A deep reinforcement learning approach for flexible multi-path routing using graph neural networks

Kun Geng [1,2], *Yingzi Zhang*[1,2]¯,*and Wenyi Liu* [1,2]

[1]North University of China, State key Laboratory of Extreme Environment optoelectronic Dynamic Testing Technology and Instrument, 030051 No. 3 Xueyuan Road, Jiancaoping, Taiyuan, China

[2]North University of China, School of Instrumentation and Electronics, 030051 No. 3 Xueyuan Road, Jiancaoping, Taiyuan, China

Abstract. Conventional dynamic routing protocols such as RIP and OSPF predominantly adopt single-path transmission strategies, which often result in inefficient bandwidth utilization and network congestion, especially under dynamic traffic demands. To solve these problems, this paper proposes a flexible multi-path routing optimization framework that integrates Deep Reinforcement Learning (DRL) with Graph Neural Networks (GNN). Specifically, the proposed approach introduces a multi-path decision-making mechanism that enables each traffic demand to be distributed across one or two selected paths combinations, with the goal of maximizing long-term network throughput. Then, a DRL agent is designed wherein the Q-value function leverages GNN-based state representations, capturing topological dependencies and link states. The action space of proposed agent is defined over combinations of candidate paths, allowing adaptive multi-path selection. Furthermore, a dynamic traffic allocation strategy is employed to proportionally split traffic based on the real-time residual capacities of selected paths which enhances load balancing and alleviates congestion. Simulation experiments conducted on realistic network topologies demonstrate that the proposed method achieves a 9.76% improvement in total allocated bandwidth compared to existing single-path DRL+GNN methods, validating its effectiveness in complex and dynamic network environments.

1 Introduction

Traditional dynamic routing protocols such as RIP, OSPF, and EIGRP typically adopt single-path transmission strategies, assigning each flow to one route, which often results in bandwidth underutilization or congestion when a single path cannot meet the traffic demand. In recent years, multi-path routing has been extensively studied to enhance throughput, load balancing, and fault tolerance. Yet, conventional approaches such as ECMP and link aggregation still lack the ability to adapt dynamically to real-time network changes.

The emergence of advanced transport technologies such as SDN (Software-Defined Networking) and MPTCP (Multipath TCP) has made it feasible to split traffic across multiple paths [1]. MPTCP, for instance, establishes multiple subflows and stripes data to aggregate bandwidth. However, its default scheduling relies on end-to-end metrics like RTT and loss rate, which may lead to inefficient resource usage and inter-path competition in complex networks [2].

To overcome these limitations, recent research has introduced machine learning techniques for multipath scheduling. In particular, combining DRL with GNN has shown strong potential in improving routing decisions and traffic engineering through intelligent, state-aware path selection. For example, Chen et al. [3] proposed GFlow, a GNN-based DRL algorithm designed to tackle the coarse-grained end-to-end scheduling in MPTCP and the potential link overlap across different services. The results demonstrated that GFlow outperformed baseline multipath mechanisms in both homogeneous and heterogeneous environments by improving average overall throughput and reducing average RTT. Almasan et al. [4] presented a DRL + MPNN-based routing optimization framework for OTNs (Optical Transport Networks), validating its effectiveness across multiple topologies and achieving strong generalization to previously unseen network structures. Shi et al. [5] proposed a GNN-based routing optimization algorithm for LEO satellite networks, which achieved 29.47% and 18.42% higher average throughput compared to Dijkstra and DQN algorithms, respectively, while reducing average end-to-end latency by 39.76% and 15.29%. Furthermore, the algorithm was adaptable to dynamic topologies. Xu et al. [6] proposed GN-DQN, a graph-based reinforcement learning routing scheme for software-defined networks, designed to maximize the long-term cumulative revenue of service providers. The approach integrates GNN blocks to jointly model link-level, node-level, and global network features. Yan et al. [7] proposed a flowlet-based multipath routing scheme under the SDN paradigm, using GNN to predict link

¯ Corresponding author:zhangyingzi@nuc.edu.cn

delays and proactively adjust flow splitting to reduce end-to-end latency and improve throughput.

