This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2877686, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

DROM: Optimizing the Routing in Software-Defined Networks with Deep Reinforcement Learning

Changhe Yu[1], Julong Lan[1], Zehua Guo[2], Yuxiang Hu[1]
[1]National Digital Switching System Engineering and Technological R&D Center, Zhengzhou 450002, China
[2]University of Minnesota Twin Cities, Minneapolis, MN 55455, USA

Corresponding author: Zehua Guo (e-mail: guolizihao@hotmail.com)

This work was supported in part by the National Natural Science Foundation of China for Innovative Research Groups under Grant 61521003, in part by the National Natural Science Foundation of China under Grant 61502530, in part by the National High Technology Research and Development Program (863 Program) of China under Grant 2015AA016102.

ABSTRACT This paper proposes DROM, a deep reinforcement learning mechanism for the Software-defined networking (SDN) to achieve a universal and customizable routing optimization. DROM simplifies the network operation and maintenance by improving the network performance, such as delay and throughput, with a black-box optimization in continuous times. We evaluate the DROM with experiments. The experimental results show that DROM has the good convergence and effectiveness and provides better routing configurations than existing solutions to improve the network performance, such as reducing the delay and improving the throughput.

INDEX TERMS Deep reinforcement learning, Routing optimization, Software-Defined Networking

I. INTRODUCTION

Software-Defined Networking (SDN) provides flexible traffic control by separating the control plane from the data plane, simplifying the network operation and maintenance management process, and thus is widely used as an emerging architecture for the network innovation. However, as the network control requirement becomes fine-grained, the network scale expands rapidly, and the network traffic grows exponentially, traditional static routing algorithms (e.g., OSPF) are not suitable for the SDN because of their slow convergence and slow response to the network changes. Therefore, it is very important to propose a new solution to optimize the routing process of the SDN while maintaining the Quality of Service (QoS).

Machine learning attracts many attentions from academia and industry because of its outstanding performance with large-scale data processing, classification, and intelligent decision-making. Some studies use it to solve the deadlock issue in the network operation and management [1-2]. Some works use its intelligent algorithms to achieve the intelligent, customizable, and fine-grained routing management for the SDN. The authors in [3] propose some heuristic algorithms to optimize the SDN routing [3], but the algorithms' performances are not stable when the network changes. Reinforcement learning (e.g., Q-learning [4]) is also an alternative solution to improve the network performance [5]. However, the Q-learning cannot be directly used to optimize the network routing because of its huge demand of the Q table. Additionally, it only works in discrete times, which does not hold for the network with dynamic changes. Designing a routing mechanism that uses the machine learning to achieve the universal and customizable optimization in continuous times is a big challenge.

To address the above challenges, we impose the Deep Deterministic Policy Gradient (DDPG) [6] mechanism to optimize the routing in the SDN with the DDPG Routing Optimization Mechanism (DROM). DROM has four advantages: first, DROM dynamically optimizes customized parameters or strategy by intelligently adjusting the reward function. Second, DROM uses neural networks instead of Q-tables and saves the storage overhead and the time cost of table lookup caused by maintaining large-scale Q tables. Third, DROM can be widely used since it does not rely on any specific network states. Fourth, DROM can effectively achieve black-box optimization in continuous times. We evaluate the performance of DROM with experiments. The experimental results show that DROM not only has good convergence and but also achieves better performance and stability than traditional static routing algorithms.

The rest of the paper is organized as follows. Section II introduces related works on intelligent routing optimizations of the SDN. Section III describes the SDN framework with the machine learning module and the general principles of DDPG. We illustrate DROM in Section IV. The performance evaluation is presented in Section V. A discussion about our

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2877686, IEEE Access

future work is described in Section VI. We conclude the paper in Section VII.

