Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051217

ARTICLE

QoS Routing Optimization Based on Deep Reinforcement Learning in SDN

Yu Song[1], Xusheng Qian[2], Nan Zhang[3], Wei Wang[2] and Ao Xiong[1,*]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

[2]Marketing Service Center, State Grid Jiangsu Electric Power Co., Ltd., Nanjing, 220000, China

[3]Customer Service Center, State Grid Co., Ltd., Nanjing, 211161, China

*Corresponding Author: Ao Xiong. Email: xiongao@bupt.edu.cn

Received: 29 February 2024 Accepted: 09 April 2024 Published: 15 May 2024

ABSTRACT

To enhance the efficiency and expediency of issuing e-licenses within the power sector, we must confront the challenge of managing the surging demand for data traffic. Within this realm, the network imposes stringent Quality of Service (QoS) requirements, revealing the inadequacies of traditional routing allocation mechanisms in accommodating such extensive data flows. In response to the imperative of handling a substantial influx of data requests promptly and alleviating the constraints of existing technologies and network congestion, we present an architecture for QoS routing optimization with in Software Defined Network (SDN), leveraging deep reinforcement learning. This innovative approach entails the separation of SDN control and transmission functionalities, centralizing control over data forwarding while integrating deep reinforcement learning for informed routing decisions. By factoring in considerations such as delay, bandwidth, jitter rate, and packet loss rate, we design a reward function to guide the Deep Deterministic Policy Gradient (DDPG) algorithm in learning the optimal routing strategy to furnish superior QoS provision. In our empirical investigations, we juxtapose the performance of Deep Reinforcement Learning (DRL) against that of Shortest Path (SP) algorithms in terms of data packet transmission delay. The experimental simulation results show that our proposed algorithm has significant efficacy in reducing network delay and improving the overall transmission efficiency, which is superior to the traditional methods.

KEYWORDS

Deep reinforcement learning; SDN; route optimization; QoS

1 Introduction

For power grid enterprises to effectively conduct power engineering infrastructure, operations and maintenance, marketing, and other business activities, seamless cross-departmental interaction is essential to ensure reliable data sharing. This is achieved by issuing electronic licenses for data storage and sharing, thereby guaranteeing trusted data sharing between departments. However, in order to achieve efficient and rapid electronic license issuance in power business scenarios, higher network Quality of Service (QoS) [1] is imperative. This entails satisfying users' requirements concerning the delay, throughput, jitter rate, and packet loss rate of the network. When dealing with large-scale

data transmission and traffic, ensuring the stability of network services is crucial to prevent network paralysis caused by congestion. Traditional network routing schemes typically rely on the shortest path algorithm for calculation, which has proven insufficient to meet the demands of current network traffic with extensive resource requirements. These traditional approaches often suffer from slow convergence speeds and are prone to network congestion [2,3].

Software Defined Networking (SDN) employs the separation of transfer and control by utilizing the control layer's development interface to offer services to the upper network layer, while ensuring unified control of the data forwarding layer for efficient traffic transmission [4]. Implementing SDN architecture can significantly enhance network performance and utilization, facilitating comprehensive network environment management [5]. The versatile features of SDN find applications in diverse scenarios such as edge computing [6], routing optimization, and others. In today's SDN routing algorithm, Dijkstra's fixed exploration strategy algorithm [7] is mainly used, which considers the shortest path problem and does not consider other network metrics, which will limit the exploration of the network environment, in the face of complex networks, there will be many limitations. The various algorithms of Reinforcement Learning (RL) involve a process of searching for the optimal solution in the process of continuous exploration and utilization, mainly solving the resource scheduling decision problem [8]. Through the design of reward functions and state action problems, the process of dynamic transition and state is determined. It can be combined with the routing optimization algorithm to continuously explore and optimize the QoS as the index to find the optimal path forwarding. Deep Reinforcement Learning (DRL) adds deep learning to the agent in reinforcement learning and uses the perception ability of deep learning to deal with complex network environment and routing decision problems more effectively, to achieve better routing strategy, DRL is mainly used in route design and resource management, and there are applications of DRL in games [9], video [10]and dynamic resource allocation [11]. Deep Deterministic Policy Gradient (DDPG) based routing algorithms are commonly employed to solve decision problems in continuous action space. When facing unknown and complex network conditions, DDPG can adjust its policy to adapt to the new environment through learning. When the network topology or load changes, DDPG can adjust the routing strategy through learning to optimize the network performance. Compared with traditional routing algorithms, DDPG has higher flexibility and adaptability to dynamic network conditions.

In the power industry, electronic licenses need to be issued quickly to ensure timely data interaction and sharing. Traditional routing algorithms fail to meet the QoS requirements of this scenario. We combine the SDN architecture and deep reinforcement learning algorithm and propose a QoS routing optimization algorithm based on deep reinforcement learning in SDN. Considering the QoS index requirements of the scene, we design the reward function to provide the optimal QoS service. The main contributions of this paper are as follows:

1. We propose an overall routing architecture based on deep reinforcement learning in SDN. SDN is combined with deep reinforcement learning and applied in routing scenarios. The agent architecture in reinforcement learning is put into the SDN controller layer, and the routing was put into the data forwarding layer. Using the characteristics of SDN transfer and control separation can effectively improve the transmission efficiency of the network.
2. We propose the DDPG-based routing optimization algorithm in SDN. Taking QoS as the optimization objective, the delay, bandwidth, jitter rate and packet loss rate as metrics are comprehensively considered, the reward function is designed, and the DDPG algorithm is used to learn the optimal routing strategy to provide the optimal QoS service set in this paper.

