Topology design for data center networks using deep reinforcement learning

Topology Design for Data Center Networks Using Deep Reinforcement Learning

Haoran Qi
Department of ECE
University of Alberta
Edmonton, Canada
hq1@ualberta.ca

Zhan Shu
Department of ECE
University of Alberta
Edmonton, Canada
zshu1@ualberta.ca

Xiaomin Chen
Department of CIS
Northumbria University
Newcastle, UK
xiaomin.chen@Northumbria.ac.uk

***Abstract*—This paper is concerned with the topology design of data center networks (DCNs) for low latency and fewer links using deep reinforcement learning (DRL). Starting from a K-vertex-connected graph, we propose an interactive framework with single-objective and multi-objective DRL agents to learn DCN topologies for given node traffic matrices by choosing link matrices to represent the states and actions as well as using the average shortest path length together with action penalty terms as reward feedback. Comparisons with commonly used DCN topologies are given to show the effectiveness and merits of our method. The results reveal that our learned topologies could achieve lower delay compared with common DCN topologies. Moreover, we believe that the method can be extended to other topology metrics, e.g., throughput, by simply modifying the reward functions.**

***Index Terms*—Low-latency Data Center Network Topology, Deep Reinforcement Learning, Multi-objective Learning**

I. Introduction

As many enterprises and corporations are gradually shifting their services and businesses into cloud environments, data center networks (DCNs) are being increasingly deployed to connect a massive number of servers and achieve large-scale data computing and storage. This trend has posed a significant challenge to network topology design. Generally, the conventional network topologies in data centers (DCs) are created by either 1) non-heuristic and mathematical methodology, which have considerable scalability with limited adaptability [1]–[3] , or 2) heuristic algorithms subject to some special scenarios [4], [5]. Apart from that, most network topologies seem to lack effective data interaction with real applications and may not adapt to the rapid-changing network traffic [6], especially when scalability is required. Therefore, developing network topologies that can be adaptive to traffic patterns is called for.

The primary objective of this paper is to develop a novel approach to topology design of DCNs utilizing Deep Reinforcement Learning (DRL). Unlike the topology design approaches mentioned above, DRL enables active searching from complicated network environment with greater network node numbers. Specifically, DRL agents can learn the improved solutions for reaching lower latency and fewer link numbers topologies based on specific traffic matrix inputs, as will be shown later.

The main contributions of the paper are summarized below:

- Development of an interactive environment called TopoWorld for DRL agents to explore adaptive topologies with high-dimensional nodes for low delay and link number.
- Design of a novel multi-objective learning algorithm for better generalization of node dynamics for topologies.
- Design of experimental benchmark to demonstrate the performance improvement of the learned topologies compared to conventional DCN ones.

II. Background

A. Data Center Network Topologies

A series of related studies have been conducted to cope with the problem of network topology design for DCs. M. Al-Fares et al. proposed a scalable Fat-Tree topology for data center [1]; Meanwhile, J. Kim et al. initialized the research on a network topology called Dragonfly by aggregating a collection of high-radix routers [2]; Further, C. Guo et al. introduced a server-centric network architecture named Bcube for DCs [3]; A. Shpiner et al. proposed Dragonfly+ topology by modifying the group routers of dragonfly [2] to achieve higher performance [7]; Apart from that, Y. Deng et al. developed an optimal low-latency and low-diameter topology in [4] by applying the Simulated Annealing (SA) algorithm [8].

However, most of the topology designs seems to lack interaction and fit with the actual network traffic, leading to some discrepancy between theoretical predictions and practical results. In addition, most topologies cannot be applied to an arbitrary number of nodes, meaning that developer must deploy servers strictly to a fixed number of nodes, which may increase the potential overhead in the data center space. In this paper, we will show an interactive and nodes-unlimited design of DCN topology using DRL.

B. Deep Reinforcement Learning

Reinforcement learning (RL), as one of the most popular optimization algorithms, is able to learn a series of actions that aims at maximizing the accumulated rewards by interacting actively with an environment in the absence of prior knowledge. However, most traditional reinforcement learning (RL) algorithms, such as Q-learning, are highly dependent on tabular and handcraft features [9], which are difficult to cope

with high dimensional state-space environments. To handle this problem, much research exploiting the capacities of deep learning (DL) into RL, aiming to train the neural networks from large and raw input data, proposes deep reinforcement learning (DRL) that can successfully deal with large volumes of discrete and continous space problems. Typically, the DRL algorithms can be roughly splited into two different classes, value-based algorithms such as DQN [9], and policy-based algorithms such as DDPG [10], PPO [11], etc. Nowadays these methods are widely used in various research tasks.

