SRv6 and Zero-Trust Policy Enabled Graph Convolutional Neural Networks for Slicing Network Optimization

Xin Wang, *Member, IEEE*, Bo Yi, *Member, IEEE*, Qing Li, *Senior Member, IEEE*, Shahid Mumtaz, *Senior Member, IEEE*, and Jianhui Lv, *Member, IEEE*

***Abstract*—With the rapid advancement of technologies such as B5G/6G and edge computing, network scenarios are becoming increasingly complex and diverse, leading to the emergence of slicing networks. Virtualizing applications into distinct categories and establishing corresponding network slices ensures performance to a certain extent. However, the challenges posed by the complex slicing environment demand more fine-grained routing control and higher costs to locate requested content or services, areas where current state-of-the-art methods fall short. To address these challenges, this work introduces a system framework that integrates the principles of Segment Routing over IPv6 (SRv6). An SRv6 optimization layer is created between the control and infrastructure layers to manage slices effectively and enhance routing control. Additionally, we propose a novel policy routing method based on zero-trust and Graph Convolutional Network (GCN) technology. This method transforms actions into policies that can be flexibly deployed on SRv6 nodes, segment by segment. These actions encompass both routing and security measures, allowing for dynamic and flexible deployment of policies on each segment to achieve the desired goals. This integration of segment routing and zero-trust principles simplifies implementation and enhances security. Comprehensive experiments were conducted to evaluate the proposed method. The results demonstrate significant improvements over state-of-the-art methods, including a higher service acceptance rate, better resource utilization, and reduced average latency and packet loss rate.**

***Index Terms*—SRv6, segment policy, routing, network slicing, graph neural network.**

I. Introduction

CURRENTLY, the global telecommunications industry has achieved rapid development through the eras from 2G to 5G, and now it's becoming B5G/6G. According to statistic data from the China Ministry of Industry and Information Technology [1], the B5G/6G users are now increasing greatly and continuously. As by the end of 2022, the number of B5G/6G users reaches 561 million, accounting for 33.3% of mobile phone users, which is 2.75 times the global average level (12.1%). Meanwhile, the number of B5G/6G base stations in China reached 2.312 million, which is around the increase of 62.25% compared to the end of 2021. The B5G/6G networks mark the expansion of the mobile internet into the field of Internet of Things (IoT) and drive the development of industrial internet. However, as the pace of informatization and the mobility of communication network terminals accelerate, B5G/6G technology cannot fully meet the current complex and diverse data traffic demands [2]. The performance requirements of some application scenarios have exceeded the capabilities of B5G/6G. Although compared to 5G networks, B5G/6G has higher speeds and lower latency, supporting new business applications such as high-definition video, augmented reality (AR), virtual reality (VR), et al., it still falls short of meeting the speed requirements for more advanced display modes such as ultimate AR, VR, holographic communication, etc.

Now, with the widespread adoption of IoT devices and the continuous emergence of applications such as artificial intelligence and virtual reality, higher demands are placed on network performance indicators such as data transmission speed, network latency, reliability, and security. Therefore, the evolution of B5G/6G networks is an inevitable trend. B5G/6G networks will build upon the foundation of 5G networks to further improve data transmission rates, reduce network latency, enhance network reliability and security, so as to meet the demands of more advanced business requirements. In addition, B5G/6G networks can also achieve the sharing and flexible scheduling of network resources, improving the overall performance and efficiency of the network, thereby meeting the higher level communication technology requirements of the future digital society.

Hence, for the purpose of meeting the complex and variable application demands from users in the B5G/6G environment, and to better implement traffic scheduling and resource allocation in high-load real network environments, the slicing technologies are introduced into B5G/6G networks, so as to divide isolated environments for applications with different demands, for example, the Enhanced Mobile Broadband (eMBB) emphasizes on super-high bandwidth;

Received 25 June 2024; revised 11 February 2025; accepted 4 March 2025. Date of publication 15 April 2025; date of current version 30 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62472078, Grant 62202247, and Grant U22A2004; in part by Liaoning Provincial Science and Technology Joint Program under Grant 2023-MSBA-079; in part by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy under Grant GML-KF-24-31; in part by State Grid Corporation of China Science and Technology Project Funding under Grant 2024YF-95; and in part by ZTE Industry-University Cooperation Funds under Grant IA20240819011. *(Corresponding authors: Bo Yi; Jianhui Lv.)*

Xin Wang is with the College of Information Science and Engineering, Northeastern University, Shenyang 110016, China.

Bo Yi is with the College of Computer Science and Engineering, Northeastern University, Shenyang 110016, China (e-mail: yibo@cse.neu.edu.cn).

Qing Li is with the Pengcheng Laboratory, Shenzhen 518066, China (e-mail: liq@pcl.ac.cn).

Shahid Mumtaz is with the Department of Engineering, Nottingham Trent University, NG1 4FQ Nottingham, U.K.

Jianhui Lv is with the Multi Modal Data Fusion and Precision Medicine Laboratory, The First Affiliated Hospital of Jinzhou Medical University, Jinzhou 1210001, China (e-mail: lvjh@pcl.ac.cn).

Digital Object Identifier 10.1109/JSAC.2025.3560000

Massive Machine Type Communications (mMTC) emphasizes on super-high throughput; while the Ultra-reliable and Low Latency Communication (URLLC) emphasizes on super-low latency [3]. Nevertheless, to fulfill all these applications with greatly varied demands, the resource allocation optimization and service function deployment over network slices become necessary for improving the network performance and guarantee the network reliability. In particular, we should be aware that to better serve these demands, the resource allocation and function deployment converge to the routing, through which differentiated requirements from different slices can be fulfilled. Hence, differentiated routing for slicing network is vital important, which is lacking currently [4] as far as we know. On the contrary, current state-of-the-art routing approaches are usually rigid, which cannot satisfy the dynamic and customize demands of users in slicing networks accordingly.

Based on the above analysis, we conclude that the routing for B5G/6G enabled slicing networks currently faces several severe challenges: *1) Slicing network is generally multi-tiered, ultra-dense and heterogeneous, which makes it becoming more structurally complex and security fragile, especially with the integration of SDN/NFV/AI [5], [6]. Thus, more efforts and cost will be required to guarantee the performance of slicing network. 2) The demands in slicing network vary a lot, due to the application of many different and new scenarios (e.g., eMBB, mMTC, URLLC), which makes it extremely hard to fit all with one single kind of routing design. 3) Traditional routing strategies mainly rely on using the network states for decision making, which may not be efficient enough when facing the real-time and multi-dimensional demands in slicing networks.*

Therefore, to solve these challenges in B5G/6G enabled slicing networks, this work tries to integrate the ideas of zero-trust and SRv6, so as to finally provide a differentiated and secure routing solution for slicing network, because the corresponding policies can be designed and deployed along each segment. The main contributions are as follows:

- This work designs a system framework jointly using the ideas of SRv6 and zero-trust, which consists of the application layer, the control layer, the network virtualization layer, the policy optimization layer and the infrastructure layer from top to bottom. In particular, the infrastructure layer is responsible for providing physical resource in the form of regular and SRv6 nodes. Then, we can deploy the corresponding routing and security policies on these nodes, according to the decisions from the control layer. It should be aware that the requests access in from the application layer and drive the following layers to function accordingly.
- This work formulates the policy routing problem in slicing networks as mathematical models, under which the node mapping, link mapping and slice deplyment are all fulfilled so as to satisfy the routing process. Then, we also propose a physical node upgrade strategy to change regular nodes to SRv6 supported ones. Then, the SRv6 enabled nodes can be used to host various policies such as the routing and security ones.
- This work proposes a policy routing by leveraging the ideas of both segment routing and zero-trust. In particular, the routing process is carried out segment by segment, where in each segment, the secure policies can be deployed so as to fulfill the zero-trust strategy. In addition, the Graph Convolutional Neural (GCN) network is applied in the proposed routing to achieve the intelligence. To establising such GCN based model, the matching data labels are firstly injected into the GCN model during data preprocessing. Then, the controller loads the pre-trained GCN model to perceive real-time traffic information and calculate the optimal path based on states and constraints. Subsequently, we deploy the policies along this path for customized and secure traffic forwarding.

The rest of this work are concluded as follows. Section II introduces the related work. Section III explains the designed system framework for policy routing. Section IV explains the policy routing method proposed in this work in details. Section V shows and analysis the experimental results. Section VI makes a conclude of this work.

II. Related Work

The proposed policy routing method relies on using the SRv6, the network slicing and the machine learning technologies. Therefore, we next provide a detailed introduction and analysis of the current research status in from the three pespectives.

A. SRv6 Based Routing

The framework of SRv6 network has been defined in IETF standards documents [7], specifying major segment routing operations, which allows network operators or applications to specify the path for processing packets by encoding instruction sequences in IPv6. For example, Refs. [8] and [9] elaborate on the characteristics and applications of SRv6 architectures. Ref. [10] provides a comprehensive overview of segment routing, detailing the implementation of SR in the MPLS data forwarding plane and IPv6 data forwarding plane and showcasing the future development of SRv6.

Generally, traditional IP networks use static or dynamic routing protocols to route network traffic, which has various issues, including routing loops, routing oscillations and network scalability challenges. Therefore, traffic engineering has become one of the most crucial issues in networks. For example, Ref. [11] explores traffic engineering methods applied in segment routing, while Ref. [12] proposes an advanced traffic engineering method based on SR. Meanwhile, Ref. [13] introduces an application-aware traffic engineering solution based on SRv6 to achieve fine-grained control of network traffic for enhanced performance and reliability. Ref. [14] applies deep reinforcement learning to traffic engineering in partially deployed IPv6 networks by extracting network status information from SR headers and learning network feedback to establish an end-to-end path mapping model.

