electronics

MDPI

Article

Research on Generalized Intelligent Routing Technology Based on Graph Neural Network

Xiangyu Zheng [1], Wanwei Huang [1,*], Hui Li [1] and Guangyuan Li [2]

[1] College of Software Engineering, Zhengzhou University of Light Industry, Zhengzhou 450001, China
[2] Henan Xin'an Communication Technology Co., Ltd., Zhengzhou 450001, China
* Correspondence: huangww@zzuli.edu.cn

Abstract: Aiming at the problems of poor load balancing ability and weak generalization of the existing routing algorithms, this paper proposes an intelligent routing algorithm, GNN-DRL, in the Software Defined Networking (SDN) environment. The GNN-DRL algorithm uses a graph neural network (GNN) to perceive the dynamically changing network topology, generalizes the state of nodes and edges, and combines the self-learning ability of Deep Reinforcement Learning (DRL) to find the optimal routing strategy, which makes GNN-DRL minimize the maximum link utilization and reduces average end-to-end delay under high network load. In this paper, the GNN-DRL intelligent routing algorithm is compared with the Open Shortest Path First (OSPF), Equal-Cost Multi-Path (ECMP), and intelligence-driven experiential network architecture for automatic routing (EARS). The experimental results show that GNN-DRL reduces the maximum link utilization by 13.92% and end-to-end delay by 9.48% compared with the superior intelligent routing algorithm EARS under high traffic load, and can be effectively extended to different network topologies, making possible better load balancing capability and generalizability.

Keywords: load balancing; generalizability; graph neural network (GNN); deep reinforcement learning (DRL); routing strategy

check for updates

Citation: Zheng, X.; Huang, W.; Li, H.; Li, G. Research on Generalized Intelligent Routing Technology Based on Graph Neural Network. *Electronics* **2022**, *11*, 2952. https://doi.org/10.3390/electronics11182952

Academic Editors: Houbing Song and Jehad Ali

Received: 29 August 2022
Accepted: 13 September 2022
Published: 17 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

As one of the core functions of network communication, routing is used to ensure that data packets are efficiently sent from source nodes to destination nodes. Traditional routing algorithms include static routing algorithms and dynamic routing algorithms. Static routing algorithms are relatively simple and easy to set up, and are suitable for networks with small network scale and relatively stable topology. Dynamic routing algorithms can adjust routing strategies in real time according to network conditions, and are suitable for network environments of larger size and more complex topology [1]. Although traditional routing algorithms have been deployed in various network environment, with the continuous complexity of network structures, traditional routing algorithms are not able to guarantee better network quality of service (QoS) [2], such as by reducing the maximum link utilization, transmission delay, and packet loss rate.

In recent years, with the rise of artificial intelligence, researchers have tried to apply machine learning technology to the field of routing optimization, which mainly includes intelligent routing schemes based on unsupervised learning, supervised learning, and reinforcement learning. In [3], the authors used the k-means algorithm in unsupervised learning to achieve network traffic classification. In [4], the authors proposed a lightweight communication network dynamic routing algorithm based on a convolutional neural network, which provides intelligent paths according to online training of traffic patterns; the overall bandwidth utilization rate is about 70%. Average network throughput increases by approximately 40% compared to existing mechanisms. The authors of [5] proposed applying the Q-Learning method to wireless sensor networks, with a Markov decision

process (MDP) used to model the data packets transmitted in the network, solving the problems of low packet transmission rate and high delay of wireless sensor network (WSN). In [6], the authors proposed the application of deep reinforcement learning (DRL) to the field of routing optimization to realize continuous iterative update routing strategy. The authors of [7] proposed a DRL routing algorithm based on SDN, which improves network performance such as delay and throughput. Their intelligent routing scheme has the advantages of strong adaptability, self-training, and no need for manual marking.

At this stage, the intelligent routing algorithm based on DRL is the most effective method for routing algorithms based on machine learning. It usually uses traditional neural networks such as multilayer perceptron (MLP) [8], recurrent neural network (RNN) [9], and long short-term memory networks (LSTM) [10] as the training subject. However, the input and output of these neural networks are relatively strict, and can only process data with a fixed-dimensional Euclidean structure, which is not suitable for dynamically changing network topologies (for example, link interruption/connection, node addition/deletion, etc.); furthermore, it is difficult to integrate the generalized training experience with other network topologies. In addition, although there are intelligent routing algorithms based on graph convolutional neural network (GCN) [11] which can process data with non-European structures, such intelligent routing algorithms only use the perceptual ability of the neural network for training, and do not integrate it with DRL's decision-making capabilities, making it is difficult to cope with complex network environments.

In response to the above problems, this paper proposes a DRL intelligent routing algorithm, GNN-DRL, based on graph neural network (GNN) [12] in the SDN environment. The algorithm adopts deep deterministic policy gradient (DDPG) [13] as the main frame, and uses GNN to replace the conventional neural network in DDPG to vectorize the attributes of network topology nodes and edges. GNN can use the dependencies in the network topology and the update function in DDPG to update the nodes and edges of the vectorized representation in real time. GNN-DRL can effectively generalize the training experience to different network topologies according to the structure of the input and output of the variable dimension of GNN while continuing to provide a more optimized routing scheme in the face of new network or network structure changes. Under different network traffic requirements, it can provide a good routing scheme to reduce the maximum link utilization and average end-to-end delay. The main contributions of this paper are as follows:

(1) Analyze the problems of poor QoS and weak generalization ability of traditional routing algorithms and current intelligent routing algorithms, and propose GNN-DRL intelligent routing algorithm to solve the problems of load balancing and generalization.

(2) A DDPG algorithm framework suitable for load balancing and generalization in intelligent routing is proposed, and GNN is adapted for network topology replacement, network interruption, or failure.

(3) Using the experimental environment of various traffic loads and constructing different network topologies, the intelligent routing algorithm is deployed to compare and verify the optimization effect of the GNN-DRL routing algorithm.

The remaining part of the article is organized as follows: Section 2 describes the current research status of intelligent routing algorithms and introduces GNN; Section 3 defines the routing rules and builds the GNN-DRL intelligent routing framework; Section 4 describes the GNN-DRL intelligent routing scheme in detail; and Section 5 verifies the load balancing ability and generalization of GNN-DRL through experiments. Finally, Section 6 summarizes the findings of the paper.

2. Related Works

2.1. Intelligent Routing Algorithms

The current intelligent routing algorithms based on machine learning are mainly divided into three categories: intelligent routing algorithms based on supervised learning, intelligent routing algorithms based on unsupervised learning, and intelligent routing

algorithms based on reinforcement learning. Among them, the unsupervised learning method only needs to model by summarizing the rules of data feature information, avoiding the process of collecting prior knowledge. In [14], the authors studied a scheme based on unsupervised learning, using principal component analysis to extract network features and complete the analysis of network traffic characteristics, although manual design of routing strategies is required and the algorithm's accuracy is low. The authors of [15] studied an opportunistic network routing algorithm based on a clustering algorithm, which was able to effectively improve the node search speed and adapt to the changing network topology quickly. However, this algorithm needs to further optimize the node clustering, and its environmental adaptability needs to be improved. For the above reasons, the performance of intelligent routing algorithms based on unsupervised learning remains difficult to guarantee.

Intelligent routing algorithms based on supervised learning mainly adopt the deep neural network model. These algorithms take network state information such as traffic demand, link utilization, delay, and throughput in the network topology as input data, which are preprocessed and input into the deep neural network model. After the training of the deep neural network is completed, the decision result is output as the output value. In [16], the authors studied a routing decision scheme based on a deep belief network (DBN) and used this scheme for the network backbone. Compared with the traditional routing scheme, this method has faster convergence and lower cost of information exchange. The authors of [17] carried out an experimental comparison of several different deep learning models and found that the combination of deep learning and topological structure feature extraction can improve accuracy compared with the DBN model. Overall, intelligent routing algorithms based on supervised learning need to manually mark a large number of network traffic characteristics, lack better scalability and the ability to make effective routing decisions, and need improved performance in terms of convergence speed, robustness, and accuracy of routing, and adaptability to faults.