II. RELATED WORKS

To intelligently optimize the routing of the SDN, lots of intelligent algorithms are imposed into the SDN routing process, and various techniques have been presented in prior researches. The authors in [7] propose a route predesign scheme based on a multi-machine learning approach. First, this scheme leverages a proper clustering algorithm, such as Gaussian mixture model or K-means clustering, to extract flow features. Then, a supervised learning mechanism, such as an extreme learning machine, is used to forecast traffic demand. Finally, an adaptive multipath routing method based on the analytic hierarchy process is proposed to deal with elephant flows according to the weights of different constraint factors. Existing works in [8-9] use heuristic algorithms, such as ant colony algorithm and genetic algorithm, to optimize the routing selection for flows. However, due to the limitations of heuristic algorithms, the algorithms are only work for specific problems. When the network state changes, the parameters of heuristic algorithms may be adjusted, leading to the potential scalability issues. The works in [5] and [10] both deploy reinforcement learning modules in the SDN. Specifically, the work in [5] uses QoS-aware reward functions to implement QoS adaptive routing in the SDN with Q-learning. The work in [10] proposes an end-to-end adaptive HTTP streaming media intelligent transmission architecture. In [10], the system modeling is based on a partially observable Markov decision process, and the Q-learning method is used as a cluster decision algorithm to maximize quality of experience.

The reinforcement learning is an alternative solution for optimizing the routing. It can achieve low latency, high throughput, and adaptive routing. However, the SDN requires fine-grained control on data flows, and traditional Q-learning algorithms could require huge storage space to maintain Q tables, each of which contains states, actions, reward information. In addition, as the Q-table scale extends, the look-up time of the Q tables could also increase significantly. These limitations greatly restrict the application of the reinforcement learning in the SDN routing. To handle the problems, the authors in [11] use a neural network to replace the Q-table in the traditional Q-learning method. DQN [12] is used to optimize the routing process but it only works for discrete-time control since it cannot converge in real time.

III. FRAMEWORK AND DDPG GENERAL PRINCIPLES

How to combine the SDN with a machine learning algorithm to improve the network performance is an interesting and valuable question [13]. This section proposes a feasible framework for incorporating a machine learning module to the SDN to realize intelligent operation and maintenance.

A. A machine learning-based SDN framework

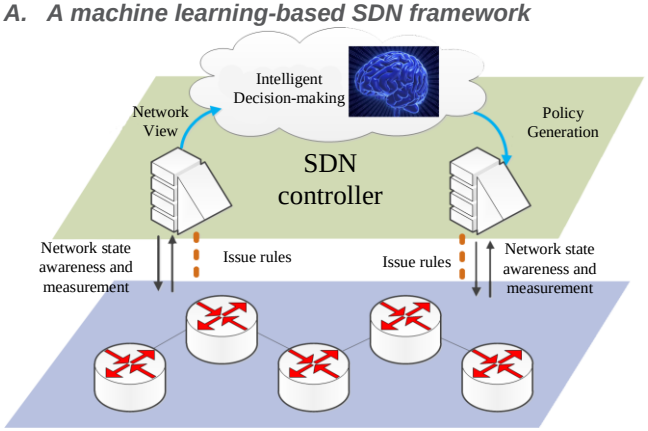

FIGURE 1. Structure of a machine learning-based SDN framework

Fig. 1 shows the structure of a machine learning-based SDN framework, which has an intelligent decision-making module of the machine learning in the SDN control plane. The intelligent decision-making module efficiently generates network policies to realize the global, real-time and customized network control and management. Specifically, by obtaining the global network status from the SDN, the intelligent decision-making module generates a network strategy, and the corresponding rules are generated by the control plane based on the strategy. From this architecture, we can intelligently optimize a series of maintenance and management operations, such as routing, resource adaptation, to improve the network performance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2877686, IEEE Access