However, the aforementioned studies have not specifically addressed issues related to network architecture optimization,

thus more organizations are beginning to build networks using SRv6. According to Ref. [15], an SDN-based SRv6 solution can effectively reduce the operational costs of SRv6, while Ref. [16] adopts segment routing to implement a network architecture supporting service function chains. As more applications need to cope with the growing demand for network performance, Ref. [17] proposes that network slicing, as a technology born in new network environments, can be one of the measures to address these challenges. In addition, Ref. [18] proposes an application aware SRv6 structure to support the deployment of 5G/6G services, which is limited to simulation experiments and lacks validation in a real environment. Such limitation also happens in Ref. [19] which proposes a multi-routing path method based on SRv6 technology.

B. Network Slicing Based Routing

With the diverse new services continuously emerge, slice technology is increasingly adopted to isolate these services and ensure their performance. References [20], [21], and [22] review network slicing in 5G/B5G environments: [20] summarizes slice architecture, key technologies, and solutions; [21] explores slice architecture, implementation methods, design, mapping, and deployment; and [22] analyzes applications in 5G networks, offering guidance for related research. However, these studies overlook economic benefits and business models, which future research must address. Network slicing implementation relies on SDN and NFV technologies, posing challenges in current research and practice. Reference [23] discusses the research status and current challenges of 5G network slicing, while [24] proposes a QoS-enabled slicing framework integrating SDN and NFV to enhance network performance and user experience. However, further optimization is needed to meet broader application requirements.

As for the slice management and orchestration, it is a practical problem that needs to be addressed. In this context, Ref. [25] proposes a service-oriented slicing deployment strategy, aiming to maximize the utilization of network resources to meet various service requirements. The work discusses the service-oriented slicing deployment strategy at a theoretical level, but its actual effectiveness needs further experimentation and validation. In comparison, Ref. [26], focusing on a small-scale 5G test platform, systematically evaluates the application of network slicing deployment strategies and summarizes the characteristics and application of the small-scale 5G test platform. However, the limitation of this work lies in its relatively narrow research approach and the absence of experimental validation and case analysis. Additionally, based on model-driven development and management architecture, Ref. [27] proposes a new end-to-end slicing orchestration method to support slicing management and orchestration. Although the method exhibits excellent performance, it lacks the evaluation in real-world scenarios, thus the practical validation is further required.

C. Machine Learning Based Routing

Due to the increase in network data volume and the growth of user demands, traditional routing methods have become inadequate to meet the various requirements of new applications. Aiming at these issues, Ref. [28] proposes an intelligent packet transmission method based on deep learning technology, which uses deep learning algorithms for packet transmission determination, so as to intelligently adapt and adjust the network environment dynamically and flexibly. In addition, Ref. [29] discusses how to use deep learning technology to implement an intelligent network traffic control system, including the summarization of deep learning applications and the corresponding basic principles as well as the categories, which can be leveraged for efficient routing. Ref. [30] designs a federated learning based time-sensitive network controller to optimize the network failure recovery.

Reviewing the current research on machine learning-based routing algorithms, the network states are crucial for routing decisions. However, due to the dynamic nature of network topology changes, traditional deep learning models often struggle to handle this information well. Hence, Ref. [31] proposes a distributed routing protocol generation method based on graph deep learning, which applies graph deep learning technology to traditional routing. Simultaneously, Ref. [32] introduces a deep learning-based routing algorithm, primarily including the design of the Graph Convolutional Network (GCN) model and the generation of routing strategies. The GCN model uses deep learning technology to learn the connectivity between nodes and transfers the learning results to the routing strategy for generating routing policies. Ref. [33], building on the basis of using graph neural network methods to model and optimize SDN networks, proposes a deep reinforcement learning-based algorithm for optimizing SDN network routing and traffic control. Ref. [34] proposes a vehicle network concept, Federated Vehicular Network, which based on federated learning and contains potential for application in routing optimization.

Moreover, Ref. [35] proposes an adaptive reinforcement learning algorithm for routing selection of packets in dynamic networks. This algorithm uses the value function and policy function of traditional reinforcement learning algorithms to autonomously learn routing decisions, so as to adapt to changes rapidly. Ref. [36] introduces a novel Q-Learning algorithm to predict the performance of routing algorithms. Compared with traditional algorithms, it proves that this method has better performance in terms of throughput and latency. On the other hand, Ref. [37] establishes a reinforcement learning-based routing optimization model, which ultimately confirms that this method can adapt to dynamic network topology changes with significant practical implications. Ref. [38] combines the GCN model and deep reinforcement learning algorithm to finally propose an intelligent network slicing and routing optimization method. Although this method theoretically improves network resource utilization and efficiency, it still faces issues such as uncertainty and scalability in actual environments.

III. System Framework and Problem Model

A. System Framework

The designed system framework is shown in Figure 1, which is mainly composed of five layers that are the infrastructure

layer, the SRv6 optimization layer, the virtualization layer, the control layer and the application layer from bottom up. As explained, the user requests access in from the application layer, which then drives the control layer to execute the corresponding routing algorithm to find the most satisfied routing path to meet the users' demands. Meanwhile, the virtualization layer and the optimization layer are in the middle of the system framework to build the relationship between strategy and resource. Moreover, the infrastructure is responsible for providing the underlying physical resource to support the slices and applications. These layers are then explained in details from bottom up.

- The infrastructure layer is responsible for providing the fundamental resource to support software execution within the B5G/6G enabled slicing network environment. In particular, it is aware that the three representative B5G/6G scenarios are also represented and prepared in this layer, in which different kinds of slices can be generated accordingly.
- The SRv6 optimization and slice deployment layer is responsible for managing the corresponding slices over SRv6 nodes. By obtaining the basic information from the infrastructure layer, we can determine the optimal solution for SRv6 node distribution, that is, we can achieve the goal of minimizing the maximum link utilization by using the smallest number of SRv6 nodes. After that, the network nodes add color attributes for the following traffic forwarding according to the pre-defined policies.
- The network virtualization layer is responsible for maximizing the average resource utilization for slices. According to the network information obtained from the underlying layers, we first design a slice mapping strategy which takes the three representative slices into consideration. Then, we propose a slice deployment method which provides corresponding deployment services based on the type of incoming slice requests and is fulfilled according to the pre-defined color templates relating to each kind of slices.
- The control layer is responsible for receiving requests from application layer and executing the policy routing method for the underlying layers. Specifically, the slice requests and network states are collected in the first place for routing decision making. Then, based on the color template, the requests will be redirected to the corresponding deployed slices. After that, we employ the GCN based policy routing algorithm to generate candidate paths and active paths. These information will be recorded in policies and associated with the active path segment by segment, so as to guide the data plane forwarding.
- The application layer is responsible for receiving the application requests from users. For different kinds of request, we will tag it with the corresponding color templates defined, so as to make a difference between them. It is note that such color template will be used for the following policy deployment. Then, this layer will send the request and analysis logic to the control layer for further processing. On the other hand, this layer can also receive the resource allocation solution and result from control layer, so as to allow the users to access the corresponding service and resource dynamically.

B. Workflow

This framework focuses on optimizing the resource utilization in B5G/6G enabled slicing networks via SRv6 based routing, in which the three representative kinds of slices (i.e., eMBB, mMTC. URLLC) are mainly considered. In particular, the slice deployment method takes the network states and resource conditions as input and outputs the slicing results. Then, based on the established slices, the proposed policy routing algorithm is carried out based on the GCN model to find and determine the optimal paths intelligently. The specific process is outlined below:

Step 1: Request generation and pre-analysis. Users will generate their requests towards the slices in the application layer. Upon receiving such a request, a pre-analysis process is initiated to establish a color template that defines the mapping relationship between requests and slices.

Step 2: Request forwarding and optimal solution finding. The request is forwarded to the control plane to determine the optimal solution. To achieve this, the SRv6 optimization layer must continuously interact with the infrastructure layer to collect real-time network state information. Based on this information, the network topology can be optimized by upgrading certain nodes to SRv6-capable nodes.

Step 3: State information storage and node availability calculation. The collected state information is stored to train the Graph Convolutional Network (GCN) model. Additionally, this information is used to calculate the node availability indicator. Using this indicator, slice mapping and slice deployment are implemented to meet the demands of the requests.

Step 4: Network slice completion and traffic matching. After the network slices are completed, application traffic labeled with predefined color tags is matched with the deployed slices according to their corresponding slice types. During this process, network resources are naturally allocated to support the traffic.

Step 5: Network state collection and label generation. Network states such as link bandwidth, transmission delay, traffic volume, and packet loss rate are collected and stored. Labels are then generated for these data points to use them in training the GCN model for policy routing.

Step 6: Training process and policy deployment. During the training process, the node feature vectors and network adjacency matrices are updated. The GCN model takes the network status as input and outputs the optimal routing path along with its associated costs and states. The identified active paths are then associated with a segment list, where detailed policies are deployed.

