DQN-based energy-efficient routing algorithm in software-defined data centers

Zan Yao, Ying Wang and Xuesong Qiu

Abstract
With the rapid development of data centers in smart cities, how to reduce energy consumption and how to raise economic benefits and network performance are becoming an important research subject. In particular, data center networks do not always run at full load, which leads to significant energy consumption. In this article, we focus on the energy-efficient routing problem in software-defined network-based data center networks. For the scenario of in-band control mode of software-defined data centers, we formulate the dual optimal objective of energy-saving and the load balancing between controllers. In order to cope with a large solution space, we design the deep Q-network-based energy-efficient routing algorithm to find the energy-efficient data paths for traffic flow and control paths for switches. The simulation result reveals that the deep Q-network-based energy-efficient routing algorithm only trains part of the states and gets a good energy-saving effect and load balancing in control plane. Compared with the solver and the CERA heuristic algorithm, energy-saving effect of the deep Q-network-based energy-efficient routing algorithm is almost the same as the heuristic algorithm; however, its calculation time is reduced a lot, especially in a large number of flow scenarios; and it is more flexible to design and resolve the multi-objective optimization problem.

Keywords
Software-defined data centers, deep Q-network, in-band control, load balance, energy-efficient routing, smart city

Date received: 30 October 2019; accepted: 28 April 2020
Handling Editor: Bo Rong

Introduction
With the rapid development of modern information technologies such as cloud computing, big data, Internet of things, and edge computing, more and more studies about smart city have emerged. It emphasizes the cooperation and coordination of urban management and achieves the deep integration of industrialization and informationization. As the physical carrier of cloud computing and big data, data centers play an important role in smart cities. The data center network undertakes the function of data transfer and exchange in the data center.

Data center networks target high performance and high reliability which often have numerous redundant links and excessive link bandwidth. Network devices typically operate 24/7 at full capacity, consuming a large amount of energy. However, the utilization of network equipment is low for the most time, resulting in extremely low network energy efficiency. Therefore, there is an urgency to study an efficient data center network energy-saving mechanism to save energy while ensuring network performance.
The energy-saving mechanisms of the current data center networks can be divided into two types: device sleep (DS)\(^4\)\(^{-11}\) and adaptive link rate (ALR)\(^12,13\). The DS-based energy-saving mechanism is based on the point that dynamically sleeping the switches and links are not needed to be active, whereas the ALR-based energy-saving mechanism can dynamically assign the bandwidth for flows and it is through minimizing the link rates to save the energy. The switches account for the majority of energy consumption of data center network. This article focuses on the DS-based energy-efficient routing technology. For example, Heller et al.\(^4\) presented Elastic Tree method, which dynamically adjusts the set of active nodes and links. While minimizing the energy consumption, it can handle traffic surges and have a good ability of fault tolerance. After software-defined network (SDN) technology is applied in the Data Center Networks, new performance needs will come together.

In order to better manage data center network resources and improve energy consumption and service quality and network performance, more and more data center networks are beginning to adopt SDN technology. SDN decouples the control plane and data plane of the network device. The control plane uses a dedicated controller to provide unified and flexible control over the network. The forwarding device of the data plane is simplified and only needs to forward data according to the flow table sent by the controller. Therefore, SDN can greatly simplify network management and improve network resource utilization. Obviously, within the SDN-enabled network, the controller is responsible for the issuing of the switch flow table, which is very important in the network.

There is a dedicated control path between the switch and the controller. The control path can be divided into two modes: out-of-band\(^14\) and in-band control modes.\(^15\)\(^{-17}\) The out-of-band control mode has a special control link, which does not require the special calculation of the control path. In the in-band control mode, the control information and the data information can share the links, so the control path needs to be calculated. Although out-of-band control mode has less latency and better fault tolerance than in-band control mode, it requires an additional budget to create additional paths. At the same time, the switches still need to add new ports and protocols. So in this article, we choose the in-band control mode. Considering the control delay problem, the switch generally selects the shortest paths as the alternative control path. The control path needs to share the links with data path. Therefore, when studying the energy-saving routing problem, it is necessary to coordinate the data path and the control path energy consumption.

Besides, when the size of network is large, one controller’s ability may not support it to process the request information of the switches in a timely manner, so the control plane can be composed of multiple controllers. Then, every switch can select one from controllers, the corresponding control path will be different, and the load of the controllers may be different. So in order to guarantee the performance of the control plane while saving energy, we use the load balancing of the controllers as another network optimization objective.

In summary, it is needed to design an algorithm to choose the appropriate data path for every flow and control path for every switch to obtain the dual goals of energy savings and load balancing between controllers.

The energy-saving routing problem is usually modeled as a mixed-integer linear programming problem. There is a large number of network flows and switches, and there are several optional paths for each flow and switch, so the solution set of the optimization problem is very large. The existing literature separately calculates the data path and control path. Wu\(^17\) proposes an energy-efficient routing strategy, which uses a heuristic algorithm to coordinatively calculate the paths for the switches and data traffic. However, the proposed dual optimal problem has a very large solution space, and heuristic algorithms are close to traversal and they will need a certain time. Moreover, there are many bursts on the network, and the actual network paths need to be calculated in time.

