frontiers | Frontiers in Computational Neuroscience

TYPE Original Research
PUBLISHED 29 April 2024
DOI 10.3389/fncom.2024.1393025

OPEN ACCESS

EDITED BY
Qihe Shan,
Dalian Maritime University, China

REVIEWED BY
He Ren,
Northeast Electric Power University, China
Yinlei Wen,
Shenyang Jianzhu University, China

*CORRESPONDENCE
Geyang Xiao
✉ xgyalan@outlook.com

RECEIVED 28 February 2024
ACCEPTED 12 April 2024
PUBLISHED 29 April 2024

CITATION
He Y, Xiao G, Zhu J, Zou T and Liang Y (2024) Reinforcement learning-based SDN routing scheme empowered by causality detection and GNN.
Front. Comput. Neurosci. 18:1393025.
doi: 10.3389/fncom.2024.1393025

Reinforcement learning-based SDN routing scheme empowered by causality detection and GNN

Yuanhao He, Geyang Xiao*, Jun Zhu, Tao Zou and Yuan Liang

Intelligent Manufacturing Computing Research Center, Zhejiang Lab, Hangzhou, China

In recent years, with the rapid development of network applications and the increasing demand for high-quality network service, quality-of-service (QoS) routing has emerged as a critical network technology. The application of machine learning techniques, particularly reinforcement learning and graph neural network, has garnered significant attention in addressing this problem. However, existing reinforcement learning methods lack research on the causal impact of agent actions on the interactive environment, and graph neural network fail to effectively represent link features, which are pivotal for routing optimization. Therefore, this study quantifies the causal influence between the intelligent agent and the interactive environment based on causal inference techniques, aiming to guide the intelligent agent in improving the efficiency of exploring the action space. Simultaneously, graph neural network is employed to embed node and link features, and a reward function is designed that comprehensively considers network performance metrics and causality relevance. A centralized reinforcement learning method is proposed to effectively achieve QoS-aware routing in Software-Defined Networking (SDN). Finally, experiments are conducted in a network simulation environment, and metrics such as packet loss, delay, and throughput all outperform the baseline.

KEYWORDS

reinforcement learning, causal inference, graph neural network, SDN routing, quality-of-service

1 Introduction

Software-Defined Networking (SDN) routing separates the routing decision process from hardware devices, such as routers, allowing routing decisions to be made through a centralized software controller. This provides greater flexibility and stronger control capabilities. SDN routing (Tu et al., 2021, 2022; He et al., 2024) plays a crucial role in application scenarios that require high-quality network service, such as online video game and video conferencing.

The routing problem has been widely abstracted into graph theory model, where routers and links in the network are represented as nodes and edges in the graph. This model allows us to determine the optimal transmission path for data packets in the network through path selection algorithms. Early routing methods, such as distance-vector algorithms and link-state algorithms, had significant limitations in terms of computation and communication overhead, slow convergence speed, and poor network scalability. Heuristic algorithms can be used for routing optimization, but they have high computational complexity and increased the computational load on the SDN controller.

In recent years, there have been numerous studies attempting to optimize routing using machine learning techniques (Xie et al., 2019; Su et al., 2023; Teng et al., 2023; Xiao and Zhang, 2023), particularly through reinforcement learning (RL) methods. By maximizing rewards through continuous interaction with the environment, the agent is able to find optimal strategies. Causal inference can help understand the causal relationship between network events, identify cause of problems, and guide control decisions. These machine learning techniques can achieve more intelligent SDN routing, enhancing network performance and stability.

In this study, a QoS-aware network routing scheme that combines deep reinforcement learning, causal inference, and GNN is designed to improve routing performance. Real-time network state is collected by the reinforcement learning agent, which aggregates neighborhood information using CensNet (Jiang et al., 2019; Jiang et al., 2020), to obtain representations of nodes and edges as input to the deep reinforcement learning (DRL) model. The agent outputs link weights and generates routing policies based on the Dijkstra algorithm. Causal inference is used to measure the causal impact of actions on the network environment and guide the agent to explore the action space more effectively. Finally, the routing performance of the network topology is tested in a simulation environment. The innovation points of this study are mainly listed as follows:

- This study is the first to effectively combine causal inference and reinforcement learning, resulting in significant performance improvement in network routing problems.
- For QoS routing problems, causal inference is used to quantify the impact of actions, and a new reward function is designed to guide effective exploration of actions.
- Graph neural network is employed to simultaneously represent nodes and edges, which is applied to SDN network routing.
- Experiments are conducted on a network topology, and performance metrics are improved significantly, verifying the effectiveness of the proposed method.