3. We have carried out a large number of experimental simulations. The experimental results show that the DDPG algorithm in the routing optimization scenario, compared with the traditional routing algorithm, can efficiently process data requests and reduce data transmission delays.

The rest of the paper is structured as follows: Section 2 summarizes the related work. Section 3 presents the system model architecture. Section 4 introduces the routing algorithm based on deep reinforcement learning in SDN. In Section 5, the experimental simulation is carried out. Conclusions are given in Section 6.

2 Related Work

To solve the problem of network congestion and meet the demands of current mass traffic. There is also many related research to solve this practical problem. At present, most of the research on QoS in SDN network is based on a single data index, which has low algorithm complexity and is simple to realize. It can also optimize routing traffic to a certain extent, but it is easy to cause local optimum. It only optimizes a single parameter without considering the constraints of multiple QoS parameters, so it can only solve a specific part of the problem [12].

Most of the traditional routing traffic forwarding methods use the Shortest Path First method to select route forwarding, such as the Open Shortest Path First (OSPF) [13] protocol, the forwarding traffic only considers the shortest path, namely delay, without considering other factors in QoS, which is easy to cause channel congestion and cannot meet the demands of today's high data traffic.

There have been extensive studies on the application research of SDN [14,15]. There have been many studies on SDN and artificial intelligence technology in routing scenarios in recent years. Gopi et al. [16] and Shirmarz et al. [17] used SDN technology to enhance the traditional routing protocol, which has limitations and does not combine network operation knowledge to realize intelligent routing. Xu et al. [18] proposed a route randomization defense method based on deep reinforcement learning to resist eavesdropping attacks. Compared with other route randomization methods, it has obvious advantages in security and resistance to eavesdropping attacks. Yu et al. [19] proposed DROM routing optimization framework, which uses DDPG algorithm to optimize SDN routing, and can deal with multi-dimensional action space. By maximizing the reward, the network state updates the link weight continuously, to select QoS routes that meet the conditions. Stampa et al. [20] proposed a deep reinforcement learning agent for routing optimization under the SDN framework, which selects routes according to the current routing state, to minimize network delay. Some related studies use Q-learning algorithm to optimize routing in QoS [21], and the whole training process uses Markov Decision Process (MDP) for training, which can achieve low delay and high throughput to a certain extent. However, Q-learning uses the Q-table value to store the reward value of the training process. When the number of routes increases, the selection of paths also increases greatly, and the storage of Q-table will bring a lot of memory overhead. Some related studies use Deep Q-Learning (DQN) algorithm for routing optimization [22], and use neural network to replace the traditional Q table. However, DQN algorithm can only be applied to discontinuous action and network state space, and today's network state transition is very fast, so it is not suitable for use in today's network transition and traffic transmission is large.

The above references can improve the performance of routing by using SDN and DRL technology, but only consider the delay parameter in the QoS index and optimize the network delay, and do not consider other indicators of network status. In this paper, SDN and DDPG algorithm are combined to optimize the routing, and the QoS of the routing is optimized by considering factors such as packet loss rate, jitter rate, delay, and bandwidth, to provide users with a better network experience. Therefore,

using the global network topology to realize intelligent QoS routing optimization in SDN architecture to improve QoS while ensuring network quality of service has become an urgent problem to be solved in current research.

The above papers can improve the performance of routing by using SDN and ML technology, but in QoS indicators, only delay is optimized without considering other indicators of network status. In this paper, SDN and DRL technology are combined to optimize the routing, and the factors of packet loss rate, jitter rate, delay and bandwidth are comprehensively considered to optimize the QoS of the route to achieve the optimal QoS. Therefore, using the global network topology to realize intelligent QoS routing optimization in SDN architecture, and ensuring the quality of network service while improving QoS, has become a problem to be solved in current research.

3 Routing Architecture Based on Deep Reinforcement Learning in SDN

The system model architecture utilizes in this paper adopts SDN architecture and DRL. SDN's separation of forwarding and control can effectively improve the problem of network congestion and low efficiency and combines the decision-making ability of reinforcement learning and the perception ability of deep learning and is well-suited for the scenario of routing optimization. In the SDN control layer, the DRL layout sends the action through the SDN controller to the data forwarding layer for path selection. In this section, the routing architecture based on deep reinforcement learning under SDN will be introduced in detail.

SDN can effectively improve the congestion and inefficiency of current networks [23]. SDN is built by separating the control and data layers of the network devices used today. The SDN framework is divided into application layer, control layer and data forwarding layer from top to bottom [24]. Information transfer between layers the control layer and the application layer are transmitted through the north interface, and the information transfer between the control layer and the data layer is the south interface. The advantages of SDN architecture are 1) the network structure is clearly layered, and the functions are clearly distributed; 2) the network transmission and configuration are unified and operated by the controller, programmable; 3) the control layer and data forwarding are structurally coupled, which can improve the efficiency of data transmission. SDN's structure of separating the transfer and control layers and centralized control can offer enhanced flexibility for data handling and can accelerate the overall network transmission efficiency more effectively, which has been widely used in recent years.

