G-Routing: Graph Neural Networks-Based Flexible Online Routing

Huihong Wei, Yi Zhao, and Ke Xu

Abstract

Deep reinforcement learning (DRL) has been widely used to find optimal routing schemes to meet various demands of users. However, the optimization goal of DRL is typically static, whereas the network environment is dynamic. Changes in traffic environment or reconfiguration of network equipment often lead to periodic changes in network performance (e.g., throughput degradation and latency peaks). The traditional static target configuration cannot reflect the importance difference of different metrics in the dynamic network environment, resulting in the inflexibility of DRL-based routing algorithms.

To address the above issue, we propose G-Routing, an online routing optimization algorithm that uses graph neural networks (GNNs) and DRL. By modeling and comprehending the relationship among different features (e.g., path, flow, and link) of the network, our proposed GNN model can predict the future development of network performance metrics (i.e., latency, throughput, and loss), thereby adjusting the routing algorithm's target promptly. Then, with the DRL model we proposed, the agent can learn the optimal path to adapt to different environmental changes. We implement the G-Routing method on the control plane and perform simulation experiments using real-world network topology and traffic data. Experimental results demonstrate that when the network environment changes significantly, our proposed G-Routing converges faster, achieves lower jitter, and generates more reliable routing scheme.

Introduction

Intelligent Internet architecture [1] has become an emerging trend. In various intelligent models based on deep neural networks (DNNs), DRL has exhibited the outstanding advantages in network decision-making and system adaptation problems, for example, congestion control [2], adaptive bitrate streaming [3], and so on. Among them, intelligent routing based on DRL [4] has received great attention.

Specifically, modern applications (e.g., in-car networking, AR/VR, and metaverse) have more fine-grained requirements for low latency, slight jitter, high throughput, and low loss. Due to the significant expansion of network scale, the traditional best-effort forwarding method cannot solve the optimization problem in a large search space[1], and it cannot meet the demands of a variety of applications at the same time. DRL brings new light to the resolution of these gaps. Depending on the characterization ability of DNN and the optimization ability based on environment interaction, DRL can well solve the large-scale spatial optimization problem. Moreover, through reasonable configuration of target policy[2], DRL can also meet the fine-grained requirements of different applications. Therefore, DRL-based online intelligent routing has attracted extensive attention from both academia and industry.

Regarding DRL-based intelligent routing, the process of route generation consists of two parts: First of all, according to the network state, the agent automatically generates routing paths that meet the policy target, and loads them to the data plane for execution. Immediately afterwards, the agent adjusts the model parameters according to the reward feedback from the data plane. Since rewards are calculated according to the target policy, the upper bound on the convergence of the DRL-based routing algorithm is closely related to the target policy.

Due to the diversity of traffic demand, existing intelligent routing algorithms often focus on a multi-objective optimization problem, where different objectives may conflict with each other (e.g., low latency and high throughput). One common approach is to utilize weighted factor programming to convert it into a single-objective optimization problem, which is the policy target of DRL. The policy target calculation is typically expressed as a combination of metrics, for example, throughput, latency, and loss. These coefficient parameters of each metric can explicitly indicate the relative importance of these metrics based on application requirements.

However, determining these weighting coefficients can be challenging and can significantly impact the system's performance. Existing DRL-based intelligent routing algorithms generate fixed and unchanged weighting coefficients through manually defined methods. Although they can achieve good convergence performance under fixed requirements and network environments, these algorithms cannot adapt when the algorithm environment changes or new application requirements emerge. This leads to various issues, such as weak generalization and poor flexibility.

To address above challenges (i.e., weak generalization and poor flexibility), we propose an online routing optimization algorithm based on GNN and DRL, namely G-Routing. The core of G-Routing is a novel dynamic-target DRL framework, which automatically adjusts the weighting

[1] The optimization problem in a large search space refers to how to effectively select the optimal routing solution from a large number of available routing options in a network with a large topology.

[2] The target policy refers to the method of configuring the reward function. Different methods of configuring the reward function can guide DRL-based algorithms to continuously optimize toward different targets.

Digital Object Identifier: 10.1109/MNET.012.2300052

The authors are with Tsinghua University, China. Yi Zhao and Ke Xu are the corresponding authors.

0890-8044/23/$25.00 © 2023 IEEE

coefficient of the policy target to enhance the flexibility of the routing algorithm. The newly proposed G-Routing consists of two parts: DRL-based routing generation algorithm and GNN-based metric prediction algorithm. The DRL-based routing generation algorithm adopts an Actor-Critic model to generate optimal paths in real-time that meet the target requirements. The GNN-based metric prediction algorithm perceives the global network environment for a certain time period and dynamically adjusts the weighting coefficient of the policy target. It is implemented based on the GNN message passing mechanism [5], which can perform relational reasoning and combinatorial generalization. By modeling the graph structure information, it predicts network performance metric accurately.