2 Related work

2.1 Reiforcement learning

DRL-based methods (Bernardez et al., 2021; Casas-Velasco et al., 2021; Dong et al., 2021; Liu et al., 2021, 2023; Sun et al., 2022; He et al., 2023) deploy the agent within SDN controller and generate control signals based on reward feedback after interacting with the data plane. Distinguishing from supervised learning algorithms, DRL methods do not require labeled datasets and can converge to optimal policies through continuous iteration with the environment, achieving automated network operation (Sun et al., 2022). Bernardez et al. (2021) combined traffic engineering with multi-agent RL to minimize network congestion and optimize routing. Casas-Velasco et al. (2021) introduced the concept of knowledge plane into SDN and applies DRL for routing decisions. Dong et al. (2021) employed a generative adversarial network to learn domain-invariant features for DRL-based routing in various network environments. Liu et al. (2021), He et al. (2023), and Liu et al. (2023) proposed multi-agent RL approaches for hop-by-hop routing.

2.2 Causal inference

Causal inference is a method used to determine and quantify causal relationship by analyzing observed data and credible hypotheses, inferring causal connection between causes and effects rather than just correlation. Causal reinforcement learning is an umbrella term for RL approaches that incorporate additional assumptions or prior knowledge to analyze and understand the causal mechanism underlying actions and their consequences, enabling agents to make more informed and effective decisions. The four key applications of causal inference to RL include improving sample efficiency, enhancing generalization and knowledge transfer, eliminating spurious correlations, and studying interpretability, safety, and fairness in RL (Deng et al., 2023). Research studies such as Sontakke et al. (2021) and Huang et al. (2022) enhance sample efficiency in RL by conducting causal representation learning. Seitzer et al. (2021) improves the efficiency of the agent, exploring the action space by measuring the causal impact on the environment based on conditional mutual information. Pitis et al. (2020) explores counterfactual data by studying local independence condition in the environment, enriching the sample dataset and enhancing the generalization capability of the agent. Lu et al. (2018) eliminate decision bias of agent and improve decision accuracy by studying confounding bias in RL.

2.3 Graph neural network

Supervised learning algorithms (Rusek et al., 2019; Xie et al., 2019; Ferriol-Galm's et al., 2023) rely on labeled training datasets, where the model take network and traffic information as input to generate routing scheme. One major challenge of supervised learning methods is feature extraction, and the existing extraction methods generally perceive the network topological structure based on Graph Neural Network (GNN). Rusek et al. (2019) and Ferriol-Galm's et al. (2023) predicted network performance metrics (e.g., packet loss, delay, and jitter) for quality-of-service routing only through GNN.

3 Problem formulation

The network traffic considered in this study originates from any node and terminates at other nodes, represented as a discrete-time model (Liu et al., 2021), where traffic arrives in a predetermined time sequence. Each traffic flow is represented as a source node and a destination node. Additionally, the network topology is modeled as a bidirectional graph consisting of a collection of routers or switches and links. A DRL agent is deployed in the SDN controller, which takes network state as input and output routing control

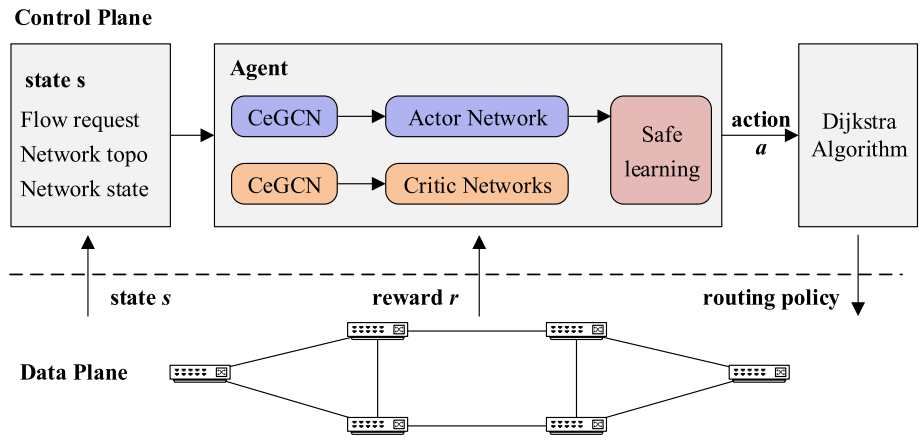

FIGURE 1
The system framework for SDN routing.

signals. The SDN controller creates routing tables and deploys them to the data plane to achieve traffic forwarding.

The objective of this study is that each traffic flow is successfully routed from the source node to the destination node, avoiding congested or failed links and maximizing the average reward for all traffic flows. It is important to note that once a routing policy is implemented on a specific traffic flow, the routing policy for that flow remains stable.