These studies collectively highlight the advantages of reinforcement learning-based routing strategies in improving bandwidth utilization, reliability, and throughput compared to traditional approaches. Among reinforcement learning-based approaches, Graph Neural Networks (GNNs) are well-suited for capturing topological structure and inter-node dependencies, enabling more expressive state representations and enhancing the routing policy's ability to adapt to dynamic network conditions [8]. While the DRL+GNN agent proposed by Almasan et al. offers a strong baseline, its restriction to single-path transmission limits the ability to fully exploit network resources, as suggested by theoretical fluid models. Meanwhile, Optical Service Unit (OSU) technology, as an advancement of traditional OTN, addresses inefficiencies in supporting fine-grained services. OSU enables dynamic, lossless bandwidth adjustment from 2 Mbit/s to 100 Gbit/s, allowing for more efficient transport of fine-grained optical traffic and reducing bandwidth waste. In such scenarios, granular traffic allocation becomes increasingly important.

Therefore, our work seeks to enhance overall network bandwidth utilization by enabling flexible flow splitting across multiple paths. This paper presents a GNN-based multi-path routing optimization DRL agent, introducing a flexible decision mechanism that dynamically splits each traffic demand over one or two paths to maximize long-term throughput, defined as total allocated bandwidth. A central challenge lies in embedding multi-path decisions into the DRL action space and policy while preserving the generalization and scalability advantages of GNN-based state representations. The main contributions are:

(a) The Q-value function architecture, initially proposed by Almasan et al. , has been extended to accommodate a multi-path action space, which includes combinations of one to four paths. By leveraging GNN-based state encoding, the agent is equipped to model path overlaps and capture the complex dependencies between routing choices. This design enhances the agent's ability to evaluate long-term rewards across multiple routing alternatives, thereby enabling more adaptive and efficient decision-making in dynamic network conditions.

(b) A dynamic traffic allocation mechanism is introduced for discrete traffic demands (8, 32, and 64 units), wherein traffic is distributed across the selected paths proportionally to their available capacities. Drawing inspiration from fluid models, this strategy facilitates effective load balancing and mitigates the risk of congestion by adapting to real-time network conditions.

(c) The proposed method is implemented within a simulation framework and evaluated against a baseline DRL+GNN agent utilizing single-path routing. Experimental results indicate that the multi-path approach yields a substantial improvement in total allocated bandwidth, thereby validating its capability to enhance overall network utilization and routing efficiency.

2 Multi-path agent design

The proposed scheme aims to maximize overall throughput by considering both bottleneck and shared bandwidth across multiple paths. The network topology is represented as a graph, enabling the use of a Graph Neural Network (GNN) to process the structured data, which facilitates the agent's learning of nonlinear relationships among sub-flows, leading to more efficient traffic scheduling.

By leveraging GNN-based state representations, the DRL agent expands its action space to encompass combinations of one or two paths, thereby enabling dynamic adaptation to multi-path transmission scenarios. (see Fig. 1).

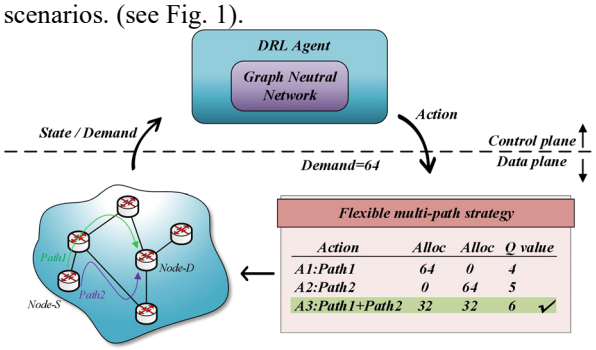


Fig. 1. Schematic representation of flexible multi-path routing agent

2.1 Input features

The input to the GNN is a joint representation of the current network state and the candidate action (the selected path combination). The network is modeled as a graph, $G=(N,L)$ where N denotes the number of nodes and L denotes the number of links. Each link $l \in L$ is associated with a feature vector that describes its current state. Node features are not explicitly considered, under the assumption that all nodes possess sufficient processing capabilities to accommodate any traffic demand without computational constraints.