Reinforcement learning is an important branch of machine learning that can be used in Markov decision processes. Deep reinforcement learning is based on the basic theory of reinforcement learning. It uses a deep neural network to replace the original decision function, and takes advantage of the powerful fitting ability of deep neural networks for training. In [18], the authors studied the semi-state-independent traffic engineering scheme SMORE, which improves the load balancing capabilities of routing algorithms such as delay and throughput to a certain extent. Subsequently, the authors of [19] studied the introduction of deep reinforcement learning technology in the field of intra-domain traffic engineering and proposed the DRL-TE scheme for multi-path traffic division, which optimizes multi-path routing by controlling the traffic split ratio and thus has better generality and robustness compared to SMORE. Existing research shows that intelligent routing schemes based on DRL are relatively mature; these schemes take the network state as input and the routing strategy as output. Because traditional neural networks in DRL have difficulty defining the number and Euclidean distance of their neighbor nodes for Non-Euclidean Structure Data (NSD), where the data are not neatly arranged, this means that traditional neural networks require different processing for different data types. Therefore, these neural networks usually have fixed requirements for the input and output formats of different data samples in order to simplify the computation, and are suitable for processing Euclidean Structure Data such as text sequences, images, etc. In real scenarios, the network topology often changes dynamically, as does the input format of the neural network. Therefore, it is necessary to readjust the input format of the neural network in order to further judge whether the neural network can output a better routing strategy after training convergence. As a result, traditional neural networks cannot generalize the training experience to different network topologies and their generalization ability is weak, making it difficult to cope with complex problems such as link interruption and node failure.

2.2. Graph Neural Networks

GNN is a new type of neural network architecture. GNN has strong generalization and reasoning ability, and is used to process graph-structured data [20]. In the basic form of a GNN, initial states are associated with a graph structure with different elements in the GNN, then these elements are iteratively updated in combination with the interrelationships in the graph. In [21,22], the authors show that GNN has better generalization and reasoning ability than neural networks such as MLP, LSTM, and RNN through relevant experiments. There are three general frameworks in GNN, namely, message passing neural networks (MPNN) [23], non-local neural networks (NLNN) [24], and graph networks (GN) [20], with the latter a more general summary of GNN compared to MPNN and NLNN.

In the GN framework, a Graph is defined as a 3-tuple $G = (u, V, E)$, where u is the global attribute, $V = \{v_i\}_{i=1:N^v}$ is the set of nodes (cardinality is N^v), the element v_i is the state of the node, $E = \{(e_k, r_k, s_k)\}_{k=1:N^e}$ is the set of edges (cardinality is N^v), the element e_k is the state of the edge, r_k is the state of the receiving node, and s_k is the state of the sending node. A GN block includes three update functions ${e'}_k = \phi^e(e_k, v_{rk}, v_{sk}, u)$, $\bar{e}' = \rho^{e\to u}(E')$, and $\bar{v}' = \rho^{v\to u}(V')$, as well as three aggregation functions ${\bar{e}'}_i = \rho^{e\to v}({E_i}')$, $\bar{e}' = \rho^{e\to u}(E')$, and $\bar{v}' = \rho^{v\to u}(V')$. These update functions cover the edge, node, and global update processes, respectively. The update process of GN is shown in Figure 1.

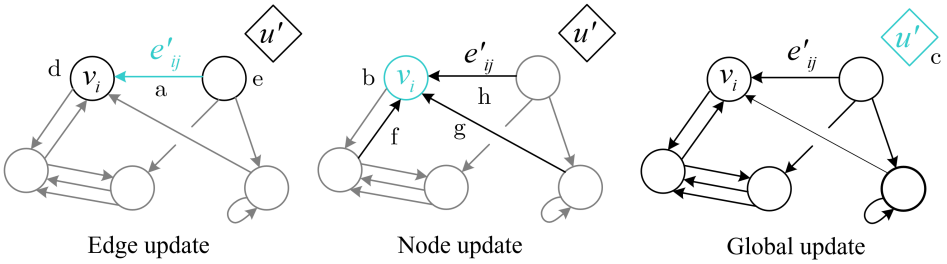

Figure 1. Update process of GN.

In Figure 1, the update process of the GN block mainly includes the following three forms: (1) edge updating, in which the edge attributes are updated through the aggregation of the edge itself as well as the sending node, receiving node, and global state (for example, the edge update in Figure 1 represents nodes d and e for updating of edge a); (2) node updating, in which the node attributes are updated through the node itself, node-adjacent edges, and the global state (for example, the node update in Figure 1 represents the update of edges f, g, and h, aggregating node b); and (3) global updating, in which the global attributes are updated through aggregation of edges and nodes and aggregation of the initial value of the global state itself (for example, the global update in Figure 1 represents the update of node aggregation and edge aggregation to global c).

3. GNN-DRL Intelligent Routing Model

3.1. Definition of Routing Rules

In order to judge the load balancing ability and generalization of different routing algorithms under the unified routing rules, we define the following five routing rules:

(1) Define the network model as a directed graph $G = (V, E, c)$, where V is a set of nodes, E is a set of links, and $c : V \to R^+$ is a mapping function with R^+ as the value range that assigns each edge a capacity for storing link information.

(2) During the traffic transmission process between the specified source node s and the destination node t, it is assumed that the traffic needs to be forwarded to node u through node v and that the traffic transmission needs to comply with the two constraints of Equations (1) and (2):

(1) There is no traffic loss between the source node and the destination node:

$$\sum_{u=\Gamma(v)} \Re_{v,(s,t)}(u)=1,\ \forall_{s,t} \in V \cap v \neq t \text{ and } u \in \Gamma(v) \quad (1)$$

(2) All traffic is absorbed by the destination node:

$$\sum_{u=\Gamma(v)} \Re_{t,(s,t)}(u)=0,\ \forall_{s,t} \in V \text{ and } u \in \Gamma(v) \tag{2}$$

where $\Gamma(v)$ is the set of all neighbor nodes of node v and $\Re_{v,(s,t)}(u)$ is the proportion of traffic forwarded by node v to node u. The above traffic transfer constraints are used to observe the optimal performance of the routing algorithm in a stable network environment.

(3) Define the traffic demand matrix between nodes as shown in Equation (3):

$$D_{m\times n} = \begin{pmatrix} d_{11} & \cdots & d_{1m} \\ \vdots & \ddots & \vdots \\ d_{m1} & \cdots & d_{mn} \end{pmatrix} \tag{3}$$

where $D_{m\times n}$ represents the traffic demand matrix of all nodes in the network topology and the element d_{mn} in the matrix represents the traffic demand of one of the nodes.

(4) Define $U(v,u)$ as the utilization rate between link (v,u) and satisfy the constraint of Equation (4):

$$\forall(v,u) \in E,\ \exists U_{max} > U(v,u) \tag{4}$$

(5) Define $delay(v,u)$ as the end-to-end delay of link (v,u); the average end-to-end delay is calculated as shown in Equation (5):

$$delay_{average} = \frac{\sum_{i=1}^{n} delay(v,u)}{n},\ n \in N^{+} \tag{5}$$

where n is the total number of links.

3.2. GNN-DRL Intelligent Routing Framework

The intelligent routing framework of GNN-DRL is implemented based on the new SDN network architecture, which decouples traditional network forwarding devices and control functions and uses distributed forwarding and centralized control to achieve more fine-grained network control with better scalability and flexibility. In this way, GNN-DRL builds an intelligent routing architecture based on the advantages of SDN, which consists of three functional planes from south to north: the data plane, control plane, and intelligent plane. The GNN-DRL intelligent framework is shown in Figure 2.