During the last few years in the academic field, much attention has been paid to deep reinforcement learning (DRL). This combines the perception ability of deep learning (DL) with the decision-making ability of reinforcement learning (RL) to achieve complementary advantages, and it is used in many fields,\(^7,18\)\(^{-20}\) such as language processing,\(^19,20\) vehicle,\(^21\) traffic signal time planning,\(^22\) resource management,\(^23\) energy efficiency,\(^24\) and communications and networking.\(^7,18\)

DeepMind’s deep Q-network (DQN) algorithm (2013, 2015)\(^27\) is the first successful algorithm of DRL for combining DL and RL. It uses a deep network to represent the value function, which is based on Q-Learning, to provide target values for deep networks, and to constantly update the network until convergence. DQN algorithm can get a good result just through training part of the states. Therefore, we attempt to solve this energy-saving problem using a DRL algorithm with high-dimensional data processing capability.

In summary, for the software-defined data center network in the in-band control mode, we mainly study the problem of energy-saving routing. Our main contributions are twofold:

1. We propose a dual objective optimization problem of energy-saving routing and load balancing between controllers.
2. We model the optimization problem as a Markov decision process (MDP) and propose a deep Q-network-based energy-efficient routing (DQN-EER) algorithm. It only trains part of the states to collaboratively obtain the optimal data path and control path, and simultaneously processes the traffic in batches instead of sorting sequentially. The computation efficiency and timeliness are greatly improved.

The rest of the article is organized as follows: Section “Related work” analyses related works. Then, the dual-objective optimization problem is presented in section “Model of network system,” and we propose the DQN-EER algorithm to solve the problem in section “DQN-based energy-efficient routing algorithm.” The simulation results are presented to verify the feasibility and effectiveness of the proposed approach in section “Simulation and results.” Finally, the conclusion is given in section “Conclusion and future work.”

Related work

This article focuses on energy-saving routing problems based on the DS-based energy-saving mechanism. In the existing studies, the network flows are aggregated in the subset T of the network topology G with constraints of network performance. The devices and links in the T/G are dormant, thereby achieving energy-saving effects. The routing problem can be modeled as a multi-commodity flow problem, and the complexity of the solution is NP-hard. This problem can be solved using the optimization problem-solving tool. However, the tool is very expensive. When the network size is slightly larger, and it takes 10 h or more, hence most of the existing feasible solutions are solved by heuristic methods. Most of the current works model the problem as a mixed-integer linear programming problem and propose a heuristic algorithm. The elastic tree-based routing method proposed in Heller et al. and the routing method based on energy consumption characteristic curve generally calculate energy-saving routing according to network load. Chen et al. proposed a time-efficient energy-aware routing algorithm, which reduced the number of used links, and considered the temporal variation in demand. Al-Tarazi and Chang considered the load balance of network. After SDN technology is applied in the Data Center Networks, new performance needs will come together. The above references do not consider the case when the control paths and the forwarding traffic routing share the physical network resources. Shang et al. formulated the network in the in-band control mode and proposed a heretical method for the energy efficiency. Wu proposed a heuristic energy-efficiency routing algorithm based on dynamic weight of nodes, in which data plane control paths and data traffic routing were coordinated. The energy-saving working path was obtained by iteratively updating the dynamic weights, thereby reducing the energy consumption of the network.

Currently, there are many attempts to solve such kind of problem in addition to heuristic algorithms. Moreover, DQN is value-based and can be updated in a single step. This structure only needs to input a state and then output the Q-value of all actions, which is suitable for scenes with small action spaces. Other DRL algorithms, such as A3C and deep deterministic policy gradient (DDPG), are both strategy and value algorithms, suitable for continuous action space. In the routing algorithm in this article, we can design a few actions. Therefore, we apply DQN to solve the routing problem of the data center network to achieve energy savings and improve the network performance.

Model of network system

Model of data center network

The data center network is an undirected graph and can be modeled as $G(V, E, C)$, where $V$ is the set of switches, $E$ is the set of links, and $C$ is the set of controllers. The set of traffic that needs to be transmitted is defined as $K = \{k_1, k_2, \ldots, k_i, \ldots, k_m\}$, where each flow $k_i \in K$ includes the following parameter $k_i = (s_i, d_i, bi)$, $s_i, d_i$ indicate the source and destination nodes separately, and $bi$ indicates bandwidth requirements, which need to be guaranteed.