C. System Models

1) Network Model: The slice network is abstracted as an undirected weighted graph denoted by $G(N, E)$, where N indicates the node set and E indicates the link set. For any node $n_i \in N$, we have $n_i = (C_i, R_i)$, where C_i means the storage

capacity and R_i means the CPU resource of the physical node i respectively. In addition, given any link $e_{ij} \in E$, it is expressed as $e_{ij} = (Cap_{ij}, Band_{ij}, Del_{ij}, Cost_{ij})$, where Cap_{ij} indicates the link capacity; $Band_{ij}$ indicates the link bandwidth; Del_{ij} indicates the link delay and $Cost_{ij}$ indicates the link cost.

2) SRv6 Deployment Model: After obtaining the underlying infrastructure information, we can calculate the shortest path set $ShortPath(n_i, n_j)$ from node n_i to n_j using the Equal-Cost Multipath Routing (ECMP) method. Since any link $e_{pq} \in E$ may appear many times in $ShortPath(n_i, n_j)$, so that we can calculate the traffic load of the traversed link e_{pq} as follows:

$$\theta^{n_i n_j}_{flow_\alpha e_{pq}} = \frac{|e^{n_i n_j}_{pq}|}{ShortPath(n_i, n_j)}, \tag{1}$$

where $e^{n_i n_j}_{pq}$ indicates that the shortest path between n_i and n_j includes the link e_{qp}.

In this work, we adopt the two-segment routing idea which needs to upgrade the regular nodes to SRv6 enabled nodes. Assuming that the intermediate node n_k has been upgraded as the SRv6 node, when the flow $flow_\alpha$ traverses this node, the corresponding traffic load over the link connecting n_k can be expressed as: $\theta^{n_i n_k}_{flow_\alpha e_{pq}} + \theta^{n_k n_j}_{flow_\alpha e_{pq}}$.

Let the size of $flow_\alpha$ be ψ_{flow_α}, the following binary variable is defined to judge whether the $flow_\alpha$ has traversed the SRv6 enabled node n_k:

$$\mu_{flow_\alpha n_k} = \begin{cases} 1, & flow_\alpha \text{traverses the node} n_k \\ 0, & otherwise \end{cases} \tag{2}$$

Now, we create a virtual SRv6 node $\dot{N}$, so that $N = N \cup \dot{N}$. If the $flow_\alpha$ does not traverse n_k, then, we assume it passes the virtual node and let the maximum link utilization be U, such that we have:

$$\sum_{flow_\alpha} \sum_{n_k \in N} \mu_{flow_\alpha n_k} \cdot (\theta^{n_i n_k}_{flow_\alpha e_{pq}} + \theta^{n_k n_j}_{flow_\alpha e_{pq}}) \cdot \psi_{flow_\alpha} \leq U \cdot Cap_{pq}, \tag{3}$$

where $\sum_{n_k \in N} \mu_{flow_\alpha n_k} = 1, \forall flow_\alpha$.

Let ε_{pq} represent the reciprocal of Cap_{pq}, then, the link utilization of e_{pq} can be expressed as:

$$util_{flow_\alpha n_k} = (\theta^{n_i n_k}_{flow_\alpha e_{pq}} + \theta^{n_k n_j}_{flow_\alpha e_{pq}}) \cdot \psi_{flow_\alpha} \cdot \varepsilon_{pq}. \tag{4}$$

Substituting the above equation into (3), we have

$$\sum_{flow_\alpha} \sum_{n_k \in N} \mu_{flow_\alpha n_k} \cdot util_{flow_\alpha n_k} \leq U. \tag{5}$$

To maximize the link utilization, the objective is formulated as follows:

$$\min_{\mu_{flow_\alpha n_k}} \left\| \sum_{flow_\alpha} \sum_{n_k \in N} \mu_{flow_\alpha n_k} \cdot util_{flow_\alpha n_k} \right\|. \tag{6}$$

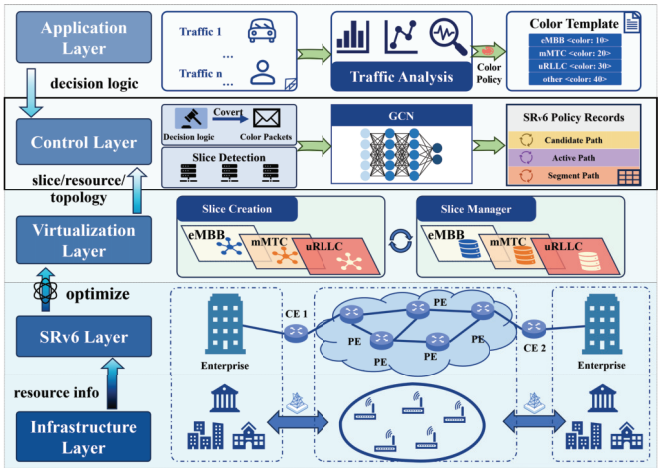


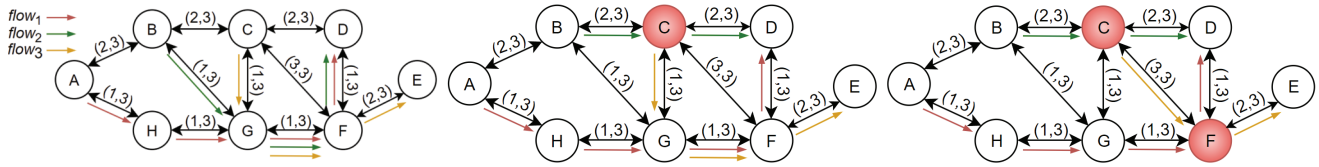
Fig. 1. System framework.

3) Virtualization Model: In the virtualization layer, the network slices are denoted by N^{NS}, which include the eMBB, mMTC, URLLC and the backup slice represented by R^e, R^m, R^u, R^o respectively. In this way, we have $N^{NS} = R^e \cup R^m \cup R^u \cup R^o$. The end to end network slice topology is abstracted as an undirected graph of $Q^R(N^R, E^R)$, where N^R is the set of virtual nodes and E^R is the set of virtual links. Given any node $n^R_i \in N^R$, it is expressed as $n^R_i = (C^R_i, R^R_i)$, where C^R_i is the node capacity requirement and R^R_i is the CPU requirement. For any virtual link $e^R_{ij} \in E^R$, it is indicated by $e^R_{ij} = (Band^R_{ij}, Del^R_{ij})$, where $Band^R_{ij}$ indicates the bandwidth demand and Del^R_{ij} indicates the delay demand from this slice request.

4) Routing Model: The routing is carried out after the SRv6 supported slice deployment. Hence, we need to firstly abstract the slice network as an undirected graph denoted by $G^D(N^D, E^D)$, where N^D and E^D are the virtual node and link for the slices. Specifically, $G^D = G^e \cup G^m \cup G^u \cup G^o$, where G^e, G^m, G^u, G^o are the slice representations of the eMBB, mMTC, URLLC and backup slices respectively. For any $n^D_i \in N^D$, it is $n^D_i = (C^D_i, R^D_i)$ that C^D_i, R^D_i are the node capacity and CPU resource. As for any $e^D_{ij} \in E^D$, it is $e^D_{ij} = (Band^D_{ij}, Del^D_{ij}, Cost^D_{ij})$, where $Band^D_{ij}, Del^D_{ij}, Cost^D_{ij}$ indicate the available bandwidth, the delay and the cost on e^D_{ij} respectively.

The routing is supported by the SRv6 which consists of the head-end, the end-point and the color identification. The control layer in the system architecture converts the decision logic sent by the application layer into color messages and corresponds to the network slices in different scenarios output by the network virtualization layer. Given the head and tail nodes in the network topology, it is necessary to find the forwarding nodes and links to get the optimal path. Hence, we first introduce the link attibute (denoted by N^D_V) to G^D with u nodes and v links, so that we have $N^{D'} = N^D \cup N^D_V$, where $|N^{D'}| = u + v$. Then, the adjacent matrix of this network is formulated by $A = (a_{ij})$, where

$$a_{ij} == \begin{cases} 1, & n_i \text{ directly connects to } n_j \\ 0, & otherwise \end{cases} \tag{7}$$



(a) Graph without SRv6 nodes (b) Graph with 1 SRv6 node (c) Graph with 2 SRv6 nodes

Fig. 2. Deployment node update illustration.

The corresponding degree matrix D can be calculated as follows:

$$D = \begin{bmatrix} d(n_1) & 0 & \cdot & 0 \\ 0 & d(n_2) & \cdot & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdot & d(n_{u+v}) \end{bmatrix}, \tag{8}$$

where $d(n_i)$ indicates the degree of $n_i, \forall i \in [1, u+v]$.

IV. Policy Routing Algorithm

To achieve more efficient, customized and secure policy routing for the B5G/6G enabled slicing network, we need first to perform the SRv6 optimization deployment environment. Then, we propose the policy routing algorithm to ahiceve the goal of minimizing the maximum resource utilization from the aspects of model transition, SRv6 node update, network slicing and policy routing respectively.

Algorithm 1 Optimization Deployment With Node Upgrade Algorithm (ODNUA)

```
Calculate maximum link utilization ∑_i δ_i/cap_e.
Transform the network into the graph model.
Set SubG = ∅
for all link pq in the network do
    if link pq satisfies Equ. (9) then
        Set μ_{flow_a n_k} = 1.
        Add link pq into SubG.
    end if
end for
return: The nodes in SubG need to be upgraded.
```

A. Model Transition

In order to fulfill the SRv6 enabled policy routing process, we should first transform the mathematical model of the problem into a graph model, after which we can then analyze this graph model to identify subgraphs that satisfy the corresponding constraints when pursuiting the goal. Hence, the first step, acting as the input to the second step, provides the theoretical foundation for selecting nodes to implement SRv6 deployment.

