electronics

MDPI

Article

Network State Aware Dual-Graph Spatiotemporal Fusion Prediction Model for SDN Dynamic Routing Optimization

Jiaxian Zhu [1], Jialing Zhao [1], Weihua Bai [1,*], Chuanbin Zhang [1], Zhizhe Lin [2,*] and Teng Zhou [2,3,4]

[1] School of Computer Science, Zhaoqing University, Zhaoqing 526061, China
[2] School of Cyberspace Security, Hainan University, Haikou 570228, China
[3] Yangtze Delta Research Institute, University of Electronic Science and Technology of China, Quzhou 324003, China
[4] School of Artificial Intelligence, Guilin Tourism University, Guilin 541006, China
* Correspondence: baiweihua@zqu.edu.cn (W.B); linzhizhe@hainanu.edu.cn (Z.L.)

Abstract

Software-defined networking (SDN) provides a flexible solution to manage complex networks on demand by centralized control and programmability. However, efficiently optimizing network configurations to achieve load balance and improve service quality remains challenging. In this paper, we propose a novel SDN network state awareness and dynamic routing optimization method, termed DGSFN-DR. Hereby, we leverage a Graph Attention Network (GAT) to model the spatial dependencies of the network topology for its link graph. Then, we employ a Recurrent Neural Network (RNN) to capture the temporal dependencies of link states, including the lagged temporal features induced by routing algorithms, to improve the prediction accuracy of future link states. Our algorithm dynamically adjusts routing strategies to optimize network performance according to the predicted link weights with the dual graph spatiotemporal fusion prediction network (DG-SFN). Experimental results demonstrate that our DGSFN-DR outperforms other methods in various network traffic intensities and topologies. Specifically, it achieves improvements of 4% to 15% in latency, jitter, packet loss, and available bandwidth. In particular, the DGSFN-DR exhibits superior adaptability and optimization potential under high traffic loads and complex network topologies. This work expands dynamic routing optimization theory for SDN and new insights for practical network management.

Keywords: line graph; spatiotemporal prediction; software-defined networking; routing optimization; graph deep learning

Check for updates

Academic Editor: Hung-Yu Chien

Received: 7 April 2026
Revised: 26 April 2026
Accepted: 28 April 2026
Published: 1 May 2026

Copyright: © 2026 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.

1. Introduction

Modern network infrastructure has become increasingly complex, posing greater demands on network management as exponential growth in network devices and the rapid increase in data traffic. Software-defined networking (SDN) offers a viable solution to address these challenges by providing centralized control and programmability [1,2]. SDN decouples the control plane from the data plane for flexibility and programmability in network operations. Network administrators can configure, monitor, and optimize network behavior through a unified control interface in SDN [3,4]. However, conventional routing algorithms based on static metrics may fail to adapt to dynamic traffic variations, leading to network congestion and a decline in Quality of Service (QoS).

Recently, deep learning techniques have shown remarkable potential for predicting network traffic, especially with complex spatiotemporal data [5]. Existing studies mainly

focus on leveraging neural networks for traffic prediction to optimize routing decisions in SDN. However, challenges remain regarding scalability and real-time performance in large-scale networks. Effectively integrating traffic predictions for future network states with routing optimization techniques remains an open problem [6].

In this paper, we propose a dual-graph spatiotemporal prediction framework for SDN network state-aware dynamic routing optimization. Our method provides a granular characterization of the network's physical and logical structure by representing the network topology as a node adjacency matrix and a link-based vertex adjacency matrix. Real-time network metrics, e.g., latency, packet loss rate, bandwidth, and jitter, are collected through the ONOS platform. We generate spatiotemporal feature embeddings for both nodes and links by a Graph Attention Network (GAT). These embeddings are organized into t-step time-series slices. Then, we fed these embeddings into a Recurrent Neural Network (RNN) for future network state prediction. We dynamically adjust the SDN routing strategies using the prediction results to determine optimal paths by minimizing link weights. Our method enhances network resource utilization to improve the QoS.