In our research, the number of nodes is a critical factor for network topology design since thousands of nodes need to be placed in modern data centers. We find that the conventional RL algorithms might show some weakness in topology exploration as the node numbers continuously increase due to the remarkable rise of the state-space dimension. To handle this issue, we uses a single-objective DQN instead of traditional RL algorithms to explore the topologies and uses a variant of the Multi-objective PPO algorithm to achieve bi-objective search, which is able to find a better topology even if the number of nodes continues to increase or decrease.

III. Topology Design by Single-Objective DQN

A. Initial guess K-connected Graph

An initial guess of the topology plays a vital role in topology learning since it may determine the starting level of the topology and thus has a great chance to enhance the efficiency of learning processes. These first topology generations would automatically be treated as the first observation at each episode. For the sake of reliability, we select K-vertex-connected topology as the initial guess graph [12].

B. State Design

The state can be loosely described as the observation of the DRL agent from monitoring the results caused by the previous action. In our work, the state would be a sparse matrix with the size of maximum link numbers, indicating the connectivity of each candidate link, with 1 representing the existence of a link while 0 illustrating nonexistence. One of the example is shown on Fig. 1. This way, models would be more efficient in applying floating-point operations (FLOPs) and obtaining acceptable accuracy parameters with sparse matrices. [13]

Node#	0	1	2	3	4
0	NaN	1	0	0	1
1	1	NaN	1	0	0
2	0	1	NaN	1	0
3	0	0	1	NaN	1
4	1	0	0	1	NaN

Fig. 1: A state example. The length of the state equals to maximum links number(solid line boxes), and thus there exists a link between nodes number (0,1), (0,4), (1,2), (2,3) and (3,4).

C. Action Design

In this paper, the action is defined as adding or removing a link from the current topology. Specifically, our action vector would consist of two units: 1) an index number referring to a specific link, e.g. the link $(0, 1)$ is marked as the first link and link (N-2, N-1) as the last link of the state; 2) an actual action taken, where 0 represents to add the link while 1 represents to disconnect the link selected. For example, an action $[0, 0]$ means adding the first link of to the current topology.

D. Reward Function Design

The goals of our single-objective DQN agent is minimizing the average delay of the networks from the source node to the destination node. Exploiting the Dijkstra shortest paths algorithm, the possible average latency could be determined from each link's previously assigned delay value. For the proposed TopoWorld, the reward is assigned for several sections:

a) Delay Reward R_t^d: the delay reward is designed as the gap between a threshold ρ and the current feedback of an average delay $\overline{\theta}_t$, assuming that the delays here are the total delays of four different delay types. And if a source node (default to the first node) can send packets to every other node in the topology without packet loss, the average delay $\overline{\theta}_t$ could be described as:

$$\overline{\theta}_t = \frac{\sum_{n=1}^{N} \sum_{i \in \delta_n} \theta_i^t}{N-1} \tag{1}$$

where N is the total node number. Note the source node(i.e. node zero) would not be taken into account when calculating average delay. The variable δ_n represents the set of passing nodes in the shortest path from the source node to node n based on Dijkstra's algorithm. In summary, the delay reward could be denoted as :

$$R_t^d = \eta(\rho - \overline{\theta}_t) \tag{2}$$

where the η is a discount number, and the ρ is a threshold that ensures the rewards are positive.

b) Redundancy Penalty R_t^l: Too many links are obviously undesirable and contrary to the learning objectives since this may significantly increase the cost of the network. To alleviate this problem as much as possible, we take a reward penalty R_t^r to reduce the number of useless links in the learned topologies.

$$R_t^l = -L \tag{3}$$

where L is the number of current links.

c) Repetition Penalty R_t^r: Applying repeated and non-useful works such as adding an existent edge or deleting a non-existent link would lead to a negative reward for the selection, which is exactly the opposite of the delay reward.

$$R_t^r = -R_t^d \tag{4}$$

d) K-connected Degree Penalty R_t^{nK}: If any node determined in the first place is no longer higher or equal to the degree specified in the initial guess graphs, the agent would be given a relatively huge penalty for that intolerable behavior.

e) Non-connected Penalty R_t^{nc}: If any nodes included in the graph are no longer being reached in the topology due to the last action, the agent would be given a huge penalty for that intolerable behavior.

