Computers, Materials & Continua
DOI: 10.32604/cmc.2022.031698
Article

Tech Science Press

A Novel Method for Routing Optimization in Software-Defined Networks

Salem Alkhalaf* and Fahad Alturise

Department of Computer, College of Science and Arts in ArRass Qassim University, Ar Rass, Qassim, Saudi Arabia
*Corresponding Author: Salem Alkhalaf. Email: s.alkhalaf@qu.edu.sa
Received: 25 April 2022; Accepted: 07 June 2022

Abstract: Software-defined network (SDN) is a new form of network architecture that has programmability, ease of use, centralized control, and protocol independence. It has received high attention since its birth. With SDN network architecture, network management becomes more efficient, and programmable interfaces make network operations more flexible and can meet the different needs of various users. The mainstream communication protocol of SDN is OpenFlow, which contains a Match Field in the flow table structure of the protocol, which matches the content of the packet header of the data received by the switch, and completes the corresponding actions according to the matching results, getting rid of the dependence on the protocol to avoid designing a new protocol. In order to effectively optimize the routing for SDN, this paper proposes a novel algorithm based on reinforcement learning. The proposed technique can maximize numerous objectives to dynamically update the routing strategy, and it has great generality and is not reliant on any specific network state. The control of routing strategy is more complicated than many Q-learning-based algorithms due to the employment of reinforcement learning. The performance of the method is tested by experiments using the OMNe++ simulator. The experimental results reveal that our PPO-based SDN routing control method has superior performance and stability than existing algorithms.

Keywords: Reinforcement learning; routing algorithm; software-defined network; optimization

1 Introduction

Software-Defined Networking (SDN) breaks the vertical structure of the traditional network model, separates control and forwarding, and provides more possibilities for network innovation and evolution. It is considered to be the main architecture of future networks, but in recent years, network scale Continuing to expand, network traffic has also grown exponentially, and the requirements for network control have become increasingly sophisticated [1–5]. Traditional network routing schemes rely primarily on the shortest path method, which has drawbacks such as sluggish convergence and difficulties coping with network congestion [6–9]. As a result, establishing an effective optimization

technique for network routing through the controller is a critical step in ensuring network services and advancing SDN [10].

Machine learning, particularly deep learning technology, has gained a lot of attention because of its outstanding performance in large-scale data processing, classification, and intelligent decision-making [11–14]. Many researchers employ SDN networks to overcome the challenges in network operation and administration [15–17]. The authors of [18] presented a machine learning-based route pre-design approach. This method first extracts network traffic characteristics using a clustering algorithm (such as Gaussian mixture model or K-means model), then predicts traffic demand using a supervised learning method (such as extreme learning machine), and finally handles traffic routing problems using analytic hierarchy process (AHP)-based automatic adapt dynamic algorithms. The authors devised a heuristic technique to improve SDN routing in [19], but the algorithm's performance does not remain steady as the network evolves. The ant colony algorithm and the genetic algorithm [20–21] are examples of heuristic algorithms that have been used to address routing problems. These algorithms, on the other hand, perform poorly in terms of generalization. When the network's condition changes, the parameters of these heuristic algorithms must also change, making the process difficult to repeat consistently.

Reinforcement learning is frequently used to tackle route optimization issues because it can conduct dynamic decision management by continually engaging with the environment. Reference [22] offers a reward scheme for network service quality measurements using Q-Learning. An end-to-end adaptive hypertext transfer protocol (HTTP) streaming intelligent transmission architecture based on a partially observable Markov decision process with a Q-Learning decision algorithm is shown in reference [23]. These Q-Learning based reinforcement learning algorithms require solving of Q-tables for control decisions [24]. However, the state space of SDN network is very huge, and the algorithm based on Q-Learning cannot describe the state well. At the same time, as a value method in reinforcement learning, the Q-Learning only works in the discrete action space for the output control action, and the decision action space is very limited. Therefore, it is a huge challenge to design a dynamic routing strategy that can perform fine-grained analysis and control of SDN networks.