Although there are many literatures on network modeling based on GNN to achieve metric prediction, most of them cannot be well adapted to specific application scenarios. G-Routing establishes a customized GNN model to better adapt to the dynamic network change scenario, and considers a new feature type (i.e., flow) into our GNN model. The algorithm can aggregate and update the information from various features (i.e., path, flow, and link) to comprehend the intricate relationship between routes and traffic. This enables accurate estimation of path-level metrics in the next time period, and assists the routing algorithm to adjust the target policy in real-time. Therefore, G-Routing can rapidly adapt to changes in the environment and select a relatively optimal path.

The application scenario of our G-Routing is illustrated in Fig. 1. Specifically, the network environment often presents large gaps in network states according to the occurrence of specific scenarios, such as network load aggravation caused by traffic change, network link failure caused by attacks or emergencies, and network topology changes caused by configuration. This leads to large changes in the state and demand of the network in the next time period. For example, during the live broadcast of the World Cup, the video backbone network traffic will surge at the beginning of the game, and the network load will increase. This new environment will last 90 minutes or even longer. To ensure the quality of live video applications, we should pay more attention to the two metrics of latency and throughput. Correspondingly, our G-Routing can flexibly adapt to this new demand. By sensing and modeling the current network environment, the predicted future information can adjust the reward weights (i.e., w_1, w_2, and w_3 illustrated in Fig. 1) in real time. Therefore, the agent can flexibly grasp the current focus of the network, and then learn a more realistic optimal routing path.

We summarize the key contributions of this article as follows:

- We propose a context-aware routing optimization algorithm framework, which utilize GNN to assist DRL agents in routing selection. By flexibly changing the configuration of target policy, the agent can adapt to environmental changes faster and more accurately.
- We propose a customized GNN-based metric prediction algorithm for dynamic network environments, which takes into account the cyclic dependency relationships among three network features (i.e., path, flow, and link).
- By using the network topology and traffic data in the real world, multiple comparison experiments are carried out under two environment change scenarios. Extensive experimental results demonstrate the reliability and stability of G-Routing.

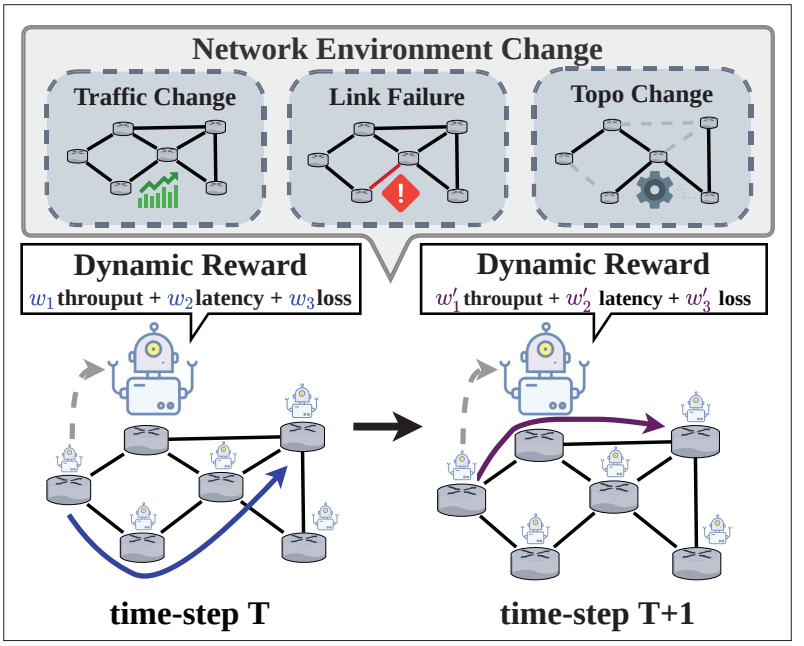

FIGURE 1. Routing optimization algorithm scenario.

Related Work

This section mainly introduces relevant research on network modeling and intelligent routing, including the development path and latest research achievements of the two fields.

Network Modeling

The task of the network modeling is to predict the network performance metrics (e.g., throughput and latency) under various "what-if" scenarios, such as network device reconfigurations. Network modeling offers the advantage of reducing the cost of online performance monitoring through real-time reasoning. Traditional network modeling can be approached in two ways. One approach involves using queuing theory to establish an analysis model with simplified assumptions, but the problem with this approach is that the assumptions may be too strong to be practical. Another approach involves using network simulators (e.g., NS3) to simulate real scenarios, but this approach has high computational costs and cannot perform real-time reasoning tasks.