The finally aggregated reward function of single-objective DQN is thus synthesized as:

$$R_t = \begin{cases} R_t^{nc} & \text{if any node is disconnected} \\ n \times R_t^{nK} & \text{if } n \text{ nodes has lower degree than } K \\ R_t^r + R_t^l & \text{if action is repeated} \\ R_t^d + R_t^l & \text{otherwise} \end{cases}$$

E. Interaction Environment for Agents

RL agents are required to construct an interactive environment for decision-making. OpenAI Gym [14] is one of the most famous RL toolkits to test and compare different algorithms on various accessible episodic benchmarks. In our work, an interactive gym-based environment called ***TopoWorld*** is contributed to focus on developing DC topologies with deterministic node dynamics, e.g., link latency. The general layout of TopoWorld is shown in Fig. 2.

The architecture of the framework can be briefly depicted as follows: 1) The DRL would first send the action selected to the ***TopoWorld***. 2) The step function would check and execute the action to generate a topological structure as the next observation of each iteration. 3) The ***TopoWorld*** would evaluate the topology produced based on the reward function. 4) The newly obtained information would be passed to the DRL model to update parameters.

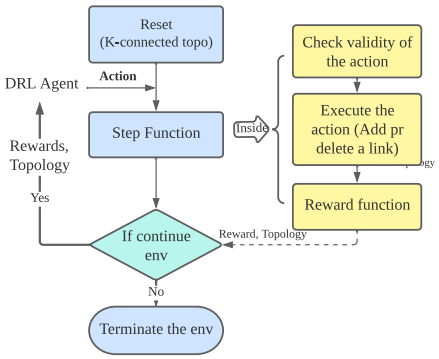

Fig. 2: Layout of TopoWorld

IV. Topologies Design by Multi-Objective DRL

A. Multi-objective Deep Reinforcement learning

Notably, the single-objective DQN learning agent is forced to explore the graphs under multiple reward constraints. As the number of nodes increases, the single-objective agent may not learn efficiently, i.e., achieving a higher accumulated reward. For example, it is kind of distinct that the learning process tends to add incredible link numbers as the nodes grow, as shown in Table III, which may not be desirable based on cost considerations. To break this bottleneck, many recent studies have focused on exploring the potential of deep reinforcement learning in multi-objective conditions (MO-DRL). [15], [16] have proposed their multi-objective algorithms and many research works like [17], [18] use MO-DRL algorithms to tackle some practical problems.In this paper, We would like to use this approach to further reduce the number of links while trying to maintain the same network performance.

B. Multi-objective Proximal Policy Optimization

As one of the most effective baselines of policy-based DRL algorithms, the Proximal Policy Optimization (PPO) Algorithm [11] creatively combines importance sampling and the Trust Region method firstly introduced in TRPO algorithm [19], thus making a good trade-off between sample complexity and algorithm efficiency. To take advantage of this, we introduce multiple value networks and previously defined weights on top of PPO to allow the algorithm to better extract characteristic attributes of multi objects in the following. We call this algorithm a multi-objective and -values PPO algorithm (MOV-PPO), which is given in Algorithm 1.

Consider a set of pre-defined weights $\boldsymbol{\omega_O}$ that determine the weighting preferences for each learning objective o. Note that this weight set should take into account the range of values of the qualified rewards as well as the priority of the objectives.The original PPO uses advantage as a relative assessment of the selected action. By utilizing generalized advantage estimation [20], the estimator of the advantage can be shown as:

$$\hat{A}_t^o = \delta_t^o + (\gamma\lambda)\delta_{t+1}^o + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1}^o \tag{5}$$