The RL model is a kind of MDP architecture. The process of RL is a continuous interaction between the intelligence and the environment for action selection and state change, and a reward value is returned, and the interaction process converges until the reward value reaches its maximum. Its representative algorithms are Q-learning [25], but as the number of environmental states increases, Q-table will occupy larger storage resources and time-consuming to find, and when the number of environmental states is immeasurable, Q-table cannot support storing all the states. DRL can solve the problems of Q-learning. Deep reinforcement learning combines the decision-making ability of RL with the perceptual ability of deep learning. The representative algorithms are DQN which combines neural network and Q-learning, but it is not applicable to the environmental situation of continuous action. Deterministic Policy Gradient (DPG) algorithm [26] can be used in the case of continuous action changes but is prone to overfitting problems. The DDPG algorithm [27] can solve these problems, which is based on the combination of DPG and DQN algorithms on the Actor-Critic method. The algorithm used in this paper for the deep reinforcement learning module is DDPG, and the DDPG algorithm is described in detail next.

The overall process of the DDPG algorithm is shown in Fig. 1 and employs the experience replay technique. The experience replay is a set of data (s_t, a_t, r_t, s_{t+1}) for each state transition into a relay buffer, where s_t is the current state of the environment, a_t is the action made in the current environment, r_t is the real reward value obtained by making the action, and s_{t+1} is the state of the environment after making the action. After reaching the set number of cache pools N a random uniform collection of M (M < N) datasets is performed to train the neural network, using randomly sampled datasets in order to eliminate the temporal correlation between datasets. The Actor-Critic algorithm is used in the DDPG algorithm, where each module has two neural networks, an online network for training and learning, and a target network, both of which are identical in structure. Firstly, the online network is initialized with $Q\left(s, a|\theta^{Q}\right)$ and $\mu\left(s|\theta^{\mu}\right)$, and the target network is initialized with Q' and μ', and the initialization assignment of the parameters of this network is $\theta^{Q'} \leftarrow \theta^{Q}$, $\theta^{\mu'} \leftarrow \theta^{\mu}$ The training dataset is adopted by the empirical playback algorithm. The detailed process of training is described in the following.

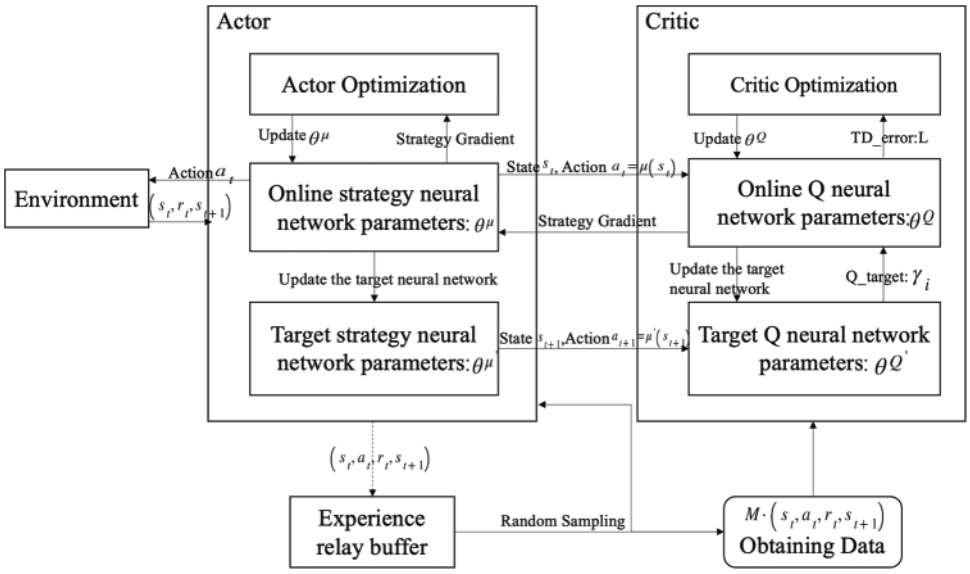

Figure 1: DDPG training process

First the agent acquires the state s_1 and performs t→1, T for cycling for the same learning and training. According to the current environment s_t, the corresponding action $a_t = \mu\left(s_t|\theta^{u}\right)$ is made according to the actor function of the online network. The action at is made, the reward r_t and the new environment state s_{t+1} are obtained, and (s_t, a_t, r_t, s_{t+1}) is put into the replay buffer R. When the amount of data in the replay buffer reaches the set number N, the parameters in the neural network are updated by randomly selecting M data sets in the replay buffer R. The reward value y_i obtained at the execution of that action, the actual reward value currently obtained plus the predicted reward value for future actions obtained with the objective function critic, uses the Time Differential (TD) method as in Eq. (1), where γ is the discount factor, representing the constant decline of the reward.

$$y_i = r_i + \gamma Q\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right) \tag{1}$$

According to the current state s_t makes the corresponding action a_t, as the input of $Q\left(s, a|\theta^{Q}\right)$, the reward value is derived, and the TD-error is performed between the y_i calculated with the TD algorithm to calculate the loss function as in Eq. (2), which is used to update the online critic network

parameters.

$$L = \frac{1}{N}\sum_{i}\left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2 \tag{2}$$

The parameters in the online actor network are updated using the product of the gradient of the online critic network and the gradient of the online actor network, as shown in Eq. (3). The equations for updating the critic and actor parameters of the target network are shown Eqs. (4) and (5).

$$\nabla_{\theta^\mu} J = grad\left[Q\right] * grad\left[\mu\right] = \frac{1}{N}\sum_{i}\nabla_a Q\left(s, a|\theta^Q\right)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu\left(s|\theta^\mu\right)|s_i \tag{3}$$

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\,\theta^{Q'} \tag{4}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\,\theta^{\mu'} \tag{5}$$