This work proposed the proximal policy optimization (PPO) technique, which is based on this algorithm, for making routing scheme judgements in the SDN control plane. The following are some of the benefits of the suggested algorithm:

- First, the approach employs a neural network to precisely determine the Q value of the SDN network state, bypassing the Q table's restrictions and inefficiencies.
- Simultaneously, the algorithm is part of the reinforcement learning strategy technique, which may produce a finer-grained control scheme for network decision-making.
- Finally, the system may adapt the reinforcement learning reward function according to different optimization targets to dynamically optimize the routing method. Because it is not dependent on any specific network state and has great generalization performance, this technique successfully performs black-box optimization.

The performance of the algorithm is evaluated by experiments based on Omnet++ simulation software. The experimental findings reveal that the proposed algorithm outperforms both the classic shortest path-based static routing approach and the QAR method reported in [25] in terms of performance and stability.

2 Background

2.1 SDN and Knowledge Plane Network

The conventional Internet architecture is primarily based on the OSI seven-layer or TCP/IP four-layer protocol model, with data transfer between network devices taking place at each layer via matching network protocols (exchange, routing, labeling, security and other protocols). The general workflow is as follows: the neighbor establishment-information sharing-path selection is realized in three steps. In addition, the transmission of information between network devices adopts a typical distributed architecture. The devices exchange information in the form of "baton", then establish database information, and then transmit data according to the relevant path algorithm (such as Dijkstra's shortest path algorithm). Each layer of equipment calculates independently, has independent controllers and forwarding hardware, and communicates through protocols.

This distributed architecture facilitated the Internet's flourishing in the past when protocol specifications were incomplete. However, with the gradual unification and improvement of communication equipment protocols, the distributed architecture has gradually reached a bottleneck, and many problems have been highlighted, such as redundant transmission table information, difficult traffic control, and inability of equipment to customize transmission. The root cause of these problems is that the data and control of network equipment are coupled in the traditional architecture, and the equipment does not have an open programmable interface, so it is impossible to separate data forwarding and data control.

The concept of SDN is proposed to solve this problem. Its core idea is to centralize all the information on the network to a core controller, so that the controller can directly manipulate the underlying infrastructure in a centralized way. This method can provide great flexibility for data transfer, so it has been widely used in recent years [26].

It separates the control plane from the data plane, allowing the control plane's controller to fully comprehend all the information in the network, resulting in a more efficient and speedy overall routing system. A knowledge plane network paradigm is presented in [27], which adds a knowledge plane to the typical SDN design. As shown in Fig. 1, the knowledge-defined networking (KDN) needs to process the information collected by the lower plane and use it. The machine learning methods are used to make decisions about network management, thereby improving the overall efficiency of SDN.

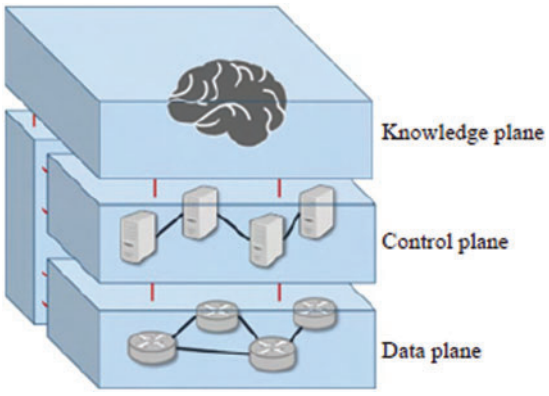

Figure 1: Illustration of KDN structure

The routing strategy optimization algorithm proposed in this paper is also developed based on this control model architecture, and the reinforcement learning method will be introduced in the knowledge plane to centrally and dynamically manage the routing scheme of the data plane.

2.2 Reinforcement Learning and PPO Algorithms

Reinforcement learning is a machine learning paradigm that uses Markov decision processes to learn the interdependence of states, actions, and rewards between a decision-making agent and its environment. Fig. 2 depicts the interaction process in reinforcement learning between the decision agent and the environment.

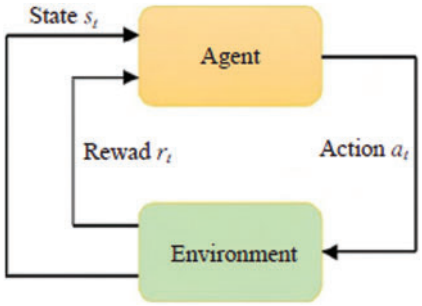

Figure 2: The interactive process of decision-making subject and environment in reinforcement learning

The PPO [28] is based on Actor-Critic architecture, deploys a TRPO-based model-free policy gradient approach [29]. The algorithm's purpose is to find a decision policy $\pi_\theta(s)$ in the aforementioned interaction process that maximizes the cumulative return. Among these, s is the model's input, which is a vector that describes the present environment. In network control, it can be the traffic matrix of the entire network and the topology adjacency matrix. The decision function outputs an action, which can be a weight describing the network link in the routing strategy, and an optimal routing scheme can be uniquely determined through the weight. A neural network can fit the decision function, and is the network parameter θ.