where δ_t^o is the temporal difference (TD) error that equals

$$\delta_t^o = R_t + \gamma V_o(S_{t+1}) - V_o(S_t) \tag{6}$$

$V_o(S_t)$ denotes the estimated value of the objective o by the value function based on current state S_t. In a multi-objective scenario, we have an advantage estimator $\hat{A}_t^o$ per objective o. For each advantage estimator, there would be a particular value approximator, e.g., value neural network, to evaluate action distribution and update corresponding values. In this way, we have a vector of estimated advantages $\boldsymbol{\hat{A}_t^O} = \{\hat{A}_t^1, \ldots, \hat{A}_t^o\}$ and values $\boldsymbol{Q_t^O} = \{V_1(S_t), \ldots, V_o(S_t)\}$ associated with O objectives. In this case, the weighted advantages estimator vector can be computed as:

$$\boldsymbol{\hat{A}_t^\omega} = \boldsymbol{\hat{A}_t^O} \times \boldsymbol{\omega_O} \tag{7}$$

and it would be used to optimize the main clipped Surrogate Objective:

$$L^{CLIP}(\theta) = E[\min(p_t(\theta)\boldsymbol{\hat{A}_t^\omega}, clip(p_t(\theta), 1-\epsilon, 1+\epsilon)\boldsymbol{\hat{A}_t^\omega}) \tag{8}$$

where θ is the policy parameter, $p_t(\theta)$ is the ratio of the probability under the new and old policies, and ϵ is a hyperparameter, which is set as ϵ = 0.2 [11] in this work.

The reward function of the MOV-PPO algorithm still follows the same goal of the single-objective learning algorithm, i.e., minimizing the average latency and the number of links. However, instead of a scalar number, the reward feedback

$\boldsymbol{R_t}$ in this algorithm is an independent vector recording the specific reward for each objective:

$$\boldsymbol{R_t} = \{R_t^1, \ldots, R_t^o\} \tag{9}$$

In the case of our topology development, the rewards vector(or objectives) of the MOVPPO algorithm includes:

- R_t^d: Minimize the average delay of the networks.
- R_t^l: Add as few links as possible to the topology to suppress the cost of the network.

Apart from this, we still apply the exact definition of action, state and same other reward penalties for both reward objectives of the MOVPPO algorithm as in the single-objective approach.

Algorithm 1 MOV-PPO for Topology Design

Input: Given O preferred weights $\boldsymbol{\omega_o}$, O value nets $V_1, ..., V_o$ for each objective, policy net $\pi_{\theta_{old}}$, replay buffer M, state S_t, action A_t, and the transition probability P_t

1: **for** $iteration = 1, 2, \ldots$ **do**
2: Initialize the S_1 with K-connected graph.
3: **for** $actor = 1, 2, \ldots$ **do**
4: Run policy $\pi_{\theta_{old}}$ in TopoWorld environment for T time steps, and push dataset $\{S_t, A_t, \boldsymbol{R_t}, S_{t+1}, P_t\}$ into M
5: Compute advantage estimates $\hat{\boldsymbol{A}}_{\boldsymbol{t}}^{\boldsymbol{o}} = \{\hat{A}_t^1, \ldots, \hat{A}_t^o\}$ based on value vector $\boldsymbol{Q_t^O} = \{V_1(S_t), \ldots, V_o(S_t)\}$, then compute weighted advantages as $\hat{\boldsymbol{A}}_{\boldsymbol{t}}^{\boldsymbol{\omega}} = \hat{\boldsymbol{A}}_{\boldsymbol{t}}^{\boldsymbol{O}} \times \boldsymbol{\omega_O}$
6: **end for**
7: Update value net by applying label $(\hat{\boldsymbol{A}}_{\boldsymbol{t}}^{\boldsymbol{\omega}} + \boldsymbol{Q_t^O})$ and optimize surrogate L w.r.t θ,
8: $\theta_{old} \leftarrow \theta$
9: **end for**

V. Experiments and Results

A. Testbed and Training Settings

The testbed in our work draws on several popular software-defined network tools: The $Mininet$ [21], which is a widely used real-time network emulator, will create a customized topology, thus providing the infrastructure for performance tests like the $Ping$ (Packet InterNet Groper) and $D-ITG$ (Distributed Internet Traffic Generator) [22] commands, which is a network workload generator as well as a standard performance metric measurement tool. Note that all the tests have considered the first node as the source node, i.e., traffic input or output entry; And all topologies used for effect comparison were tested under the same network traffic matrix, i.e., same pre-settings of delay values for the links in the $Mininet$.