The main contributions of this paper are as follows:

- We design a dual-graph spatiotemporal prediction framework for dynamic network state prediction.
- We present a dynamic routing optimization algorithm to integrate the network state predictions for the SDN.
- Extensive experiments are conducted to evaluate the proposed method's effectiveness in improving network performance. The experimental results demonstrate our method achieves 4% to 15% improvements compared with commonly-used methods.

The rest of this paper is organized as follows. In Section 2, we briefly review the related works, and the problem description is presented in Section 3. Section 4 presents a dual-graph spatiotemporal prediction framework (DG-SFP) and the routing optimization method named DGSFN-DR. In Section 5, we validate the accuracy of the prediction model and the effectiveness of the prediction-based DGSFN-DR algorithm through extensive experiments conducted under various network topologies and different traffic intensity scenarios. In Section 6, we conclude and discuss future work.

2. Related Work

In software-defined networking (SDN), network state prediction technologies, graph neural network (GNN)-based temporal network state data processing, and dynamic routing optimization enable efficient network management [7,8]. Existing research has made notable progress in these areas, particularly leveraging deep learning techniques for traffic prediction and routing optimization [9,10]. However, these approaches face limitations of scalability in large-scale networks and comprehensive optimization of multiple performance metrics. Network state awareness for SDN dynamic routing optimization by spatiotemporal fusion network state prediction promises to provide a more intelligent and efficient solution [11].

2.1. Network State Awareness and Prediction Technologies

Network State Awareness and Prediction Technologies play a pivotal role in SDN, enabling network administrators to monitor and predict traffic dynamics in real-time. In recent years, deep learning techniques have been widely applied to network traffic prediction. They mainly use recurrent neural networks (RNNs) and their variants, such as gated recurrent units (GRUs), to process time-series data. These techniques analyze historical network performance metrics, such as latency, jitter, and bandwidth, to predict future network states, thereby supporting dynamic routing decisions.

For instance, Tang et al. [12] propose a novel deep learning algorithm based on a deep belief architecture and convolutional neural networks (CNNs) to predict potential traffic loads and network congestion.

They further integrated the deep learning-based prediction algorithm with a channel allocation algorithm to intelligently control routing traffic.

Similarly, Ferriol et al. [13] introduce a customized GNN model, RouteNet-Fermi, which aligns with queuing theory principles and can accurately predict network latency, jitter, and packet loss. Almasan et al. [14] incorporate graph neural networks into a deep reinforcement learning (DRL) agent, designing a problem-specific action space to enable learning and generalization across arbitrary network topologies. These machine learning and deep learning-based approaches are also frequently utilized to address state awareness and prediction challenges in 5G/6G networks [15–17].

However, existing research has not fully integrated real-time and continuous network state monitoring data, leading to reduced prediction accuracy in dynamically changing network environments.

2.2. Spatiotemporal Data Fusion Techniques Based on Graph Neural Networks

The introduction of Graph Neural Networks (GNNs) and Graph Convolutional Networks (GCNs) has marked a transformative shift in modeling relationships within non-Euclidean spaces [18–20]. Spatiotemporal data fusion models based on GNNs leverage graph structures to represent the topology of real-world networks, effectively capturing logical spatial dependencies that conventional Convolutional Neural Networks (CNNs) cannot process. Models such as Spatiotemporal Graph Convolutional Networks (STGCN) [21] and those employing Dynamic Time Warping (DTW) matrices for semantic understanding, as well as ASTGCN [22], have demonstrated outstanding performance in extracting spatial features.

To address spatiotemporal fusion in sequential data, Li et al. [23] proposed an innovative Spatiotemporal Fusion Graph Neural Network (STFGNN), which enhances the modeling capacity for complex spatiotemporal data by introducing temporal graphs and spatiotemporal fusion mechanisms. This approach overcomes the limitations of existing models in learning spatiotemporal dependencies. Similarly, Yang et al. [24] introduce an improved Spatiotemporal Adaptive Fusion Graph Network (STFAGN). This model integrates convolutional layers with a novel adaptive dependency matrix for end-to-end training, capturing hidden spatiotemporal dependencies in the data to supplement incomplete information.