4 The proposed SDN routing scheme

4.1 System framework for SDN routing

In this study, the network state and global topology are obtained by the SDN controller and used as inputs for the RL agent. The scheme utilizes soft actor-critic (Haarnoja et al., 2018) integrated with nodes and links co-embedding (Jiang et al., 2019; Jiang et al., 2020) and applies causal action influence modeling (Seitzer et al., 2021) to reward feedback, called SAC-CAI-EGCN. Routing control policies are generated as outputs and prune action through a safe learning mechanism (Mao et al., 2019). The final routing strategy is then generated using the Dijkstra algorithm and deployed to the data plane, as shown in Figure 1.

4.2 Design of SAC-CAI-EGCN

SAC-CAI-EGCN includes an actor net, two critic nets and two target critic nets. The structure of actor and critic nets is shown in Figure 2. CeGCN part is employed to represent nodes and links, and then, the resulting link embedding and node embedding are concatenated as input to the actor and critic nets.

4.2.1 CeGCN part

To achieve simultaneous embedding of nodes and links, a three-layer network structure called CeGCN is designed, as shown in Figure 2. It consists of two edge-wise layers and a node-wise layer. The node-wise layer updates node embedding by combining the updated link embedding with propagation process referring to Equation (1). The edge-wise layer updates link embedding based on the input data with information propagation referring to Equation (2).

The node-wise propagation process of node features is shown in Equation (1).

$$H_v^{(l+1)} = \sigma(T\Phi(H_e^{(l)}P_e)T^T \odot \widetilde{A}_v H_v^{(l)} W_v) \tag{1}$$

The edge-wise propagation process of edge features is shown in Equation (2).

$$H_e^{(l+1)} = \sigma(T^T\Phi(H_v^{(l)}P_v)T \odot \widetilde{A}_e H_e^{(l)} W_e) \tag{2}$$

In which, T is a transformation matrix and $T_{i,m}$ represents whether node i connects edge m. Φ denotes the diagonalization operation of a matrix. P_e and P_v, respectively, represent the learnable weights of edge and node feature vectors. $\odot$ denotes the element-wise product operation. W_v is the network parameter in the node-wise propagation process, so as W_e.

In Equations (1, 2), $\widetilde{A}_e$ and $\widetilde{A}_v$ are calculated as Equation (3). A_i represents the adjacency matrix of nodes or edges, and i represents the node or edge. I_{N_i} is an identity matrix and D_i is the diagonal degree matrix of $A_i + I_{N_i}$.

$$\widetilde{A}_i = D_i^{-\frac{1}{2}}(A_i + I_{N_i})D_i^{-\frac{1}{2}} \tag{3}$$

4.2.2 DRL model

The agent is trained based on a quadruple data structure $< S, A, R, S' >$, which is defined in detail as follows:

- State S: the current state mainly includes (1) the representation of nodes and links generated by the CeGCN part; (2) the topology of network; and (3) the flow request. Specifically, the raw features for representation include the remaining available bandwidth and packet loss rate of each link, the number of flows, and the total size of data packets of each node.