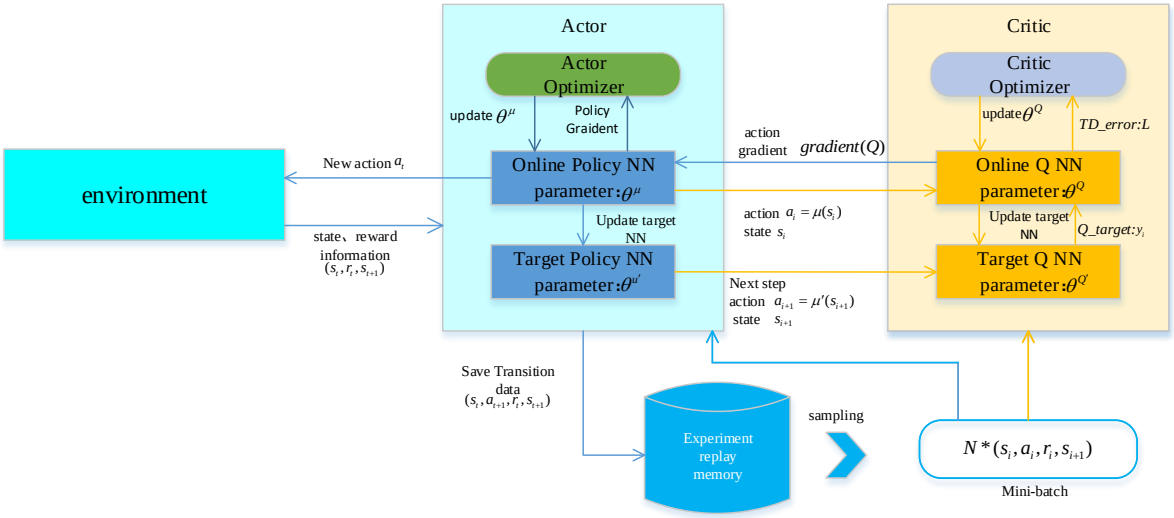

FIGURE 2. **DDPG Framework.**

B. The general principle of DDPG

The reinforcement learning enables agents to learn actions in **the environment** to maximize its reward value. Deep reinforcement learning combines the perception ability of deep learning with the decision-making ability of enhanced learning. However, the general value-based deep reinforcement learning mechanism, such as DQN[], cannot solve the modeling and control of continuous actions, which is not suitable for the dynamic and real-time network system. The policy-based reinforcement learning methods, such as Deterministic Policy Gradient (DPG) [14], can realize the continuous-time control and optimization, but they only generate policy functions for linear functions and have the overfitting issue caused by the correlation of the training data. To solve these problems, the DeepMind propose a new deep reinforcement learning approach named DDPG [6], which combines the DQN method with the DPG method in an actor-critic framework. DDPG uses neural networks to generate the strategy function and Q function and forms an efficient and stable discrete action control model.

Fig. 2 shows the DDPG framework. Notably, μ and Q represent the deterministic strategy function and Q function generated by the neural network, respectively. In the actor-critic architecture of DDPG, the actor module adopts the DPG method while the critic module adopts the DQN method. Each actor module consists of two neural networks, one is an online network for training and learning, and the other one is a target network for blocking the correlation of the training data. The target network has the same structure of the online network but uses the previous parameters of the online network. The online network regularly passes its parameters to the target network, and the target network use the parameters to update its own parameters. In a training period, the transition information of each interaction with the environment is stored in an experience pool, and the learning batch of the neural network is composed of transition process information sampled from the experience pool.

The parameter updating process of the DDPG consists of updating the actor module of the neural network and updating of the critic module of the DQN neural network. The two updates are related to each other. For the neural network of the actor module, the policy gradient should be backpropagated. The policy gradient can be calculated by Eq.(1).

$$\nabla_{\theta^{\mu}} J = grad[Q] * grad[\mu] \approx \frac{1}{N}\sum_i \nabla_a Q(s, a \mid \theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu})|_{s_i} \quad (1)$$

In the equation, the first half part grad[Q] is the action gradient from the critic network and used to characterize the movement direction of the actor to obtain a higher reward. The second half part grad[μ] is the parameter gradient from the actor network and used to characterize how the neural network of the actor should adjust its parameters to choose the high reward actions. The equation shows that the neural network of the actor module modifies its parameters to obtain high rewards.