TABLE I: Experimental Parameters

Command	Experimental parameters			
Ping	***Protocol***		***Compartment***	
	Internet Control Message Protocol		30 ms	
D-ITG	***Protocol***	***Rate***	***Packet Size***	***Duration***
	UDP	10 pkt/s	512 bytes	60 s

For neural networks(NNs) design, we use multiple linear, fully connected layers to form our NNs for training purposes, which can be roughly divided into three parts: 1) Input layer: The input layer includes an equal number of neurons with a maximum number of links. 2) Hidden layers: Usually, we have two hidden layers in our networks. Each of them has exactly the same neurons, as depicted in Table III. 3) Output layers: This layer composition would depend on the selected algorithm. For DQN, it only includes one neuron, while for MOV-PPO, it typically contains a softmax layer as well as a linear layer with the neurons equaling actions in the action space. After training, this superimposed linear network allows agents to learn optimized results under different node dimensions and network traffic. The specific experimental parameters and DRL training settings are listed in Table I and Table II.

TABLE II: Training Settings

Algorithms selected	**Parameters**			
	Learning rate	***Discount***	***Epsilon***	***GAE factor***
DQN	1e-4	0.99	0.88	None
MOV-PPO	3e-4	0.99	None	0.95
Algorithms selected	**Neurons per hidden layer for different node numbers**			
	N10	***N16***	***N20***	***N32***
DQN	64	128	256	384
MOV-PPO	64	128	256	384

B. Results

The learned topologies and test results will be provided with the corresponding analysis in this section. However, we only test the learned topologies in the number of nodes N equal to 10, 16, 20, and 32 due to hardware limits. The accumulated rewards of learning processes are showed on the Fig. 3 and 4.

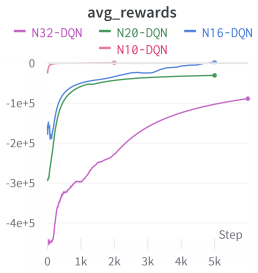

Fig. 3: DQN Rewards

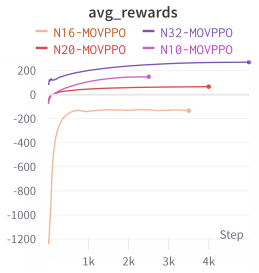

Fig. 4: MOVPPO Rewards

1) Single-Objective Learned Topology: To compare the actual performance of topologies learned by single-objective DQN agent, we also test several existing topologies under the same traffic environment.

a) Multi-Ping Test: In this sub-section, the source node will be set as a server to receive packets from other nodes synchronously. The results in Fig. 5 show the average value obtained after 5 $Ping$ tests. The graph reveals that the topologies learned by DQN dominate the lowest delay among all the multi-ping test results for all node dimensions, except

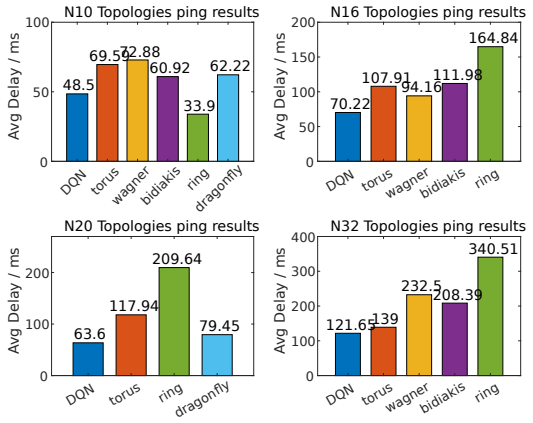

Fig. 5: Ping Test for Single-Objective Learned Topologies

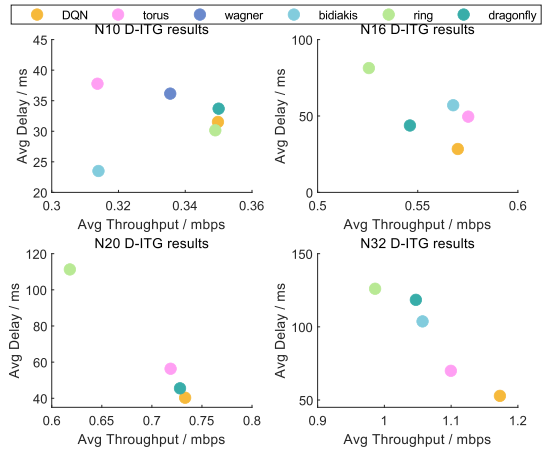

Fig. 6: D-ITG Test for Single-Objective Learned Topologies

for the ten-nodes scenario where the Bidiakis [23] topology unexpectedly takes the lead.