The policy gradient algorithm works by estimating the policy gradient and using gradient ascent to update the policy parameters. The policy loss $J(\theta)$ and its gradient are estimated by running the policy in the environment to obtain samples:

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\sum_t R(s_t, a_t)\right] = E_{\tau-\pi_\theta(\tau)}[R(\tau)] \tag{1}$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum\nolimits_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\right) R(\tau)\right] \tag{2}$$

The primary goal of policy gradient approaches is to minimize the variation of gradient estimations so that better policies may be developed. The end results after deploying such architecture is as follows:

$$Q^\pi(s, a) = \sum_t E[R(s_t, a_t)|s, a] \tag{3}$$

$$V^\pi(s) = \sum_t E_{\pi_\theta}[R(s_t, a_t)|s] \tag{4}$$

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{5}$$

where $A^{\pi}(s,a)$ is the benefit function in the above formula which assesses the quality of a certain choice in the state, the algorithm's ultimate purpose is to optimise to constantly raise this function. The Actor network is the ultimate decision-making strategy network in the Actor-Critic architecture, the goal of the Critic network is to examine the current strategy's benefit function. On this basis, the PPO method transforms the optimization goal into the surrogate objective function below, which improves the model's stability and efficiency during training.

$$L(\theta) = E\left[\min\left(r_t(\theta) A_t, clip\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) A_t\right)\right] \tag{6}$$

Among them, A_t is the advantage function, ϵ is the hyperparameter, $r_t(\theta)$ is the important sampling correction parameter, and the expression is as follows:

$$r_t(\theta) = \frac{\pi_{\theta}(a_t, s_t)}{\pi_{\theta_{\text{old}}}(a_t, s_t)} \tag{7}$$

Finally, when the algorithm is working, it is necessary to set up two networks for Actor and Critic. To get the interaction history, the Actor network takes decisions for the agent. The agent objective function can be updated according to the above formula.

3 Algorithm Design

This section introduces the proposed algorithm. By implementing the algorithm in SDN, performance variables like as throughput, latency, and jitter may be automatically modified, allowing for real-time network control and considerably reducing network operational and maintenance strain.

3.1 Task Modeling

The most significant aspect of using reinforcement learning to enhance SDN routing is determining the environmental state, reward, and decision-action space gained by the decision-making agent. As a dynamic optimization algorithm, the reinforcement learning does not need to perceive the amount that will not change in the whole process when making a decision. For example, when routing optimization, the specific physical link and the corresponding network topology have been given, the SDN controller only needs to continuously plan the routing scheme according to the current network traffic information. Therefore, for a specific SDN network, the state s input by the agent can be represented by the vector transformed by the traffic matrix of the current network load. Representation learning has already been obtained during the training process of the body.

After the agent perceives the traffic situation, it needs to give an action a, which can determine the only optimal solution for the current network environment. Here, the action is set to a vector representing all link weights. Through the link weight vector, the Floyd algorithm can be used to uniquely determine a set of optimal routing schemes for the network.

The reward obtained by the agent is related to the overall performance indicators of the network, such as network delay, throughput, or a comprehensive reward considering various parameters. For example, Eq. (8) is a reward function that comprehensively considers multiple performance indicators.

$$R_{i \to j} = R(i \to j | s_t, a_t) = -h(a_t) - \alpha delay_{ij} + \beta BW_{ij} + \gamma loss_{ij} \tag{8}$$

Among them, $s_t$represents the current state of the network, and a_t is the action generated by the control agent, through which the network link weight is adjusted, and to achieve the unique optimum

routing method, the network's point-to-point optimal path is recalculated according to the link weight. Assume that the best path after adjusting the link weights is p_{ij}. The function h represents the cost of adjusting this path, such as the action effect on the switching operation. $\alpha, \beta, \gamma \in [0, 1)$ are adjustable weights, *delay* represents the delay time under the path, BW is the bandwidth, loss is the packet loss rate, and he operation and maintenance approach can determine the reward function in a variety of ways.