The energy consumption of SDN network equipment with a small load impact is similar to that of traditional network equipment. Therefore, the impact of traffic load parameters on energy consumption can be neglected in a high-bandwidth and low-latency data center network. This simplifies the network energy consumption (NEC) model. The total NEC is directly related to the number of active switches and the number of links. The calculation formula for NEC is shown in equation (1)

$$NEC = E_{base} \sum_{v \in V} x_v + E_{link} \sum_{e \in E} y_e$$

where $x_v$ represents the state of the switch $v$, and $y_e$ indicates the state of the link $e$, both of which have two values: 1 or 0, where 1 means active, 0 means dominant. $E_{base}$ and $E_{link}$ are separately the energy consumption of the device and the energy consumption of the link resource configuration. Since current “rich-connected” data center network topologies typically use homogeneous switch interconnect servers, it is assumed that the SDN switches used by the data center network have inherent energy consumption $E_{base}$ and all link...
energy consumption is $E_{\text{link}}$. Based on the NEC model, an energy-efficient routing algorithm is designed to control the number of devices and links to achieve an ideal energy consumption state.

Suppose there are $l$ flows, the set of flows is $K = \{k_1, k_2, \ldots, k_i, \ldots, k_l\}$. In order to facilitate subsequent data center network routing optimization, we first use depth-first traversal (DFS) or breadth-first traversal (BFS) for each flow to find all their shortest paths, represented by a set with $p_i \in P_i(i = 1, 2, \ldots, l)$, where $p_i$ represents one possible path of the flow $k_i$, where the start and end points are, respectively, $s_i$ and $d_i$.

\[ P_i = \begin{pmatrix}
    s_1 & \cdots & d_i \\
    s_i & \cdots & d_i \\
    \vdots & \ddots & \vdots \\
    s_i & \cdots & d_i
\end{pmatrix} \]  

(2)

The possible control path for each switch $v_i \in V$ which choose the controller $c_j$ is represented as $v p_{i,j} \in VP_{i,j}$.

**Problem formulation**

The topology design of the data center network has high redundancy. In this article, we calculate the traffic-aware energy-saving routing by considering both the transmission paths of flows and the control paths of switches. In addition, the influence of the load balancing between the controllers in the in-band control mode is also considered. The weighted sum of NEC and controller load balancing is the objective function of the optimization strategy

\[ s^* = \arg \min_{s} (\alpha \text{NEC}' + \beta \text{AB}') \]  

(3)

\[ \text{NEC}'_{c_j} = \frac{\text{NEC}_{s_j} - \min_{1 \leq j \leq n} \{\text{NEC}_{s_j}\}}{\max_{1 \leq j \leq n} \{\text{NEC}_{s_j}\} - \min_{1 \leq j \leq n} \{\text{NEC}_{s_j}\}} \]  

(4a)

\[ \text{AB}'_{c_j} = \frac{\min_{1 \leq j \leq n} \{\text{AB}_{s_j}\}}{\max_{1 \leq j \leq n} \{\text{AB}_{s_j}\} - \min_{1 \leq j \leq n} \{\text{AB}_{s_j}\}} \]  

(4b)

\[ \text{AB} = \sqrt{\sum_{c=1}^{[C]} \left( \sum_{v \in V} \lambda_{v,c} - |V|/|C| \right)^2 / |C|} \]  

(4c)

With constraints

\[ \sum_{c \in C} \lambda_{v,c} = 1, \forall v \in V \]  

(5a)

\[ \lambda_{v,c} = \begin{cases} 
    1 & \text{if } c \text{ is selected to control } v \\
    0 & \text{else}
\end{cases} \]  

(5b)

\[ \sum_{p_i \in P_i} \gamma_{p_i} = \sum_{v \in V} \gamma_{vp_{i,j}} = \sum_{c \in C} \gamma_{sp_{i,j}} = 1, \forall k_i \in K \]  

(6a)

\[ \gamma_{p_i} = \begin{cases} 
    1 & \text{if } p_i \text{ is selected to route } k_i \\
    0 & \text{else}
\end{cases} \]  

(6b)

\[ \sum_{c \in C, v p_{i,j} \in VP_{i,j}} \gamma_{vp_{i,j}} = 1, \forall v_i \in V \]  

(7a)

\[ \gamma_{vp_{i,j}} = \begin{cases} 
    1 & \text{if } v p_{i,j} \text{ is selected to connect } c \text{ and } v \\
    0 & \text{else}
\end{cases} \]  

(7b)

\[ \sum_{n \in V} \sum_{k \in E} f_k^{uv} \leq \delta C_{uv}, \forall uv \in E \]  

(8)

\[ x_u x_v y_{uv} = 1, \forall uv \in E, \forall f_k^{uv} > 0 \]  

(9)

where in the objective function (3) and (4a)–(4c), AB is a standard deviation used to measure the load balancing effect between the controllers, and the variables $\text{NEC}'$ and $\text{AB}'$ are the normalized ones, and $\alpha$ and $\beta$ represent the ratio between the energy consumption and the load balancing, respectively, and $s$ represents one feasible solution of data paths and control paths, and will be included in the $S$ as one row of the solution matrix, which will be given in section “DQN-based energy-efficient routing algorithm.” Equation (5a) indicates that only one controller can be connected per switch. The symbol $\lambda_{v,c}$ represents the connection between the switch and the controller. Equations (6a) and (7a) indicate that only one path can be selected for each flow and control information of each switch, and equations (6b) and (7b) describe the selection relation between the flow and route, and selection relation between the switch and the controller. Formula (8) is the constraints of link capacity, that is, the bandwidth used by network traffic cannot exceed the available bandwidth of the link. In order to ensure the availability of the link, the available bandwidth of the link is $\delta$ times the link bandwidth capacity, and $(1 - \delta)$ times the link bandwidth needs to be reserved for the emergency. Equation (9) is a traffic deployment restriction. It indicates that the corresponding switch $uv$ and link $uv$ are active when the network flow is assigned to the link $uv$. The symbols $uv$ indicate the working state of switch, which are binary values, 1 means active, 0 means sleep; and $uv$ means the working state of the link, which is also a binary value, 1 means active, 0 means sleep.