With the development of deep learning, there have been many relevant network modeling works. For example, Deep-Q[6] uses the generative models to infer the network quality of service (QoS). However, its model is relatively simple and cannot be generalized. GNN has been found to be very suitable for modeling graph structure information. Moreover, GNN-based network modeling can improve generalization and accuracy. For example, RouteNet [7] performs accurate path-level performance estimation and generalizes to unseen topologies and routing schemes, while

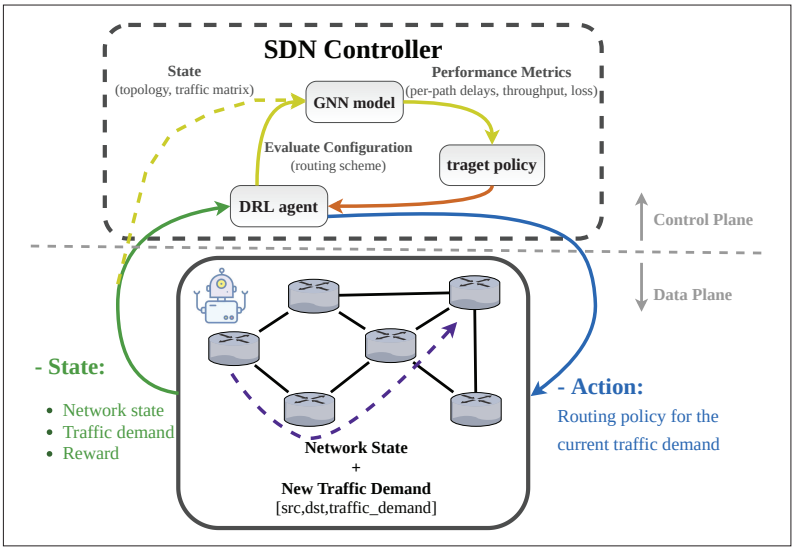

FIGURE 2. Architecture of routing optimization algorithm.

xNet [8] fully considers the fine-grained requirements of network model prediction and provides flow-level modeling capabilities. However, the problem with RouteNet is that it is too simple to be applied to complex scenarios. On the other hand, although xNet is detailed, the model is too complex and considers too many elements, making it also unsuitable for simple scenarios.

Taking into account the above considerations, we propose a customized GNN architecture for dynamic network changes, which endows DRL-based routing with generalization capabilities to adapt to specific scenarios.

Intelligent Routing

Intelligent routing [9] has become a research hotspot. For example, Mao *et al.* [10] envision a supervised deep learning system to construct the routing tables. Geyer *et al.* [11] propose a distributed routing algorithm that uses GNN to learn shortest paths and max-min routing strategies. However, the routing method based on supervised learning can only learn the policy of a specific network. It is difficult to adapt and generalize under the dynamic change of the network.

To improve the generalization ability, DRL-based routing methods are proposed. For example, Liu *et al.* [12] propose an online routing algorithm that uses multi-agent DRL to meet various service requirements. It models the route generation process as a multi-agent Markov decision process. However, it mainly focuses on a variety of application requirements, and its target policy is fixed. Therefore, it still cannot adapt well to dynamically changing network environments. Paul *et al.* [4] propose a DRL+GNN architecture for routing optimization. It optimizes network routing by modeling the Q-value in deep Q-network (DQN) with GNN, and generalizes on arbitrary topologies never seen before. However, the DRL algorithm used by Paul *et al.* [4] is too simple and cannot be well adapted to complex network environments.

To solve the above contradiction, our proposed DRL-based routing can easily select the appropriate action in the continuous action space and achieve a single-step fast update. At the same time, the target policy used in G-Routing can dynamically change with the environment to enhance flexibility.

G-Routing Scheme

In this section, we first introduce the architecture of G-Routing in detail, including the input and output of each part. Then we introduce the design details of DRL-based routing generation algorithm and GNN-based metric prediction algorithm. Finally, we introduce the weight update method of the policy target.

Architecture

Our proposed G-Routing requires the control plane to quickly perceive changes in the network environment. It explores the potential of routing algorithm to find the optimal target policy in the current environment, and chooses a better path for each incoming traffic demand. The architecture of the algorithm is illustrated in Fig. 2.

The control plane algorithm consists of two parts: a GNN-based metric prediction algorithm that predicts future path-level information, and a DRL-based routing generation algorithm that learns the optimal route in real-time. Both GNN model and DRL agent receive real-time network state information from the data plane. The DRL agent generates routing scheme based on traffic demand and sends it to the GNN model and data plane simultaneously. The GNN model generates the next-stage predictive information of metrics based on routing scheme and network state (i.e., topology, traffic matrix). It also updates the target policy in real-time based on the predicted values, and guides the DRL agent's routing behavior in the next time period.

Design

Here, we provide a detailed introduction to the control plane in Fig. 2. including internal architecture details and the network design of G-Routing, illustrated in Fig. 3.

DRL-Based Routing Generation Algorithm: It is deployed on the DRL agent, which selects the appropriate next hop based on the network state and traffic demand. When the selected next hop reaches the destination, the source-destination path is generated, and loaded to the data plane for execution.

The DRL agent adopts the proximal policy optimization (PPO) algorithm [13], and it solves the problem of low sample utilization. As illustrated in Fig. 3, PPO algorithm belongs to Actor-Critic algorithm and includes two models (i.e., actor and critic). The actor model probabilistically selects an action, while the critic model assesses the score of that action based on the reward feedback from the environment. Moreover, the actor model modifies the probability of action selection according to the critic's score.