3.2 Proposed Framework

Fig. 3 depicts the proposed framework. The three variables that the PPO agent uses to interact with the environment are Status, Action, and Reward. The traffic matrix (TM) of the present network load is one of them, and the agent's action to the environment is to modify the weight vector of the network's connections. Through the weight vector, a routing scheme in a network can be uniquely determined. The pseudo code of the training process is shown in Algorithm 1.

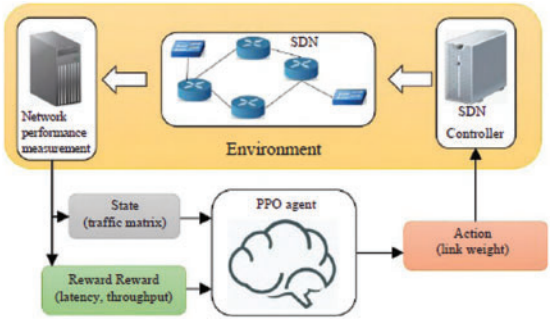

Figure 3: Overall model architecture

Algorithm 1: Proposed PPO method

1: Initialize the PPO decision function $\pi_\theta(s)$ and the value function $V_\omega(s)$
2: While task $\neq$ done
3: $k = 0$
4: $\theta_{\text{old}} \leftarrow \theta$
5: Randomly initialize environment s_k and cache list *buffer*
6: While $k < K$ do:
7: Get action $a \sim \pi_\theta(s_k)$
8: $s_{k+1}, r_k = Env(s_k, a_k)$
9: Store $[s_k, a_k, r_k]$ in buffer
10: $k = k + 1$
11: end while
12: Calculate A_k and Q_k based on the data in the buffer (s_k, a_k)
13: Calculate $L(\theta)$ according to Eq. (6)
14: $J(\theta) = \frac{1}{K}(Q_k - V_\omega(s_k))^2$
15: $\theta \leftarrow \theta + \alpha_\theta \nabla L(\theta)$
16: $\omega \leftarrow \omega - \alpha_\omega \nabla J(\omega)$
17: end while

In the above algorithm, the 6th row to the 11th row is the sampling process, and the 12th row to the 16th row is the training process of the model parameters. The *Env* in line 8 encapsulates

the environment other than the PPO decision body. In addition, the decision function $\pi_\theta(s)$ usually represents a high-dimensional normal distribution, and the action a is sampled from this distribution.

The PPO agent's training goal is to discover the optimum action a based on the input states s in order to maximize the cumulative reward. The following is a summary of the whole working process: The PPO agent can collect precise network status s and identify the best operation, that is, given a set of link weights $[\omega_1, \omega_2, \ldots, \omega_n]$, using the SDN controller's network analysis and measurement. The new flow path is recalculated using updated weights, and the Floyd shortest path method may be used to solve the particular path creation using the weights. As a result, the SDN controller generates new rules in order to build a new route. Following the path update, the following network analysis and measurement produces the reward r and the new network state, allowing for iterative network optimization.

The traditional machine learning-based routing optimization algorithm learns the routing scheme from the network data of a specific configuration, so it can only work under the corresponding configuration. When the network hardware equipment changes, the routing scheme becomes invalid. The suggested approach is adaptive because it can update the agent gradient in interactions with the network environment using current experience fragments. That is, when the physical equipment in the network changes locally, the agent may make a rational judgement to acquire the best routing scheme for the optimization aim. Furthermore, as compared to existing algorithms, the proposed algorithm may directly output the actions, allowing for more precise network routing management. Furthermore, the suggested technique establishes a direct relationship between the output action vector and routing performance, making it easier to train the agent's neural network parameters.

4 Experimental Results

4.1 Environment

The computer hardware configuration used in the experiment is NVIDIA Geforce1080Ti GPU, 32 GB memory, i9-9900k CPU, and Ubuntu16.04 as the operating system. We used tensorflow as the code framework to build the algorithm, and use OMNeT++ [30] as the software for network simulation. The differences in routing performance between the proposed algorithm proposed and the traditional shortest-path-based mainstream routing algorithm and the randomly generated routing strategy are compared.