Since the solution space of the network topology optimization problem is very large, it is not advisable to use a method that is close to traversal. The DRL algorithm only trains some state data to get better results. Therefore, for this problem, the DRL method can not only approximate the optimal solution well but also greatly improve the computational efficiency.
DQN-based energy-efficient routing algorithm

For the energy-saving routing optimization model established above, the DQN is adopted to seek the most energy-saving data path and control path for each flow from the network as much as possible while maintaining the load balancing between the controllers. We will describe these in two parts. First, we propose a DQN-EER algorithm architecture and describe the components and the interaction between them in the architecture where the design of the state, action, and reward are outlined. Second, the process of the DQN-EER is presented.

DQN-EER algorithm architecture

DQN-EER algorithm architecture is shown in Figure 1. RL algorithm mainly includes two parts: environment and agent. Moreover, the problem can be modeled as an MDP with a state space, action space, and reward function, which will be designed in the next part.

Although the RL algorithm can learn from the surrounding environment itself, it still needs to design the corresponding features manually for it to be able to converge. In practical applications, the number of states may be large and in many cases, the features are difficult to be designed manually. The neural network happens to have very good processing for massive data, so the neural network is considered to replace the matrix $Q$ of the Q-Learning algorithm. DQN algorithm is modified on the basis of Q-Learning, and it has been improved on three aspects: (1) using deep convolutional neural networks (CNNs) to approximate value functions, (2) using empirical replay to train the agent, and (3) setting up independent target networks to determine target values. Therefore, the architecture includes two components as indicated below.

Environment. As shown in the architecture of Figure 1, SDN-enabled data center network consists of switches, controllers, and the links. Our goal is to reduce energy consumption and improve network performance, such as load balancing between the controllers. Data center network is modeled as the environment of the RL algorithm. The state is used to describe the situation of the SDN-enabled data center network, which covers two elements that include the paths of the traffic and the controlling paths of the switches. It will be designed in the next part.

Agent. When the DQN is used in the system, the overall SDN controller 0 has a global view of the network and can collect the environment state. So it can be seen as the agent. Based on the observation, it can carry out an action to react to the current state and offer a flexible way of policy deployment. There are three main parts as below.

MainNet. The DQN algorithm model is a combination of a multi-layer neural network model and an RL model. A CNN is used to approximate the action value function. The value function is the cumulative discount bonus when the action $a$ is performed in the state $s$. The approximation method of the action value function uses a parametric nonlinear approximation as shown. For a $n$-dimensional state space $S$, the action space has $m$ actions, and the neural network is a function that maps it from the $n$-dimensional space to $m$-dimension. Given state $s$ as an input, vector $Q(s, a; \theta)$ of action values output, where the parameters of the network are $\theta$.

Replay buffer. The concept of experience replay in the RL algorithm is used when extracting training samples during neural network training. Replay buffer means that the observed state transition process is first stored. After the sample has accumulated to a certain extent, it is randomly sampled from it to update the network. The main reason is that the samples obtained by randomly exploring the surrounding environment by different flows are a sequence associated with time and have a correlation. Due to the temporal correlation, if the data are directly used as a sample for training and updating the $Q$-value, the system convergence will be greatly affected, thereby the random sampling method solves the time correlation problem. This random extraction approach disrupts the correlation between experiences and makes neural network updates more efficient. In summary, the replay buffer is a very
important part of DRL method, which greatly improves the system performance of DRL.

**TargetNet.** The DeepMind team proposed to set up a separate target network called TargetNet, which is the same as the current network model. As represented in Figure 1, the output of the current network MainNet is \( Q(s, a; \theta) \) used to evaluate the value function of the current state action pair; \( \hat{Q}(s, a; \theta^-) \) represents the output of TargetNet, where the Target\( Q \) value is obtained by Formula (10)

\[
TargetQ = r + \gamma \max_a \hat{Q}(s', d'; \theta^-) \tag{10}
\]

And the calculation formula of the loss function in the neural network is

\[
L(\theta) = E \left[ (TargetQ - Q(s, a; \theta))^2 \right] \tag{11}
\]

The MainNet parameters are updated according to the Loss Function and then the MainNet parameters are copied to TargetNet every \( C \) iteration. Therefore, the current network model is updated once every time it interacts with the environment, and the target network is updated once every several times, which reduces the correlation between the current \( Q \) value and the target \( Q \) value to some extent, thereby improving the stability of the algorithm.