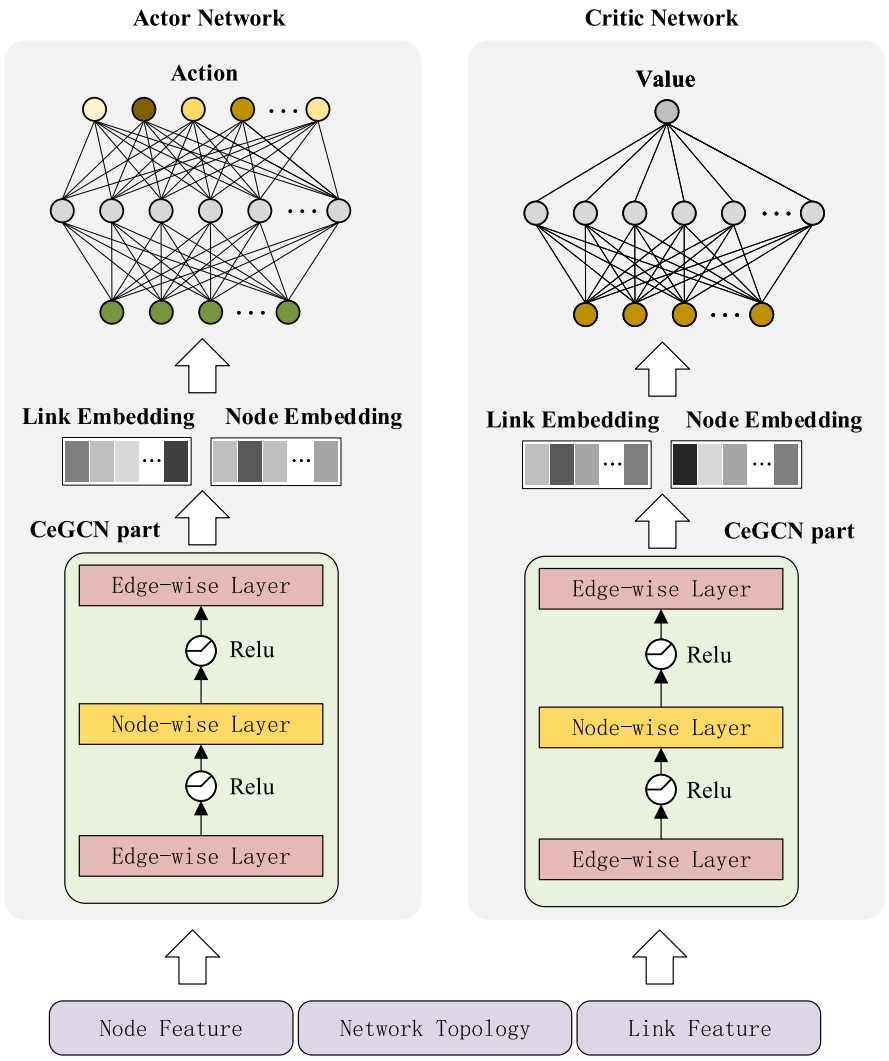

FIGURE 2
The structure of SAC-CAI-EGCN.

- Action A: the weights of links in the network which are decimals and belong to the interval (0, 1].
- Reward R: for comprehensive calculation of packet loss rate, delay, and throughput, a reward function is designed as follows:

$$r_t = -\gamma_1 x_t - \gamma_2 y_t + \gamma_3 z_t + \gamma_4 r_t^{cai} \tag{4}$$

$$r_t^{cai} = \frac{1}{N}\sum_{j}^{N} C^j(s_t) \tag{5}$$

$$C^j(s_t) = I(S'_j; A \mid S = s_t) = \mathbb{E}_{a\sim\pi}\left[\mathrm{D_{KL}}\left(P_{S'_j|s_t,a} \middle\| P_{S'_j|s_t}\right)\right] \tag{6}$$

In Equation (4), let x_t represents the packet loss rate, y_t represents the delay, and z_t represents throughput at t time slot, respectively. γ_1, γ_2, γ_3, and γ_4, respectively, represent the weight of packet loss rate, delay, throughput, and causal influence in the reward function. In this study, the reward function assigns weights to prioritize packet loss rate, delay, and throughput in the following order: γ_1, γ_2, γ_3, and γ_4 are assigned to be 2, 1.5, 1, and 1, respectively. In Equations (5, 6), S'_j represents the j-th component of S', D_{KL} denotes the KL divergence, and $C^j(s)$ quantifies the causal influence of action A on S'_j given the state $S = s_t$.

In specific scenarios, packet loss rate, delay, and throughput are not on the same scale, so normalization is required. The normalization operation for packet loss rate and delay is as follows:

$$x_t = \frac{x_t}{\bar{x}_t} \tag{7}$$

$$y_t = \frac{y_t - y_{base}}{\bar{y}_t} \tag{8}$$

Between Equations (7) and (8), y_{base} represents the average delay of all links in the network, which is set to be 5 ms. $\overline{x}_t$ and $\overline{y}_t$ denote the average loss rate and delay of the recent flows with the same source and destination as the flow at time slot t, which are approximated by the following equation.

$$\overline{m}_t = \begin{cases} m_t, & t = 1 \\ \epsilon \cdot \overline{m}_{t-1} + (1-\epsilon) \cdot m_t, & t \geq 2 \end{cases} \tag{9}$$

ϵ is a constant used to control the update rate of Equation (9). In this study, ϵ is 0.8.

For the throughput, after normalizing according to the bandwidth requirement z_{demand}, it needs the logarithmic change. The process is presented as Equation (10):

$$z_t = log(\frac{z_t}{z_{demand}} + b) \tag{10}$$

In Equation (10), in order to avoid abnormal values in the log operation, the parameter $b = 0.5$ is added.

- State S': after executing action A, state S' is acquired and it contains the same type of information as state S.

The experience of interacting with the environment is stored in the replay buffer and sampled by prioritized experience replay. The policy $\pi_\theta(s)$ is updated by the temporal difference method, in which θ represents the parameter of policy network. The loss function of actor net and critic net is as follows:

$$\widetilde{a}_t = f_{\theta,\epsilon_t \sim \mathcal{N}}(\epsilon_t; s_t) \tag{11}$$

$$L_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{R}}[\alpha log \pi_\theta(\widetilde{a}_t|s_t) - \min_{j=1,2} Q_{\omega_j}(s_t, \widetilde{a}_t)] \tag{12}$$

$$y_t = r_t + \lambda[\min_{j=1,2} Q_{\omega_j^-}(s_{t+1}, a_{t+1}) - \alpha log \pi(a_{t+1}|s_{t+1})] \tag{13}$$

$$L_Q(\omega) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1}) \sim \mathcal{R}}[\frac{1}{2}(Q_\omega(s_t, a_t) - y_t)^2] \tag{14}$$

As shown in the above equations, ϵ_t is a random noise variable sampled from the unit gaussian distribution $\mathcal{N}$. $\widetilde{a}_t$ is obtained through the reparameterization trick of Equation (11). The loss of actor net is calculated based on Equation (12). a_{t+1} is obtained by $\pi_\theta(\cdot|s_{t+1})$, and the loss of any critic net is calculated by Equations (13, 14).