The Sprint structure network [31] is used in the experiment, which has 25 nodes and 53 connections with the same bandwidth on each link. The experiment sets several different levels of traffic load (TL) for the network structure to simulate real network scenarios. Each distinct TL level represents a proportion of a network's overall capacity. For the same level of traffic load, the gravity model is used [32] to generate a variety of different traffic matrices.

The PPO decision agent is trained and tested on various levels of traffic load to verify the efficiency of the proposed method.

4.2 Convergence and Efficiency Evaluation

The experiment initially tested the suggested algorithm's training on different levels of traffic loads, utilizing four traffic loads: 10%, 40%, 70%, and 100% of the full network capacity, respectively. 250 traffic matrices are randomly generated under each traffic load, of which 200 are used as training environments and 50 are used as test environments. When training the model, the average performance of the model in the test environment is tested every 1000 steps of training for each level of traffic load.

The OMNeT++ is used to obtain performance parameters such as network latency given the traffic matrix and routing system. The final PPO model is trained for the network delay test results under different traffic loads as shown in Fig. 4. As can be seen from Fig. 4, as the training continues, the routing scheme given by the model can reduce the network delay continuously, and finally converge to the optimal value.

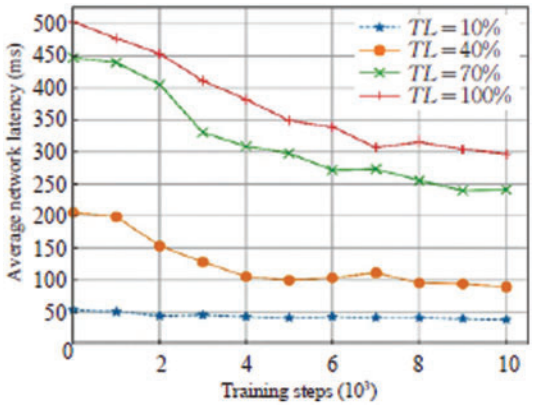

Figure 4: Comparison of the average network latency of the training process

Next, in order to validate the impact of the proposed algorithm, the delay performance of the proposed algorithm and 50000 randomly generated routing schemes under the above four different levels of traffic load is compared. After the proposed model is trained in the training environment under these several traffic loads until the performance converges, it is tested multiple times in the test environment. These 50,000 random routing schemes can provide representative comparative data for the PPO performance data. Finally, the network delay performance data is obtained through the simulation of these schemes, and the box plot is drawn as shown in Fig. 5.

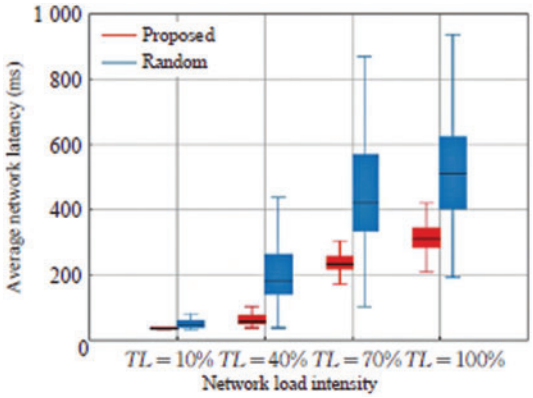

Figure 5: Comparison of network delay of the proposed and random routing algorithms

As can be seen from Fig. 5, the upper and lower quartiles of network latency are represented by the top and bottom of the rectangle, respectively, while the median of latency is represented by the line in the center of the rectangle. The greatest and minimum retardation values are represented by the top and lower ends of the straight line extending from the rectangle. The experimental findings demonstrate that the proposed algorithm's shortest latency is extremely near to the ideal result of

a randomly generated routing configuration, and the variation is very small, demonstrating the suggested algorithm's usefulness in SDN optimization.

4.3 Model Performance Comparison

The experiment compares the performance difference between the proposed algorithm and the QAR routing algorithm [25] in optimizing the average delay and maximum delay of the network, and gives the performance of the traditional shortest path-based routing algorithm as a reference. The QAR algorithm builds a model using the Q-Learning algorithm in reinforcement learning and generates routing decisions by constructing a state-action value function for the network state, which is a typical technique for SDN network routing.