**The interaction between environment and agent.** The problem can be modeled as a MDP with a state space, action space, and reward function. An agent is used to interact with the environment. Based on the observations, it learns to alter the behavior and action in response to the received reward. In this part, we construct three elements for the interaction between the Environment and Agent.

**State.** For the traffic \( K = \{k_1, k_2, \ldots, k_i, \ldots, k_l\} \), all possible data paths and the resulting control paths are calculated by the DFS algorithm under the constraints of Formulas (5)-(7). When each flow \( k_i (i = 1, 2, \ldots, l) \) selects different paths \( p_i \), the networks can be combined into different topologies. Let \( dp, dp \in DP \) represents one of the topological states, with a set \( DP \) representing all possible topological states

\[
dp = [p_1, \ldots, p_i, \ldots, p_l]
\]

\[
DP = [P_1, \ldots, P_i, \ldots, P_l]
\]

Define \( W, W \subseteq V \) as the set of all the switches that need to be activated in each possible topology \( dp \), with the number of activated switches as \( m \). One possible corresponding control path of \( w_j \) is represented as \( wp_j \). The corresponding control path of all the \( w_j \in W \) can

| Table 1. | The pseudo-code of constructing the state space. |
|----------|-----------------------------------------------|
| **Input:** | \( G(V, E, C) \), \( K \) |
| **Output:** | \( S \) |
| 1. for \( k_i \) in \( K \) | 2. Compute the path set \( P_i \) using the DFS algorithm. |
| 3. end for | 4. Get the possible path combination \( dp \). |
| 5. Get the set of possible path combinations \( DP \). | 6. for \( dp_i \) in \( DP \) |
| 7. Get the corresponding set of switches \( W \). | 8. for \( w_i \) in \( W \) |
| 9. Compute the control path \( wp_i \) using the DFS algorithm. | 10. end for |
| 11. Get the possible control path combination \( cp \). | 12. Get the set of possible path combinations \( CP \). |
| 13. for \( cp_j \) in \( CP \) | 14. \( s \leftarrow dp_i \cup cp_j \) |
| 15. \( S \leftarrow S \cup s \) | 16. end for |
| 17. end for | 18. return \( S \) |

be placed in a set of \( cp \). Each \( dp \) corresponds to several states \( cp \)

\[
cp = [wp_1, \ldots, wp_i, \ldots, wp_m]
\]

\[
CP = [CP_1, \ldots, CP_i, \ldots, CP_m]
\]

Then there may be more switches to be activated. For a newly activated switch, because it is the switch that is open on the control path, the same control path can be selected without having to activate other switches. We then update the collections of switches \( N \) and the set of corresponding control paths \( CP \).

Let \( s \) represent the set of one possible data path for every flow and one possible control path for every associated switch. And it is used as input to CNN in the DQN algorithm. \( s \) is stored in the set \( S \) in rows

\[
s = [p_1, \ldots, p_i, \ldots, p_l, wp_1, \ldots, wp_i, \ldots, wp_m]
\]

\[
S = [DP, CP]
\]

In order to better define the action space, we first sort all the data paths of the path space of each data stream and the corresponding control paths by row. The pseudo-code of constructing the state space of the environment is shown in Table 1.

**Action.** The agent focuses on mapping out the space of state to the space of action and in identifying the optimal policy. The action space for each possible combined state can be defined as \( a_i \). The size of the entire action space is 3, that is, there are three optional actions for each state.
where $-1$ indicates that the state $s$ moves up one line, 1 means move down, and 0 means unchanged.

**Reward.** The immediate reward is defined by analyzing the objective function

$$ r = \begin{cases} \frac{1}{\text{NEC}} + \frac{1}{\text{BEP}}, & \text{if meet the constraints} \\ 0, & \text{otherwise} \end{cases} \quad (12) $$

Since the objective function is to find the minimum energy consumption, and the smaller the energy consumption, the larger the reward, so the reciprocal of energy consumption can be immediately used as an immediate reward. For those that do not satisfy the bandwidth constraints (8) and (9), the immediate reward is 0.

**DQN-EER algorithm process**

DQN algorithm includes offline construction of the network phase and online DL phase.\(^2\)

The offline network construction phase. CNN is used to obtain the relationship between the pair of state action $(s, a)$ and the value function $Q(s, a; \theta)$, which is the cumulative discount reward when performing the action $a$ in the state $s$. Offline construction requires the accumulation of sufficient value estimates and corresponding samples $(s, a)$ and uses relay memory to smooth the training process.

The online DL phase. The $\epsilon$ greedy strategy is used to select actions $a, \epsilon$ of which randomly select actions, and $(1-\epsilon)$ of which choose the action with the largest estimated value $Q$. In the interaction with the environment, the immediate reward $r$ and the next state $s'$ are observed. Then the state transition $(s, a, r, s')$ is stored in the replay buffer, and finally some of which are sampled from the replay buffer to update the CNN parameters.