As shown in Algorithm 1, lines 1–3 are to initialize actor net, two critic nets, two target critic nets, and replay buffer. Lines 4–10 collect experience, line 8 calculates the causal action influence r_t^{cai}, then lines 9–10 calculate reward r_t and store them to the replay buffer $\mathcal{R}$. Line 13 updates two critic nets $Q_{\omega_1}(s,a)$, $Q_{\omega_2}(s,a)$, and then lines 14–15 update actor net $\pi_\theta(s)$. Moreover, line 17 softly synchronizes parameter to the two target critic nets $Q_{\omega_1^-}(s,a)$, $Q_{\omega_2^-}(s,a)$.

Input: Replay buffer $\mathcal{R}$, Actor Network $\pi_\theta(s)$, Critic Networks $Q_{\omega_1}(s,a)$, $Q_{\omega_2}(s,a)$, Target Critic Networks $Q_{\omega_1^-}(s,a)$, $Q_{\omega_2^-}(s,a)$, Entropy regularization coefficient α, Parameter synchronization weighted τ

Output: Actor Net $\pi_\theta(s)$

1: Randomly initialize network parameters $\theta, \omega_1, \omega_2$ of Actor network and two Critic Networks
2: Copy the same parameters $\omega_1 \rightarrow \omega_1^-, \omega_2 \rightarrow \omega_2^-$ to initialize Target Critic Networks
3: Initialize the replay buffer $\mathcal{R}$
4: **for** Episode = 1, E **do**
5: **for** t in Trajectory **do**
6: obtain state s_t from the environment
7: execute action a_t, then get new state s_{t+1}
8: calculate r_t^{cai} according to Equations (5, 6)
9: calculate reward r_t according to Equation (4)
10: add experience $< s_t, a_t, r_t, s_{t+1} >$ into $\mathcal{R}$
11: **for** Training rounds = 1, K **do**
12: sample mini batch B from $\mathcal{R}$ by Prioritized Experience Replay
13: calculate loss with Equations (13, 14) and update Critic Networks $Q_{\omega_1}(s,a)$, $Q_{\omega_2}(s,a)$ by Adam optimizer
14: Sample action by reparameterization trick based on Equation (11)
15: calculate loss with Equation (12) and update Actor network by Adam optimizer
16: update Entropy regularization coefficient α
17: update Target Critic Networks by:

$$\tau\omega_i + (1-\tau)\omega_i^- \rightarrow \omega_i^-, i \in \{1,2\}$$

18: **end for**
19: **end for**
20: **end for**

Algorithm 1. SAC-CAI-EGCN routing algorithm.

4.2.3 Safe learning mechanism for routing

To prevent the degradation of performance caused by unsafe strategies, such as passing through failure or heavily congested links, a safe learning mechanism is designed. As shown in Figure 3, for each decision-making process, the control plane will determine whether the *safe condition* is met. For the current action and status s, the safe condition containing the following two items needs to be met simultaneously: (1) not passing through failure links and (2) not going through heavily congested links. If the safe condition is satisfied, the action will be output directly. If not, a fallback stable action will be output. Specifically, the weights of failure links or heavily congested links will be modified to the maximum value, which is 1. At the same time, an extra reward penalty will be fed back to guide the actor net to generate safer routing policies.

5 Experiments

5.1 Simulation setup

A public network topology is used, namely, GEANT2. It has 24 nodes and 37 bidirectional links. In the simulation environment,

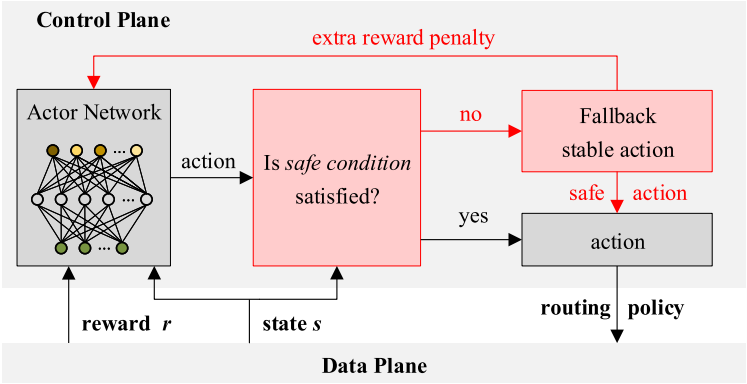

FIGURE 3
The safe learning mechanism for SDN routing.