For example, as shown in Figure 2, there are eight nodes in the topology and each link is equipped with a binary attribute denoted by (cst_e, cap_e), where cst_e is the cost and cap_e is the maximum link capacity. Besides, there are three flows $\{flow_1, flow_2, flow_3\}$ and the size of each flow can be denoted by $\delta_i (i = \{1, 2, 3\})$. Thus, the maximum link utilization is expressed by $\sum_i \frac{\delta_i}{cap_e}$. Given the paths for the three flows, that is, $\{A, H, G, F, D\}$ for $flow_1$, $\{B, G, F, D\}$ for $flow_2$ and C, G, F, E for $flow_3$, then, assuming that there are no SRv6 enabled nodes in Figure 2(a), we can see that e_{GF} has been used by the three paths, so that the maximum link utilization is 100%. However, when we update node C as the SRv6 enabled node, as shown in Figure 2(b), the path of $flow_2$ is updated as B, C, D, so that the maximum link utilization is 66.67%. Moreover, when nodes C and F become the SRv6 enabled node, the corresponding maximum link utilization is reduced to 33.3%, as shown in Figure 2(c). Therefore, the number of SRv6 enabled nodes as well as their distributions can affect the maximum link utilization apparently.

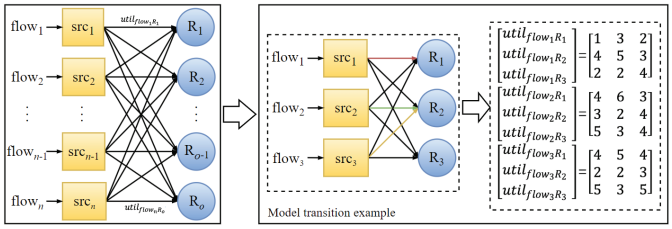


Fig. 3. Model transition illustration.

The illustration diagram of the first step in transforming the graph model is shown in Figure 3, where we can focus on the left side. There are n nodes representing the source nodes of n flows and o nodes representing forwarding nodes. Among them, the first $o-1$ nodes represent physical nodes that can be upgraded to SRv6 nodes, and the node o represents the virtual SRv6 node. Each link from the source node to the forwarding node has the attribute denoted by $util_{flow_a R_k}$, indicating the utilization of all links when $flow_a$ is forwarded via R_k. Besides, the attribute $util_{flow_a R_o}$ attached to the virtual router R_o represents the utilization of all links when the corresponding flow is forwarded along the shortest path.

The second step involves analyzing the transformed graph model. According to equation (6), the conditions that the subgraph must satisfy to be considered as a solution include the following: 1) The degree of each source node on the left side should be 1; 2) The number of upgrade nodes included in the subgraph must not exceed the specified quantity β; 3) The sum of all attributes $util_{flow_a R_k}$ in the subgraph is minimized in the sense of its infinite norm.

B. SRv6 Node Update

In order to determine which nodes to be upgraded as the SRv6 nodes, the greedy strategy is adopted. Firstly, we set

the target subgraph as $SubG = \emptyset$, that is, $\mu_{flow_{\alpha n_k}} = 0$. To satisfy the constraints of (3) and (5), we add a link in *SubG*, which can maximize $\mu_{flow_{\alpha n_k}}$ greatly. If

$$(\alpha, k) = \arg\min_{pq} \left\| \sum_{flow_\alpha} \sum_{n_k \in N} \mu_{flow_\alpha n_k} \cdot util_{flow_\alpha n_k} \right\| \quad (9)$$

is satisfied, we set $\mu_{flow_\alpha n_k} = 1$, that is, the link $e_{\alpha k}$ is added to *SubG*. However, we should be aware that such setting is conflict with the constraint of equation (1), such that the links connecting the header node of this link should be deleted from this sub-graph. As for the constraint in equation (2), it is satisfied by making sure that the number of nodes in *SubG* would not exceed a specific threshold. Once it is violated, the links between the SRv6 enabled nodes not in *SubG* should be deleted. Repeating the above process, we can find the sub-graph, in which the nodes are the ones determined to be upgraded.

For example, we give a detailed caseof small-scale sub-graph in the right side of Figure 3. In particular, there are three flows starting from the source nodes of src_1, src_2, src_3. R_1, R_2 are physical nodes that can be upgraded and R_3 is the virtual SRv6 node. If $Flow_\alpha$ is forwarded by R_3, the path is the shortest one. Now, assuming that threshold is set to 2 and $\{util_{flow_1 R_1}, util_{flow_2 R_2}, util_{flow_3 R_3}\}$ has the minimum infinite norm, to create the target sub-graph, the links e_{11}, e_{22}, e_{32} will be added into *SubG* subsequently. Specifically, this method selects the link of $e_{src_1 R_1}, e_{src_2 R_2}, e_{src_3 R_2}$ in strict order, such that the maximum link utilization is expressed as:

$$\begin{aligned} &\|util_{flow_1 R_1} + util_{flow_2 R_2} + util_{flow_3 R_3}\|_\infty \\ &= \max\{6, 9, 7\}, \end{aligned} \quad (10)$$

which satisfy the object to minimize the maximum link utilization.

C. Network Slicing

The network slicing needs to take both the resource mapping and slice deployment into consideration, after virtualizing the physical network and dividing physical resources into multiple virtual slices.

1) Resource Mapping: As explained, it is necessary to map the virtual nodes in slices to physical nodes to achieve resource isolation and service partitioning. To this end, this work designs a comprehensive node mapping scheme for network slicing. The primary goal of this scheme is to optimize the allocation of virtual nodes to physical nodes, ensuring efficient resource utilization and meeting the diverse requirements of different network slices. To facilitate effective node mapping, we introduce a Node Availability Level Indicator (NALI), which evaluates the suitability of physical nodes for hosting virtual nodes. The NALI encompasses two critical aspects of physical nodes: capacity and importance.

Firstly, the node capacity refers to the computational and storage resources available on a physical node. It is quantified by metrics such as CPU power, memory, and storage capacity. Higher capacity indicates a greater ability to handle resource-intensive tasks, ensuring that virtual nodes are mapped to high-performance physical nodes.

Secondly, the node importance reflects the strategic significance of a physical node within the network topology. It is assessed based on factors such as the node's centrality, connectivity, and role in critical paths. Nodes with high importance are those that play pivotal roles in maintaining network integrity and performance, making them ideal candidates for hosting critical virtual nodes. The node mapping scheme aims to better meet the requirements of network slicing resource mapping and deployment by optimizing resource utilization, enhancing service quality, and supporting dynamic adaptation. By considering both node capacity and importance, the scheme ensures that physical resources are used efficiently, reducing waste and improving overall system performance. Mapping virtual nodes to appropriate physical nodes based on their availability levels helps to maintain high service quality, ensuring that network slices meet their performance and reliability targets. Additionally, the NALI can be dynamically updated based on real-time network conditions, allowing the system to adapt to changing resource demands and network topologies.

Generally, larger node capacity indicates a greater ability to handle traffic, which is denoted by C_i. Then, the node importance indicator denoted by $Node\_Imp$, can be used to measure the significance of nodes in the network topology, which can be jointly defined by the node degree centrality (Deg_Cen) and the node betweenness centrality (Bet_Cen). Given any node with the degree denoted by Deg_{n_i}, it can be expressed as follows:

$$Deg_{n_i} = \sum_{n_i, n_j \in N} \delta_{n_i n_j}, \quad (11)$$

where $\delta_{n_i n_j} = \{0, 1\}$ indicates whether n_i and n_j are connected directly.

The larger the node degree, the higher the degree centrality, which means that this node is more important in the network. Hence, for a network with *n* nodes, the degree centrality is calculated by:

$$Deg\_Cen_{n_i} = \frac{Deg_{n_i}}{n-1}. \quad (12)$$

Besides, we also introduce the concept of the node betweenness centrality which indicates the number of shortest paths that traverses the same node and can also reflect the importance of this node in the network in a certain aspect. It is expressed as:

$$Bet\_Cen_{n_i} = \sum_{n_p \neq n_i \neq n_q} \frac{srt\_path_{n_p n_q}(n_i)}{srt\_path_{n_p n_q}}, \quad (13)$$

where $srt\_path_{n_p n_q}$ represent the shortest path set between n_p and n_q; $srt\_path_{n_p n_q}(n_i)$ is a variable used to determine whether n_i is in this shortest path set.

The larger the value of $\sum_{n_p \neq n_i \neq n_q} srt\_path_{n_p n_q}(n_i)$, the larger the node betweenness centrality of n_i, and naturally the more important of this node. Hence, the betweenness centrality of n_i can be normalized as follows:

$$Bet\_Cen_{n_i} = \frac{2Bet\_Cen_{n_i}}{(n-1)(n-2)}. \quad (14)$$

Nevertheless, we should be aware that the physical node capacity and connected link capacity should both be considered when evaluating the importance of the node. Hence, $Node\_Imp_{n_i}$ is formulated as:

$$Node\_Imp_{n_i} = C_i \times \sum_{l \in L_i} Cap_i \times \left(\frac{Deg\_Cen_{n_i} + Bet\_Cen_{n_i}}{2} \right), \quad (15)$$

where L_i is the set of links connecting to n_i.

