant-flows are also guaranteed. Albeit complicated in computation, the improvement of network performance is obvious.

VII. CONCLUSION

In this paper, we focus on solving routing problems in SDN-based data center networks. DQL is employed to achieve the optimal routing. In the learning phase, we constantly adjust our routing strategies through trial and error, and train CNNs to generate the optimal paths. It spends a lot of time and computing resources. For the subsequent routing phase, we can obtain the optimal routing strategies according to the trained CNNs accurately and quickly without extra calculations. Aiming at the two types of flows in data center networks, elephant-flows and mice-flows, two DQNs are built to train and generate the corresponding routing strategy respectively. Meanwhile, the flow table utilization and port rate are both taken into account to describe the network state in the scheme. We have successfully verified the effectiveness of the proposed mechanism in a simulated data center network. Simulation results show that, the proposed routing scheme can not only provide optimized routing strategy intelligently, but also improve the network performance. In the future, we will improve the neural network structure to reduce the path computation time. Furthermore, the routing problem will be researched in a more complex scenario, where multiple flows arrive at the network at the same interval.

REFERENCES

[1] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and operations," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 640–656, 1st Quart., 2017.
[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. USENIX NSDI*, San Jose, CA, USA, 2010, pp. 281–296.
[3] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1629–1637.
[4] J. Liu, J. Li, G. Shou, Y. Hu, Z. Guo, and W. Dai, "SDN based load balancing mechanism for elephant flow in data center networks," in *Proc. Int. Symp. Wireless Pers. Multimedia Commun. (WPMC)*, Sep. 2014, pp. 486–490.
[5] H. Long, Y. Shen, M. Guo, and F. Tang, "LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks," in *Proc. IEEE 27th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2013, pp. 290–297.
[6] G. Xiao, W. Wenjun, Z. Jiaming, F. Chao, and Z. Yanhua, "An OpenFlow based dynamic traffic scheduling strategy for load balancing," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Dec. 2017, pp. 531–535.
[7] Y.-C. Wang and S.-Y. You, "An efficient route management framework for load balance and overhead reduction in SDN-based data center networks," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1422–1434, Dec. 2018.
[8] Z. Guo, Y. Xu, R. Liu, A. Gushchin, K.-Y. Chen, A. Walid, and H. J. Chao, "Balancing flow table occupancy and link utilization in software-defined networks," *Future Gener. Comput. Syst.*, vol. 89, pp. 213–223, Dec. 2018.
[9] C. Wang, G. Zhang, H. Chen, and H. Xu, "An ACO-based elephant and mice flow scheduling system in SDN," in *Proc. IEEE 2nd Int. Conf. Big Data Anal. (ICBDA)*, Mar. 2017, pp. 859–863.
[10] W. Wang, Y. Sun, K. Zheng, M. A. Kaafar, D. Li, and Z. Li, "Freeway: Adaptively isolating the elephant and mice flows on different transmission paths," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols*, Oct. 2014, pp. 362–367.
[11] F. Amezquita-Suarez, F. Estrada-Solano, N. L. S. da Fonseca, and O. M. C. Rendon, "An efficient mice flow routing algorithm for data centers based on software-defined networking," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
[12] J. Ho, D. W. Engels, and S. E. Sarma, "HiQ: A hierarchical Q-Learning algorithm to solve the reader collision problem," in *Proc. Int. Symp. Appl. Internet Workshops (SAINTW)*, 2006, pp. 88–91.
[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[14] H. Yao, T. Mai, X. Xu, P. Zhang, M. Li, and Y. Liu, "NetworkAI: An intelligent network architecture for self-learning control strategies in software defined networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4319–4327, Dec. 2018.
[15] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Trans. Emerg. Topics Comput.*, early access, Feb. 14, 2019, doi: 10.1109/TETC.2019.2899407.
[16] B. Mao, F. Tang, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "A novel non-supervised Deep-Learning-Based network traffic control method for software defined wireless networks," *IEEE Wireless Commun.*, vol. 25, no. 4, pp. 74–81, Aug. 2018.
[17] 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.
[18] P. Sun, Y. Hu, J. Lan, L. Tian, and M. Chen, "TIDE: Time-relevant deep reinforcement learning for routing optimization," *Future Gener. Comput. Syst.*, vol. 99, pp. 401–409, Oct. 2019.
[19] *Mininet*. Accessed: Aug. 28, 2018. [Online]. Available: http://mininet.org/
[20] *Ryu*. Accessed: May 23, 2020. [Online]. Available: http://ryu.readthedocs.io/en/latest/
[21] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM Conf. Data Commun. SIGCOMM*, Aug. 2008, pp. 95–104.
[22] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with Equal-Cost-multipath: An algorithmic perspective," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2014, pp. 1590–1598.
[23] W. Sehery and T. Charles Clancy, "Load balancing in data center networks with folded-clos architectures," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, Apr. 2015, pp. 1–6.

QIONGXIAO FU received the B.E. degree from the Faculty of Information Technology, Beijing University of Technology (BJUT), Beijing, China, in July 2017, where she is currently pursuing the M.S. degree. Her research interests include software defined networks and deep reinforcement learning.

ENCHANG SUN (Senior Member, IEEE) received the Ph.D. degree in information and telecommunications engineering from Xidian University, Xi'an, China, in 2008. Since 2008, he has been with the School of Electronic Information and Control Engineering (now Faculty of Information Engineering), Beijing University of Technology, Beijing, China, where he is currently an Associate Professor. He visited the University of Warwick, Carleton University, and the University of Houston, in 2012, 2015, and 2019, respectively. He is a Senior Member with the Chinese Institute of Electronics (CIE) and China Institute of Communications (CIC), and an MIET Member of the Institution of Engineering and Technology (IET). He has published more than 50 articles in famous journals and academic conferences, and hold eight patents, one second prize for science and technology in Shaanxi Provincial Higher Education, and one third prize for Beijing Science and Technology. His current research interests include communication and information theory with special emphasis on cognitive and collaborative communications, the Internet of Things and data processing, distributed machine learning, and blockchain.

KANG MENG received the B.E. degree from the Faculty of Information Technology, Beijing University of Technology (BJUT), Beijing, China, in July 2018, where he is currently pursuing the M.S. degree. His research interests include blockchain and deep reinforcement learning.

MENG LI received the B.E., M.E., and Ph.D. degrees in electronic information engineering, electronics and communication engineering, and electronic science and technology from the Beijing University of Technology, Beijing, China, in 2011, 2014, and 2018, respectively. He joined the Beijing University of Technology in 2018, where he is currently a Lecturer. From September 2015 to September 2016, he visited Carleton University, Ottawa, ON, Canada, as a Visiting Ph.D. student funded by the China Scholarship Council (CSC). His current research interests include M2M communications, industrial Internet, mobile edge computing, resource management, and so on.

YANHUA ZHANG received the B.E. degree from the Xi'an University of Technology, Xi'an, China, in 1982, and the M.S. degree from Lanzhou University, Lanzhou, China, in 1988. From 1982 to 1990, he was with the Jiuquan Satellite Launch Center, Jiuquan, China. During the 1990s, he was a Visiting Professor with Concordia University, Montreal, QC, Canada. In 1997, he joined the Beijing University of Technology, Beijing, China, where he is currently a Professor. His research interests include quality-of-service-aware networking and radio resource management in wireless networks.