To update of the critic module in the DQN network, we calculate the TD error of the critic module as the mean square error of the online network's Q value and the target network's Q value. It is shown in Eq.(2).

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i \mid \theta^Q))^2 \quad (2)$$

In the above equation, y_i is the Q value of the target network based on the next state s_{i+1} and next action a_{i+1}. The target network selects the next action a_{i+1} and the next state s_{i+1} with the parameter μ' from the previous online network. Eq.(3) is the equation for calculating y_i.

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'}) \quad (3)$$

In the above equation, the Q value of the online network evaluates the actor network with the current state to select the current action. The online network of the actor module passes its selected current action to the online network of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2877686, IEEE Access

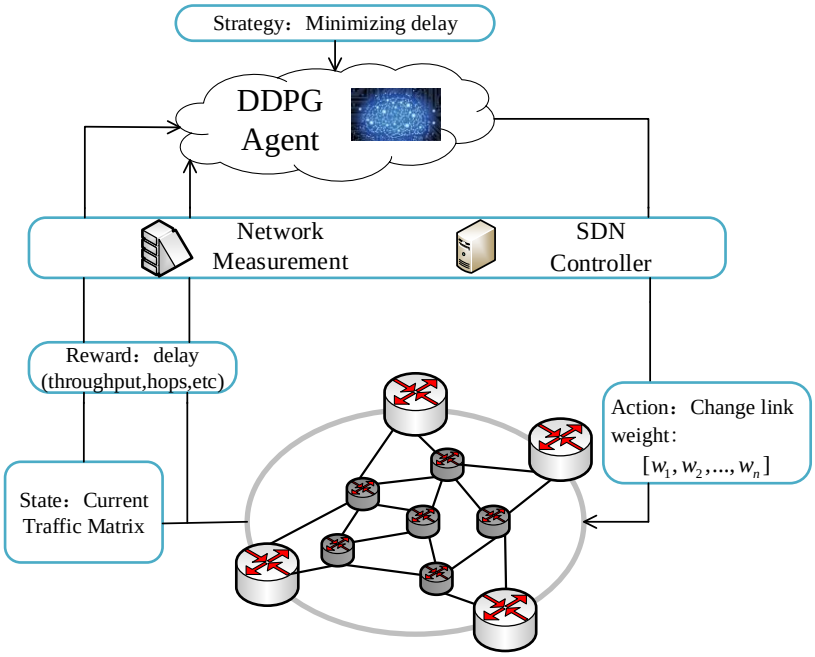

FIGURE 3. DROM Framework.

critic module, and the online network of the critic module evaluates the action to generate the Q value.

IV. MECHANISM OF DROM

The basic principles of DDPG and the framework of SDN deploying machine learning modules are introduced in the previous section. In this section, we introduce the DROM. By running DROM, the customized performance parameters, such as delay, forwarding path length, throughput, can be automatically optimized to realize the real-time control of the network in continuous time, thus effectively alleviating the pressure of operation and maintenance.

Fig. 3 shows DROM framework. DROM agents interact with the **environment** through three signals: state, action, and reward. Among them, state *s* is the Traffic Matrix (TM) of the current network load, and the action *a* taken by the agent to the environment is to change the weights of links in the network. By changing the weights of the links, the agent can change the paths of the data flows. The reward *r* of the agent is related to the network operation and maintenance strategy. It can be a single performance parameter, such as delay, throughput, or a comprehensive strategy to balance multiple parameters. For instance, Eq. (4) takes multiple parameters into consideration.