The SDN architecture itself, with its inherent separation of the transfer and control layers, offers a platform for network optimization. By integrating DRL, specifically the DDPG algorithm, into the routing optimization process, we aim to achieve QoS routing optimization. The overall architecture is depicted in Fig. 2. In this architecture, the network topology resides within the data forwarding layer of the SDN architecture, serving as the environment element in the MDP model. The DRL agent, utilizing DDPG, operates within the control layer of SDN, functioning as an agent element within the MDP. The agent receives real-time network state information from the environment, serving as input to the DDPG neural network. It then outputs actions with maximum Q-value to the SDN controller, which centrally routes the next hopping action. Subsequently, the network state transitions to the next state, with actual network data provided to the agent as a parameter for the reward value calculation based on QoS metrics. This reward value is crucial for updating the neural network parameters within the DDPG algorithm employed by the agent. The neural network parameters in the DDPG algorithm are continuously trained and updated to achieve convergence, thereby obtaining the optimal routing and forwarding policy. Consequently, this approach enables the identification of the optimal path for route forwarding to meet the specified QoS targets.

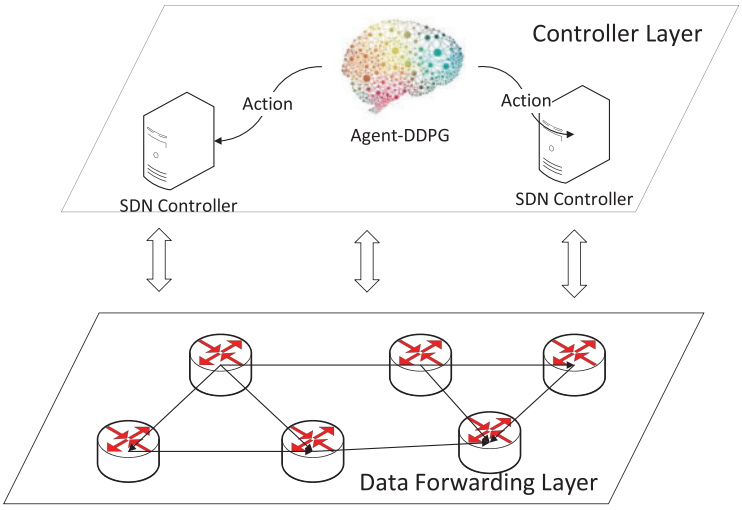

Figure 2: Overall system architecture for deep reinforcement learning-based routing in SDN

4 Deep Reinforcement Learning Based Routing Algorithm in SDN

The deep reinforcement learning used in this paper uses the DDPG algorithm, which has been described in detail about the algorithm training process in the previous section. The DDPG algorithm can be used for continuous action space scenarios, and the neural network used to record the reward values is used for training records. The design and implementation details of each element of DDPG are described in detail below. A directed graph G (V, E) is used to represent the routing network, where V denotes the set of all routes and E denotes the set of links between routes. The number of nodes N = |V|, and the number of links L = |E|.

State: In the RL process, the state reflects the characteristics of the current environment in which the agent is located. In DRL based routing scenarios, states represent the transmission status of packets in the network. It starts from the source node to the destination node. The total number of nodes in the network is N, and the packet goes through each node in the network. For each QoS metric, a |N| ∗ |N| two-dimensional matrix R is defined. d_{ij} stands for bandwidth, delay, packet loss rate and jitter rate in the current network packet in the QoS metric per unit time from the source node d_i to d_j. The state matrix is shown in Eq. (6).

$$R = \begin{bmatrix} d_{11} & \cdots & d_{1N} \\ \vdots & \ddots & \vdots \\ d_{N1} & \cdots & d_{NN} \end{bmatrix} \tag{6}$$

Action: Action is the route that the agent chooses for the next hop based on the current state and reward. In routing as is the specific routing rules issued by the agent to the network. Suppose the network is equipped with E edges, the set of actions is defined as $A = [a_1, a_2, \ldots, a_{|E|}]$. Each communication link $(i,j) \in E$ in this network.

Reward Function: According to the current network state and the behavior made by the agent the shift to the next network state feedback to the reward, the reward can be set according to different networks with different indicators of the reward function. In this paper the reward design parameters are delay D, bandwidth B, packet loss rate L, jitter rate J of QoS, delay, jitter rate, packet loss rate in QoS the smaller represents the better network quality. Then the reward function is shown in Eq. (7), where w_1, w_2, w_3, and w_4 take values in the range of (0,1).

$$Reward = -d_{ij} * w_1 + b_{ij} * w_2 - l_{ij} * w_3 - j_{ij} * w_4 \tag{7}$$

Routing Optimization: The routing optimization algorithm based on DDPG under SDN is shown in Algorithm 1. The algorithm initializes the parameters of critic network and actor network as well as the target network parameters and replay buffer. In the training process of DDPG algorithm, the current routing network state s_t is obtained from the SDN controller, and the action a_t is made according to the current policy and noise, and the current state is converted into s_{t+1}, and the reward value r_t obtained currently is calculated. The size of the reward value r is influenced by the routing QoS metrics delay, bandwidth, packet loss rate, and jitter rate. After that, a set of data (s_t, a_t, r_t, s_{t+1}) is put into the replay buffer, and when the set target number is reached, the specified number is randomly taken from the replay buffer to update the parameters of the critic network, actor network and each parameter in the target network until each parameter reaches convergence. If the source node i to the destination node wants to find the QoS optimal path, the agent obtains the routing network information from the SDN controller and outputs the path that satisfies the largest value of y_{ij} in node i to j. Thus, the routing is optimized to improve the QoS so that the network can provide services more stably.