Then, the implementation process of DQN-EER algorithm in software-defined data center network is given in this part. The flow chart of DQN-EER algorithm is shown in Figure 2, and the steps are as follows:

1. Initialize the replay buffer and set the minibatch (the number of samples collected in a training session);
2. Initialize state $s$ randomly;
3. On the basis of the current state $s$ select an action $a$, then obtain the corresponding reward value $r$, and the state $s'$;
4. Then save the relevant parameters $(s, a, r, s')$ in the replay buffer;
5. Check whether the amount of data stored in the memory pool exceeds minibatch. If not, go to 6; otherwise, perform the following steps:
   a. Randomly select some samples from the replay buffer;
   b. Use the randomly taken sample state $(s, a)$ as a training sample of MainNet, and obtain the $Q$ value in the corresponding state;
   c. Calculate the $TargetQ$ value corresponding to the $Q$ value according to the Formula (12);
   d. Train the neural network using the $Q$ value and the $TargetQ$ value using Formula (13);
   6. $Update Q = Q$ every $C$ steps.
   7. Determine whether the search process ends or not (set the maximum number of search steps before searching). If the maximum number of search steps is reached, perform step 8; otherwise, update the current state $s$ to $s'$, and return to step 3;
   8. Determine whether the training number reaches a certain Maximum. If it reaches, return to step 2; otherwise, end the whole training.

**Simulation and results**

To verify the effectiveness of the proposed DQN-EER algorithm, simulation is conducted in a Fat-Tree SDN-enabled data center network. We build a simulator by Python.

**Simulation environment**

Under the Windows 10 system, the Python language is used to program the algorithm. The hardware platform is configured as a 2.4 GHz CPU and 64 GB memory. This work selects the commonly used Fat-Tree data center network topology which consists of 20 four-point switches, 16 hosts and 48 links. In order to simulate the load balancing between the controllers and considering the numbers of switches and the ability of controllers, we design three controllers 0, 1, and 2. The controller 0, the overall controller, is connected with controller 1 and 2, which respectively connects with switch 0 and 2, and, as shown in Figure 1.

**Simulation results and analysis**

In order to verify the validity and performance of the proposed DQN-EER algorithm, we mainly design the simulation from two parts. First, we chose a small number of flows and the DQN-EER algorithm is applied to solve the dual optimal objectives problem. It is verified that the algorithm effectively achieves the dual goals of energy-saving and load balancing between controllers. We mainly use the network energy-saving percentage $P$
as the evaluation matrix of energy-saving effectiveness, that is, the NEC saved by using the method \( A \) accounts for the percentage of the total NEC when all the switches are active without using any method. The specific definition is as shown in Formula (13)

\[
P = 1 - \frac{NEC_A}{NEC_{full}}
\]  

(13)

Then, we will describe the experimental design and the results in two parts.

First, in order to verify the effectiveness of the algorithm, we design eight flows, which belong to four different pods and any two of them do not belong to the same edge switch; and there are four flows that need to go through the core switch. There are two or four alternative paths for each flow, and the magnitude of the state is between \( 2^8 \) and \( 4^8 \). In addition to the control paths for the switches to be activated, the state level is between \( 4^8 \) and \( 8^8 \). Our goal is to find the data path and the corresponding control path from the alternate paths for the eight flows to make the objective function minimized.

In order to achieve a better energy-saving effect and more reserved bandwidth is conducive to dealing with emergencies and failure recovery, the parameter of redundancy \( \delta \) is taken as 0.8 in our experiment.

**Figure 2.** The flow chart of DQN-EER algorithm.
Through learning and constantly adjusting various parameters, we finally obtain the actual parameters in the stable convergence algorithm. The parameters in the algorithm are given in Table 2.

After the training of DQN-EER algorithm is completed, the model is saved and then tested, and the network will find a relatively ideal path. For the test results, every 100 steps, the energy consumption of data path plus control path and the load balancing between controllers are counted, which are shown in Figure 3. It is found that before the 1600 steps, the rate of decline is very fast. After reaching 2300 steps, the algorithm approaches convergence and the objective function tends to be stable. At this time, the solution tends to the optimal solution. Therefore, the algorithm is stopped at around 2300 steps. The paths of the eight flows are regarded as the optimal ones.

Table 3 gives the optimal data path for the eight flows obtained using the DQN-EER algorithm, along with the activated switches and links (which include the links of control paths). The number of all activated switches is 17, including switches (0, 1, 2, 4, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19), and the number of all active links in green is 18, including links (0, 1, 3, 5, 8, 9, 12, 14, 15, 16, 18, 20, 21, 24, 25, 26, 30, 31), except the two links which are used to directly connect the switches and controllers and the two links between the controllers with the overall controller 0. And the link 21 is the added one when calculating the control path.

And we evaluate the effectiveness of the load balancing between the controllers by comparing both cases: first consider the optimization goal of the load balancing between controllers and second, no consideration. Both of them are calculated by the DQN-EER algorithm. The experimental results are shown in Figures 4 and 5. The switches controlled by controller 1 are marked blue, and the ones controlled by controller 2 are marked green. All the activated links are marked green. Obviously, the load balancing performance of the DQN-EER algorithm considering load balancing between controllers is better.