Based on the above definition, the specific process can be described as follows. Firstly, we should rank physical nodes according to their impact level indicators. Then, the breadth-first searching method is used to traverse the virtual network topology and generate a searching tree for the request denoted by Res_Req for node mapping. However, during the mapping process, the node capacity, resource demand and importance indicators should all be considered, so as to offer the most optimal solution for improving resource utilization. To fulfill the slice mapping, physical links should also be mapped, during which factors such as bandwidth and latency need to be considered to ensure the communication efficiency and stability of virtual links. This work proposes a virtual link mapping scheme based on the K-shortest path method to build the mapping relationship between physical links and virtual links.

2) Slicing Deployment: After completing the virtual resource mapping, network slices need to be deployed to provide services. Given the frequent need for slice recycling and reallocation, dynamic deployment often results in lower stability and increased management costs. To address these challenges, we propose a slice deployment approach that integrates both static and dynamic methods during the slice deployment process.

Specifically, static deployment is performed during the initial network planning phase to establish a stable and efficient baseline configuration. This ensures that the core network infrastructure is robust and can handle typical traffic patterns. On the other hand, dynamic deployment is carried out during the slice working process to adapt to changes in the network environment. This includes releasing and reallocating resources that exceed the slice capacity threshold, ensuring that the network remains stable and reliable while maximizing resource utilization.

By combining static and dynamic deployment methods, our approach leverages the strengths of both traditional approaches. Static deployment provides a solid foundation, while dynamic deployment offers flexibility and adaptability to network changes. This comprehensive strategy not only enhances network stability and reliability but also significantly improves resource utilization and network efficiency. Compared to using a purely static or dynamic deployment approach, this integrated method is more flexible and better suited to handle dynamic network conditions, thereby enhancing overall network performance.

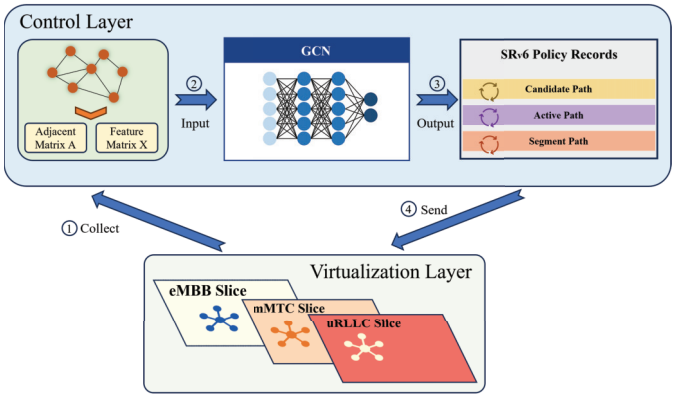


Fig. 4. GCN and slicing based Routing illustration.

D. GCN and Slice Based Policy Routing

As explained, the goal is to define routing by policies for the slicing network, in which the GCN model is leveraged. The overall GCN and slice based routing is illustrated in Figure 4, and we then explain this approach accordingly.

1) Training Label Construction: As far as we know, there is currently no publicly available dataset from real slicing network and the private datasets collected or generated within companies or organizations often lack transparency and review processes, making their credibility susceptible to quality and security challenges. Therefore, we first try to generate emulated traffic data for GCN training.

Firstly, The shortest path algorithm is executed based on the acquired network topology to perform initial routing. The shortest routing path from the source node to the destination node will be recorded as a training item used to finally form the dataset. Secondly, the calculated routing paths contain not only connection information between nodes but also information about resource usage on nodes. Hence, each node should monitor these information and associate them with the corresponding routing path, so as to enrich the training dataset features. Lastly, after collecting the key information towards network metrics, we should generate a vector containing these metrics (e.g., node capacity, node CPU resources, link bandwidth, delay, cost, etc.), so as to correspond to different application scenarios for network slices. Next, storing these vectors as data labels in the database and injecting them into the GCN model to finish the supervised training.

Based on the above process, the generated data label vectors for eMBB, mMTC, URLLC and backup slices working independently to satiasfy the user demands respectively, can be represented by the following equation:

$$\begin{cases} label_e = \frac{C_i^D + R_i^D + Band_{ij}^D + Cost_{ij}^D}{4} \\ label_m = \frac{C_i^D + R_i^D + Band_{ij}^D + Del_{ij}^D + Cost_{ij}^D}{5} \\ label_u = \frac{C_i^D + R_i^D + Del_{ij}^D + Cost_{ij}^D}{4} \\ label_o = \frac{label_e + label_m + label_u}{3} \end{cases} \quad (16)$$

where $C_i^D, R_i^D, Band_{ij}^D, Del_{ij}^D, Cost_{ij}^D$ are in the normalization of the corresponding resource and indicators. Such formulation transforms the problem into a classification task

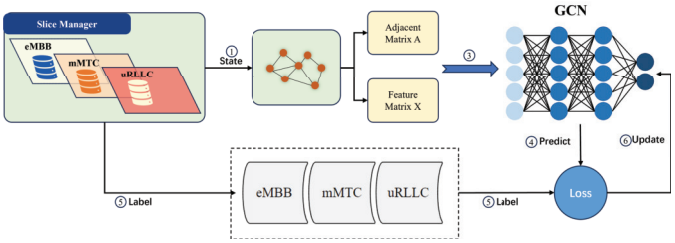


Fig. 5. GCN model training.

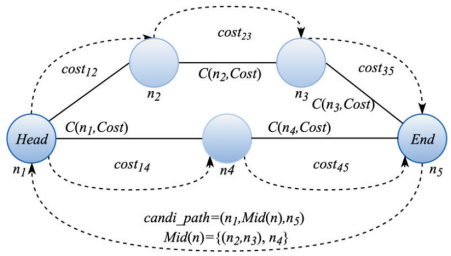


Fig. 6. Global minimum cost path iteration.

rather than regression, which lies in its ability to handle non-linear relationships and avoid sensitivity to outliers. Additionally, classification aligns with practical deployment scenarios where operators often require discrete decisions.

2) GCN Model Construction: The GCN model training can simultaneously utilize the structural information of the network topology and node features. Based on the generated data labels, the GCN model is employed to output the routing cost for the differentiated routing paths. However, several issues need to be addressed for GCN model training.

First of all, when using the GCN to predict routing cost, the network topology and resource information need to be represented in a graph which describes the slicing network with nodes and links. However, since the GCN model cannot handle complex multi-parameter link issues in the network topology, nodes are introduced to represent link attributes, so that the network topology is updated accordingly. Besides, the parameters and layers of GCN should be set. As explained, the first layer is an input layer where node and link information from the network topology are collected and fed into the model. Following the input layer is the hidden layer, where GCN transforms the feature matrix of node characteristics. Finally, there is the output layer, where GCN transforms node features into the predicted results of link costs in the routing problem. To guarantee the performance of the model, we should note that deeper models do not necessarily lead to better performance. Hence, the $(l+1)$-th layer node features and graph convolution operator for the GCN model can be derived from the following quations:

$$\begin{aligned} X^{l+1} &= \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X^{(l)}W^{(l)}\right) \\ x_i^{l+1} &= \sigma\left(\sum_{j\in V_j}\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}x_j^{(l)}W^{(l)}\right) \end{aligned} \tag{17}$$

where x_j^l represents the node feature of n_j at the l-th layer and V_j is the neighbor node set of n_j.

From equation (18), we can observe that it is a two-layer GCN model. In particular, the first GCN layer is primarily used to extract features of the first-order neighbor nodes of the target node, while the second GCN layer aggregates features to perceive the second-order neighbor nodes of the target node, thereby obtaining the corresponding routing cost for the target node. Therefore, for the first layer of the GCN network, ReLU is used as the activation function, expressed by the function as follows:

$$f(x) = \begin{cases} x, & \text{if} x > 0 \\ 0, & \text{if} x \le 0 \end{cases} \tag{18}$$

For the second layer of the GCN network, we employ Softmax, which can reduce the model fluctuation and improve model robustness, as the activation function, as follows:

$$softmax_{n_i} = \frac{e^{n_i}}{\sum_{n_i}^{N} e^{n_i}}. \tag{19}$$

When the activation is finished, the prediction result for the node can be expressed as follows:

$$\begin{aligned} pre\_node &= softmax\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}ReLU(*)W^{(l)}\right) \\ s.t. ReLU(*) &= ReLU\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X^{(0)}W^{(0)}\right). \end{aligned} \tag{20}$$

Lastly, we need to design the loss function for GCN. In particular, we adopt the cross entropy function as the loss function for GCN, since it can well evaluate the difference between the actual result and the predicted result, as follows:

$$loss = -\sum_{c=1}^{C} y_c \cdot \log \hat{y}_c, \tag{21}$$

where y_c means the real value and $\hat{y}_c$ means the output value of GCN.

The designed GCN model mentioned above requires continuous training iterations to obtain the optimal solution for cost prediction, where the training process is shown in Figure 5. During the iterative training process of the model, it is necessary to use the loss function to calculate the loss value between the predicted routing cost and the actual routing cost. The model weight parameters are then updated until the model converges, ultimately yielding a reliable routing cost output from the GCN model.

3) Routing: After completing the above design process, the GCN model is supervisedly trained offline based on data labels and other parameters to generate prior data, avoiding the issue of inaccurate predictions due to insufficient model training. Thus, the control layer inputs real-time network information collected to the pre-trained GCN model, which then generates node prediction results to obtain a candidate path set. Subsequently, based on link costs and the adjacency matrix A, the active path is iteratively derived, illustrating the iterative process of obtaining the globally minimum cost path, as shown in Figure 6.