In Figure 2, the functions of the data plane, control plane, and intelligent plane of the GNN-DRL intelligent routing architecture are introduced: (1) the data plane is the bottom layer of the SDN architecture, and consists of several network forwarding devices (routers, switches) which are mainly responsible for data processing and forwarding, although it does not have the ability to control the data; (2) the control plane is composed of several SDN controllers, which obtain the network information of the data plane in real time through the south interface, allowing flexible completion of centralized control and management of network resources according to different user requirements and thereby realizing the full utilization of network resources; (3) the intelligent plane is composed of the GNN-DRL agent, which can interact with the network information of the control plane through the north interface to complete the training of the routing strategy and realize the optimal routing strategy.

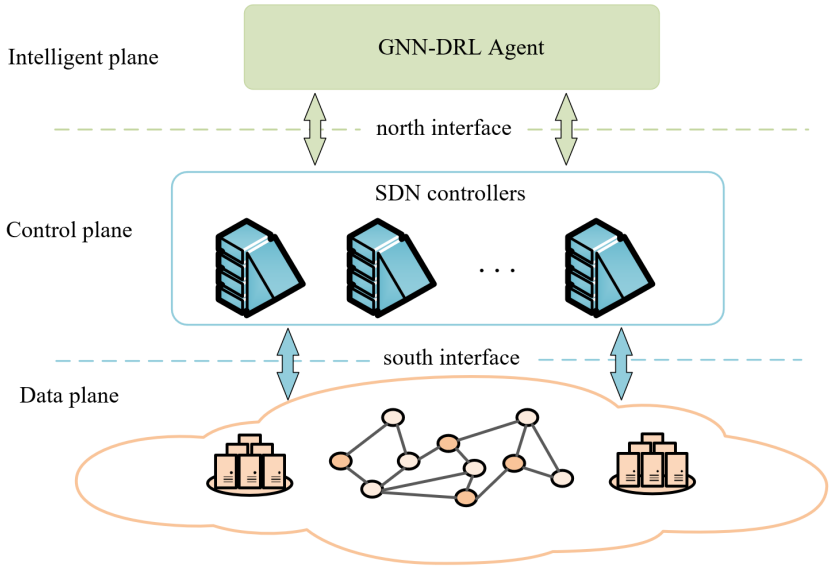

Figure 2. GNN-DRL intelligent routing framework.

4. GNN-DRL Intelligent Routing Solution

4.1. GNN-DRL Intelligent Routing Algorithm

This paper combines the DDPG algorithm in deep reinforcement learning with GNN to construct an intelligent routing algorithm based on GNN-DRL. GNN-DRL utilizes the advantages of DDPG's online network and target network as well as the application of a soft update algorithm which can promote the learning process make it more stable and ensure model convergence. Moreover, DDPG requires fewer samples and does not need to integrate the action space, which effectively reduces the algorithm's complexity [25]. However, traditional Actor and Critic frameworks in DDPG algorithms usually use MLP, RNN, LSTM, etc., as neural networks. Such neural networks often lack flexibility, and the training process can only perform input and output in a fixed format, making it difficult to adapt suitable routing strategies under different network topologies. In order to improve the generalization of routing policies for different network topologies, this paper replaces the traditional neural network in DDPG with a GNN. This allows the GNN update process to be integrated with the online network and the target network in DDPG, taking advantage of the better combination and generalization capability of GNN and enhancing the generalization of the network topology in case of link interruption or failure. The GNN-DRL algorithm framework is shown in Figure 3.

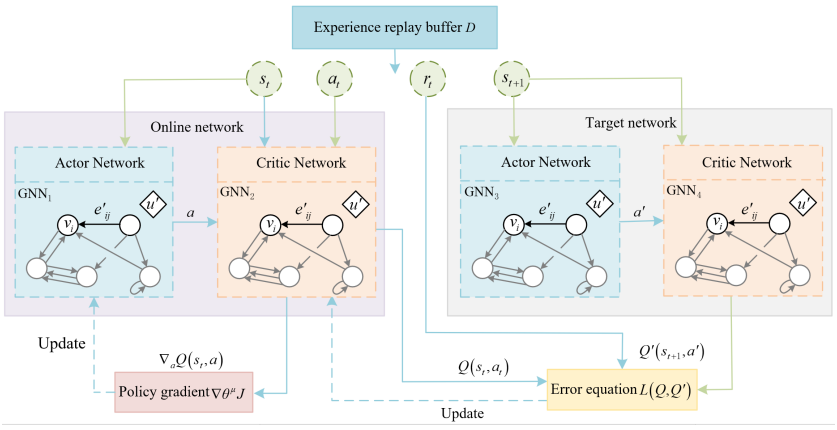

Figure 3. GNN-DRL algorithm framework.

In Figure 3, Actor and Critic networks are constructed with the same network structure and initialization parameters in the online network and target network. Among them,

GNN_1 and GNN_2 represent the improved online Actor and Critic network, respectively, and GNN_3 and GNN_4 represent the improved target Actor and Critic network, respectively, which can effectively receive the dynamically changing node and link information in the network topology and promote policy update. The experience playback buffer D is used to store the state transition storage tuple (s_t, a_t, r_t, s_{t+1}) during the interaction between the agent and the environment, which can be randomly sampled for error value calculation when updating the policy. The update process of the target network and online network in the GNN-DRL algorithm framework is as follows:

(1) Update the target network: First, input the state s_{t+1} at the next moment into the target Actor network GNN_3 from the experience playback buffer D obtain the action a' after iterative training, and combine the action a' and the state s_{t+1} as the input of the target Critic network. Then, the target value is obtained by iterative training of GNN_4; the calculation process of the target reward value is shown in Equation (6):

$$y_t = r_t + \gamma Q'(s_{t+1}, u'(s_{t+1}|\theta^{u'})|\theta^{Q'}) \tag{6}$$

Here, $\theta^{u'}$ is the target Actor network parameter, which is used to generate the output policy a'=$u'(s_{t+1}|\theta^{u'})$, and $\theta^{Q'}$ is the target Critic network parameter, which is used to evaluate the value of the current policy.

(2) Update the online network; the online Critic network can calculate the error through the error equation, and the calculation process is shown in Equation (7):

$$L = \frac{1}{N}\sum_t \left(y_t - Q(s_t, a_t|\theta^Q)\right)^2 \tag{7}$$

Here, y_t is the target return value obtained by the target network, while Q is the actual value obtained by the GNN_2 iteration through the combination of s_t and a_t input to the online Critic network. The network is updated by minimizing the error, and GNN_1 in the online Actor network is updated by the policy gradient. The overall GNN-DRL intelligent routing algorithm process is shown in Algorithm 1.

Algorithm 1: GNN-DRL Intelligent Routing Algorithm

Input: Network topology status information s
Output: Link weight w
  Initalize θ^u, θ^Q, $\theta^{u'}$, $\theta^{Q'}$ and D
  For episode = 1, M do
    Initalize action noise $\aleph$, state s_1 and soft update parameter τ
    For t = 1, Tdo
      Function graph (u, V, E)
        ${e'}_k = \phi^e(e_k, v_{rk}, v_{sk}, u)$
        ${v'}_i = \phi^v(\bar{e}_i, v_i, u)$
        $u' = \phi^u(\bar{e}', \bar{v}', u)$
        $s \leftarrow (u, V', E')$
      End function
      $w_t \leftarrow a_t = u(s_t|\theta^u) + \aleph$
      $y_t = r_t + \gamma Q'(s_{t+1}, u'(s_{t+1}|\theta^{u'})|\theta^{Q'})$
      $L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2$
      $\nabla_\theta uJ \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=u(s_i)} \nabla_{\theta^u} u(s|\theta^u)|s_i$
      $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
      $\theta^{u'} \leftarrow \tau\theta^u + (1-\tau)\theta^{u'}$
    End for
  End for

4.2. GNN-DRL Agent Interacts with the Environment

The GNN-DRL agent can complete routing policy training by interacting with the network environment. The GNN-DRL agent can adapt to the variable-dimensional network environment using the GNN, which enables the routing algorithm to be updated, making the load balancing of the network close to optimal and ensuring that the routing algorithm training converges. Thus, GNN-DRL remains highly adaptable and scalable in the face of new network topologies, network topology link interruption/failure, etc. The interaction process between the GNN-DRL agent and the environment is shown in Figure 4.