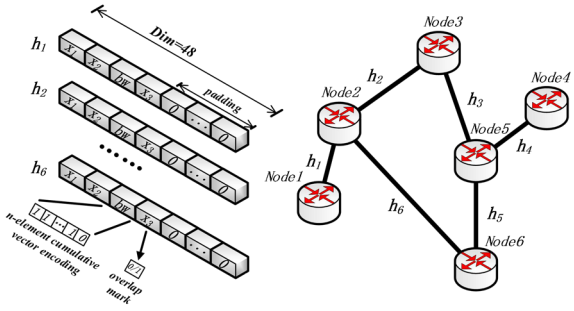


Fig. 2. Action representation in the link hidden states.

Fig. 2. Shows the feature representation embedding of the GNN. The link features include:

(i) Remaining Capacity X_1 -The available bandwidth of the current link;

(ii) Link Betweenness Centrality X_2 - A static metric that measures the frequency of a link appearing in the

shortest paths, helping the agent quickly identify critical links;

(iii) Traffic Demand *bw* - The traffic assigned to each link after the agent's action, using an n-element cumulative vector encoding. Traffic intensity is represented by a 32-dimensional vector, where the number of "1" vectors indicates the traffic intensity. At maximum intensity, there are 32 "1" vectors.

(iv) Overlap Mark X_3 - used to indicate whether there are overlapping links between paths.

In addition, zero-padding is applied to reserve space in the hidden state vector for encoding information from neighboring links, while ensuring consistency in link feature dimensions. These features together form the initial hidden state h_l^0 of each link, with a total dimension set to 48.

2.2 MPNN based graph embedding

The GNN architecture employed in this study adheres to the Message Passing Neural Network (MPNN) paradigm, wherein at each propagation step, each link node generates messages through a lightweight multilayer perceptron (MLP), aggregating information from its neighbors to iteratively update hidden states.

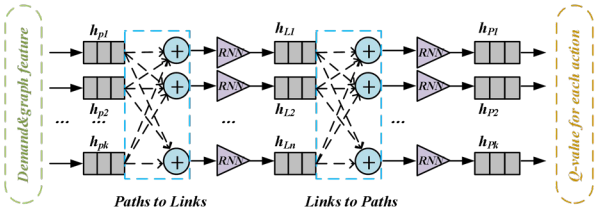


Fig. 3. Message passing architecture.

Upon receiving messages, each neighboring node aggregates them—typically via summation—and updates its hidden state using another neural network. This iterative process facilitates effective information propagation throughout the network. As a result, links along multiple candidate paths can exchange information. For example, if several selected sub-paths converge at the same node, the links connected to that node can share information, reflecting that they serve the same traffic flow by T rounds of message passing. After T iterations, each link obtains a final hidden state h_l^T that captures not only local features but also integrates global contextual information.

Subsequently, a readout function aggregates the hidden states of all links into a global representation, which is then used to predict a scalar Q-value. Specifically, the hidden states are summed across all links and passed through a fully connected layer, like:

$$Q(s,a) = f\left(\sum_{l\in L} h_l^T\right) \tag{1}$$

where f denotes a small feedforward neural network. The summation-based pooling ensures that the output remains invariant to graph size, thereby enhancing the model's generalization capability across diverse network topologies.

2.3 DQN based routing optimization Agent

2.3.1 DQN algorithm

The proposed multi-path routing optimization agent is fundamentally built upon the Deep Q-Network (DQN) algorithm.

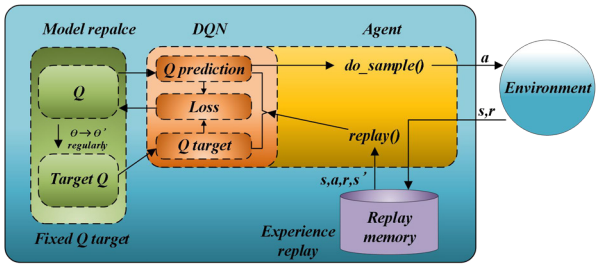


Fig. 4. DQN architecture.