TABLE I
PARAMETERS

Network parameters	
Number of nodes and links	25, 29
Average node degree	2.32
Node capacity	[100,1000] GB
Node CPU resource	[2,16] GHz
Link bandwidth	[1,40] Gbps
Link delay	[5,20] ms
Slice parameters (eMBB | mMTC | URLL)	
Minimum delay	<10ms | <1ms | <0.5ms
Maximum delay	200ms | 20ms | 10ms
Minimum dbandwidth	100Mbps | 10Mbps | 10Mbps
Maximum bandwidth	1Gbps | 100Mbps | 100Mbps
Preference level	[2,4] | [2,4] | [1,2]
Assigned weight	0.76 | 0.12 | 0.04
Model parameters (ReLU + Softmax)	
Batch size	256
Number of epoches	30
Drop out rate	0.25
Number of filters	32
Kernel size	(5,5)
Pool size	(2,2)
Number of patience	3
The weight decay	0.001
The early stopping count	10
The learning rate	0.001

As shown in this figure, we can group the candidate paths with the same source and destination in the set denoted by $candi\_path = (Head, Mid(n), End)$, where $Mid(n)$ is the set of the intermediate nodes. Now, defining $C(n_i, Cost) = \{(n_i, Cost_{ij}, \ldots, (n_i, Cost_{ik}))\}$ to represent the costs between n_i to its neighbors. Specifically, from the *Head* node, we can calculate iteratively the costs to the *End* according to the definitions of N_V^D and $C(n_i, Cost)$. After that, the forwarding nodes with the minimum cost can be obtained to formulate the active path denoted by act_path. Then, we associate the act_path to the *Color* identification using the SRv6 policy, so that it can be updated dynamically according to the network information collected.

V. PERFORMANCE EVALUATION

A. Setup

In order to evaluate the proposed methods, we carry out the experiments based on PyCharm and iperf as the tools. Specifically, the CERNET topology with 25 nodes and 29 links is selected as the testing scenario, where the node degree is 2.32 on average; the node capacity is set to [100,1000] GB; the CPU resource is set to [2], [16]GHz; the link bandwidth is set to [1], [40]Gbps and the link delay is set to [5,50]ms following the uniform distribution. Besides, for the network slices, we also set the corresponding parameters for the eMBB, mMTC, URLLC and backup slice respectively. For simplicity, the parameters are summarized in Table I.

In addition, the ReLU and softmax are used as the activation functions for GCN. In particular, the corresponding batch size is set to 256 and the number of epoches is set to 30. For the purpose of avoiding over-fitting, the drop out rate is set to 0.25 and the number of filters is set to 32. Moreover, the kernel size of this model is set to (5,5) and the pool size is set to (2,2) with the number of patience set to 3. As for the determination of this model, the corresponding weigh decay, early stopping and learning rate are set to 0.001, 10 and 0.001 respectively. These model parameters are also summarized and shown in Table I.

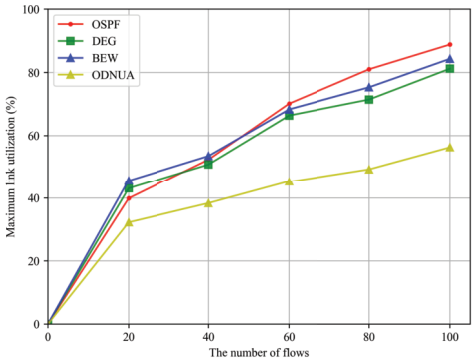


Fig. 7. Maximum link utilization.

B. Results

1) SRv6 Deployment Optimization Result: The SRv6 deployment optimization method proposed in this work is referred to as the Optimization Deployment with Node Upgrade Algorithm (ODNUA). To validate the performance of ODNUA, three comparison benchmarks are used, which are the open shortest path first algorithm (OSPF), the degree based method (DEG) and the betweenness based method (BEW) [14]. Apparently, OSPF tries to find the routing path using directly the shortest path first pricinple. As for DEG, it arranges nodes in descending order based on their degrees, prioritizing nodes with higher degrees as upgrade nodes, while BTW arranges nodes in descending order based on their betweenness centrality. It is important to note that since the DEG and BTW methods only play a role in selecting nodes and cannot determine the routing path, the OSPF algorithm is used after node selection to find the default shortest path for routing.

The corresponding results are shown in Figure 7. where the maximum link utilization is chosen as the evaluation metric. Generally, the smaller the the maximum link utilization, the stronger the load balancing. According to the experimental results, the maximum link utilization shows a gradual increase with the growth of traffic in the network. This is because as the number of flows increases, the traffic on the links also increases, leading to more consumption of link bandwidth resources and consequently increasing the maximum link utilization. In specific, the ODNUA method proposed in this work obtains a maximum link utilization approximately 20% lower than that of the DEG and BTW methods. This suggests that the ODNUA method has significant superiority in SRv6 network optimization deployment and can effectively achieve the goal of minimizing the maximum link utilization.

2) Network Slice Optimization Result: To validate the proposed slicing architecture, we compare it with the Multi-Stage [39] algorithm and the Greedy algorithm [40]. In particular, the pre-defined slice requests are randomly generated with the

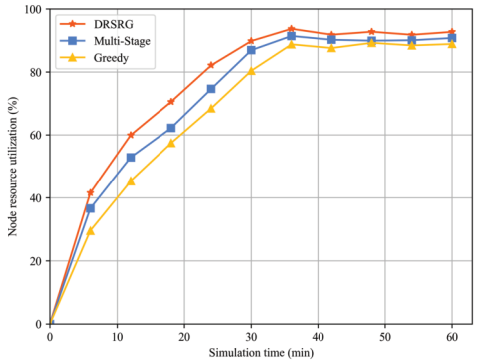


Fig. 8. Node resource utilization.

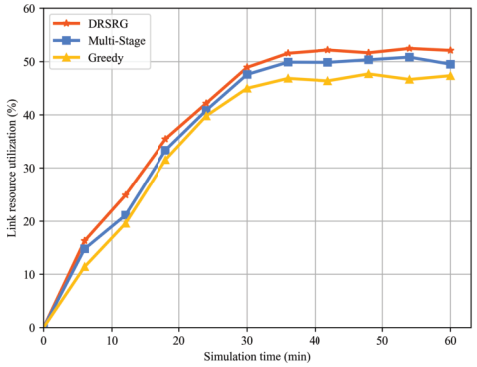


Fig. 9. Link resource utilization.

number of 500 and we run the experiments 100 times to obtain the average results. The corresponding results are summarized in Figures 8–10, where Figure 8 shows the node resource utilization, Figure 9 shows the link resource utilization and Figure 10 shows the slice acceptance rate respectively.

Apparently, we can observe that both the node and link resource utilization show a noticeable initial upward trend, gradually stabilizing over time. This is because, with the increase in simulation time, the number of incoming user requests gradually rises, leading to an increase in the mapped nodes and links. The stable value of resource utilization indicate that the resources on the slices are adequately allocated, and the resource utilization of the algorithm proposed in this work is superior to the two comparative algorithms, which directly validates the effectiveness of the proposed algorithm. Besides, it is also worth noting that node resource utilization is significantly higher than link resource utilization, primarily for two reasons: 1) communication between nodes may use multiple links and different nodes may use different links; 2) during the slice deployment, the network resources that the slice can bear are first verified against a set threshold. If exceeded, the network resources of that slice are released to avoid network congestion.

The resource utilization of nodes and links can jointly contribute to the utilization of slices, as shown in Figure 10. As explained, the proposed network slicing architecture consists of resource mapping and slicing deployment. Hence, this work evaluates from two aspects, that is, the resource utilization and

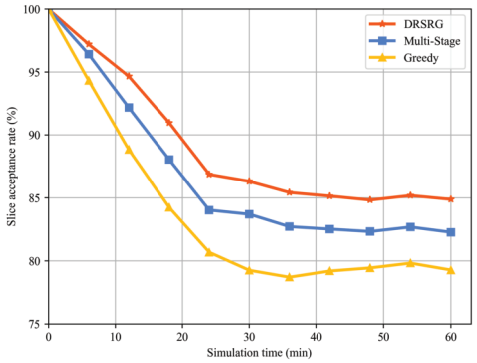
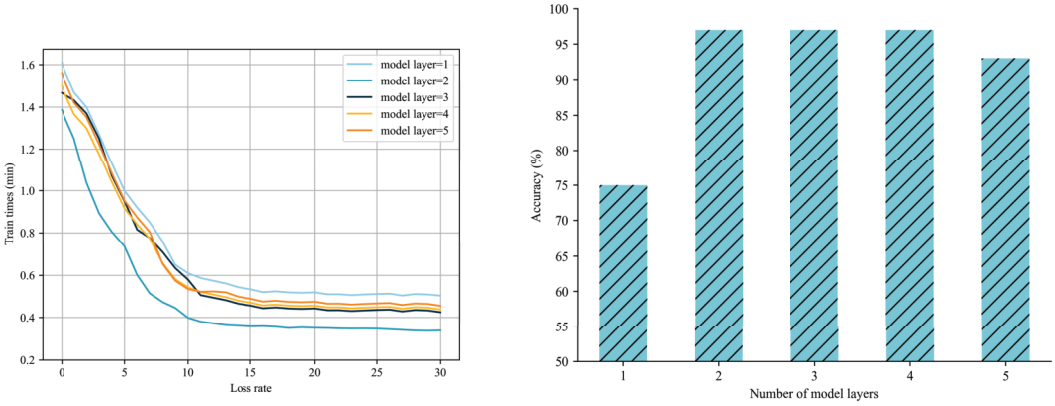


Fig. 10. Slice request acceptance rate.

the slicing request acceptance rate, aiming to fully demonstrate the efficient flexibility and optimization capabilities of the proposed network slicing architecture. Nevertheless, the resource utilization has been explained in Figures 8 and 9, thus we mainly focus on discussing the slice acceptance rate which refers to the proportion of valid requests accepted by the slice to the total requests. Generally, the higher the rate, the faster the response speed.