The further development of graph learning frameworks, such as GMAN [25] and AGCRN [26], establish mechanisms for dynamically learning optimal graph structures. The Graph Multi-Attention Network (GMAN) employs an encoder-decoder architecture, where the encoder and decoder consist of multiple spatiotemporal attention modules to model the influence of spatiotemporal factors. The encoder encodes the input traffic features, while the decoder predicts the output sequence. A transform attention layer is also applied to mitigate error propagation across prediction time steps [25].

Bai et al. [26] proposed the Adaptive Graph Convolutional Recurrent Network (AGCRN), which enhances the capability of GCNs by introducing two key modules: the Node Adaptive Parameter Learning module and the Data Adaptive Graph Generation module. These modules allow AGCRN to automatically capture fine-grained spatiotemporal correlations in time-series data without requiring a predefined spatial connectivity graph.

While these methods excel at optimizing node embeddings and graph matrices to capture the evolving relationships in temporal data, their real-time responsiveness to sudden events or anomalies remains an area for further improvement. In addition to

latency, anomaly-driven decision and explanation pipelines may also introduce privacy and data-leakage risks through AI outputs, especially when telemetry-derived information is processed or exposed in real time. A survey on data security and privacy protection in artificial intelligence models summarizes key challenges and defense mechanisms against data leakage in AI outputs [27], which supports the need to consider privacy-aware data handling when deploying network-intelligent systems that respond to anomalies.

2.3. Dynamic Routing Optimization Techniques

Heuristic and optimization-based methods are widely used in dynamic routing optimization. Binsahaq et al. [28] propose a QoS-aware routing algorithm that incorporates latency constraints into the objective function and reduces the solution search space. Their method supports latency-sensitive SDN services and achieves results with low computational overhead. Zhang et al. [29] evaluate the performance of Dijkstra's shortest path algorithm on two SDN controllers, RYU and POX, using QoS metrics as evaluation criteria. Their study examines packet capture and transmission strategies based on shortest path and minimum load path, offering valuable insights into SDN routing performance.

Mohammadi et al. [30] propose a novel and efficient algorithm to address link failure issues, utilizing the Bidirectional Forwarding Detection(BFD) protocol to detect and proactively update backup paths. The Shared Risk Link Group (SRLG) algorithm was employed to identify disjoint paths by considering latency and link utilization as cost metrics.

Heuristic and optimization-based methods have made significant advancements in dynamic routing optimization. However, these methods encounter challenges when applied to large-scale networks, particularly in terms of scalability and efficiency. Specifically, challenges remain in optimizing routing decisions under dynamically changing network environments while considering multiple performance metrics.

2.4. Summary of Limitations in Prior Work and Our Key Distinction

Although existing studies provide valuable designs for traffic/state prediction and routing optimization, they often share three limitations with respect to SDN network state-aware dynamic routing. First, many prediction-oriented approaches model spatiotemporal dependencies from *one* structural view (e.g., node-centric topology graphs) or rely on graph representations that are not explicitly aligned with the edge-level coupling required by path construction. As a result, even when prediction accuracy is high, the induced link-weight patterns along candidate paths may be inconsistent with routing objectives. Second, several graph-based spatiotemporal fusion methods focus on improving node/link embeddings and dynamic graph learning, but do not provide a mechanism to translate the learned representations into a coherent, routing-relevant link-state space for multi-QoS weight construction. Third, routing optimization methods (including heuristic/optimization and shortest-path variants) commonly compute decisions based on current link weights or handcrafted metrics, and do not explicitly integrate a prediction-to-weight coupling pipeline.