Fig. 4 illustrates the standard DQN workflow. At each decision step, the agent samples a traffic demand and computes the Q-values for all possible actions using the Bellman equation (see Equation (2)), where $Q(s_t,a_t)$ denotes the Q-value function at time step t, α is the learning rate, $r(s_t,a_t)$ is the reward obtained by taking an action at from a given state st, and $\gamma \in [0,1]$ is the discount factor.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r(s_t,a_t) + \gamma \max_{a'} Q(s_t',a') - Q(s_t,a_t)) \tag{2}$$

Then, an action is selected using the ε-greedy strategy based on the Q-values, favoring the action with the highest estimated value. After executing the selected action, the environment returns a reward and the next state. The transition tuple—comprising the current state, action, reward, and next state—is stored in the experience replay buffer. Subsequently, a batch of transitions is sampled to perform gradient descent and update the network parameters. The loss function is given by:

$$Loss = (r + \gamma \max_{a'} Q_{t\,\text{arget}}(s_t',a') - Q(s_t,a_t))^2 \tag{3}$$

Here, Q_{target} denotes the target network, while Q represents the main network; they are responsible for estimating the Q-values of the next state s_t' and the current state s_t, respectively. In DQN, the target network's weights are fixed for a certain number of steps and periodically updated from the main network to improve training stability.

Since both states and actions involve graph-structured information, the proposed GNN architecture is used to evaluate each candidate path combination individually. To balance long-term rewards, the discount factor is set to γ=0.95.

In addition, by introducing path combination action encoding, the agent can autonomously determine whether traffic splitting is necessary under dynamic network conditions, and adapt its path combination strategy accordingly based on link-sharing patterns.

2.3.2 Reward and objective

The reward function guides the learning process of the multi-path agent. To maximize long-term returns, the reward is proportional to the total amount of allocated traffic. The normalized reward, which is the sole type of reward used, is defined as follows:

$$r_t = \frac{bw_t}{bw_{\max}} \tag{4}$$

The immediate reward for successfully allocating a traffic demand is equal to the bandwidth of that demand. If the allocation fails, the reward is zero and the episode terminates. There is no explicit penalty term; however, since a blocked allocation ends the episode early and limits the accumulated reward, the agent is implicitly penalized. The objective of this work is to maximize the total allocated bandwidth within each episode, which aligns with maximizing the overall network throughput.

2.3.3 Training method

The study employs a reinforcement learning approach based on DQN. A replay buffer is used to store state-action-reward-next state transitions, with a total capacity of 8000. An Ɛ-greedy strategy is employed for exploration, starting with Ɛ = 1.0 (fully random), which gradually decays to 0.1 over the first 20 episodes and remains at 0.1 thereafter. The ADAM optimizer is used with a learning rate of 0.0005. During training, a random batch of transitions is sampled from the replay buffer to perform gradient descent. The objective function is defined by Equation (2), where Q_{target} is computed by a target network that is updated every 100 steps to stabilize training. A total of 350 training iterations are conducted (each iteration corresponds to 20 rounds within an episode), which is sufficient for policy convergence.

For the GNN component, two layers of message passing are applied (T = 2). The hidden state dimension is set to 48. Each message and update function consists of a two-layer MLP with 32 neurons per layer. The readout layer sums the hidden states of all links and passes the result through two fully connected layers to output the Q-value.

3 Flexible multi-path transmission

3.1 Action space design

Compared to the original single-path agent, the action space is expanded to include multi-path combinations. For each source-destination pair, the agent selects a path combination from a candidate set of the 4 shortest paths which results in 10 possible multi-path actions formed by different path subsets like:

$$\sum_{r=1}^{2}\binom{4}{r} = 10 \tag{5}$$

The agent selects one of these actions based on the current network state. By limiting the maximum number of paths in a combination to 2, the action space remains manageable while aligning with practical scenarios where traffic is typically split across a small number of paths.