Now, observing the results in Figure 10, we can see that the slice request acceptance rates for all three algorithms initially show a decreasing trend followed by a tendency to stabilize. Initially, due to a relatively low number of incoming requests, both node and link resources are abundant, resulting in slice request acceptance rates close to 100%. As time advances, the increasing number of incoming requests intensifies resource competition, leading to a noticeable decline in slice request acceptance rates. In the latter stage, slices have completed the initial resource allocation process and released some occupied resources. At this point, performance requirements become the primary influencing factor, causing slice request acceptance rates to gradually stabilize. The experimental results indicate that the proposed algorithm's slice request acceptance rate is higher than the two comparative algorithms. After almost half of the simulation time, the slice request acceptance rate for the proposed algorithm is approximately 84%, while the slice request acceptance rates for the two comparative algorithms are around 82% and 79%. Therefore, the network slicing architecture proposed can more effectively respond to service requests.

3) GCN Model Optimization Result: To measure the accuracy and generalization ability of the designed GCN model, evaluations are conducted from the perspectives of loss rate and accuracy. Accuracy and loss rate are commonly used evaluation metrics in machine learning classification problems. Accuracy reflects the model's prediction accuracy, while the loss rate indicates the stability and convergence speed during model training. Jointly taking the two metrics into consideration can provide a more comprehensive evaluation of the model's performance and help in selecting the most suitable model. Additionally, evaluating the accuracy and loss rate of the GCN model contributes to optimizing and improving the model, enhancing its performance and generalization ability.



(a) Loss change curve

(b) Model accuracy

Fig. 11. Model optimization results.

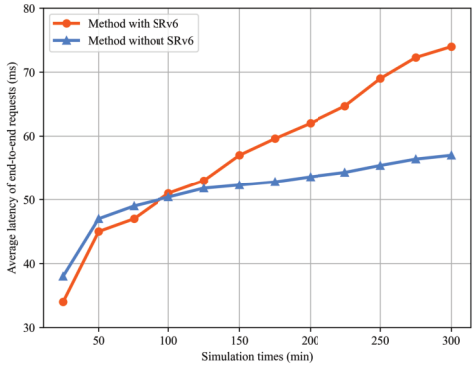


Fig. 12. Average latency of end-to-end requests.

Hence, the corresponding results are evaluated and shown in Figure 11.

Using the number of training iterations as the independent variable, the loss rates of different models are compared to explore the differences in loss rate performance as the number of training iterations increases for GCN models, as illustrated in Figure 11(a). During the experiment, the number of training iterations is set to 30 epochs which is enough to ensure the model convergence while meeting efficiency and data quality requirements, because most of the models can converge after 30 epochs of training. In this regard, the five models with different layers are used and trained for 30 epochs, and the corresponding results are shown in Figure 12, where it evaluates the changes in loss rates as the number of training iterations increases for different models. It can be observed that as the number of training iterations increases, the model's loss rate quickly decreases and converges. Through comparison, it is found that the model with a layer depth of 2 has a significantly lower loss rate than the other models. This result further validates the necessity of choosing a layer depth of 2 when designing the GCN model and indicates that adjusting the number of training iterations to some extent can optimize the model's performance.

The corresponding results of accuracy are presented in Figure 11(b). It is easy to note that the results have shown the relationship between the accuracy and the model layers. Particularly, as the number of layers in the GCN model increases, the accuracy shows a trend of initially increasing and then decreasing. When the number of layers exceeds two, the accuracy remains relatively stable. However, when the number of layers exceeds four, the accuracy starts to decline. This is because a higher number of layers lead to increased algorithm complexity, which then results in a decrease in accuracy. Such phenomenon means that large number of model layers cannot lead to good results and only the proper ones be selected can achieve better generalization ability and performance stability.

4) Policy Routing Optimization Result: To evaluate the performance of the proposed policy routing comprehensively, two kinds of comparison has been done. The first one is that we can compare the proposed method with the one not using SRv6 to show how SRv6 affects the performance. Then, the second one is that we compare the proposed policy routing (short for Differentiated Routing algorithm based on SRv6 and GCN, DRSRG) with Equal-Cost Multipath (ECMP), DRL-TE [41] and SmartRoute [42], so as to evaluate their performance towards different cases. In particular, the ECMP develops from the traditional routing, while DRL-TE and SmartRoute are based on the deep reinforcement learning to sense and predict the network traffic. All the experiments are executed for 100 times and the average results are calculated and presented.

Firstly, for the results of the average end-to-end latency, they are shown in Figure 12, where the proposed method with SRv6 and the compared one without SRv6 are compared and discussed. Specifically, with the increase in simulation time, the average latency of end-to-end requests for both methods shows an increasing trend. In the early stages (e.g., beginning half of the simulation time) of the experiment, the average delay achieved by the routing algorithm with SRv6 (i.e., the proposed one) is slightly higher than that achieved by the method without using SRv6. This is because the former needs to execute the node upgrade algorithm to complete the node selection and upgrade process. As time progresses, it can be observed that the average delay calculated by the proposed method significantly decreases after upgrading some nodes to SRv6 nodes. Such reduction is attributed to SRv6 traffic

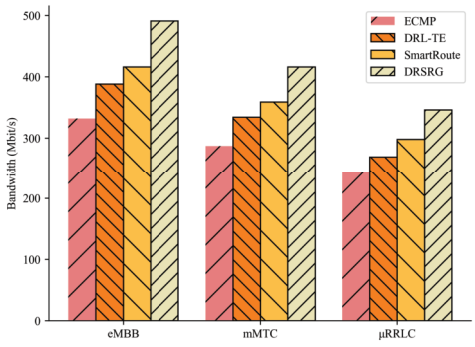


Fig. 13. Slice bandwidth capacity.

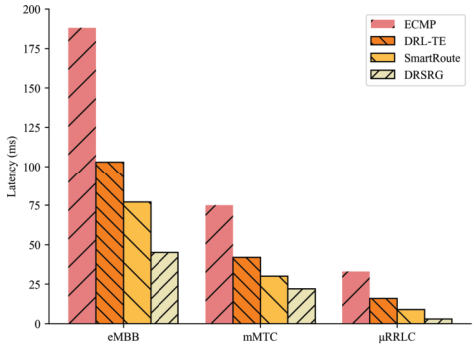


Fig. 14. Latency in slices.

engineering policy, which enables fine-grained control of end-to-end forwarding paths. After directing traffic into the SRv6 Policy, the head node forwards packets along the path specified by the control layer. On the contrary, the non-SRv6 nodes merely serve as packet forwarders, so that the time required for packet processing is reduced and then the delay becomes naturally lower.

Secondly, as for the slice bandwidth capacity, it is shown in Figure 13, where different bandwidth capacity allocated to different kinds of slices by using different methods are presented separately. Due to the highest bandwidth requirements in the eMBB slice, the bandwidth capacity on the routing paths of the eMBB slice is significantly higher than the other two slices. This indicates that the eMBB slice performs the best in terms of bandwidth performance, making it more suitable for tasks like high-definition video transmission. Additionally, since the three comparative algorithms do not adapt their chosen paths based on network load and link conditions, while the proposed DRSRG method can analyze multiple attributes of links, it performs the best in the metric of bandwidth capacity in different slices. This demonstrates that the designed routing mechanism has better path selection capabilities, which makes it more adaptable to different network environments and varied load conditions.

Thirdly, as for the average delay over different slices, the corresponding results are shown in Figure 14. As explained, the three slices have their own special focus, so that we need to take that into consideration. On one hand, due to the low-latency requirements in the uRLLC slice, the delay on

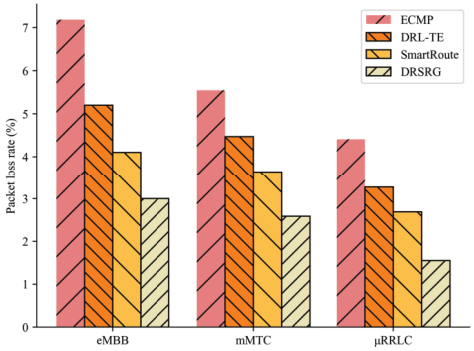


Fig. 15. Packet loss rate in slices.

the routing paths of the uRLLC slice is significantly lower than that achieved in the other two slices. On the other hand, we should note that the mMTC slice has lower delay than that in the eMBB slice, as the massive machine-type communication scenario also imposes certain requirements on latency. Nevertheless, the above analysis is mainly based on the slice categories, and we need to discuss the algorithm's performance as well. Thus, let's look deep into the results, we can see that the delay achieved by the proposed method is noticeably lower than the three comparative algorithms. This is because, after the infrastructure network undergoes SRv6 optimization deployment and node upgrades, the DRSRG method can reduce network forwarding delays by resolving and forwarding packets through SRv6 nodes. This improvement enhances data transfer speed, shortens data transmission time and consequently reduces latency.