$$R_{i->j} = R(i->j \mid s_t, a_t) = -h(a_t) + \alpha delay_{ij} + \beta BW_{ij} + \gamma loss_{ij} + \theta TP_{ij} \qquad (4)$$

This equation illustrates that the network is at state s_t with the received action *r*. Suppose the path calculated after the link weight adjustment is $p_{i,j}$. In the equation, function *h* denotes the cost to implement action a_t that represents the action influence to switch operations. α, β, γ, θ∈ [0, 1) are the tunable weights and determined by the operation and maintenance strategy. In addition, cost *h* can be set to a constant value over actions typically. The adjustment of these control strategies can be achieved by changing the reward settings. In the later section, we use the minimizing delay as an example to explain how DROM works.

The training purpose of the DDPG agent is to find the optimal action *a* according to an input state *s* to maximize a reward *r*. The general process of DROM can be summarized as follows: with network analysis and measurement from the SDN controller, the DROM agent can obtain the accurate network state *s* and determine an optimal action: a set of link weights [w_1, w_2, ... w_n]. The new paths of flows are recalculated based on the set of updated weights, and the SDN controller generates new rules to establish the new paths. After the path update, the reward *r* and the new network state are obtained through the next network analysis and measurement. The performance of the required network is iteratively optimized.

V. EXPERIMENTS AND EVALUATION

In this section, we evaluate the performance of DROM, verify the effectiveness and convergence of DROM agent and present the performance advantages of DROM agent compared with traditional routing protocols. Section V.A introduces the experimental methods in detail, and Section V.B presents experimental results.

A. Experimental setup

The hardware environment of the experiment is NVIDIA Tesla P100 GPU and 24 GB DDR4 memory, and the operating system is Ubuntu 16.04. We use Keras and TensorFlow as the machine learning framework and built the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2877686, IEEE Access

simulation network environment with OMNeT++ [15]. Under the same topology, this paper tests the self-convergence and effectiveness of DDPG agents and compares the performance of DROM with OSPF, the current mainstream routing protocol, along with a large number of randomly generated routing configurations.

In the experiment, we use a real backbone network topology-Sprint network [16], which consists of 25 nodes and 53 links. Fig.4 shows the topology. The bandwidth of each link is the same. We set up several different Traffic Load (TL) levels to simulate the actual network scenario. Each TL level is a specific percentage of the total network capacity. We use the gravity model [17] to generate several different traffic matrices under the same TL. We generate several traffic matrices with different total traffic or distribution for training and testing.

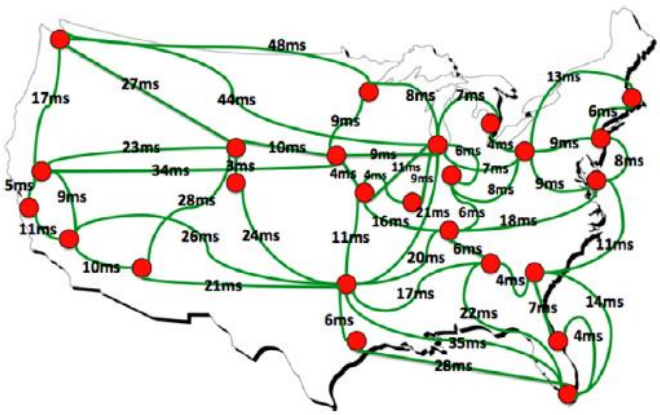

FIGURE 4. **Sprint backbone network.**

To verify the validity and convergence of DROM, we train DROM agents with different steps under different traffic intensity levels and test the performance of the trained agents to verify the effectiveness and convergence. Similarly, to verify the performance advantages of DROM, we design two experimental scenarios: (1) Under different traffic intensity, we compare the performance of DROM with 200000 randomly generated valid routing configurations. These random routing configurations ensure the representativeness and validity of the test dataset through the large quantity. (2) In the same network environment, we compare the routing performance of DROM and OSPF, which always uses the initial weight for each link.