Algorithm 1: Routing optimization based on DDPG in SDN

Input: The origin d_i and the destination d_j;
Output: d_i to d_j to the optimal path p_{ij};

Set critic-network $Q(s,a|\theta^Q)$ and actor-network $u(s|\theta^u)$ randomly generated weight θ^Q and θ^u;
Set target network parameters $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{u'} \leftarrow \theta^u$;
Empty experience replay buffer R;
Set $p_{ij} = \{\}$;
for episode 1 **to** M **do**:
    Initializes a random process N for action exploration;
    None example initialize the current route status s_1;
    for t←1 **to** T **do**:
        the current network information is obtained from SDN controller;
        choose the $a_t = u(s_t|u) + N_t$ based on the current strategies and detection noise;
        execute action a_t and the environment state changes to s_{t+1};
        update the network information from SDN controller;
        calculate the current reward value r_t according to Eq. (7);
        if number of relay buffer < N **then**:
            store (s_t, a_t, r_t, s_{t+1}) data to R;
        else:
            M data sets are randomly drawn from relay buffer R;
            calculate value y_i according to Eq. (1);
            update critic θ^Q according to Eq. (2) and actor θ^u according to Eq. (3);
            update target network parameters according to Eqs. (4) and (5);
            until the network parameters converge;
        end
    end
end
while $d_k == d_j$:
calculate the $reward_{\max}$ acquired from the next hop path d_k;
    p_{ij} adds a routing path d_k;
    k++;
end
return p_{ij}

5 Experiment

5.1 Environment and Parameter Setting

The network environment simulated in this paper is barabasi-albert network type with 500 nodes and the average node degree is 3. The DRL module is implemented based on PyTorch framework. The relevant parameters for using deep reinforcement learning are set as follows: The random sampling training capacity is M = 16, the capacity of the cache pool is N = 1000, the discount factor $\gamma = 0.99$, and the neural network parameters are 0.01,$\tau = 0.5$. The number of training rounds is 30, and the iteration step of each round is 2000. The reward function parameters w_1, w_2, w_3, and w_4 are set to be 0.9. Specific parameter settings are shown in Table 1.

Table 1: Parameter setting

Parameter	Set
Number of network nodes	500
Node degree	3
N	1000
M	16
γ	0.99
τ	0.5
Learning rate	0.005
Steps	50

5.2 Experimental Result

Regarding the DDPG algorithm, the number of network packets is 5000 for 50 rounds of training and learning, and the Shortest Path uses Dijkstra algorithm to find the shortest route. The meaning of the experimental performance index is as follows:

Average delay time per episode: The average of the delay times (s) in each training round.

Percent of empty nodes per episode: Percentage of idle nodes in each training round.

Average delay time: The average of the delay time (s) experienced by the packet transmission

Average packet idle time: The average time (s) that a packet is idle between sending and receiving.

Average non-empty queue length: The average number of non-empty elements in a queue.

Maximum number of packet nodes held: The maximum number of packet nodes held.

Percent of working nodes at capacity: The percentage of worker nodes that have reached their maximum capacity.

Fig. 3 shows that with the increase of training rounds, the average delay time of DDPG algorithm shows a downward trend. Before the number of rounds reaches 10, the downward trend is large; after the number of rounds reaches 10, the downward trend gradually decreases; when the number of rounds reaches 20, the average delay time gradually reaches convergence. The results indicate that DDPG algorithm is more effective in routing environment.

Fig. 4 shows that the percentage of idle nodes increases with the number of training times. When the training round reaches 10, the percentage of idle nodes increases greatly, and when the training reaches 20, the proportion of idle nodes gradually converges. After the training of DDPG algorithm reaches a certain convergence, the optimal path of routing is selected, that is, the path with relatively fewer routing hops and faster transmission will be selected, so the proportion of occupied nodes will be reduced.

Fig. 5 shows the average delay time comparison as the number of packets increases. It can be seen from the figure that with the increase of data packets, the delay of Shortest Path algorithm increases greatly, and the delay of DDPG algorithm increases slowly. When the data packet reaches 5000, the average delay of DDPG algorithm is 182.55 units less than that of Shortest Path algorithm. DDPG algorithm has better performance in terms of delay.