TABLE 1 Performance results under GEANT2 network.

Method	Avg.lossRate	Avg.delay (ms)	Avg.thrpt (Kbps)
SPR	0.269	302.811	1025.295
SAC-CAI	0.090	108.107	1264.046
SAC-CAI-EGCN	**0.089**	**106.653**	**1266.710**

The bold values indicate optimal performance in the current column.

most of the links have a data rate of 10 Mbps, while there is two bottleneck links in GEANT2 with a data rate of 2.5 Mbps. Overall, 10% packet loss is added to each bottleneck link. Moreover, each link has a transmission delay of 5 ms.

In this study, the shortest path routing (SPR) is selected as the typical method for comparison, which only calculates the shortest hop number without considering the network state. The other one is SAC with causal action influence modeling called SAC-CAI.

5.2 Experiment results

5.2.1 Performance under given network

For a given network topology, the starting and ending nodes of flows are generated randomly, and the exact same traffic is used to test the three methods. For the GEANT2 network, the duration of flows is set to be 35 time slots, the global steps for the three methods are, respectively, set to be 30,000, 100,000, and 100,000.

Table 1 presents a comparison of three methods in terms of performance metrics including packet loss rate, latency, and throughput under the GEANT2 network topology. From the model reward curves, SAC-CAI and SAC-CAI-EGCN converge ~100,000 steps, while SPR exhibits congestion and latency at ~30,000 steps, so the SPR method only runs 30,000 steps. SAC-CAI-EGCN outperforms SPR and SAC-CAI significantly in all metrics under the same network topology, flows, and traffic intensity. First, the superior performance of SAC-CAI over SPR indicates the positive impact of causal inference in guiding action exploration for network routing. Second, SAC-CAI-EGCN exploits link and node co-embedding to effectively aggregate neighborhood features, thereby enhancing network routing performance in comparison with SAC-CAI.

5.2.2 Performance under different traffic intensities

To investigate the performance of SAC-CAI-EGCN, SAC-CAI, and SPR under different traffic intensities, an additional experiment with 25 time slot (light-load) flows was conducted. However, due to the poor performance of SPR and its significant difference in data scale compared with the other two methods, only the experimental results of SAC-CAI-EGCN and SAC-CAI are presented, as shown in Figure 4. First, as the traffic intensity increases, the packet loss rate and latency increase, while the throughput decreases. Second, from light to heavy traffic intensity, SAC-CAI-EGCN demonstrates superior performance in terms of packet loss rate, latency, and throughput compared with SAC-CAI.

6 Conclusion

In this study, based on action influence quantification and GNN a reinforcement learning method is proposed, enabling efficient SDN routing. Experimental results on publicly available network topology and different traffic intensities demonstrate significant improvement in QoS metrics, such as packet loss rate, latency, and throughput compared with baselines. This validates the effectiveness of SAC-CAI-EGCN in quantifying the causal impact of actions on the environment and simultaneously embedding edges and node features, guiding the generation of efficient SDN routing policies. In the future, we will continue exploring the application of causal reinforcement learning in improving network service quality, such as leveraging counterfactual data augmentation to improve sample efficiency and addressing confounding bias in RL.