The state input of the DRL agent consists of the environment state S and the condition state C. The environment state S includes traffic demands (e.g., source node, destination node, and data rate), topology information, and state information (e.g., remaining link capacity). The conditional state C indicates the selected nodes for the soon-to-be generated path. The output of the actor model is the probability distribution of which neighbor node to choose as the next hop,

and the next hop is selected randomly based on the set probability distribution. After the path is generated, it will be loaded into the data plane for execution, and the model will be updated based on the reward feedback from the environment. R_t is defined as the reward value of the action taken at time t. To calculate the reward R_t, the utility function combines multiple performance metrics (i.e., throughput *Thr*, latency *Lat*, and loss *Loss*), which can be compatible with various flows. Let Thr_t, Lat_t and $Loss_t$ be the throughput, latency and loss of the flow at time t, respectively. The utility function is illustrated in Eq. 1.

$$(1)\ R_t = w_1 \log Thr_t - w_2 Lat_t - w_3 Loss_t$$

where w_1, w_2, and w_3 represent the weighting coefficient of each performance metric. The predicted values generated by our GNN-based metric prediction algorithm will be used to update w_1, w_2, and w_3, as detailed below.

GNN-Based Metric Prediction Algorithm: As illustrated in Fig. 3, our GNN model considers three network features, namely path, link, and flow. It learns the complex relationships among these three features to accurately estimate the performance metrics of each source-destination pair's path.

Our prediction algorithm periodically collects various environmental information from the network, including topology, global routing schemes, path-level information (e.g., average latency, throughput, and loss), link-level information (e.g., bandwidth, usage of the link), and flow-level information (e.g., maximum data rate). It uses a fixed-dimensional vector to encode the state of path, link, and flow, defined as h_p, h_l, and h_f, respectively. The information is propagated and updated among states using the message passing mechanism. This mechanism for each feature state consists of two parts: message passing and updating. Message passing collects and aggregates hidden states from different features. And then, updating encodes the new aggregated information and the hidden state of that feature together to generate the next stage's hidden state.

More specifically, our message passing mechanism is based on three considerations:

- The state of a path depends on the states of all the links in the path and the flows flowing through the path.
- The state of a flow depends on the state of the path it travels.
- The state of a link depends on the states of all the flows passing through the link.

Based on the cyclic dependency relationship among the features described above, we depict the computation path of the GNN model as illustrated in Fig. 3. We also provide a detailed description of the forward propagation process of the GNN via Algorithm 1.

In Algorithm 1, our GNN model takes the route description P, flow description F, link description L, initial path feature h_p^0, initial flow feature h_f^0, and initial link feature h_l^0 as input, and outputs the inferred metric $\hat{y}_p$ for each path.

Specifically, lines 2 to 10 represent the message passing process of the path. Lines 4 and 7 represent that each path collects information from all the links it contains and all the flows flowing through it, respectively. Due to the sequential dependencies of links within each path, a recurrent neural network (RNN) is utilized to aggregate link-level messages. Specifically, the RNN employs a gate recurrent unit (GRU) network for this purpose. Since there is no sequential correlation for all flows along this path, summation is used directly to aggregate flow-level messages. Line 9 represents the update process of path-level information, and the update function uses three different trainable neural networks ϕ, μ, and π. Lines 11 to 13 represent the message passing process of the flow. It is set up in a simple way by directly specifying the path-level information that the flow flows through as the collected flow-level information. And then, a trainable neural network ψ is used to update the flow-level information. Lines 14 to 17 represent the message passing process of the link, where the states of all flows passing through the link is aggregated by summing, and another RNN is used to update the link-level information. The algorithm loop the three-stage message passing process T times to reach a convergent state.

Finally, *Readout* (line 19) represents a readout function that uses the path hidden state h_p^T to predict some path-level metrics ($\hat{y}_p$). The GNN model outputs three predicted performance metrics for *Thr*, *Lat*, and *Loss*. Correspondingly, we use the triplet $\varphi(i, j) = [Thr, Lat, Loss]$ to stores the prediction information (i.e., *Thr*, *Lat*, and *Loss*) on the path from source node i to destination node j in the next time period. The global average performance of each metrics is calculated and standardized. Further more, w_1, w_2, and w_3 are automatically configured according to the proportion of different metrics, as illustrated in Eq. 1.