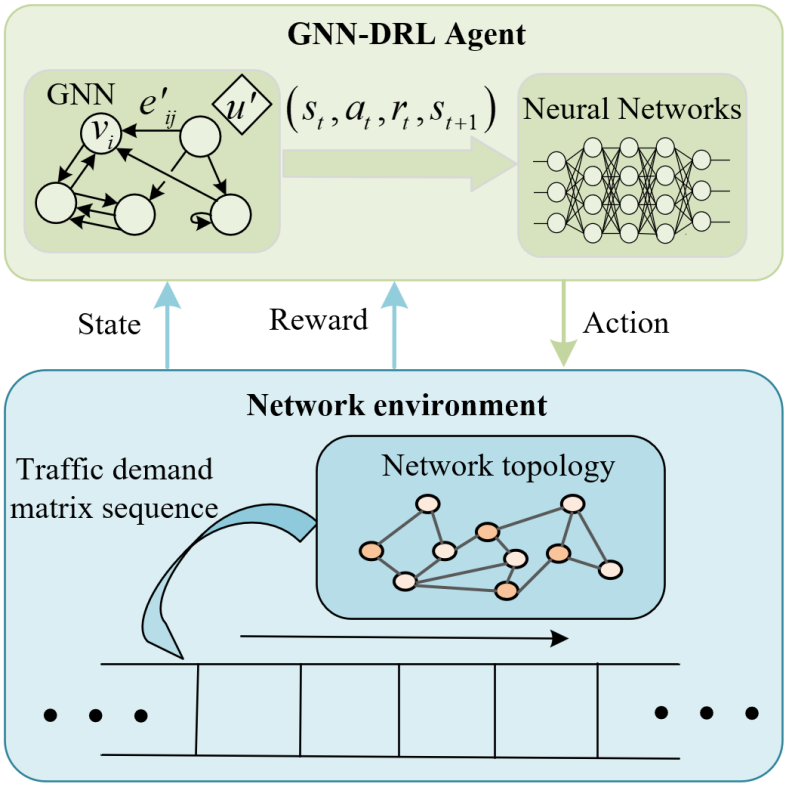

Figure 4. The interaction process between the GNN-DRL agent and the environment.

In Figure 4, the interaction process between the agent and the environment is as follows: (1) the agent takes the basic information, such as the delay, bandwidth, throughput, and link utilization of network as the state, then maps the state to the GNN structure in the agent after pre-processing; (2) after iterative training of the GNN-DRL agent, the link weight is calculated as an action, meaning that the action acts on the network environment; (3) the reward is calculated according to the effect of the action, then the reward is fed back to the agent. The agent and the network environment complete the interaction of (s_t, a_t, r_t, s_{t+1}) with changing time t. In this process, the neural network parameter in $Q(s, a|\theta)$ is updated until the loss function converges to a certain value, thereby maximizing the profit by accumulating the reward $\max \sum_{t=0}^{r} \gamma^t r_t$ (γ is the discount factor), finally obtaining a near-optimal routing strategy. The mapping process of state, action, and reward is as follows.

(1) State mapping

The state is used to reflect the characteristics of the network environment. This algorithm uses the traffic demand, delay, packet loss rate, and link utilization rate in the network topology as the state feature, and records the traffic demand between nodes at time t as $D_{m\times n_t} = \begin{pmatrix} d_{11_t} & \cdots & d_{1m_t} \\ \vdots & \ddots & \vdots \\ d_{m1_t} & \cdots & d_{mn_t} \end{pmatrix}$, where $D_{m\times n_t}$ represents the traffic demand between all network nodes at time t and d_{mn_t} represents the traffic demand of one of the nodes at time

t. We denote the delay of the link at time t as d_t, the packet loss rate of the link at time t as l_t, and the utilization rate of the link at time t as u_t. Next, we use the learning function to map the state of the network topology nodes and links with each node v_i and directed edge e_k of the GNN, where the node v_i=d_{mn_t} and the edge e_k=(d_t, l_t, u_t). After the mapping is completed, the GNN encode block is first used for message passing and core calculation, then the process block is used for iterative update, and finally the decode block outputs the weights of each link. The network topology with GNN mapping and updating process is shown in Figure 5.

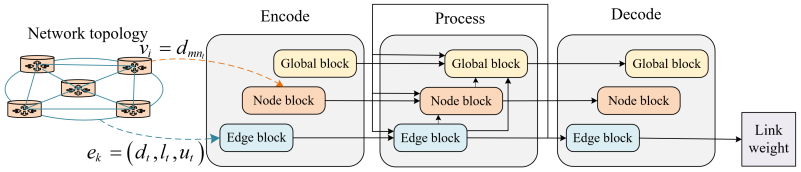

Figure 5. Mapping and updating process of network topology to GNN.

In Figure 5, each element in the update process is related to both the state of the surrounding elements and to the state of its own previous round. The update process is not static, and can be updated from the global block to the node block and the edge block. The update process is iterative.

(2) Action mapping

Actions are policies that the agent produces based on the state and reward. This algorithm is trained by inputting the state and reward at the current moment, with the set of link weights output after iterative convergence used as the action value. The action is defined as the set of global link weights trained by the GNN-DRL agent at time t, and the calculation process is shown in Equation (8):

$$a_t = < W_1, W_2, \cdots W_x \cdots W_M > \tag{8}$$

Here, $W_x = < w_{x1}, w_{x2}, ...w_{xy}...w_{xk} >$ represents the link weight set of node x and k neighbor nodes and w_{xy} represents the link weight corresponding to node x and node y. After the action provides the global link weight, the Dijkstra algorithm in OSPF is used to calculate the shortest path. Because the link weight of the traditional OSPF routing algorithm generally defaults to 1 or a fixed value related to delay and bandwidth, it has difficulty coping with complex network environments. Therefore, this method first calculates the action value using the network state and reward through the GNN-DRL algorithm, with the action value being the set of global link weights. Next, after the training of the GNN-DRL algorithm converges, Dijkstra can use the shortest path first algorithm to calculate a weighted shortest path from the source node to the destination node by the set of global link weights, then use it as the final traffic transmission path to realize intelligent forwarding and optimization of routing.

(3) Reward mapping

The reward is the feedback of DRL based on the current network state and the action of the agent. It is usually a manually defined scalar value; the reward function can be adjusted according to different optimization objectives. The adjustment direction of the reward is the direction of network performance optimization. In this paper, the maximum link utilization rate and delay are used as the reward optimization indicators. Becuase the optimal route does exist, and the optimal maximum link utilization rate $U_{max_{optimal}}$ in the network topology can be calculated by means of linear programming, this paper uses $U_{max_{optimal}}$, the maximum link utilization $U_{max_{agent}}$, and the average end-to-end delay $delay_{average}$ generated by the agent to define the reward; the specific definition is shown in Equation (9):

$$reward = \frac{U_{max_{optimal}}}{\alpha U_{max_{agent}} + \beta delay_{average}} \quad (9)$$

Here, α and β are the weight parameters and have a value range between 0 and 1. During the experiment, the parameter weight can be adjusted according to the importance of the performance index. After the reward calculation is completed, the reward result is returned to the agent to adjust the link weight at the next moment.