Lastly, as for the packet loss rate in slices, the corresponding results are shown in Figure 15, where the packet loss rate is achieved against different kinds of slices. In particular, from the overall perspective, we can see that the packet loss rate in eMBB is relatively higher than that in the other two slices, which means that supre-high bandwidth demands would have a large impact on this metric. In addition, as for the details achieved by different method, we can also observe that the proposed method calculates the lowest packet loss rate in all the three slices, which is reasonable. On one hand, due to the ultra-reliable nature required in the uRLLC scenario, the packet loss rate on the routing paths of the uRLLC slice is significantly lower than the other two slices. On the other hand, since the ECMP algorithm, DRL-TE algorithm and SmartRoute algorithm do not consider the real-time status of links, they have higher packet loss rates. The proposed routing algorithm based on graph convolutional neural networks exhibits a noticeable reduction in packet loss rate, due to its strong generalization ability of using the graph convolutional operator and allowing to dynamically adjust forwarding paths.

VI. Conclusion

To address the complex and dynamic application requirements in B5G/6G enabled slicing networks, this work first designs a framework with the ideas of SRv6 and zero-trust taken into consideration jointly. Under this framework, a GCN based policy routing algorithm is proposed to construct the

policy routing paths accordingly, where each policy can be customized and deployed in the network flexibly and dynamically. Experimental results demonstrate that the proposed policy routing algorithm outperforms the other state-of-the-art methods in terms of the end-to-end delay, bandwidth consumption and packet loss rate. Future work include exploring more potentials of policy routing in other scenarios.

REFERENCES

[1] S. Boutalbi, N. Kheir, R.-A. Koutsiamanis, M. Sudholt, and Y. Dan Moussa, "Mobile edge slice broker: Mobile edge slices deployment in multi-cloud environments," in *Proc. IEEE 7th Int. Conf. Fog Edge Comput. (ICFEC)*, Bangalore, India, May 2023, pp. 58–63.

[2] D. Mishra and E. Natalizio, "A survey on cellular-connected UAVs: Design challenges, enabling 5G/B5G innovations, and experimental advancements," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107451.

[3] X. Lv, S. Rani, S. Manimurugan, A. Slowik, and Y. Feng, "Quantum-inspired sensitive data measurement and secure transmission in 5G-enabled healthcare systems," *Tsinghua Sci. Technol.*, vol. 30, no. 1, pp. 456–478, Feb. 2025.

[4] B. Dai, Y. Cao, Z. Wu, Z. Dai, R. Yao, and Y. Xu, "Routing optimization meets machine intelligence: A perspective for the future network," *Neurocomputing*, vol. 459, pp. 44–58, Oct. 2021.

[5] C. Tipantuña and P. Yanchapaxi, "Network functions virtualization: An overview and open-source projects," in *Proc. IEEE 2nd Ecuador Tech. Chapters Meeting (ETCM)*, Oct. 2017, pp. 1–6.

[6] Z. Zhang and L. Dai, "Reconfigurable intelligent surfaces for 6G: Nine fundamental issues and one critical problem," *Tsinghua Sci. Technol.*, vol. 28, no. 5, pp. 929–939, Oct. 2023.

[7] C. Filsfils, P. Camarillo, and J. Leddy, *Segment Routing Over IPv6 (SRv6) Network Programming*, document RFC 8986, Internet Engineering Task Force (IETF), 2021.

[8] C. Filsfils, P. C. Garvia, and J. Leddy, *SRv6 Network Programming*, document IETF InternetDraft, 2017.

[9] Z. Li, Z. Hu, and C. Li, *SRv6 Network Programming: Ushering in a New Era of IP Networks*. Boca Raton, FL, USA: CRC Press, 2021.

[10] P. L. Ventre et al., "Segment routing: A comprehensive survey of research activities, standardization efforts, and implementation results," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 182–221, 2nd Quart., 2020.

[11] D. Wu and L. Cui, "A comprehensive survey on segment routing traffic engineering," *Digit. Commun. Netw.*, vol. 9, no. 4, pp. 990–1008, Aug. 2023.

[12] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 657–665.

[13] T. Miyasaka, Y. Hei, and T. Kitahara, "NetworkAPI: An in-band signalling application-aware traffic engineering using SRv6 and IP anycast," in *Proc. Workshop Netw. Appl. Integration/CoDesign*, Aug. 2020, pp. 8–13.

[14] Y. Tian et al., "Traffic engineering in partially deployed segment routing over IPv6 network with deep reinforcement learning," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1573–1586, Aug. 2020.

[15] P. L. Ventre, M. M. Tajiki, S. Salsano, and C. Filsfils, "SDN architecture and southbound APIs for IPv6 segment routing enabled wide area networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1378–1392, Dec. 2018.

[16] A. Abdelsalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a linux-based NFV infrastructure," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Jul. 2017, pp. 1–5.

[17] M. Gramaglia, V. Sciancalepore, F. J. Fernandez-Maestro, R. Perez, P. Serrano, and A. Banchs, "Experimenting with SRv6: A tunneling protocol supporting network slicing in 5G and beyond," in *Proc. IEEE 25th Int. Workshop Comput. Aided Model. Design Commun. Links Netw. (CAMAD)*, Sep. 2020, pp. 1–6.

[18] C. Li, J. Mao, S. Peng, Y. Xia, Z. Hu, and Z. Li, "Application-aware G-SRv6 network enabling 5G services," in *Proc. IEEE Conf. Comput. Commun. Workshops*, May 2021, pp. 1–2.

[19] L. Song, Y. Jin, P. Wang, D. Ma, W. Chen, and L. Cui, "Multi-path routing deployment method based on SRv6," in *Proc. IEEE Int. Conf Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Sep. 2021, pp. 723–730.

[20] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[21] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[22] S. Zhang, "An overview of network slicing for 5G," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 111–117, Jun. 2019.

[23] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 106984.

[24] Z. Shu and T. Taleb, "A novel QoS framework for network slicing in 5G and beyond networks based on SDN and NFV," *IEEE Netw.*, vol. 34, no. 3, pp. 256–263, May 2020.

[25] W. Guan, X. Wen, L. Wang, Z. Lu, and Y. Shen, "A service-oriented deployment policy of end-to-end network slicing based on complex network theory," *IEEE Access*, vol. 6, pp. 19691–19701, 2018.

[26] A. Esmaeily and K. Kralevska, "Small-scale 5G testbeds for network slicing deployment: A systematic review," *Wireless Commun. Mobile Comput.*, vol. 2021, no. 1, pp. 1–26, Jan. 2021.

[27] D. Harutyunyan, R. Fedrizzi, N. Shahriar, R. Boutaba, and R. Riggio, "Orchestrating end-to-end slices in 5G networks," in *Proc. 15th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2019, pp. 1–9.

[28] B. Mao et al., "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[29] Z. M. Fadlullah et al., "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 4th Quart., 2017.

[30] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "Fed-TSN: Joint failure probability-based federated learning for fault-tolerant time-sensitive networks," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 1470–1486, Jun. 2023.

[31] F. Geyer and G. Carle, "Learning and generating distributed routing protocols using graph-based deep learning," in *Proc. Workshop Big Data Analytics Mach. Learn. Data Commun. Netw.*, Aug. 2018, pp. 40–45.

[32] Z. Zhuang, J. Wang, Q. Qi, H. Sun, and J. Liao, "Graph-aware deep learning based intelligent routing strategy," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, Oct. 2018, pp. 441–444.

[33] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of graph neural networks for network modeling and optimization in SDN," in *Proc. ACM Symp. SDN Res.*, Apr. 2019, pp. 140–151.

[34] J. Posner, L. Tseng, M. Aloqaily, and Y. Jararweh, "Federated learning in vehicular networks: Opportunities and solutions," *IEEE Netw.*, vol. 35, no. 2, pp. 152–159, Mar./Apr. 2021.

[35] J. Boyan and M. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 6, 1993, pp. 1–8.

[36] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntes-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," 2017, *arXiv:1709.07080*.

[37] Z. Yuan, P. Zhou, S. Wang, and X. Zhang, "Research on routing optimization of SDN network using reinforcement learning method," in *Proc. 2nd Int. Conf. Saf. Produce Informatization (IICSPI)*, Nov. 2019, pp. 442–445.

[38] T. Dong et al., "Intelligent joint network slicing and routing via GCN-powered multi-task deep reinforcement learning," *IEEE Trans. Cognit. Commun. Netw.*, vol. 8, no. 2, pp. 1269–1286, Jun. 2022.

[39] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[40] F. Rezazadeh, S. Barrachina-Muñoz, E. Zeydan, H. Song, K. P. Subbalakshmi, and J. Mangues-Bafalluy, "X-GRL: An empirical assessment of explainable GNN-DRL in B5G/6G networks," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Dresden, Germany, Nov. 2023, pp. 172–174.

[41] Z. Xu et al., "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 1871–1879.

[42] H. Cao, H. Zhao, D. X. Luo, N. Kumar, and L. Yang, "Dynamic virtual resource allocation mechanism for survivable services in emerging NFV-enabled vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 22492–22504, Nov. 2022.