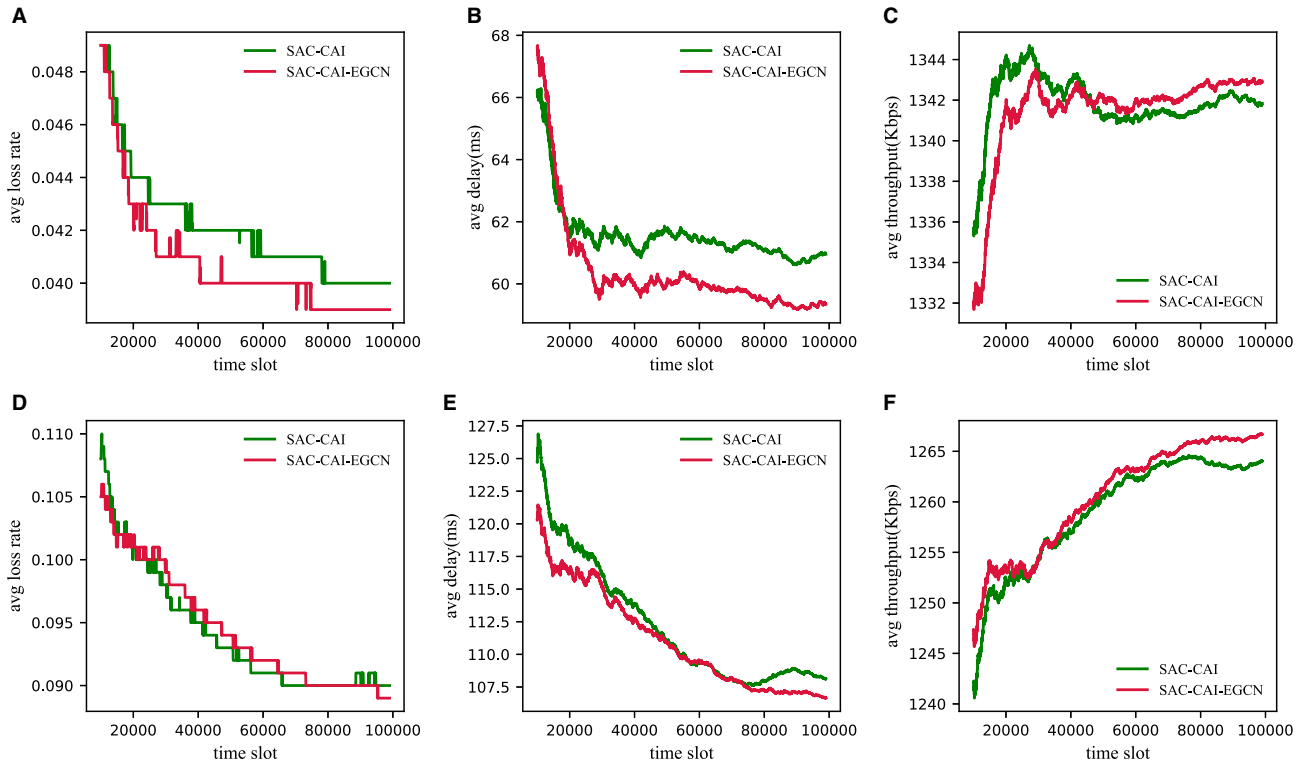

FIGURE 4
Performance results of three methods under different traffic intensities (light-load and heavy-load) in the GEANT2 network. **(A)** Loss rate under light-load. **(B)** Delay under light-load. **(C)** Throughput under light-load. **(D)** Loss rate under heavy-load. **(E)** Delay under heavy-load. **(F)** Throughput under heavy-load.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found at: https://www.cedefop.europa.eu/nl/news/geant2-world-beating-infrastructure-research.

Author contributions

YH: Conceptualization, Data curation, Methodology, Resources, Software, Validation, Writing – original draft, Writing – review & editing. GX: Formal analysis, Software, Supervision, Writing – review & editing. JZ: Visualization, Writing – review & editing. TZ: Writing – review & editing. YL: Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This study was supported in part by the National Natural Science Foundation of China under Grant U22A2005 and Grant 62203403.

Acknowledgments

The authors would like to thank the reviewers for their constructive and valuable suggestions on the earlier drafts of this manuscript.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Bernardez, G., Suarez-Varela, J., Lopez, A., Wu, B., Xiao, S., Cheng, X., et al. (2021). "Is machine learning ready for traffic engineering optimization?" in *2021 IEEE 29th International Conference on Network Protocols (ICNP)* (Dallas, TX: IEEE), 1–11. doi: 10.1109/ICNP52444.2021.9651930

Casas-Velasco, D. M., Rendon, O. M. C., and Fonseca, N. L. S. (2021). Intelligent routing based on reinforcement learning for software-defined networking. *IEEE Trans. Netw. Serv. Manag.* 18, 870–881. doi: 10.1109/TNSM.2020.3036911

Deng, Z., Jiang, J., Long, G., and Zhang, C. (2023). Causal reinforcement learning: a survey. *arXiv* [Preprint]. arXiv:2307.01452. doi: 10.48550/arXiv.2307.01452

Dong, T., Qi, Q., Wang, J., Liu, A. X., Sun, H., Zhuang, Z., et al. (2021). Generative adversarial network-based transfer reinforcement learning for routing with prior knowledge. *IEEE Trans. Netw. Serv. Manag.* 18, 1673–1689. doi: 10.1109/TNSM.2021.3077249

Ferriol-Galm's, M., Paillisse, J., Suárez-Varela, J., Rusek, K., Xiao, S., Shi, X., et al. (2023). Routenet-fermi: network modeling with graph neural networks. *IEEE/ACM Trans. Netw.* 31, 3080–3095. doi: 10.1109/TNET.2023.3269983

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). "Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning, ICML* 2018 (New York, NY: PMLR).

He, B., Wang, J., Qi, Q., Sun, H., and Liao, J. (2023). Rthop: real-time hop-by-hop mobile network routing by decentralized learning with semantic attention. *IEEE Trans. Mob. Comp.* 22, 1731–1747. doi: 10.1109/TMC.2021.3105963

He, Y., Xiao, G., Liang, Y., Lu, D., and Cheng, X. (2024). "A graph reinforcement learning routing scheme based on nodes and edges co-embedding," in *ICC 2024 - IEEE International Conference on Communications*. Piscataway, NJ: IEEE.