5. Experimental Evaluation

5.1. Experimental Environment and Parameter Configuration

In this paper, the performance indicators of GNN-DRL are tested using the network simulation software Mininet [26]. In order to reduce the training time and ensure the validity of the training results, the experiment uses the more moderate OS3E in the Topology Zoo [27] dataset as the network topology. The OS3E network topology is shown in Figure 6. The figure includes 38 routing nodes and 48 links, and the bandwidth of each link is uniformly set to 100 Mbps. In the GNN-DRL algorithm, DDPG and GNN are used to implement routing updates. In the experimental process, GNN-DRL is based on Tensorflow1.8.0 and Python3.5.0. In the network framework of GNN-DRL, it adopts the RYU controller to realize centralized network management and adopts Open vSwith to realize the networking of the data plane. The experimental hardware implementation platform uses the Linux operating system Ubuntu18.04, i5-10600KF-CPU, 16GB-DDR4 memory, and two GTX-1080 8G graphics cards.

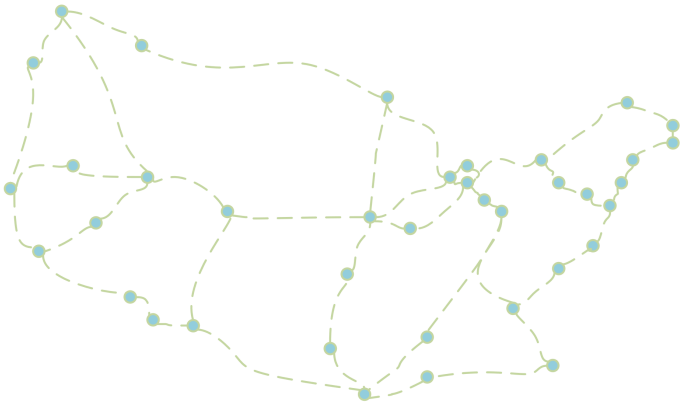

Figure 6. OS3E network topology.

During the training process based on the GNN-DRL routing strategy, the number of iterations of the DRL algorithm is set to 150,000, the learning rate lr is set to 5×10^{-4}, the initial value of ε in strategy ε-greedy is set to 0.01, the experience replay buffer D is set to 1500, and the discount factor λ is set to 0.8. The GNN forward propagation process takes 32 samples as a batch, and the number of iterations is set to 10 for message passing. The neural network uses the Adam optimizer and the Relu activation function, and the training is completed in this simulation environment over about 18 h.

5.2. Analysis of Experimental Results

The purpose of this proposal is to minimize the maximum link utilization and reduce the average end-to-end delay in the case of heavy network load in order to avoid link congestion while effectively generalizing the routing algorithm to different network topologies. In the experimental process, GNN-DRL is compared with traditional routing algorithms and better intelligent routing algorithms for network performance indicators. In order to ensure the experimental comparison of different routing algorithms under unified network environment and traffic transmission constraints, the routing rules defined in Section 3.1 are used in the experimental process. The main comparison routing algorithms include: (1) the

traditional routing algorithm Open Shortest Path First (OSPF), which can provide a good routing algorithm without considering specific network requirements; (2) the traditional routing algorithm Equal-Cost Multi-Path (ECMP), which can realize equal-cost multi-path traffic transmission according to network traffic; and (3) the intelligence-driven experiential network architecture for automatic routing (EARS), which is a high-performance routing algorithm based on SDN [28].

5.3. Routing Optimization under Different Traffic Loads

Traffic loads in real network scenarios often change dynamically. This experiment designs experimental environment with different traffic loads. Because the default bandwidth of each link is set to 100 Mbps in the experimental environment, this experiment adopts a relatively high traffic load environment of 80 Mbps and 120 Mbps for each link. During the experiment, the weight parameter α in the reward function is set to 1 and β is set to 0.5 when verifying the minimized maximum link utilization. The weight parameter α in the reward function is set to 0.5 and β is set to 1 when verifying the minimized average end-to-end delay. The experimental comparison results under different network loads are shown in Figures 7–10. The results show that GNN-DRL has better advantages in minimized maximum link utilization and average end-to-end delay with an increasing number of training steps for both 80 Mbps and 120 Mbps traffic loads, and has faster convergence and stability. Among the alternatives, OSPF is a traditional single-path routing algorithm, meaning that the routing optimization effect does not change with the increase in training time and the optimization effect under the high traffic load at 120 Mbps is obviously lower than that of the traffic load at 80 Mbps. ECMP is a traditional multi-path routing algorithm, meaning that while the effect does not change with the increase of training times, the multi-path traffic transmission mode of ECMP reduces the maximum link utilization and average end-to-end delay to a certain extent compared with OSPF. The effect with the EARS intelligent routing algorithm increases with the number of training steps, and the routing optimization continues to increase. After the training converges, the optimization effect is obvious compared to the traditional routing algorithms. Due to the limitations of the traditional neural network perception ability in EARS, however, its optimization efficiency needs to be further improved. GNN-DRL utilizes the dependency relationship between nodes and edges in graph neural networks to perform real-time state updates, fully explore the potential information between different samples, and maximize the perception of structural features and state changes of the network topology. These advantages ensure the strong adaptability of GNN-DRL under different traffic loads, especially in the case of large traffic loads, effectively improving the convergence speed and reducing both the maximum link utilization and average end-to-end delay.

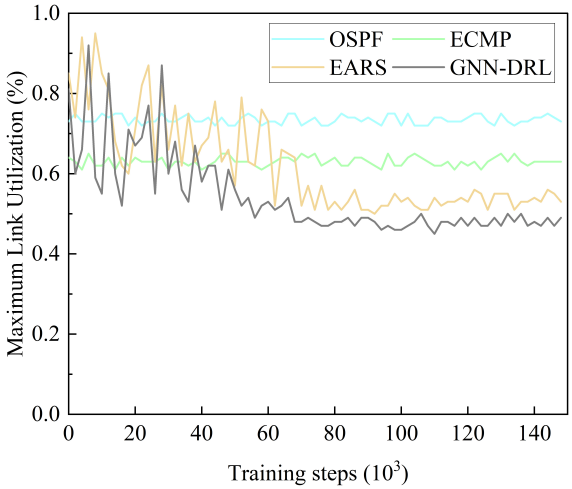

Figure 7. Optimization effect of maximum link utilization under 80 Mbps traffic load.