Table 1. Action space dimension for path combinations.

K'	Dimension	R=1	R=2
4	10	4	6
3	6	3	3
2	3	2	1
1	1	1	—

Table 1. lists the dimension of path combinations in the action space for each case where 1 to 4 distinct shortest paths exist between a source-destination (S-D) pair. Here, K' denotes the number of distinct paths, and R represents the number of paths in each combination, with each path traversing each node only once. When there are 4 distinct paths, the state space contains 10 possible combinations: 4 single-path and 6 double-path combinations. In cases where only one shortest path exists between the S-D pair, the multi-path action degenerates into a single-path action.

Based on current network congestion and demand size, the agent dynamically selects either a single path or multiple paths. For light traffic and small demands, a single path may be preferred to reduce resource overhead. For large 64 units demands where no single path suffices, the agent splits traffic across multiple paths to meet the requirement which enables adaptive subpath selection under varying network conditions.

The link-level GNN captures path overlap and joint capacity availability by propagating messages that encode flow demand and residual link capacity. Such message propagation enables the agent to reason about combined capacities across multiple paths.

To simplify the action space and guide learning, constraints are introduced: (1) duplicate paths in a selected action are disallowed to avoid redundancy; (2) if any path in a multi-path action cannot carry its share of the demand, the entire allocation fails, yielding zero reward and terminating the episode. While such logic is not hardcoded, the reward design encourages the agent to avoid unnecessary splitting. For example, excessive path splitting may lead to resource fragmentation, reducing future rewards due to failed allocations.

3.2 Network-state-aware traffic allocation

To reduce the dimensionality of the decision space, the agent does not directly output the splitting ratio across selected paths. Instead, it selects the paths to use, and a predefined proportional allocation strategy is

applied. After selecting a set of paths, the total demand V is allocated among the chosen paths in proportion to the available capacity of each path. This aligns with fluid model principles, where flow is divided to avoid bottlenecks. Specifically, suppose a demand of traffic is to be routed over two selected paths, P_1 and P_2. Let C_1 and C_2 denote the bottleneck capacity along each path, f_1 and f_2 denote allocated traffic on P_1 and P_2.

Basically, the demand is split proportionally by:

$$f_1 = \frac{C_1}{C_1 + C_2} \times V \quad f_2 = \frac{C_2}{C_1 + C_2} \times V \tag{6}$$

When the total capacity of the path combination is insufficient, the allocation fails regardless of splitting, making the action invalid with zero reward.

This ensures no path is over-allocated and more load is assigned to paths with higher capacity. The allocation strategy is heuristic, resembling max-flow or water-filling algorithms, aiming to balance the load and reduce the risk of path overload. The agent receives positive feedback for successful splits and none for failures, gradually learning effective multi-path scheduling policies. It is important to note that this work abstracts away packet-level concerns such as reordering and delay. Flow-level modeling is used, assuming that upper-layer protocols handle reassembly of split traffic.

4 Experimental settings and results

4.1 Experimental setup

This section presents the training and evaluation settings of the GN-DQN agent in the context of SDN-based routing optimization. In the experiments, three types of traffic demands are considered, each requiring 8, 32, 64 units of bandwidth. During the training phase, source-destination node pairs and bandwidth demands are sampled from a uniform distribution.

4.2 Benchmark schemes

To demonstrate the effectiveness of the proposed GN-DQN scheme, the following routing optimization strategies are employed as baseline methods:

DQN+MPNN Agent: This approach, proposed in [4], is also a GNN-based routing optimization method. Specifically, it utilizes a MPNN to learn link representations across the network.

Load Balancing (LB): A widely adopted network optimization strategy that aims to prevent congestion by evenly distributing traffic across the network.

Shortest Available Path (SAP): This strategy selects the shortest path (in terms of hop count) by default. If the shortest path is unavailable, it sequentially attempts the next shortest available paths.

To ensure a fair and rigorous comparison, the proposed multi-path DQN model and the DQN+MPNN baseline are evaluated under the same experimental configuration. Specifically, both models are configured with a learning rate of 0.0001, a batch size of 32, and an exploration strategy based on an Epsilon-Greedy approach with a discount rate of 0.95.