Second, in order to evaluate the energy-saving effect of using the DQN-EER algorithm, we select the CPLEX solver and the control-path-based energy-aware routing algorithm (CERA) greedy algorithm as the comparison algorithms. Since CERA does not consider the problem of load balance between controllers, we will use the DQN algorithm to compare energy savings without considering the load balance problem. The experimental results verify the network energy-saving percentage of the three methods under different network traffic strengths. As shown in Figure 6, we mainly use the network energy-saving percentage $P$ as an evaluation index of the energy-saving effect.

We use the energy cost when the full network element is in the opening state as the benchmark of

### Table 2. The parameters of DQN-EER algorithm.

| Parameters | Values |
|------------|--------|
| Parameter in the objective function, $\alpha, \beta$ | 0.8, 0.2 |
| Learning rate, $\lambda$ | 0.0, 0.2 |
| Discount factor, $\gamma$ | 0.8 |
| Greedy value, $\epsilon$ | 0.01–1 |
| Buffer update iterations, $t$ | 150 |
| Number of observation steps, $s$ | 200 |
| Buffer unit size, $D$ | 800 |
| Training round, $e$ | 2300 |

DQN-EER: deep Q-network-based energy-efficient routing.

### Table 3. The result of DQN-EER algorithm.

| Flows | Switches (resource-target) | Flow path | Activated switches | Activated links |
|-------|----------------------------|-----------|-------------------|-----------------|
| 1     | 12-13                      | 8-9       | 0, 1, 2, 4, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19 | 0, 1, 3, 5, 8, 9, 12, 14, 15, 16, 18, 20, 21, 24, 25, 26, 30, 31 |
| 2     | 12-14                      | 8-0-1-14  | 14, 15, 16, 17, 18, 19 | 16, 18, 20, 21, 24, 25, 26, 30, 31 |
| 3     | 14-15                      | 14-15     |                   |                 |
| 4     | 13-16                      | 9-0-16-26 |                   |                 |
| 5     | 16-17                      | 24-25     |                   |                 |
| 6     | 14-16                      | 12-3-18-26|                   |                 |
| 7     | 18-19                      | 30-31     |                   |                 |
| 8     | 14-17                      | 12-5-20-25|                   |                 |

DQN-EER: deep Q-network-based energy-efficient routing.
comparison in the network energy saving percentage. We can see that under the same load conditions, the energy-saving effect of the optimal solution obtained by CPLEX is better than that of the CERA method and DQN-EER method. By comparing the energy-saving effects of CERA and DQN-EER algorithm, it is found that both methods give appreciable results.

Then, we compare the computation time of the above algorithms, as shown in Figure 7. The state scale of the horizontal axis indicates the number of states,
from 10,000 states to 200,000 states. The vertical axis indicates the time taken by the algorithms when the objective function is reached.

It can be seen that when the number of states is small, the time of the DQN-EER algorithm is the longest because it requires certain training traffic to support. The other two algorithms are solved in a manner, which is close to traversal. When the state quantity is small, it is desirable to select these two methods. When the number of states gradually becomes larger, the time of the solver increases almost linearly, and the one of DQN-EER algorithm tends to be stable. Because it only needs some data for training to draw the model, it can predict the situation corresponding to most states. In addition, it can be seen that CPLEX costs more than 6000 s to get the optimal solution, DQN takes about no more than 700 s, and CERA method only takes less than 10 s. The DQN-EER algorithm only counts the training time here. The training process time of DQN-EER does not have an effect on the decision, and it will give the result timely. And compared with CERA method, the design of DQN-EER is more flexible, especially for multi-objective optimization problems. So it will be satisfactory to use the proposed DQN-EER algorithm.

**Conclusion and future work**

SDN is proposed as a promising technology in data center networks, and it can provide centralized network management and traffic control. In this article, in the in-band control mode, we proposed the dual optimization goals of the energy-saving and load balancing between controllers, and design the DQN-EER algorithm to solve it, which learns directly from experience and make decisions quickly. The energy-saving routing is selected for the arriving flows, and the energy-saving control path is coordinatively selected at the same time. Compared to other heuristic algorithms, it is easy to design and implement dual optimization goals using the DQN algorithm. The effectiveness of the proposed DQN algorithm is verified by simulation.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Key R&D Program of China (2018YFE0205502).

**ORCID iD**

Zan Yao [https://orcid.org/0000-0002-7004-1807](https://orcid.org/0000-0002-7004-1807)

**References**

1. Mohammadi M, Al-Fuqaha A, Guizani M, et al. Semi-supervised deep reinforcement learning in support of IoT and smart city services. *IEEE Internet Things J* 2018; 5(2): 624–635.
2. Yu P, Zhou F, Zhang X, et al. Deep learning-based resource allocation for 5G broadband TV service. *IEEE T Broadcast*. Epub ahead of print 10 February 2020. DOI: 10.1109/TBC.2020.2968730.
3. Peng Y, Wang X, Shen D, et al. Design and modeling of survivable network planning for software-defined data center networks in smart city. *Int J Commun Syst* 2018: 31: e3509.
4. Heller B, Seetharaman S, Mahadevan P, et al. Elastic-Tree: saving energy in data center networks. In: *Proceedings of the 7th USENIX conference on networked systems*
5. Chen Y-H, Chin T-L, Huang C-Y, et al. Time efficient energy-aware routing in software defined networks. In: 2018 IEEE 7th international conference on cloud networking (CloudNet), Tokyo, Japan, 22–24 October 2018, pp.1–7. New York: IEEE.