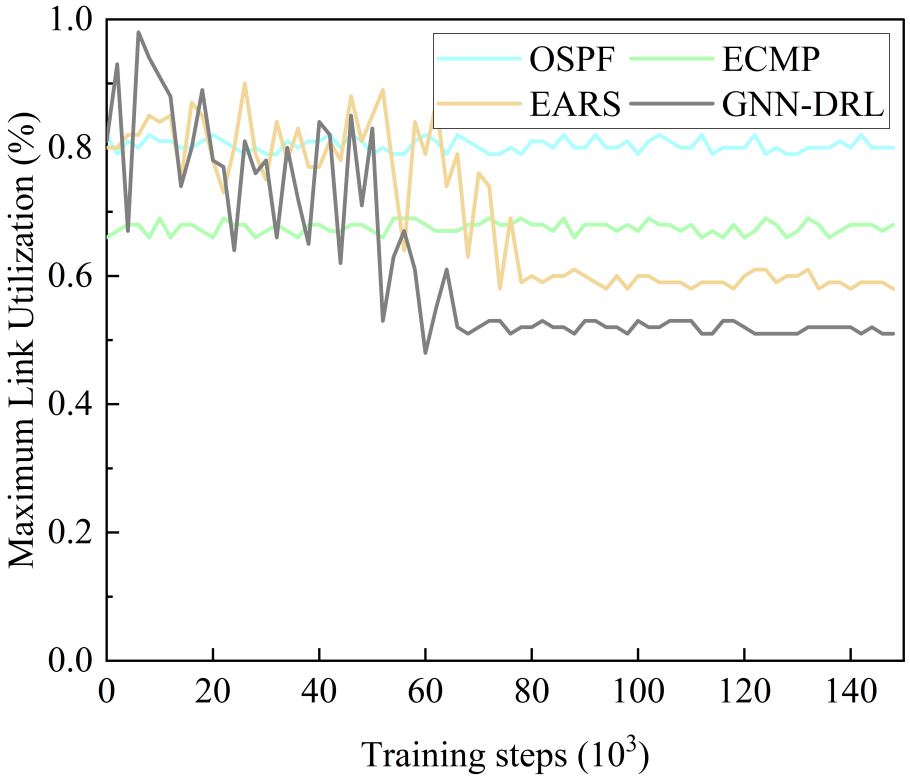

Figure 8. Optimization effect of maximum link utilization under 120 Mbps traffic load.

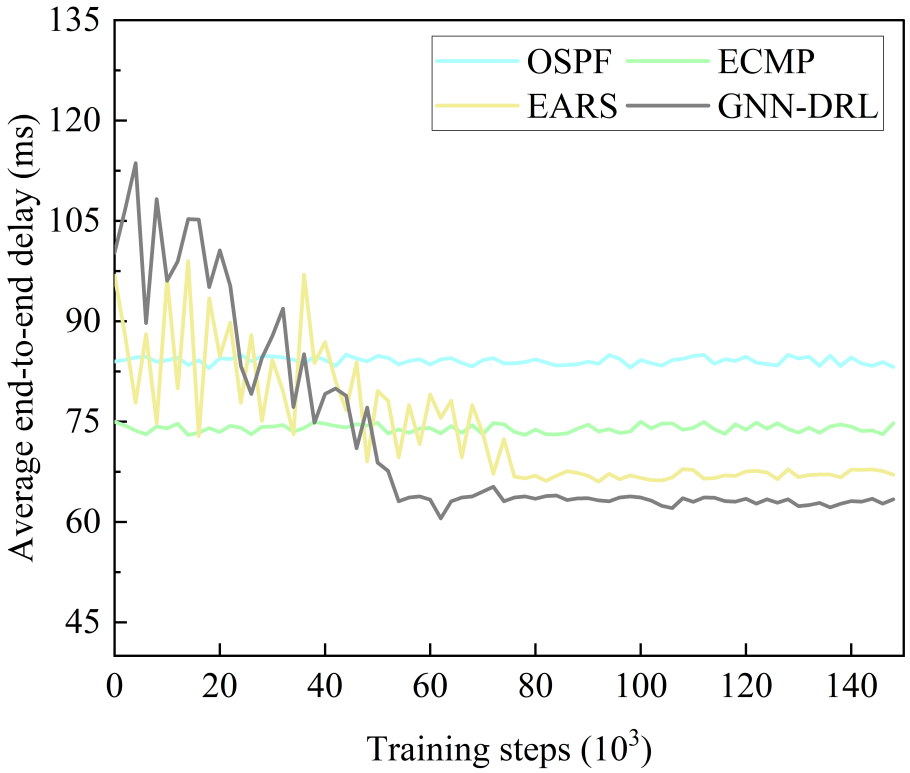

Figure 9. Optimization effect of average end-to-end delay under 80 Mbps traffic load.

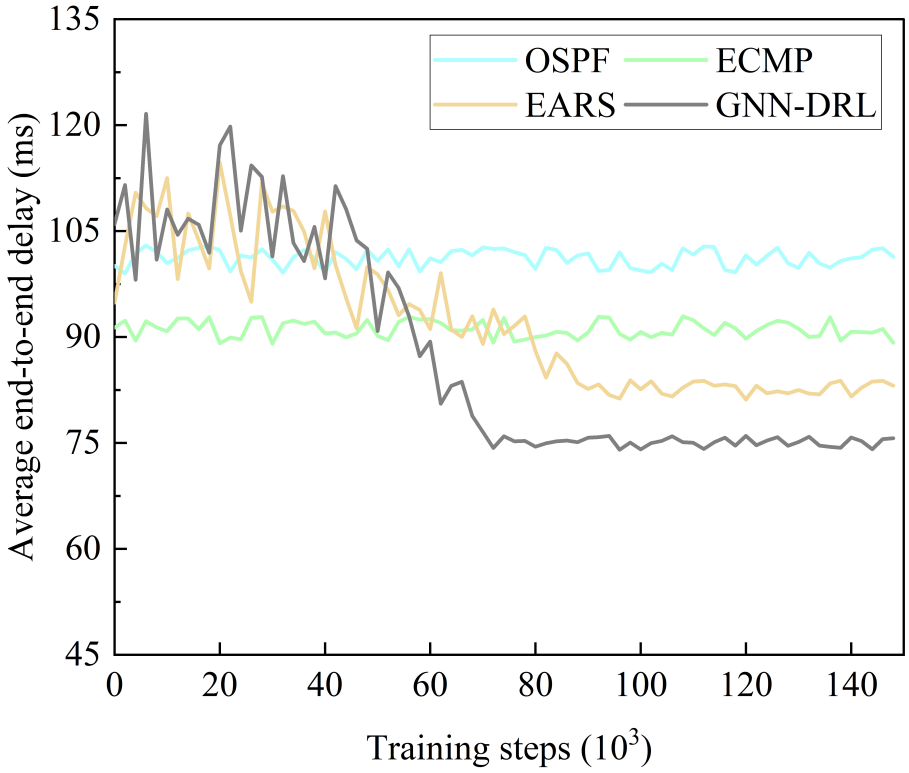

Figure 10. Optimization effect of average end-to-end delay under 120 Mbps traffic load.

5.4. Routing Optimization under Given Traffic Demand

The purpose of this experiment is to verify whether the routing algorithm can transmit traffic with low maximum link utilization and average end-to-end delay given the traffic demand matrix. The experiment provides four types of traffic demand matrices. One is the original traffic demand matrix, and the other three are randomly modified by 30%, 60%, and 90% from the original traffic demand matrix. During the experiment, the weight parameter α in the reward function is set to 1 and β is set to 0.5 when verifying the minimized maximum link utilization. The weight parameter α in the reward function is set to 0.5 and β is set to 1 when verifying the minimized average end-to-end delay. We set the traffic load to 80 Mbps and test each routing algorithm iteratively 50 times. The experimental comparison results are shown in Figures 11 and 12. The results show that GNN-DRL is better than the OSPF, ECMP, and EARS routing algorithms in terms of minimum maximum link utilization average and average end-to-end delay under the both same traffic demand matrix and different traffic demand matrices. The reason for this is that OSPF and ECMP are traditional fixed routing algorithms, which makes it difficult to dynamically adjust the routing scheme according to network traffic demand. The EARS training process uses traditional neural networks, which can only traverse all possible sequences in the network nodes as model inputs when facing a network topology with non-sequential representation features, which requires high computational power of the neural network model. However, GNN-DRL uses GNN as the neural network, and uses the non-European structure data processing capability of GNN to directly establish a mapping relationship with the network topology, thus ignoring the input order between nodes such that the output results do not change with the input order of the nodes. Therefore, GNN-DRL's ability to minimize the maximum link utilization and the average end-to-end delay advantage in a graph-structured network environment are guaranteed.