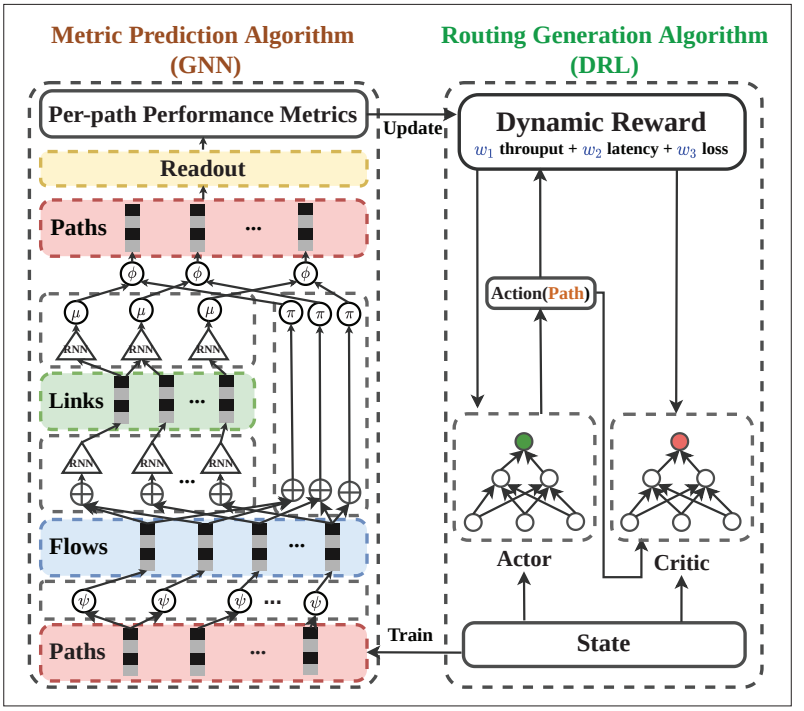

FIGURE 3. The network design of G-Routing.

Weight Update

The initial weights are set to $w_1 = 0.4$, $w_2 = 0.3$, and $w_3 = 0.3$. The experiment is designed to generate a sample per 100 source-destination pair

Input: $P, F, L, h_p^0, h_f^0, h_l^0$
Output: $\hat{y}_p$

for $t = 1:T$ do
  // Message passing of paths
  foreach p_i in P do
    foreach l_j in p_i do
      $m_{p,l} = RNN(h_{p_i}^t, h_{l_j}^t)$
    end
    foreach f_j in p_i
      $m_{p,f} = \sum h_{f_j}^t$
    end
    $h_{p_i}^{t+1} = \phi(\mu(m_{p,l}), \pi(m_{p,f}), h_{p_i}^t)$
  end
  // Message passing of flows
  foreach f_i in F do
    $h_{f_i}^{t+1} = \psi_{p:f \in p}(h_{p_i}^{t+1}, h_{f_i}^t)$
  end
  // Message passing of links
  foreach l_i in L do
    $m_{l_i} = \sum_{l_i \in f} h_{f_j}^{t+1}$
    $h_{l_i}^{t+1} = RNN(h_{l_i}^t, m_{l_i})$
  end
end
// Readout function
$\hat{y}_p = Readout(h_p^{t=T})$

ALGORITHM 1. Internal architecture of GNN.

requests, which includes the routing scheme on each path, path-level information (e.g., throughput, latency, and loss), and link-level information (e.g., link utilization). When 10 samples have been generated, the function is called to pack the 10 samples into a dataset for the current time period. The pre-trained GNN model uses these samples of the dataset and predicts the metrics of the next time period. The GNN model updates the parameters based on the feedback of the next time period.

The average performance of each metrics returned by the GNN is computed and saved in their respective lists. Take *Thr* as an example, the change rate Thr_{change} between the prediction and the current metric is calculated. Thr_{change} is multiplied by the weight w_1^t of Thr from the previous stage, and then the weight $w_1^{t+1} = w_1^t \times Thr_{change}$ of Thr_{t+1} can be obtained. The weights (i.e., w_2^{t+1} and w_3^{t+1}) of Lat_{t+1} and $Loss_{t+1}$ are updated in the same way, and new weights can be used to calculate the reward for the next time period of DRL agent.

Experiments

This section focuses on the evaluation method and experimental results of G-Routing.

Experimental Configuration

We use Ryu and Mininet [14] to build SDN architecture, in which Ryu is used as the controller and Mininet is used to simulate data plane. On the control plane, we use Pytorch 1.0 to implement our proposed G-Routing.

Parameter Setting: In the experiment, the DRL model we used is composed of three parts, namely the input layer, the feature extraction layer, and the policy layer. Our GNN model chooses a size of 32 for the hidden state of the path (h_p), 32 for the hidden state of the flow (h_f), and 16 for the hidden state of the link (h_l). In addition, for each forward propagation, we perform $T = 8$ iterations. We utilize the Adam optimizer to minimize losses, and the learning rate is set to 0.001.

Datasets: We use two real-world network topologies (i.e., Abilene [14] and GEANT [15]), where Abilene is a topology with 11 nodes and 14 bidirectional links, and GEANT is a larger topology with 23 nodes and 37 bidirectional links. Each nodes is connected to a host, and requests for flows are randomly generated based on the measured real traffic matrix. The sending rate of flow is fixed at 1500 kb/s, and the flow duration is 10 time slots. The environmental information (e.g., latency, throughput ratio, loss, and transmitted packets) is recored per 20 packets on each path.

Evaluation Criteria: Regarding comparison baselines, we first evaluate shortest path routing (SPR), which always uses the route with the least number of hops for each flow request. Second, we evaluate a DRL-based routing algorithm (DRL-OR) [12], which configures a fixed target through expert experience and uses it to learn a DRL model. We evaluate the average latency, throughput ratio, and loss of flows over a period of time when using each method in our experiments.