In the specific experiment process, after the convergence training of the proposed model and the QAR model at different load intensity levels, 1000 traffic matrices of the same level are used as network inputs to test the average network delay and network end-to-end of these different models. Figs. 6 and 7 shows the results after taking the average value.

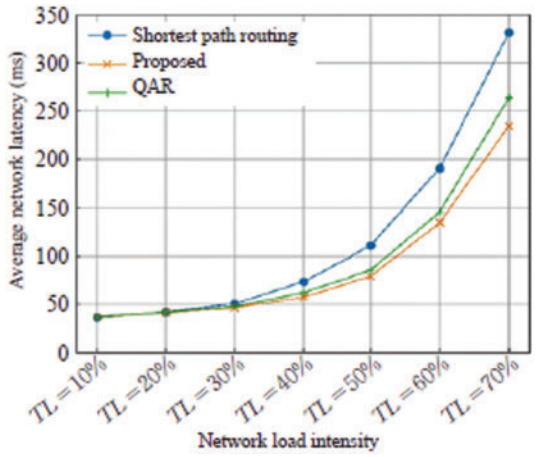

Figure 6: Comparison of the average network latency of the proposed and existing algorithms

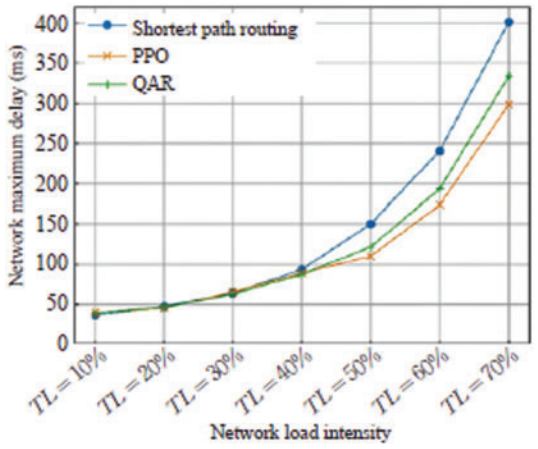

Figure 7: End-to-end maximum delay

The experiments on network throughput are also undertaken to prove the model's broad applicability for other network optimization measures. The experimental configuration is basically the same as above, the only difference is that the reward function is positively correlated with the throughput rate. The experimental results are shown in Fig. 8.

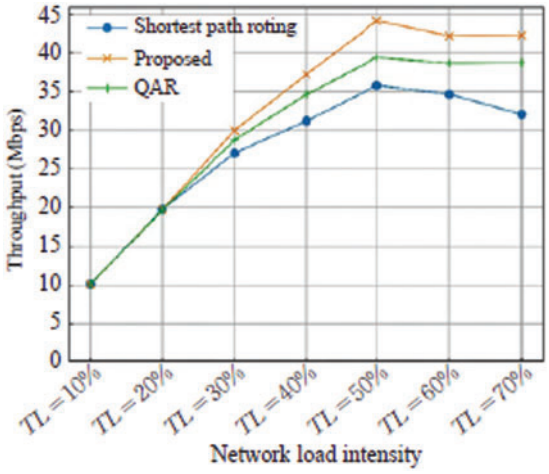

Figure 8: Comparison of network throughput of the proposed and existing algorithms

The experimental results show that the three algorithms have little difference in performance when the load intensity is low. However, when the traffic intensity is high, the traditional shortest path-based routing algorithm will cause problems such as traffic congestion, while the QAR algorithm and the proposed algorithm can effectively avoid such problems. It can also be found from the experimental data that with the traffic strength continues to increase, the routing schemes given by these two types of reinforcement learning-based models improve the performance of traditional algorithms more and more. Especially in the optimization of network throughput, when the traffic intensity exceeds 50%, the throughput of the static shortest path routing algorithm decreases rapidly, and the two routing methods based on reinforcement learning avoid this problem to a certain extent. Therefore, by introducing machine learning methods such as reinforcement learning in SDN routing management, the routing efficiency can indeed be improved.