6. Al-Tarazi M and Chang JM. Performance-aware energy saving for data center networks. IEEE Trans Netw Serv Manage 2019; 16(1): 206–219.

7. Luong NC, Hoang DT, Gong S, et al. Applications of deep reinforcement learning in communications and networking: a survey. IEEE Commun Surv Tutor 2019; 21(4): 3133–3174.

8. Hu Y, Li Z, Lan J, et al. EARS: intelligence-driven experiential network architecture for automatic routing in software-defined networking. China Commun 2020; 17(2): 149–162.

9. Zhang L, Guo S, Yang Y, et al. RE-FPR: flow preemption routing scheme with redundancy elimination in Software Defined Data Center Networks. Sustain Comput: Inform Syst 2018; 18: 14.

10. Liu W. Intelligent routing based on deep reinforcement learning in software-defined data-center networks. In: 2019 IEEE symposium on computers and communications (ISCC), Barcelona, 29 June–3 July 2019, pp.1–6. New York: IEEE.

11. Hossain MB and Wei J. Reinforcement learning-driven QoS-aware intelligent routing for software-defined networks. In: 2019 IEEE global conference on signal and information processing (GlobalSIP), Ottawa, ON, Canada, 11–14 November 2019, pp.1–5. New York: IEEE.

12. Xu G, Dai B, Huang B, et al. Bandwidth-aware energy efficient flow scheduling with SDN in data center networks. Fut Gener Comput Syst J 2017; 68: 163–174.

13. Fan K, Wang Y, Ba J, et al. An approach for energy efficient deadline-constrained flow scheduling and routing. In: 2019 IFIP/IEEE symposium on integrated network and service management (IM), Washington, DC, 9–12 April 2019, pp.469–475. New York: IEEE.

14. Lamport L. Lower bounds for asynchronous consensus. Distrib Comput 2006; 19(2): 104–125.

15. Wang Y, Zhong Q, Qiu X, et al. Resource allocation for reliable communication between controllers and switches in SDN. J Netw Syst Manage 2018; 26(4): 966–992.

16. Sharma S, Staessens D, Colle D, et al. Fast failure recovery for in-band OpenFlow networks. IEEE Des Reliab Commun Netw 2013; 8875: 52–59.

17. Wu Z. A routing optimization considering both control and data traffic in SDN. Thesis [D], Beijing University of Posts and Telecommunications, Beijing, China, 2018.

18. Machine learning finds new ways for our data centers to save energy, https://blog.csdn.net/real_myth/article (2016, 23 December 2016).

19. Narasimhan K, Kulkarni T and Barzilay R. Language understanding for text-based games using deep reinforcement learning. arXiv preprint arXiv:1506.08941v2 [cs.CL], 2015.

20. He J, Chen J, He X, et al. Deep reinforcement learning with a natural language action space. arXiv preprint arXiv:1511.04636v5 [cs.AI], 2015.

21. Zhao D, Chen Y and Lv L. Deep reinforcement learning with visual attention for vehicle. IEEE T Cogn Develop Syst 2016; 9: 356–367.

22. Li L, Lv Y and Wang F-Y. Traffic signal timing via deep reinforcement learning. IEEE/CAA J Autom Sinica 2016; 3(3): 247–254.

23. Mao H, Alizadeh M, Menache I, et al. Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM workshop hot topics networks, Atlanta, GA, 9–10 November 2016, pp.50–56. New York: ACM.

24. François-Lavet V, Taralla D, Ernst D, et al. Deep reinforcement learning solutions for energy microgrids management. In: Proceedings of the European workshop reinforcement learning, Barcelona, 3–4 December 2016, pp.1–7. arxiv.org.

25. Huang X, Yuan T, Qiao G, et al. Deep reinforcement learning for multimedia traffic control in software defined networking. IEEE Netw 2018; 32(6): 35–41.

26. Xu Z, Tang J, Meng J, et al. Experience-driven networking: a deep reinforcement learning based approach. In: Proceedings of the IEEE INFOCOM, Honolulu, HI, 16–19 April 2018. New York: IEEE.

27. Mnih V, Kavukcuoglu K, Silver D, et al. Human level control through deep reinforcement learning. Nature 2015; 518(7540): 529–533.

28. Guo J, Wang Z, Shi X, et al. A deep reinforcement learning based mechanism for cell outage compensation in massive IoT environments. In: IWMC, Tangier, Morocco, 24–28 June 2019, pp.284–289. New York: IEEE.