4.3 Illustration of training evolution

Figure 5 illustrates the training progression of our proposed GN-DQN model compared to the baseline DQN+MPNN on the 14-node NSFNet topology [9], over 14,000 training episodes. To evaluate performance, both models are assessed every 40 training episodes (one iteration) by averaging the revenue obtained over the following 40 evaluation episodes. The light-colored shading represents the raw fluctuations in the loss values, while the darker lines show the results smoothed using a Savitzky-Golay filter, highlighting the overall trend. The blue and red curves correspond to the baseline model and the proposed model, respectively. Simulation results demonstrate that, compared to the DQN + MPNN agent, the proposed model achieves lower converged loss values, indicating better performance.

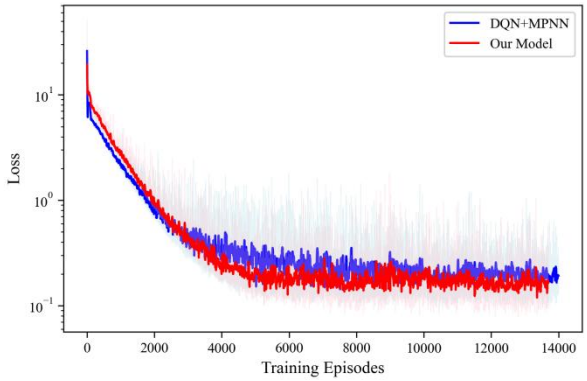


Fig. 5. Illustration of training convergence through loss evolution.

Figure 6 illustrates the training scores of the proposed model compared to the baseline DQN model, where the score is measured by the total amount of successfully allocated traffic. As shown, the proposed model requires fewer training episodes to converge and achieves higher scores. The improvement can be attributed to the agent's flexible composite traffic allocation strategy by making more efficient use of link capacities.

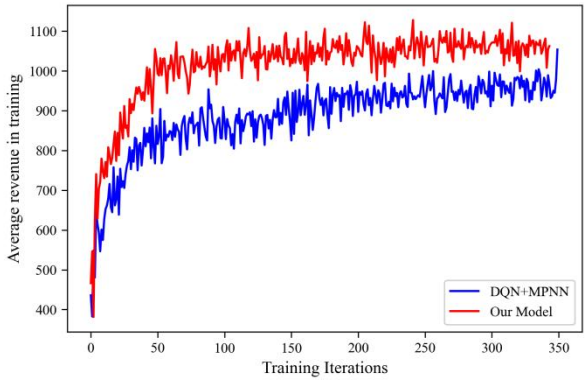
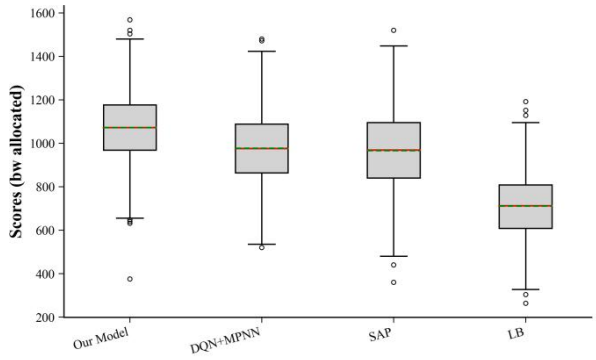


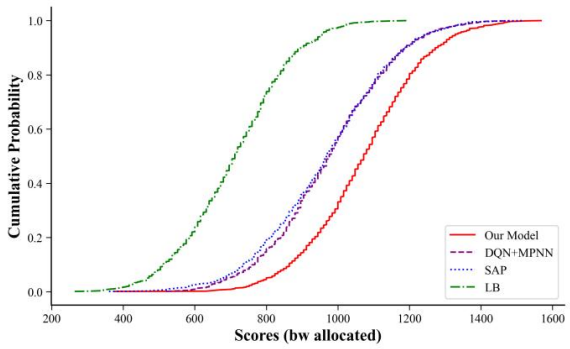
Fig. 6. Evolution of average revenue during training.