We compare the performance (i.e., latency, throughput ratio, and loss) of G-Routing with two baselines under environment change scenarios, and evaluate the adaptability of two DRL-based algorithms. Environment change scenarios include:

- *Traffic change:* Change the load in the network environment. A light load scenario with a flow duration of 10 time slots at initial time t_0 changes to a heavy load scenario with 30 time slots at time t_1.
- *Link failure:* Link failure occurs. All links are intact at the initial time t_0, and a small number of links are disconnected at time t_1.

Analysis of Performance

We evaluated the stability, generalization, and adaptability of G-Routing under link failure and traffic change. Specifically, for Fig. 4, the upper subplots and the lower subplots represent the experimental results under Abilene topology and GEANT topology, respectively. Likewise, the two row subplots in Fig. 5 are also the same configuration. Note that the analysis on stability is based on Abilene topology (i.e., the upper subplots in Figs. 4 and 5). Experiments on GEANT topology are used to compare the generalization ability between two DRL-based algorithm (i.e., DRL-OR and G-Routing).

Stability: Figures 4 and 5 illustrate the stability of different algorithms under two scenarios: traffic change and link failure. For traffic change, we simulate traffic bursts at time slot 10,000. As illustrated in Fig. 4a, when using DRL-OR based on a fixed policy target, the latency would increase sharply, even gradually entering a stable state, up to 75ms, which cannot adapt well to the new environment. Since the path selected by SPR is always fixed, that is, only the shortest path is selected, the change in the environment will not cause a significant fluctuation in performance of SPR. Our proposed G-Routing can quickly adapt to changes in the environment and maintain a lower latency after stabilization, approximately 13.5ms. Therefore, G-Routing reaches a stable state more quickly and obtains more stable routing schemes.

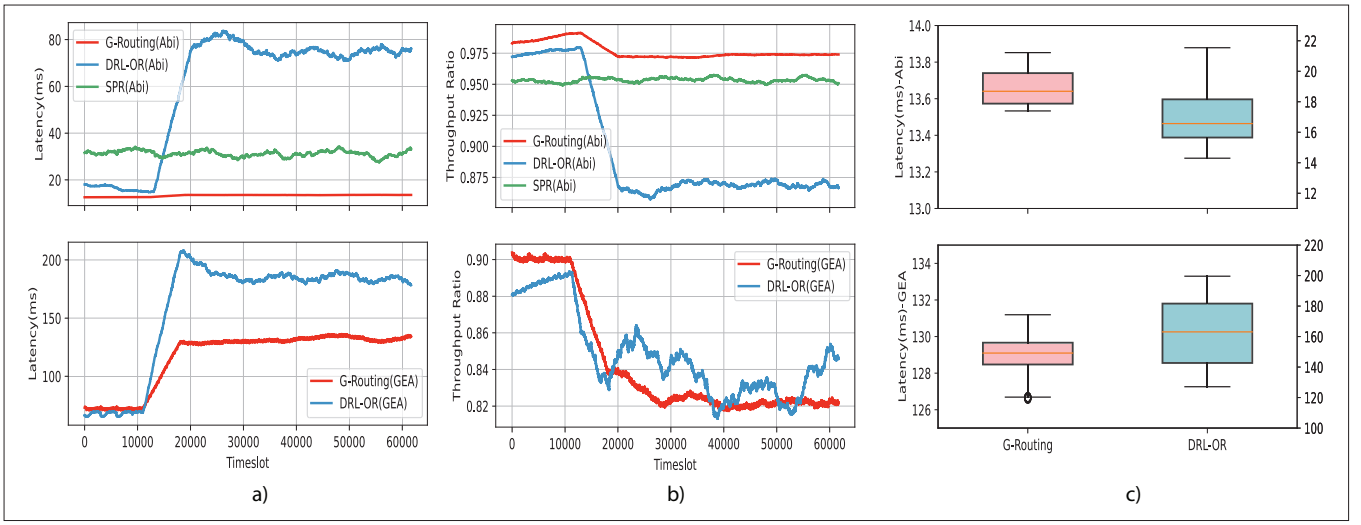

FIGURE 4. Results on flow latency and throughput ratio under traffic change: a) latency under traffic change; b) thr. ratio under traffic change; c) boxplot of the latency under traffic change.