When the proposed method is compared to the QAR algorithm, it is obvious that the proposed approach outperforms QAR in terms of overall performance. The reasons for the analysis are as follows. During the training of the QAR algorithm, the Q table needs to be continuously optimized, and the state space of the complex network is very large, and it is difficult to handle only relying on the Q table. At the same time, when the QAR outputs actions, due to the limitation of the value method, it only relies on discrete actions to generate a single action for the SDN controller in the specific control route, and cannot perform fine control. On the one hand, the proposed algorithm uses a neural network to fit the network state value, and at the same time directly outputs the action vector related to the network routing to complete more refined control, thus having better optimization performance.

In conclusion, the proposed algorithm reduces the average network delay by 29.3%, the end-to-end maximum delay by 17.4%, and the throughput by 31.77% when TL = 70% as compared with the shortest path routing method. Such superior performance indicated that the proposed algorithm has significant impact in the SDN.

5 Conclusion

On the basis of a knowledge plane network, this research provides an SDN routing optimization technique based on reinforcement learning. It uses a PPO deep reinforcement learning technique to improve SDN network routing, allowing for real-time intelligent control and administration. The suggested method exhibits high convergence and efficacy, according to the experimental findings. The suggested method, when compared to existing routing solutions, can increase network performance by providing robust and high-quality routing services.

The proposed algorithm can deal with the network routing optimization problem relatively effectively. However, because it is an on-policy technique, it is difficult to collect a significant number of training samples for large-scale complex networks in practice. Although the proposed model incorporates the importance sampling during training, the overall sample utilization during training is still low. Therefore, if some model-based reinforcement learning algorithms can be combined, there must be a more efficient solution to the problem of SDN routing optimization.

Acknowledgement: We would like to thank the anonymous reviewers for their valuable and helpful comments, which substantially improved this paper. We also would also like to thank all of the editors for their professional advice and support.

Funding Statement: The researchers would like to thank the Deanship of Scientific Research, Qassim University for funding the publication of this project.

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

References

[1] Y. Wang, C. Li, Q. Hu, J. Flor and M. Jalalitabar, "Routing and spectrum allocation in spectrum-sliced elastic optical networks: A primal-dual framework," *Electronics*, vol. 10, no. 22, pp. 1–22, 2021.

[2] X. Yu, X. Ning, Q. Zhu, J. Lv, Y. Zhao *et al.,* "Multi-dimensional routing, wavelength, and timeslot allocation (RWTA) in quantum key distribution optical networks (QKD-ON)," *Applied Sciences*, vol. 11, no. 1, pp. 1–18, 2021.

[3] E. Virgillito, A. Ferrari, A. Damico and V. Curri, "Statistical assessment of open optical networks," *Photonics*, vol. 6, no. 2, pp. 1–16, 2019.

[4] S. Ricciardi, D. Sembroiz and F. Palimieri, "A hybrid load-balancing and energy-aware RWA algorithm for telecommunication networks," *Computer Communications*, vol. 77, no. 3, pp. 85–99, 2015.

[5] F. Muro, M. Garrich, I. Castreno, S. Zahir and P. Marino, "Emulating software-defined disaggregated optical networks in a containerized framework," *Applied Sciences*, vol. 11, no. 5, pp. 1–17, 2021.

[6] Y. Zhao, Y. Ji, J. Zhang, H. Li, Q. Xiong *et al.,* "Software-defined networking (SDN) controlled all optical switching networks with multi-dimensional switching architecture," *Optical Fiber Technology*, vol. 20, no. 2, pp. 353–357, 2014.

[7] S. Zhang, X. Xue, E. Tangdiongga and N. Calabretta, "Low-latency optical wireless data-center networks using nanoseconds semiconductor-based wavelength selectors and arrayed waveguide grating router," *Photonics*, vol. 9, no. 3, pp. 1–17, 2022.

[8] S. Yan, R. Nejabati and R. Simeonidou, "Data-driven network analytics and network optimization in SDN-based programmable optical networks," in *IEEE Int. Conf. on Optical Network Design and Modeling*, Dublin, Ireland, pp. 234–238, 2018.

[9] Q. Zhou, T. Zhao, X. Chen, Y. Zhong and H. Luo, "A Fault-tolerant transmission scheme in SDN-based industrial iot (IIoT) over fiber-wireless networks," *Entropy*, vol. 24, no. 2, pp. 1–15, 2022.