b) D-ITG Test: In this part, the source node is instead treated as a client and thus receives packets from other nodes synchronously. Note that we choose a smaller flow stream at each sending node to avoid network congestion.

The test results in Fig. 6 show that, although the solution brought by our DQN agent does not carry the same equivalent benefit in the 10-node case, the learned topologies still win in most of the latency tests. Likewise, we can also observe that in most test scenarios, single-objective learned topologies come out on top in terms of throughput measurements.

2) Multi-objective Learned Topology: In this section, we test our topologies explored by a multi-objective (MO) DRL algorithm dictated in section IV. To visualize the performance of this topology more precisely, We use both the existing research results obtained by the Simulated Annealing (SA) search [4] and the consequences of single-objective (SO) learning as baselines. All topologies generated by three algorithms are tested under the same networking environments and traffic metrics. In addition, Table IV shows the difference in link number between SA [4], SO, and MO topologies.

What can be seen from Fig. 7 is that above all the $multi - ping$ tests for both kinds of topologies, the MO topologies succeed in reaching less latency in all of the node numbers. This trend seems to become more apparent as the number of nodes increases. Apart from that, although the MO topologies have more links in the $N16$ test than the SA and SO topologies, we can still see a significant reduction in latency compared to the latter two in Fig. 8 and Table III; on the other hand, in $N20$ and $N32$ tests, The MO topologies achieve the double advantage of including fewer links and lower latency compared to SA and SO topologies. It is also noticed that $N32$ MO topology outperforms the SA and SO baselines in all three objects (delay, bandwidth, and links number).

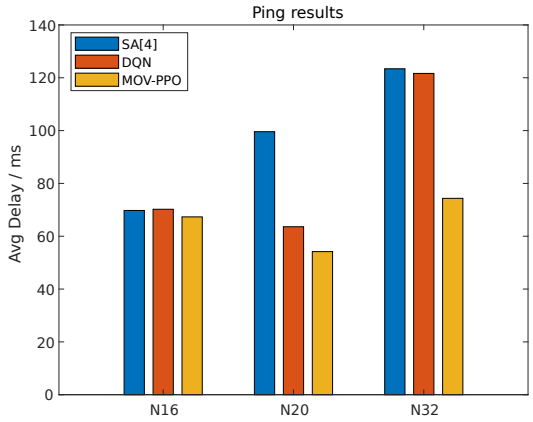

Fig. 7: Ping Test for Multi-Objective Learned Topologies

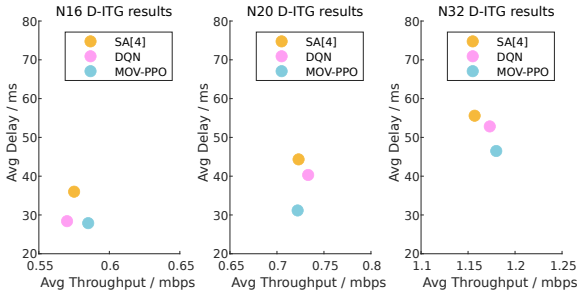

Fig. 8: D-ITG Test for Multi-Objective Learned Topologies

TABLE III: Comparison of links number

The number of nodes	Topology links		
	***SA* [4]**	***SO***	***MO***
16	32	32	34
20	40	60	37
32	64	85	58

VI. Discussion

Regularity of The Learned Topologies: What is evident in the results is that all the learned topologies, unsurprisingly, are

irregular graphs. Compared to those conventional topologies based on spatial rules, the irregular graphs are more likely to offer significant reduction of average hops. This trend is expected to be more beneficial in complex networks [24]. We could add a penalty term to the reward function such that the number of links to a node is limited to a certain number for regularization purpose, but this may affect the actual performance of the topologies.