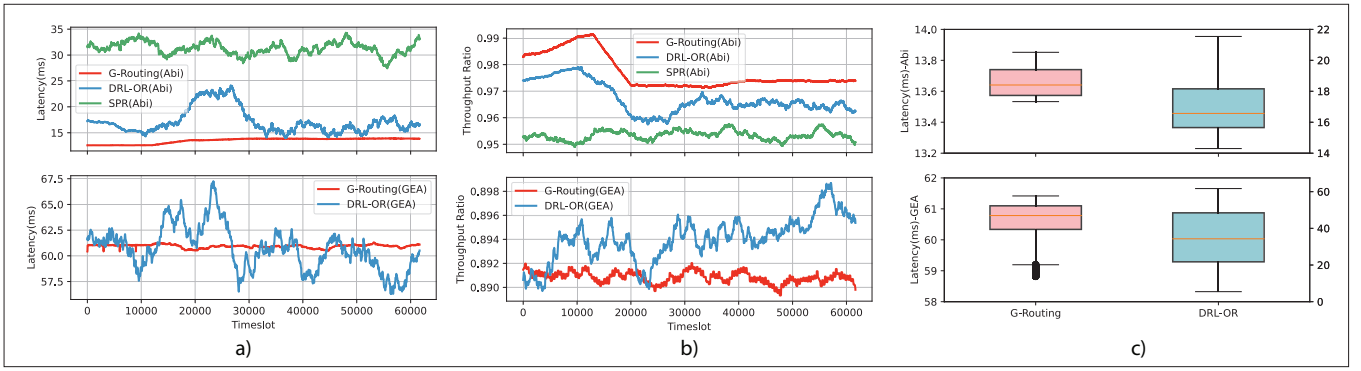

FIGURE 5. Results on flow latency and throughput ratio under link failure: a) latency under link failure; b) thr. ratio under link failure; c) boxplot of the latency under link failure.

By comparing the throughput ratio of different algorithms in Fig. 4b under traffic change, except for the SPR algorithm, both DRL-based routing algorithms (i.e., DRL-OR and G-Routing) have a decreased throughput ratio after traffic change, but G-Routing still has a relatively small change and higher throughput ratio after stabilization, indicating that our proposed G-Routing is more stable.

Similarly, for link failures, we simulate bottleneck link failures at time slot 10,000. In Figs. 5a and 5b, it can be found that the G-Routing algorithm is still the most stable among the three algorithms.

We also use boxplots to illustrate the data jitter under traffic change and link failure, illustrated in Figs. 4c and 5c, respectively. The boxplots illustrate the latency and throughput ratios from the beginning of environmental changes to the algorithm convergence (the data were extracted from time slots 10,000 to 20,000). It can be found that the variance of DRL-OR is significantly higher than that of G-Routing, indicating that our algorithm has stronger stability.

Generalization: Figures 4a, 4b, 5a, and 5b contain experimental results on different topologies (i.e., Abilene and GEANT), with the same experimental environment. They illustrate that our algorithm can achieve better stability on different topology structures, proving that our algorithm can generalize to different topologies.

Adaptability: To demonstrate the adaptability of G-Routing to new environments, we compare it to DRL-OR. Under traffic change, Fig. 6 illustrates the change in rewards of G-Routing and DRL-OR over time. The reward curve of G-Routing illustrates that the convergence speed of G-Routing is significantly faster than that of DRL-OR. Specifically, the reward of G-Routing basically converges at around the 25,000th time slot, while the convergence of DRL-OR is around the 40,000th time slot, indicating that G-Routing has a 2X faster convergence speed. Furthermore, the final convergence reward of G-Routing is higher than that of DRL-OR.

Conclusion and Future Work

In this article, we propose G-Routing, an online routing algorithm based on GNN and DRL. The algorithm employs GNN to predict the evolution of network performance metrics in the future time period, and assist DRL-based routing algorithm in selecting the optimal path. We implement G-Routing on the control plane, and simulate it under two environment change scenarios. Experimental results demonstrate that when the network environment changes significantly, our proposed G-Routing converges faster and produces a more dependable routing scheme.

The proposed GNN-based online routing sheds new light on better adapt to changes in the network environment. Due to the lack of inter-

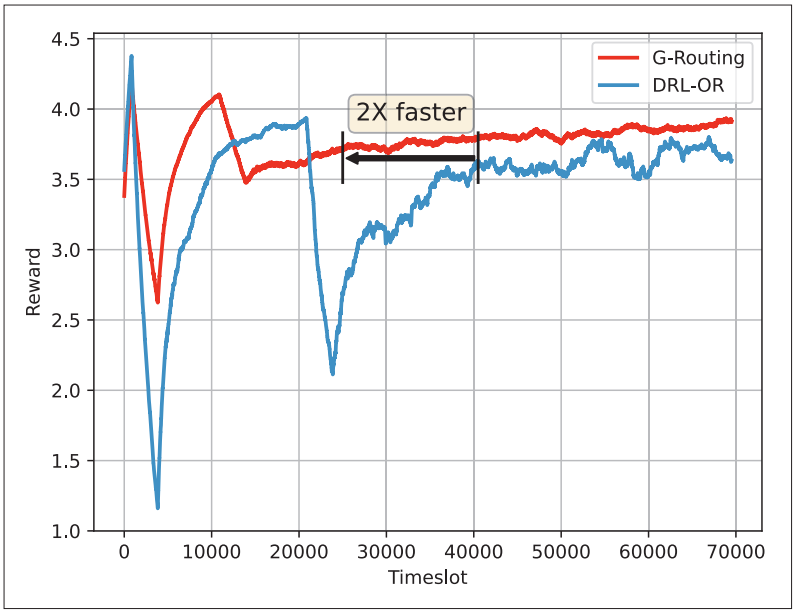

FIGURE 6. Adaptability to new environment.

pretability of the neural network on which GNN relies, there are still potential pitfalls in large-scale deployment. In the future, we will focus on the interpretability of such methods. With a clearer explanation of the internal mechanism of GNN, we can further improve the efficiency of intelligent routing.