[10] K. Kondepu and C. Jackson, "Fully SDN-enabled all-optical architecture for data center virtualization with time and space multiplexing," *Journal of Optical Communications and Networking*, vol. 7, no. 10, pp. 90–101, 2018.
[11] C. Jackson, K. Kondepu and Y. Ou, "COSIGN: A complete SDN enabled all-optical architecture for data centre virtualization with time and space multiplexing," in *IEEE European Conf. on Optical Communication (ECOC)*, Gothenburg, Sweden, pp. 1–3, 2017.
[12] R. Luis, H. Furukawa, G. Rademacher, B. Puttnam and N. Wada, "Demonstration of an SDM network testbed for joint spatial circuit and packet switching," *Photonics*, vol. 5, no. 3, pp. 1–15, 2018.
[13] J. Sun, G. Huang, G. Sun, H. Yu, A. Sangiah *et al.,* "A Q-learning-based approach for deploying dynamic service function chains," *Symmetry*, vol. 10, no. 11, pp. 1–12, 2018.
[14] Z. Beheshti and S. Shamsuddin, "Memetic binary particle swarm optimization for discrete optimization problems," *Information Sciences*, vol. 299, no. 3, pp. 58–84, 2015.
[15] F. Cugini and N. Sambo, "Enhancing GMPLS signaling protocol for encompassing quality of transmission (QoT) in all-optical networks," *Journal of Lightwave Technology*, vol. 26, no. 19, pp. 3318–3328, 2008.
[16] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Perez. *et al.*, "A survey on machine learning techniques for routing optimization in SDN," *IEEE Access*, vol. 9, pp. 104582–104611, 2021.
[17] Z. Fadlullah, F. Tang and B. Mao, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.
[18] E. Genesan, I. Hwang, A. Liem and M. Rahman, "SDN-Enabled FiWi-IoT smart environment network traffic classification using supervised ML models," *Photonics*, vol. 8, no. 6, pp. 1–18, 2021.
[19] F. Wang, B. Liu and L. Zhang, "Dynamic routing and spectrum assignment based on multilayer virtual topology and ant colony optimization in elastic software-defined optical network," *Optical Engineering*, vol. 56, no. 7, pp. 8877–8884, 2017.
[20] M. Parsaei, R. Mohammad and R. Javidan, "A new adaptive traffic engineering method for tele-surgery using ACO algorithm over software defined networks," *European Research in Telemedicine*, vol. 6, no. 4, pp. 173–180, 2017.
[21] Z. Zhang, G. Wu and H. Ren, "Multi-attribute-based QoS-aware virtual network function placement and service chaining algorithms in smart cities," *Computers & Electrical Engineering*, vol. 96, no. 2, pp. 3077–3089, 2021.
[22] J. Jiang, L. Hu and P. Hao, "Q-FDBA: Improving QoE fairness for video streaming," *Multimedia Tools and Applications*, vol. 77, no. 9, pp. 10787–10806, 2018.
[23] J. Guerrero and L. Lamata, "Reinforcement learning and physics," *Applied Sciences*, vol. 11, no. 18, pp. 1–24, 2021.
[24] C. Liu, W. Ju and G. Zhang, "A SDN-based active measurement method to traffic QoS sensing for smart network access," *Wireless Networks*, vol. 2, pp. 592–604, 2020.
[25] S. Lin, I. Akydiz and P. Wang, "QoS-Aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *IEEE Int. Conf. on Service Computing*, New York, USA, pp. 25–33, 2016.
[26] F. Xiong, A. Li and H. Wang, "SDN-MQTT based communication system for battlefield UAV swarms," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 41–47, 2019.
[27] A. Mestres, A. Rodrigueznatal and J. Carner, "Knowledge-defined networking," *ACM Special Interest Group on Data Communication*, vol. 47, no. 3, pp. 2–10, 2017.
[28] 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.
[29] W. Meng, Q. Zheng, Y. Shi and G. Pan, "An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 4, no. 5, pp. 1–13, 2021.
[30] D. Ramos, L. Almeida and U. Moreno, "Integrated robotic and network simulation method," *Sensors*, vol. 19, no. 20, pp. 1–18, 2019.

[31] N. Josbert, W. Ping, M. Wei and Y. Li, "Industrial networks driven by SDN technology for dynamic fast resilience," *Information Journal*, vol. 12, no. 10, pp. 1–18, 2021.
[32] M. Roughan, "Simplifying the synthesis of internet traffic matrices," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 93–106, 2005.