4.4 Experimental results

Specifically, Figure 7(a) presents the distribution of per-episode rewards across 1,000 episodes using boxplots. The green dashed line denotes the mean, while the red solid line indicates the median. The box boundaries correspond to the 25th and 75th percentiles, with whiskers extending to the most extreme non-outlier points (within 1.5 times the interquartile range). Outliers are plotted individually. The proposed method achieves an average reward of 1,072, representing a 9.76% improvement over the DQN+MPNN baseline (976.64), an 11% improvement over the SAP method (965.76), and a striking 150.82% improvement over the Load Balancing (LB) baseline (710.4). Figure 7(b) shows the cumulative distribution of per-episode rewards. With the proposed method, only 30.50% of the episodes yield a reward below 1000.00, compared to approximately 54.90% and 55.80% for DQN+MPNN and SAP, respectively, and up to 97.20% for LB.



(a)



(b)

Fig. 7. Boxplot and cumulative distribution of revenue per episode under uniform traffic distribution.

5 Conclusion

This paper proposes a flexible multi-path adaptive routing method based on DRL and GNN to improve bandwidth utilization. By enabling each traffic demand to be distributed over one or two paths using discrete units (0-64), the approach supports fine-grained allocation and addresses the limitations of single-path routing. The DRL agent jointly incorporates multi-path selection and network-aware allocation strategies, while GNNs encode topological structure and action representations. As a result, the proposed agent effectively captures complex inter-path interactions, enabling precise, state-dependent decision-making. Simulation results demonstrate a 9.76% improvement in total allocated bandwidth compared to the DRL+GNN baseline, underscoring the significant potential of integrating multi-path flexibility into DRL-based routing optimization to enhance overall network resource efficiency.

References

1. Z. Shu, H. B. Du, X. Y. Zhu, et al., Research on the control strategies of data flow transmission paths for MPTCP-based communication networks. PeerJ Comput. Sci. **9**, e1716 (2023). https://doi.org/10.1016/j.nanoen.2020.104652
2. F. Jowkarishasaltaneh, J. But, An Analysis of MPTCP Congestion Control. Telecom **3**, 581–609 (2022). https://doi.org/10.3390/telecom3040033
3. D. Chen, W. Zhang, D. Gao, et al., GFlow: GNN-Based Optimal Flow Scheduling for Multipath Transmission with Link Overlapping. IEEE Trans. Netw. Sci. Eng. **11**(6), 6244–6258 (2024). https://doi.org/10.1109/TNSE.2024.3481413
4. P. Almasan, J. Suárez-Varela, K. Rusek, et al., Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. Comput. Commun. **196**, 184–194 (2022). https://doi.org/10.1016/j.comcom.2022.09.029
5. Y. Shi, W. Wang, X. Zhu, et al., Low Earth Orbit Satellite Network Routing Algorithm Based on Graph Neural Networks and Deep Q-Network. Appl. Sci. **14**(9), 3840 (2024). https://doi.org/10.3390/app14093840
6. J. Xu, Y. Wang, B. Zhang, et al., A Graph reinforcement learning based SDN routing path selection for optimizing long-term revenue. Future Gener. Comput. Syst. **150**, 412–423 (2024). https://doi.org/10.1016/j.future.2023.09.017
7. B. Yan, Q. Liu, J. L. Shen, et al., Flowlet-level multipath routing based on graph neural network in OpenFlow-based SDN. Future Gener. Comput. Syst. **134**, 140–153 (2022). https://doi.org/10.1016 /j.future.2022.04.006
8. K. Rusek, J. Suárez-Varela, P. Almasan, et al., RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN. IEEE J. Sel. Areas Commun. **38**(10), 2260–2270 (2020). https://doi.org/10.1109/JSAC.2020.3000405
9. J. B. Hamrick, K. R. Allen, V. Bapst, et al., Relational inductive bias for physical construction in humans and machines. arXiv preprint arXiv:1806.01203 (2018). https://doi.org/10.48550/arXiv.1806.01203