B. Experimental results

In this paper, we use the minimizing delay as the operation and maintenance strategy and as the performance measure metric of DROM. In the experiment, we mainly focus on the convergence and effectiveness of DROM and compare the delay performance of DROM with OSPF and random routing configuration. To verify scalability, we also use the maximizing throughput as another operation and maintenance strategy and compare the performance of DROM with existing throughput solutions [18] [5]. SDN-LB [18] can achieve throughput optimization by dynamically scheduling load to avoid congestion based on the SDN. QAR [5] uses QoS-aware reward functions to implement QoS adaptive routing. In this experiment, the reward function of QAR is only related to the throughput.

1) CONVERGENCE AND EFFECTIVENESS OF DROM

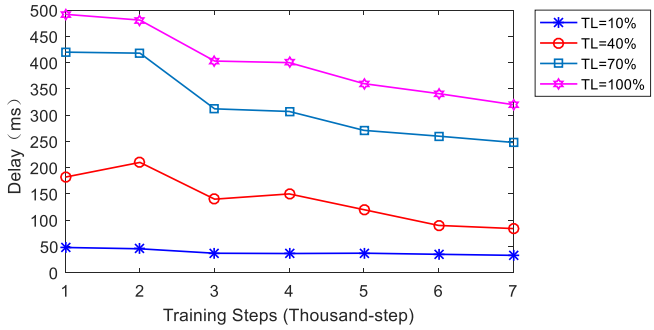

FIGURE 5. **Network delay of training steps under different traffic loads.**

In the experiment, we use four different TLs: 10%, 40%, 70% and 100% of the whole network bandwidth, respectively. For each TL, we generate 250 traffic matrices. We train DDPG agent under different TLs for 2000, 4000, 10000, 20000, 50000, 80000, and 100000 times with the same traffic matrices. Then, we test the performance of the DROM agent with 1000 testing traffic matrices as the input and obtain the routing results from the trained agent for each TL. Given the traffic matrix and routing solution, we can obtain network performance parameters, such as network delay, directly from the OMNeT++. To ensure the validity of the data, we use the average results of 1000 experiments to reflect the training results.

Fig. 5 shows the experimental results. In the figure, the DROM agent effectively reduces the network delay. As training steps increase, the network delay reduces. Under TL = 70%, the delay performance of DROM with 100000 steps training improves 40.4%, which is lower than the performance of DROM with training 2000 steps training.

2) PERFORMANCE ADVANTAGES OF DROM

To verify the advantages of DROM, we compare DROM with 200000 randomly generated routing configurations and OSPF.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2877686, IEEE Access

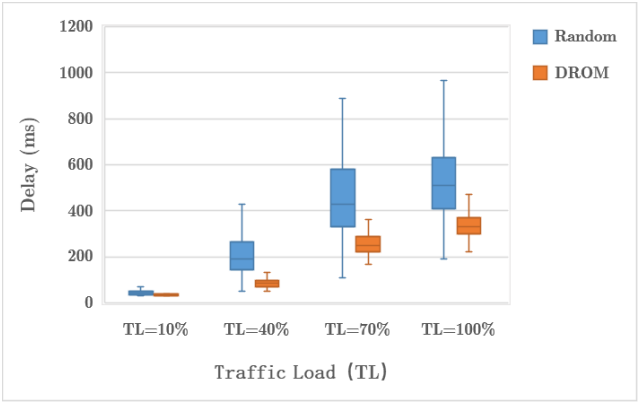

FIGURE 6. Comparison of DROM and random routing configuration under different traffic loads.

We use DROM with 100000 steps training as the comparison baseline. Fig. 6 shows the comparison results in the form of box plots.