Acknowledgments

This work was in part supported by the National Key R&D Program of China with No. 2022YFB3102302, National Science Foundation for Distinguished Young Scholars of China with No. 61825204, National Natural Science Foundation of China with No. 61932016, No. 62132011 and No. 62202258, Beijing Outstanding Young Scientist Program with No. BJJWZYJH012019100030011, China Postdoctoral Science Foundation with No. 2021M701894, China National Postdoctoral Program for Innovative Talents, and Shuimu Tsinghua Scholar Program. In this work, Huihong Wei and Yi Zhao made equal contributions. We also thank our editors and anonymous reviewers for their comments and guidance.

References

[1] Y. Zhao *et al.*, "Collaboration-Enabled Intelligent Internet Architecture: Opportunities and Challenges," *IEEE Network*, vol. 36, no. 5, 2022, pp. 98–105.
[2] N. Jay *et al.*, "A Deep Reinforcement Learning Perspective on Internet Congestion Control," *Proc. ICML*, 2019, pp. 3050–59.
[3] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," *Proc. ACM SIGCOMM*, 2017, pp. 197–210.
[4] P. Almasan *et al.*, "Deep Reinforcement Learning Meets Graph Neural Networks: Exploring a Routing Optimization Use Case," *Computer Commun.*, vol. 196, 2022, pp. 184–94.
[5] F. Scarselli *et al.*, "The Graph Neural Network Model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, 2009, pp. 61–80.
[6] S. Xiao, D. He, and Z. Gong, "Deep-Q: Traffic-Driven QoS Inference using Deep Generative Network," *Proc. Workshop on Network Meets AI & ML*, 2018, pp. 67–73.
[7] K. Rusek *et al.*, "Unveiling the Potential of Graph Neural Networks for Network Modeling and Optimization in SDN," *Proc. ACM Symposium on SDN Research*, 2019, pp. 140–51.
[8] M. Wang *et al.*, "xNet: Improving Expressiveness and Granularity for Network Modeling with Graph Neural Networks," *Proc. IEEE INFOCOM*, 2022, pp. 2028–37.
[9] N. Kato *et al.*, "The Deep Learning Vision for Heterogeneous Network Traffic Control: Proposal, Challenges, and Future Perspective," *IEEE Wireless Commun.*, vol. 24, no. 3, 2016, pp. 146–53.
[10] B. Mao *et al.*, "Routing or Computing? The Paradigm Shift Towards Intelligent Computer Network Packet Transmission Based on Deep Learning," *IEEE Trans. Computers*, vol. 66, no. 11, 2017, pp. 1946–60.
[11] F. Geyer and G. Carle, "Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning," *Proc. Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 2018, pp. 40–45.
[12] C. Liu *et al.*, "DRL-OR: Deep Reinforcement Learning-Based Online Routing for Multi-Type Service Requirements," *Proc. IEEE INFOCOM*, 2021, pp. 1–10.
[13] J. Schulman *et al.*, "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.
[14] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," *Proc. ACM HotNets*, 2010, pp. 1–6.
[15] S. Uhlig *et al.*, "Providing Public Intradomain Traffic Matrices to the Research Community," *ACM SIGCOMM Computer Commun. Review*, vol. 36, no. 1, 2006, pp. 83–86.

Biographies

Huihong Wei received her B. Eng. degree from the School Ph.D. degreeof Software and Microelectronics, Harbin Institute of Technology, Harbin, China, in 2017. Currently, she is pursuing a MA.Sc degree in the Institute of Network Science and Cyberspace at Tsinghua University. Her research interests include Intelligent Routing and machine learning.

Yi Zhao [S'19, M'21] received his B. Eng. degree from the School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, China, in 2016, and his Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2021. Currently, he is an Assistant Researcher and Postdoctoral Fellow at the Department of Computer Science and Technology, Tsinghua University. He is a recipient of the Shuimu Tsinghua Scholar Program. His research interests include next-generation Internet, network security, machine learning, and game theory. He is a member of IEEE, and a member of ACM.

Ke Xu [M'02, SM'09] received his Ph.D. from the Department of Computer Science and Technology of Tsinghua University, Beijing, China, where he serves as a Full Professor. He has published more than 200 technical papers and holds 11 US patents in the research areas of next-generation Internet, Blockchain systems, Internet of Things, and network security. He is a member of ACM and senior member of IEEE. He has guest-edited several special issues in IEEE and Springer Journals. He is an editor of IEEE IoT Journal and has served as a steering committee chair of IEEE/ACM IWQoS.