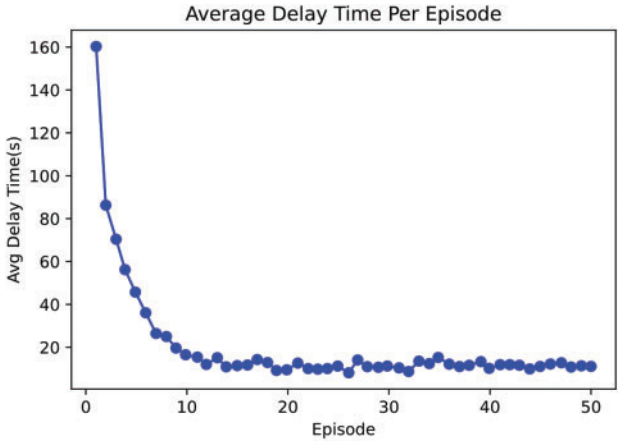

Figure 3: Average delay time per episode

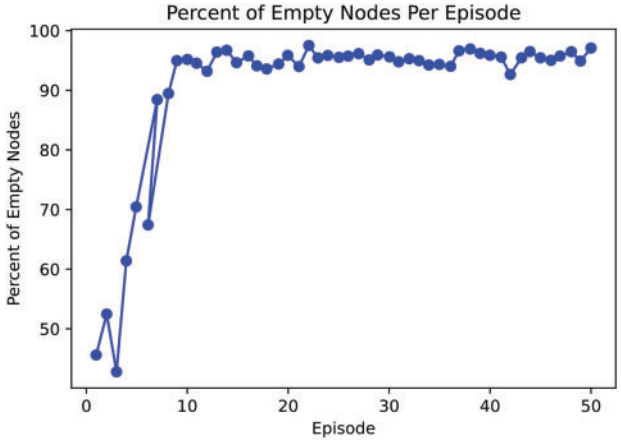

Figure 4: Percent of empty nodes per episode

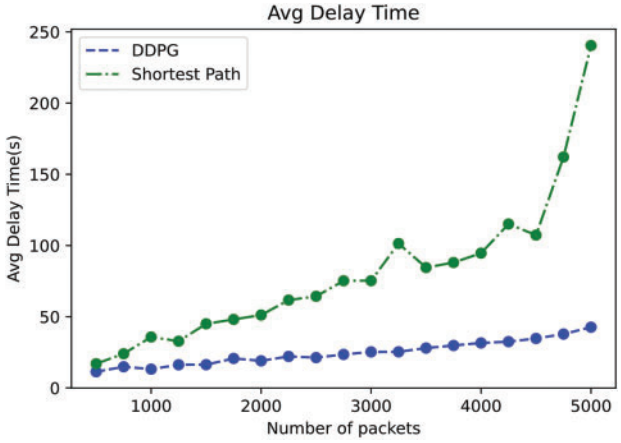

Figure 5: Average delay time

Fig. 6 shows the average waiting time of data packets with the increase of data packets. It can be seen from the figure that the waiting time of data packets in Shortest Path algorithm increases greatly with the increase of the number of data packets, while the waiting time of data packets in DDPG algorithm increases slowly with the average waiting time of data packets less. The average packet waiting time of DDPG algorithm is 82.2 units less than Shortest Path algorithm.

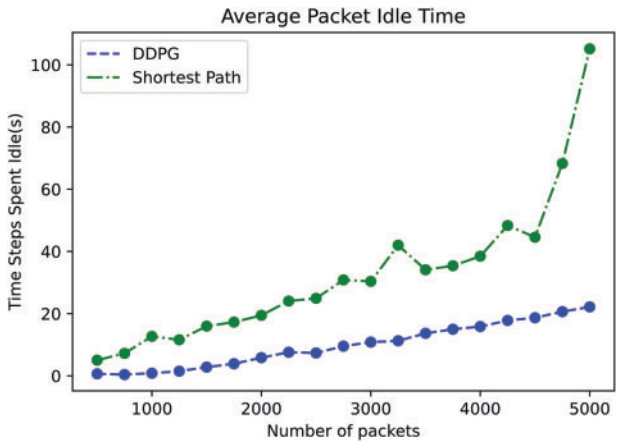

Figure 6: Average packet idle time

Fig. 7 illustrates the change in the length of the average nonempty queue as the number of packets increases. It can be seen from the figure that with the increase of the number of data packets, the change of the average non-empty queue length in the Shortest Path algorithm increases greatly, and the average non-empty queue length in the DDPG algorithm is stable without large fluctuations. When the data packet reaches 5000, the average non-empty queue length of DDPG algorithm is 61.481 less than that of Shortest Path algorithm.

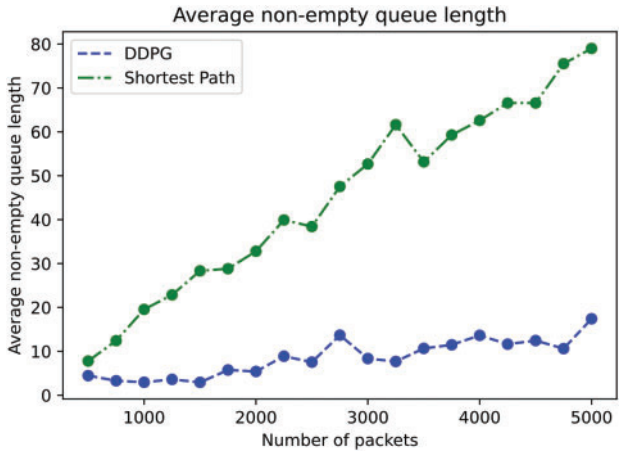

Figure 7: Average non-empty queue length

Fig. 8 illustrates the variation of the maximum number of packet nodes as the number of packets increases. It can be seen from the figure that with the increase of data packets, the maximum number of packet nodes in the Shortest Path algorithm changes little, but is above 140, while the maximum

number of packet nodes in the DDPG algorithm increases gradually, but the number of packet nodes in the DDPG algorithm is less than the maximum number of packet nodes in the Shortest Path algorithm.

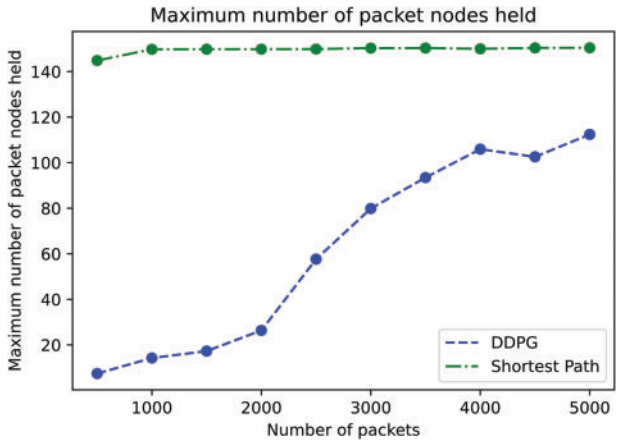

Figure 8: Maximum number of packet nodes held

Fig. 9 illustrates the capacity percentage of the working nodes as the packets increase. It can be seen from the figure that the capacity percentage of working nodes of the Shortest Path algorithm increases more than that of the DDPG algorithm. When the data packet is 5000, the workload percentage of DDPG is nearly 25% less than that of the Shortest Path algorithm.