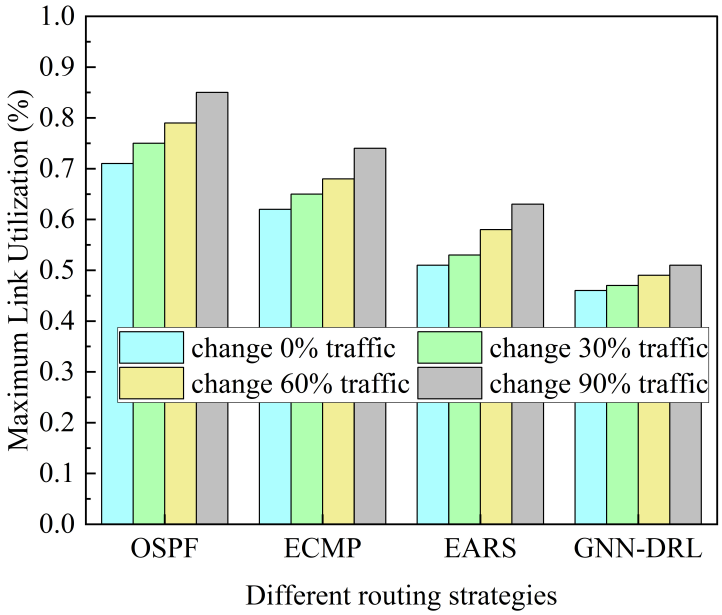

Figure 11. Optimized performance of maximum link utilization under different traffic loads.

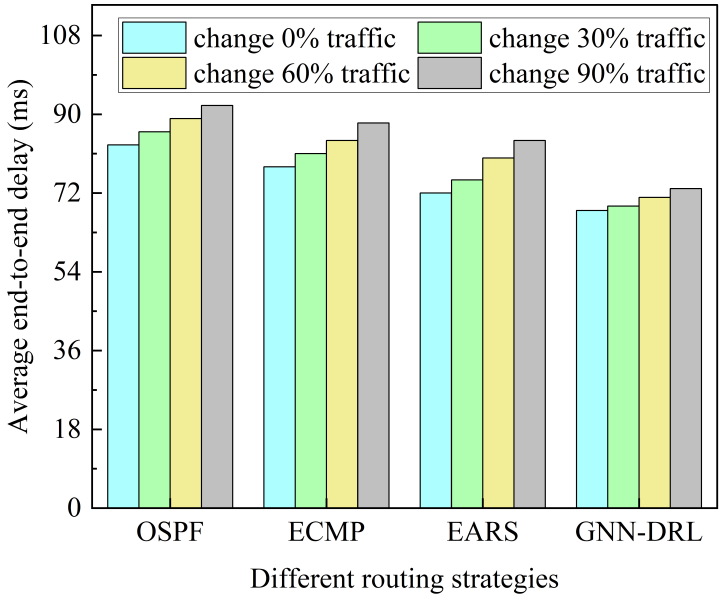

Figure 12. Optimized performance of average end-to-end delay under different traffic loads.

5.5. Generalization Capability of Intelligent Routing

The traditional neural networks need to maintain a fixed input and output format during the training process, which usually leads to overfitting; furthermore, it causes difficulties when generalizing to other network topologies after training is completed. In order to verify the generalization of the GNN-DRL intelligent routing algorithm, this experiment provides four types of network topologies. One is the original network topology, and the other three are randomly modified from the original network topology in four, eight, and twelve places. During the experiment, the weight parameter α in the reward function is set to 1 and β is set to 0.5 when verifying the minimized maximum link utilization. The weight parameter α in the reward function is set to 0.5 and β is set to 1 when verifying the minimized average end-to-end delay. We set the traffic load to 50 Mbps and test each routing algorithm iteratively 50 times. The experimental comparison results are shown in Figures 13 and 14. The results show that GNN-DRL both have smaller maximum link utilization and average end-to-end delay after random modification of the network

topology, with the generalization being the most obvious. The reason for this is that traditional routing algorithms OSPF and ECMP have difficulty choosing according to topology changes. EARS uses traditional neural networks, which usually have overfitting defects, resulting in poor optimization results in the case of network topology changes. However, GNN-DRL utilizes the strong generalization reasoning ability of GNN to ensure that its maximum link utilization and average end-to-end delay have better values even in cases of changing network topology. Therefore, GNN-DRL can dynamically adapt to complex and changeable network environments such as link interruption/connection, node addition/deletion, etc., and the training experience can be effectively extended to new network topologies in order to achieve routing optimization.

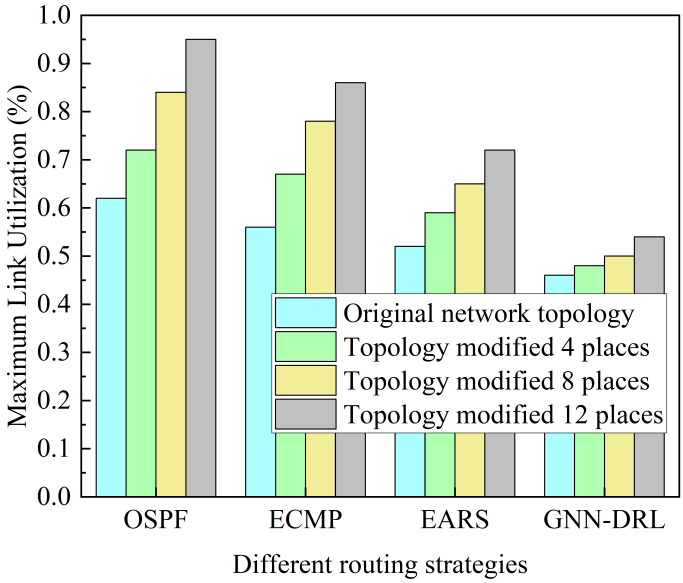

Figure 13. The effect of generalization on maximum link utilization.

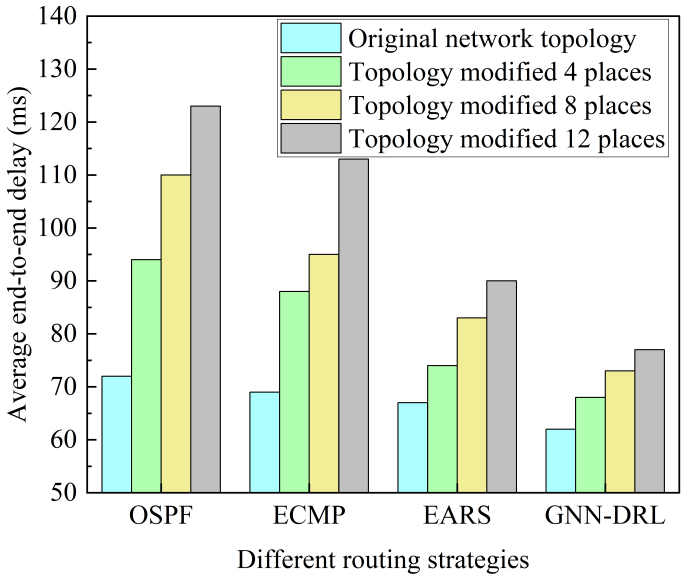

Figure 14. The effect of generalization on average end-to-end delay.