In the figure, the upper and bottom parts of the rectangular respectively represent the upper quartile and the lower quartile values of the delays, and the line in the middle of the rectangular represents the median of the delays. The upper and lower ends of the straight line extending from the rectangle represent the maximum and minimum values of delays, respectively. For simplicity, the outliers are not shown in the figure. Experimental results show that the delays of DROM are less than the lower quartile delay of the randomly generated routing configurations, and the minimum delay of DROM is very close to the best result of the randomly generated routing configurations. The above results fully verify the superiority and effectiveness of the DROM.

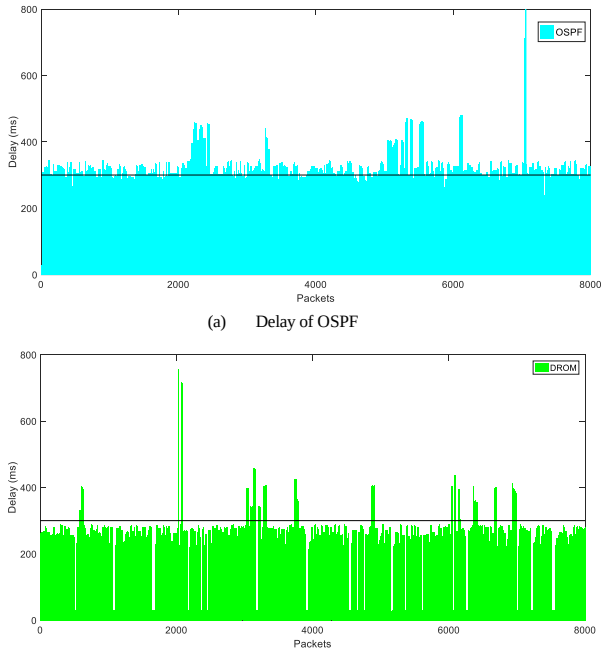

FIGURE 7. Comparison of DROM and OSPF under different traffic loads.

Subsequently, we run DDPG agents and OSPF as routing mechanisms under the same TL and the same link weights of the experimental topology. DDPG agents undergone 100000 steps training. Fig.7 shows the transmission delay of 8000 packets of DROM and OSPF. In Fig.7 (a), the peak delays of OSPF are larger than 300 ms, while in Fig.7 (b), DROM only has a few of the peak delays over 300 ms, The result indicates that DROM outperforms OSPF and verifies the effectiveness and performance advantages of DROM.

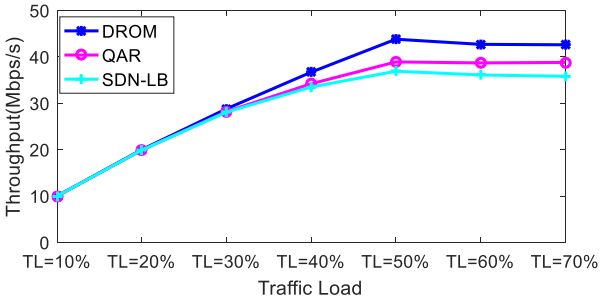

FIGURE 8. Comparison of DROM and two existing solutions under different traffic loads.

We also use the throughput maximization as the operation and maintenance strategy. The DROM agents are trained at TL levels with the same steps (i.e., 100000 steps), and 1000 testing traffic matrices are used as the input to obtain the throughput of each TL. We compare DROM with two existing throughput optimal solutions: SDN-LB [18] and QAR [5]. Similarly, we use the average results of the 1000 test experiments as the performance metric. Fig.8 shows the results of DROM, QAR and SDN-LB. In the figure, when the TL exceeds 30%, DROM's throughput is larger than that of the two comparison schemes. This experiment result further illustrates that DROM outperforms the existing solutions.

VI. Future Work

From the above results, we can see the DROM can achieve the customizable routing optimizations with the different operation and maintenance strategies. How to generate a strategy is still an open question. If we can combine the QoS-aware traffic classification and the network measurement with the DROM， we can generate a strategy that is adaptively generated to realize the QoS-aware, reliable and effective end-to-end transport. We leave this interesting problem as our future work.