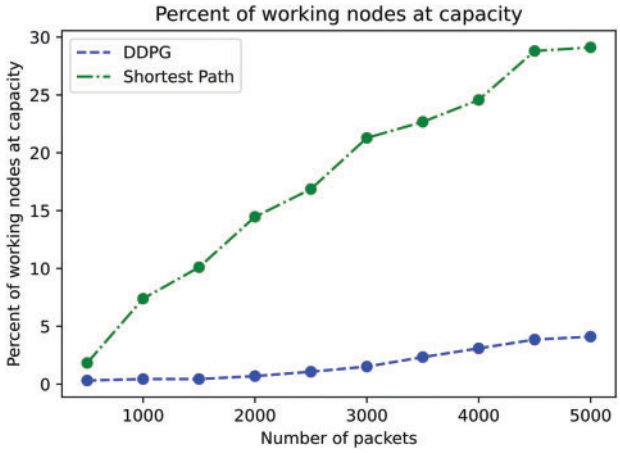

Figure 9: Percent of working nodes at capacity

To show the impact of different w values in the reward function on the performance, we conduct comparative experiments. Fig. 10 illustrates the effect of different values of w on the delay. Among them, when w_1 is set to 1, only the delay factor is considered, while the other w values are set to 0. When w_1 is set to 1/4, the other values of w are also set to 1/4, indicating that all factors have the same weight. When w_1 is set to 0, the delay factor is not considered. According to the adjustment to different performance requirements, the w value can be adjusted accordingly.

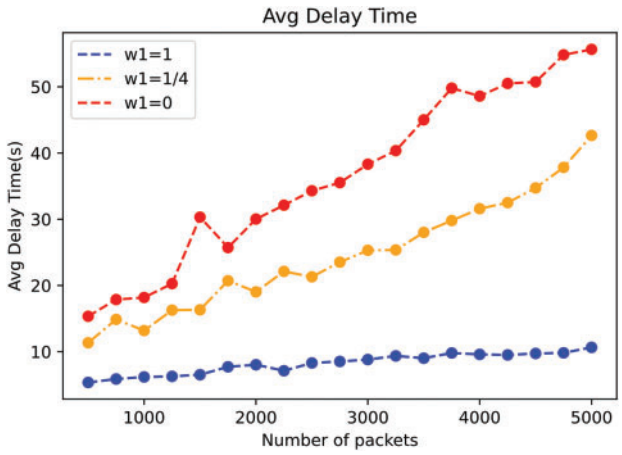

Figure 10: Effect of different w values on latency

In summary, DDPG algorithm is employed in the routing optimization scenario, in the face of a large number of traffic data requests, compared with the traditional routing algorithm, the optimal routing path is customized, the QoS optimal path is selected comprehensively, the packet queue length is reduced, the data request can be processed in time, the packet waiting time is reduced, and the delay is reduced.

6 Conclusion

To achieve efficient and fast electronic certificate issuance in power business scenarios, traditional network routing schemes pose significant challenges. To solve the existing problems, we propose a QoS routing optimization method based on deep reinforcement learning in SDN. Through the combination of SDN architecture and DDPG algorithm, a reward function based on QoS measurement was designed to obtain the optimal network transmission path. Through the characteristics of SDN transfer and control separation, the optimal network transmission path was uniformly communicated to the data transmission layer, so as to improve the network transmission rate and provide the optimal QoS. The experimental results show that the algorithm can effectively improve the efficiency of network transmission, reduce the delay of processing data packets, and effectively reduce network congestion. In the future, we will consider multiple objectives such as security, performance and resource utilization into the routing optimization algorithm, find the optimal solution to the multi-objective problem, and compare multiple routing optimization algorithms at the same time.

Acknowledgement: All authors sincerely thank all institutions for providing resources and research conditions. We would like to thank all the members of the research group for their suggestions and support, which have provided important help and far-reaching influence on our research work.

Funding Statement: This work has been supported by State Grid Corporation of China Science and Technology Project "Research and Application of Key Technologies for Trusted Issuance and Security Control of Electronic Licenses for Power Business" (5700-202353318A-1-1-ZN).

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Yu Song, Ao Xiong; analysis and interpretation of results: Xuesheng Qian, Nan Zhang,

Wei Wang; draft manuscript preparation: Yu Song. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The authors confirm that the data supporting the findings of this study are available within the article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Ali and B. H. Roh, "Quality of service improvement with optimal software-defined networking controller and control plane clustering," *Comput. Mater. Contin.*, vol. 67, no. 1, pp. 849–875, 2021. doi: 10.32604/cmc.2021.014576.

[2] Tomovic, Slavica, I. Radusinovic, and N. Prasad, "Performance comparison of QoS routing algorithms applicable to large-scale SDN networks," in *IEEE EUROCON 2015-Int. Conf. Comput. Tool (EUROCON)*, Salamanca, Spain, IEEE, 2015.

[3] E. Akin and T. Korkmaz, "Comparison of routing algorithms with static and dynamic link cost in software defined networking (SDN)," *IEEE Access*, vol. 7, pp. 148629–148644, 2019. doi: 10.1109/ACCESS.2019.2946707.

[4] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *J. Netw. Comput. Appl.*, vol. 67, no. 4, pp. 1–25, 2016. doi: 10.1016/j.jnca.2016.03.016.