6. Conclusions

This paper proposes an intelligent routing algorithm, GNN-DRL, which uses GNN to dynamically sense changes in links/nodes in the network topology. It achieves the minimum maximum link utilization and minimum average end-to-end delay under high

traffic loads as well as in cases involving new network topologies or network topology changes thanks to its better generalization ability. In this paper, we compare the GNN-DRL intelligent routing algorithm with existing routing algorithms such as OSPF, ECMP, and EARS. The experimental results are as follows: (1) GNN-DRL has better network performance under both 80 Mbps and 120 Mbps traffic loads, and reduces the maximum link utilization by 13.92% and the average end-to-end delay by 9.48% compared to the superior intelligent routing algorithm EARS; (2) GNN-DRL has better advantages in terms of minimizing the maximum link utilization and average end-to-end delay for both the same traffic demand matrix and for random modifications of 30%, 60%, and 90% on the original traffic demand matrix; (3) GNN-DRL can effectively generalize the training experience to different network topologies when the network topology is randomly modified in four places, eight places, and twelve places, and has good generalization. In summary, we have experimentally verified the load balancing capability and generalizability of the GNN-DRL intelligent routing algorithm. Nonetheless, while the GNN-DRL routing algorithm mainly considers load balancing and generalization, it does not consider energy saving. Given the continuous complexity of the network topology, network energy consumption is becoming an ever more serious issue, and thus the energy saving effect of the routing algorithm has great research value. In our forthcoming research, we therefore intend to optimize network energy saving.

Author Contributions: Conceptualization, X.Z. and W.H.; methodology, X.Z., H.L. and G.L.; software, X.Z. and H.L.; validation, X.Z., W.H., H.L. and G.L.; formal analysis, X.Z. and W.H.; investigation, X.Z., H.L. and G.L.; data analysis, X.Z., W.H. and H.L.; writing—original draft preparation, X.Z. and W.H.; writing—review and editing, X.Z., W.H., H.L. and G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation of China (62002382, 62072416), in part by the Project of Science and Technology in Henan Province (222102210175, 222102210111), and in part by the Postgraduate Education Reform and Quality Improvement Project of Henan Province (YJS2022AL035).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jesús-Azabal, M.; García-Alonso, J.; Soares, V.N.; Galán-Jiménez, J. Improving Delivery Probability in Mobile Opportunistic Networks with Social-Based Routing. *Electronics* **2022**, *11*, 2084. [CrossRef]
2. Zheng, X.; Huang, W.; Wang, S.; Zhang, J.; Zhang, H. Research on Energy-Saving Routing Technology Based on Deep Reinforcement Learning. *Electronics* **2022**, *11*, 2035. [CrossRef]
3. Liu, Y.; Li, W.; Li, Y. Network traffic classification using k-means clustering. In Proceedings of the Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007), Iowa City, IA, USA , 13–15 August 2007; pp. 360–365.
4. Modi, T.M.; Swain, P. Intelligent routing using convolutional neural network in software-defined data center network. *J. Supercomput.* **2022**, *78*, 13373–13392. [CrossRef]
5. Hu, T.; Fei, Y. QELAR: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks. *IEEE Trans. Mob. Comput.* **2010**, *9*, 796–809.
6. Stampa, G.; Arias, M.; Sánchez-Charles, D.; Muntés-Mulero, V.; Cabellos, A. A deep-reinforcement learning approach for software-defined networking routing optimization. *arXiv* **2017**, arXiv:1709.07080.
7. Yu, C.; Lan, J.; Guo, Z.; Hu, Y. DROM: Optimizing the routing in software-defined networks with deep reinforcement learning. *IEEE Access* **2018**, *6*, 64533–64539. [CrossRef]
8. Du, Y.; Yin, X.; Lai, J.; Ullah, Z.; Wang, Z.; Hu, J. Routing Control and Fault Recovery Strategy of Electric Energy Router Under the Framework of Energy Internet. In Proceedings of the International Conference on Emerging Internetworking, Data & Web Technologies, Chiang Mai, Thailand, 25–27 February 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 256–268.
9. Woo, M.H.; Lee, S.H.; Cha, H.M. A study on the optimal route design considering time of mobile robot using recurrent neural network and reinforcement learning. *J. Mech. Sci. Technol.* **2018**, *32*, 4933–4939. [CrossRef]
10. Kumar, J.; Goomer, R.; Singh, A.K. Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters. *Procedia Comput. Sci.* **2018**, *125*, 676–682. [CrossRef]
11. Huang, R.; Huang, C.; Liu, Y.; Dai, G.; Kong, W. LSGCN: Long Short-Term Traffic Prediction with Graph Convolutional Networks. In Proceedings of the IJCAI, Yokohama, Japan, 11–17 July 2020; pp. 2355–2361.

12. Rusek, K.; Suarez-Varela, J.; Almasan, P.; Barlet-Ros, P.; Cabellos-Aparicio, A. RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 2260–2270. [CrossRef]
13. Yao, Z.; Wang, Y.; Meng, L.; Qiu, X.; Yu, P. DDPG-based energy-efficient flow scheduling algorithm in software-defined data centers. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6629852. [CrossRef]
14. Yan, R.; Liu, R. Principal Component Analysis Based Network Traffic Classification. *J. Comput.* **2014**, *9*, 1234–1240. [CrossRef]
15. Zhili, H.; Daru, P.; Hui, S. An Opportunistic Network Routing Algorithm Based on Clustering Algorithm. *J. South China Norm. Univ. (Natural Sci. Ed.)* **2019**, *51*, 120–128.
16. Mao, B.; Fadlullah, Z.M.; Tang, F.; Kato, N.; Akashi, O.; Inoue, T.; Mizutani, K. Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning. *IEEE Trans. Comput.* **2017**, *66*, 1946–1960. [CrossRef]
17. Zhuang, Z.; Wang, J.; Qi, Q.; Sun, H.; Liao, J. Graph-aware deep learning based intelligent routing strategy. In Proceedings of the 2018 IEEE 43rd Conference on Local Computer Networks (LCN), Chicago, IL, USA , 1–4 October 2018; pp. 441–444.
18. Kumar, S.; Miikkulainen, R. Dual reinforcement Q-routing: An on-line adaptive routing algorithm. In Proceedings of the Artificial Neural Networks in Engineering Conference, Citeseer, Stockholm, Sweden, 16–18 June 1997; pp. 231–238.
19. Xu, Z.; Tang, J.; Meng, J.; Zhang, W.; Wang, Y.; Liu, C.H.; Yang, D. Experience-driven networking: A deep reinforcement learning based approach. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; pp. 1871–1879.
20. Almasan, P.; Suárez-Varela, J.; Badia-Sampera, A.; Rusek, K.; Barlet-Ros, P.; Cabellos-Aparicio, A. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *arXiv* **2019**, arXiv:1910.07421.
21. Yoon, K.; Liao, R.; Xiong, Y.; Zhang, L.; Fetaya, E.; Urtasun, R.; Zemel, R.; Pitkow, X. Inference in probabilistic graphical models by graph neural networks. In Proceedings of the 2019 53rd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA , 3–6 November 2019; pp. 868–875.
22. Sanchez-Gonzalez, A.; Heess, N.; Springenberg, J.T.; Merel, J.; Riedmiller, M.; Hadsell, R.; Battaglia, P. Graph networks as learnable physics engines for inference and control. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 4470–4479.
23. Hoang, T.; Vien, N.A. Graph-Based Motion Planning Networks. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Ghent, Belgium, 14–18 September 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 557–573.
24. Li, J.; Sun, P.; Hu, Y. Traffic modeling and optimization in datacenters with graph neural network. *Comput. Netw.* **2020**, *181*, 107528. [CrossRef]
25. Qiu, C.; Hu, Y.; Chen, Y.; Zeng, B. Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications. *IEEE Internet Things J.* **2019**, *6*, 8577–8588. [CrossRef]
26. Sood, M. SDN and mininet: Some basic concepts. *Int. J. Adv. Netw. Appl.* **2015**, *7*, 2690.
27. Knight, S.; Nguyen, H.X.; Falkner, N.; Bowden, R.; Roughan, M. The internet topology zoo. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 1765–1775. [CrossRef]
28. Hu, Y.; Li, Z.; Lan, J.; Wu, J.; Yao, L. EARS: Intelligence-driven experiential network architecture for automatic routing in software-defined networking. *China Commun.* **2020**, *17*, 149–162. [CrossRef]