VII. CONCLUSION

In this paper, we propose a machine learning-based SDN framework, which uses a novel deep reinforcement learning mechanism called DDPG to optimize the routing process of the SDN. Based on the framework, we propose a routing optimization mechanism called DROM to realize the global, real-time and customized network intelligence control and management in continuous times. The experiment results show that DROM has a good convergence and effectiveness, and compared with existing routing solutions, DROM can

improve the network performance with stable and superior routing services.

REFERENCES

[1] R. Boutaba, M. A. Salahuddin, N. Limam, *et al*. "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services & Applications*, vol.9, no.1, pp.16, May. 2018.

[2] Z. M. Fadlullah, F Tang, and B Mao, *et al*. "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, May. 2017.

[3] F. Wang, B. Liu, and Q. Zhang. "Dynamic routing and spectrum assignment based on multilayer virtual topology and ant colony optimization in elastic software-defined optical networks," *Optical Engineering*, vol.56, no.7 pp.076111, July. 2017.

[4] R. SUTTON and A. BARTO, Reinforcement Learning: An Introduction. Cambridge, MA: The MIT Press, 1988.

[5] S. C. Lin, I. F. Akyildiz, P. Wang, *et al*. "QoS-aware adaptive routing in multi-layer hierarchical software defined networks: a reinforcement learning approach," in *Proc*. IEEE International Conference on Services Computing, 2016, pp.25-33.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al*. "Continuous control with deep reinforcement learning," U.S. Patent, WO/2017/019555, Feb. 2, 2017.

[7] W. Li, G. Li, and X. Yu. "A fast traffic classification method based on SDN network," in *Proc*. the 4th International Conference on Electronics, Communications and Networks, 2015, pp. 223-229.

[8] M. R. Parsaei, R. Mohammadi, R. Javidan. "A new adaptive traffic engineering method for telesurgery using ACO algorithm over Software Defined Networks," *European Research in Telemedicine / La Recherche Européenne en Télémédecine*, vol. 6, no.3-4, pp.173-180, Nov. 2017.

[9] J. Wang, C. D. Laat, and Z Zhao. "QoS-aware virtual SDN network planning," in *Proc*. IFIP/IEEE Symposium on Integrated Network and Service Management, 2017, pp. 644-647.

[10] J. Jiang, L. Hu, P. Hao, *et al*. "Q-FDBA: improving QoE fairness for video streaming," *Multimedia Tools & Applications*, no.2, pp.1-20, July.2017.

[11] S. Sendra, A. Rego, J. Lloret, *et al.* "Including artificial intelligence in a routing protocol using Software Defined Networks," in *Proc*. the IEEE International Conference on Communications Workshops, 2017, pp. 670-674.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.* "Human-level control through deep reinforcement learning," *Nature*, vol.518, no.7540, pp. 529-533, Feb. 2015.

[13] A. Mestres, A. Rodriguez-natal, J. Carner, *et al.* "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no.3, pp.2–10, June. 2017.

[14] D. Silver, G. Lever, N. Heess, *et al*. "Deterministic policy gradient algorithms," in *Proc*. International Conference on Machine Learning, 2014, pp.387-395.

[15] V. András and R. Hornig. "An overview of the omnet++ simulation environment," in *Proc*. the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, 2008, pp. 60.

[16] Sprint, Overland Park, KS. "Sprint IP network performance," 2011, [Online]. Available: http://www.sprint.net/performance.

[17] R. Matthew. "Simplifying the synthesis of internet traffic matrices," *ACM SIGCOMM Computer Communication Review*, vol.35, no.5, pp.93–96, Oct. 2005.

[18] Y. Wang, X. L. Tao, Q. He, *et al.* "A Dynamic Load Balancing Method of Cloud-Center Based on SDN," *China Communication*, vol.13, no.2, pp.130-137, Feb. 2016.