The Price of Our Design: Although DRL has made a big splash in various research and industrial fields in recent years, there are still some difficulties when solving practical problems using it. In our case, the generalization of the topologies design process is not trivial, as the algorithm always needs to relearn the optimized topology when the network scale or network nodes change. This could bring some extra time costs. However, some recent studies [25], [26] have shown that the transferable features of neural networks may alleviate this problem to some extent.

Another Possible Objective: Apart from the latency and number of links, throughput is another essential network performance indicator. Specifically, low latency and high throughput are usually preferred for DCNs. Following our workflow, there is a definite possibility to improve this indicator by building an evaluation reward function, e.g., the upper bound of topology throughput from the recent research [27]. Furthermore, if such a multi-objective learning model is used, the DRL agents would have a good chance of discovering the Pareto Frontier for all of the above optimization particles.

VII. Conclusion

In this paper, we have proposed a learning approach to search DCN topologies with lower latency and limited link numbers. A simulation environment called TopoWorld has been developed to allow DRL agents to perform learning activities, and a single-objective and proposed multi-objective DRL algorithm have been applied. Experiments with Mininet and D-ITG have been carried out to show the advantages of the learned topologies in achieving better performance compared to traditional topologies.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM 2008*, ser. SIGCOMM '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 63–74.

[2] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *2008 International Symposium on Computer Architecture*, 2008, pp. 77–88.

[3] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," in *ACM SIGCOMM 2009*, ser. SIGCOMM '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 63–74.

[4] Y. Deng, M. Guo, A. F. Ramos, X. Huang, Z. Xu, and W. Liu, "Optimal low-latency network topologies for cluster performance enhancement," *The Journal of Supercomputing*, pp. 1–27, 2020.

[5] B. Dengiz, F. Altiparmak, and A. Smith, "Local search genetic algorithm for optimal design of reliable networks," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 3, pp. 179–188, 1997.

[6] S. Salman, C. Streiffer, H. Chen, T. Benson, and A. Kadav, "Deepconf: Automating data center network topologies management with machine learning," in *Proceedings of the 2018 Workshop on Network Meets AI and ML*, ser. NetAI'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 8–14.

[7] A. Shpiner, Z. Haramaty, S. Eliad, V. Zdornov, B. Gafni, and E. Zahavi, "Dragonfly+: Low cost topology for scaling datacenters," in *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, 2017, pp. 1–8.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[12] D. B. West, *Introduction to Graph Theory*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2001.

[13] T. Gale, M. Zaharia, C. Young, and E. Elsen, "Sparse gpu kernels for deep learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '20. IEEE Press, 2020.

[14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[15] T. Tajmajer, "Modular multi-objective deep reinforcement learning with decision values," 09 2018, pp. 85–93.

[16] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] Y. Zheng, X. Xie, T. Su, L. Ma, J. Hao, Z. Meng, Y. Liu, R. Shen, Y. Chen, and C. Fan, "Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning," in *2019 IEEE/ACM ASE*, 2019, pp. 772–784.

[18] K. Ryu and W. Kim, "Multi-objective optimization of energy saving and throughput in heterogeneous networks using deep reinforcement learning," *Sensors (Basel, Switzerland)*, vol. 21, 2021.

[19] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[20] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *CoRR*, vol. abs/1506.02438, 2016.

[21] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: Association for Computing Machinery, 2010.

[22] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-itg distributed internet traffic generator," in *QEST 2004. Proceedings.*, 2004, pp. 316–317.

[23] E. W. Weisstein, "Bidiakis cube." *MathWorld–A Wolfram Web Resource.* [Online]. Available: https://mathworld.wolfram.com/BidiakisCube.html

[24] J. Murray, P. Wettin, P. Pande, and B. Shirazi, *Current Research Trends and State-of-the-Art NoC Designs*, 12 2016, pp. 11–21.

[25] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *The 32nd ICML - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 97–105.

[26] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2019.

[27] P. Namyar, S. Supittayapornpong, M. Zhang, M. Yu, and R. Govindan, "A throughput-centric view of the performance of datacenter topologies," in *2021 ACM SIGCOMM Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 349–369.