Huang, B., Lu, C., Leqi, L., Hernández-Lobato, J. M., Glymour, C., Schölkopf, B., et al. (2022). "Action-sufficient state representation learning for control with structural constraints," in *International Conference on Machine Learning* (New York, NY: PMLR), 9260–9279.

Jiang, X., Ji, P., and Li, S. (2019). "Censnet: convolution with edge-node switching in graph neural networks," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (International Joint Conferences on Artificial Intelligence Organization). San Francisco, CA: Morgan Kaufmann, 2656–2662. doi: 10.24963/ijcai.2019/369

Jiang, X., Zhu, R., Li, S., and Ji, P. (2020). Co-embedding of nodes and edges with graph neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 7075–7086. doi: 10.1109/TPAMI.2020.3029762

Liu, C., Wu, P., Xu, M., Yang, Y., and Geng, N. (2023). Scalable deep reinforcement learning-based online routing for multi-type service requirements. *IEEE Trans. Parallel Distrib. Syst.* 34, 2337–2351. doi: 10.1109/TPDS.2023.3284651

Liu, C., Xu, M., Yang, Y., and Geng, N. (2021). "DRL-or: deep reinforcement learning-based online routing for multi-type service requirements," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications* (Vancouver, BC: IEEE), 1–10. doi: 10.1109/INFOCOM42981.2021.9488736

Lu, C., Schölkopf, B., and Hernández-Lobato, J. M. (2018). Deconfounding reinforcement learning in observational settings. *arXiv* [Preprint]. arXiv:1812.10576. doi: 10.48550/arXiv:1812.10576

Mao, H., Schwarzkopf, M., He, H., and Alizadeh, M. (2019). "Towards safe online reinforcement learning in computer systems," in *NeurIPS Machine Learning for Systems Workshop*. New York, NY: Curran Associates.

Pitis, S., Creager, E., and Garg, A. (2020). "Counterfactual data augmentation using locally factored dynamics," in *Advances in Neural Information Processing Systems 33*. New York, NY: Curran Associates, 3976–3990.

Rusek, K., Suárez-Varela, J., Almasan, P., Barlet-Ros, P., and Cabellos-Aparicio, A. (2019). Routenet: leveraging graph neural networks for network modeling and optimization in SDN. *IEEE J. Sel. Areas Commun.* 38, 2260–2270. doi: 10.1109/JSAC.2020.3000405

Seitzer, M. Schölkopf, B., Martius, G. (2021). "Causal influence detection for improving efficiency in reinforcement learning," in *Advances in Neural Information Processing Systems 34*. New York, NY: Curran Associates, 22905–22918.

Sontakke, S. A., Mehrjou, A., Itti, L., and Schölkopf, B. (2021). "Causal curiosity: RL agents discovering self-supervised experiments for causal representation learning," in *International conference on machine learning* (New York, NY: PMLR), 9848–9858.

Su, Y., Shan, Q., Li, T., and Chen, C. P. (2023). Variable separation-based fuzzy optimal control for multiagent systems in nonstrict-feedback form. *IEEE Trans Fuzzy Syst.* 34, 547–561. doi: 10.1109/TFUZZ.2023.3302293

Sun, P., Guo, Z., Li, J., Xu, Y., Lan, J., Hu, Y., et al. (2022). Enabling scalable routing in software-defined networks with deep reinforcement learning on critical nodes. *IEEE/ACM Trans. Netw.* 30, 629–640. doi: 10.1109/TNET.2021.3126933

Teng, F., Zhang, Y., Yang, T., Li, T., Xiao, Y., Li, Y., et al. (2023). Distributed optimal energy management for we-energy considering operation security. *IEEE Trans. Netw. Sci. Eng.* 11, 225–235. doi: 10.1109/TNSE.2023.3295079

Tu, H., Zhao, G., Xu, H., Zhao, Y., and Zhai, Y. (2021). "Robustness-aware real-time SFC routing update in multi-tenant clouds," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)* (Tokyo: IEEE), 1–6. doi: 10.1109/IWQOS52092.2021.9521350

Tu, H., Zhao, G., Xu, H., Zhao, Y., and Zhai, Y. (2022). A robustness-aware real-time SFC routing update scheme in multi-tenant clouds. *IEEE/ACM Trans. Netw.* 30, 1230–1244. doi: 10.1109/TNET.2021.3137418

Xiao, G., and Zhang, H. (2023). Convergence analysis of value iteration adaptive dynamic programming for continuous-time nonlinear systems. *IEEE Trans. Cybern.* 54, 1639–1649. doi: 10.1109/TCYB.2022.3232599

Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C., et al. (2019). A survey of machine learning techniques applied to software defined networking (SDN): research issues and challenges. *IEEE Commun. Surv. Tutor.* 21, 393–430. doi: 10.1109/COMST.2018.2866942