[5] W. Li, Y. Wang, Z. Jin, K. Yu, J. Li and Y. Xiang, "Challenge based collaborative intrusion detection in software-defined networking: An evaluation," *Digital Commun. Netw.*, vol. 7, no. 2, pp. 257–263, 2021. doi: 10.1016/j.dcan.2020.09.003.

[6] X. Xu *et al.*, "Secure service offloading for internet of vehicles in SDN-enabled mobile edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3720–3729, 2021. doi: 10.1109/TITS.2020.3034197.

[7] M. Cicioğlu and A. Çalhan, "Energy-efficient and SDN-enabled routing algorithm for wireless body area networks," *Comput. Commun.*, vol. 160, pp. 228–239, 2020.

[8] D. Yang *et al.*, "DetFed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 5162–5178, 2023. doi: 10.1109/TMC.2023.3303017.

[9] R. Chen *et al.*, "A three-party hierarchical game for physical layer security aware wireless communications with dynamic trilateral coalitions," *IEEE Trans. Wirel. Commun.*, Early Access, 2023.

[10] W. Zhu *et al.*, "Edge-Assisted video transmission with adaptive key frame selection: A hierarchical DRL approach," in *2023 Biennial Symp. Commun. (BSC)*, Montreal, QC, Canada, IEEE, 2023, pp. 89–94.

[11] R. Chen *et al.*, "A DRL-based hierarchical game for physical layer security with dynamic trilateral coalitions," in *ICC 2023–IEEE Int. Conf. Commun.*, Rome, Italy, IEEE, 2023.

[12] S. Tomovic and I. Radusinovic, "Fast and efficient bandwidth-delay constrained routing algorithm for SDN networks," in *2016 IEEE NetSoft Conf. Workshops (NetSoft)*, Seoul, Korea (South), IEEE, 2016, pp. 303–311.

[13] A. Verma and N. Bhardwaj, "A review on routing information protocol (RIP) and open shortest path first (OSPF) routing protocol," *Int. J. Future Gener. Commun. Netw.*, vol. 9, no. 4, pp. 161–170, 2016. doi: 10.14257/ijfgcn.2016.9.4.13.

[14] S. S. Vinod Chandra, and S. Anand Hareendran, "Modified smell detection algorithm for optimal paths engineering in hybrid SDN," *J. Parallel Distr. Comput.*, vol. 187, no. 1, pp. 104834, 2024. doi: 10.1016/j.jpdc.2023.104834.

[15] A. R. Ananthalakshmi, P. C. Sajimon, and S. S. Vinod Chandra, "Application of smell detection agent based algorithm for optimal path identification by SDN controllers," in *Advances in Swarm Intelligence,* Fukuoka, Japan: Springer International Publishing, Jul. 27, Aug. 1, 2017, vol. 10386, pp. 502–510.
[16] D. Gopi, S. Cheng, and R. Huck, "Comparative analysis of SDN and conventional networks using routing protocols," in *2017 Int. Conf. Comput., Inform. Telecommun. Syst. (CITS)*, Dalian, China, IEEE, 2017, pp. 108–112.
[17] A. Shirmarz and A. Ghaffari, "An adaptive greedy flow routing algorithm for performance improvement in software-defined network," *Int. J. Numerical Model.: Elect. Netw., Devices Fields*, vol. 33, no. 1, pp. 2676, 2020. doi: 10.1002/jnm.2676.
[18] X. Xu, H. Hu, Y. Liu, J. Tan, and H. Zhang, "Moving target defense of routing randomization with deep reinforcement learning against eavesdropping attack," *Digital Commun. Netw.*, vol. 8, no. 3, pp. 373–387, 2022.
[19] 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. doi: 10.1109/ACCESS.2018.2877686.
[20] G. Stampa *et al.*, "A deep-reinforcement learning approach for software-defined networking routing optimization," arXiv preprint arXiv:1709.07080, 2017.
[21] O. J. Pandey, T. Yuvaraj, J. K. Paul, H. H. Nguyen, K. Gundepudi and M. K. Shukla, "Improving energy efficiency and QoS of lpwans for IoT using q-learning based data routing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 365–379, 2021. doi: 10.1109/TCCN.2021.3114147.
[22] C. Zhao *et al.*, "DRL-M4MR: An intelligent multicast routing approach based on DQN deep reinforcement learning in SDN," arXiv preprint arXiv:2208.00383, 2022.
[23] H. Zhang and J. Yan, "Performance of SDN routing in comparison with legacy routing protocols," in *2015 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Disc.*, Xi'an, China, IEEE, 2015.
[24] S. Singh and R. K. Jha, "A survey on software defined networking: Architecture for next generation network," *J. Netw. Syst. Manage.*, vol. 25, no. 2, pp. 321–374, 2017. doi: 10.1007/s10922-016-9393-9.
[25] J. Clifton and E. Laber, "Q-learning: Theory and applications," *Annu. Rev. Stat. Appl.*, vol. 7, no. 1, pp. 279–301, 2020. doi: 10.1146/annurev-statistics-031219-041220.
[26] S. Han, W. Zhou, S. Lü, and J. Yu, "Regularly updated deterministic policy gradient algorithm," *Knowl. Based Syst.*, vol. 214, no. 7540, pp. 106736, 2021. doi: 10.1016/j.knosys.2020.106736.
[27] H. Tan, "Reinforcement learning with deep deterministic policy gradient," in *2021 Int. Conf. Artif. Intell., Big Data Algor. (CAIBDA)*, Xi'an, China, IEEE, 2021, pp. 82–85.