In contrast, this paper proposes DGSFN-DR with an explicit dual-graph spatiotemporal fusion design and a routing-aligned coupling mechanism. We construct a physical topology graph G_P and a line/edge adjacency graph G_L, and introduce a feature alignment/fusion strategy to produce coherent edge-level representations for future link-state prediction. Subsequently, the predicted future link states are deterministically mapped to link weights via $\theta(\cdot)$, and routing decisions are obtained by minimizing the predicted end-to-end path cost using a Dijkstra-based procedure. Importantly, the routing stage follows an identical rule across variants in our ablation studies, so the downstream QoS gains can be attributed to how the dual-graph fusion improves prediction-to-weight consistency rather than changing the routing mechanism.

3. Preliminaries

3.1. Problem Description

Figure 1 illustrates an application scenario of an SDN network, where the SDN controller separates the network control plane from the data plane. This separation enables flexible network management and configuration, centralized network control, and dynamic resource allocation, enhancing the network's manageability and scalability. The dual-graph spatiotemporal prediction framework for the intelligent management of SDN networks aims to predict and optimize network states by collecting and analyzing historical data.

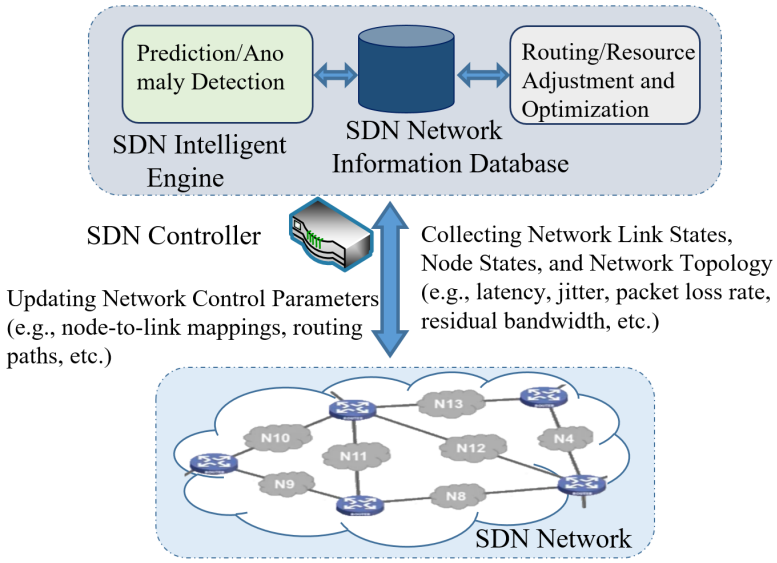


Figure 1. Application Scenario of SDN Networks Based on a Prediction Framework.

In operational SDN settings, routing issues and abnormal QoS patterns are not only decision-triggering signals but also require actionable interpretation by network operators. Therefore, beyond predicting future link states, the SDN intelligent control engine should provide clear, evidence-backed explanations grounded in telemetry data. A recent line of work on explainable anomaly detection in cloud-native systems shows that telemetry can be used as a basis for anomaly explanation, and retrieval-enhanced techniques can support transparent and case-specific rationale [31]. Motivated by this, our SDN Intelligent Engine is designed to treat telemetry as both (i) the input evidence for spatiotemporal fusion prediction and (ii) the underlying support for interpreting detected routing problems, improving the practical usability of prediction-guided routing decisions. As shown in Figure 1, the primary task is accomplished by deploying and implementing the prediction framework in the SDN network as follows.

3.2. Real-Time Data Collection

In the SDN architecture, the network controller collects real-time performance data from various nodes and links within the network. These data include the state of nodes $N(t)$ (e.g., CPU utilization $U_c(t)$, memory usage $U_R(t)$), the attributes of links $Link(t)$ (e.g., bandwidth $B_{ij}(t)$, latency $L_{ij}(t)$, packet loss rate $P_{ij}(t)$, jitter $J_{ij}(t)$), and other relevant performance metrics. The network state at time t is represented as a state vector:

$$X(t) = \{N(t), Link(t)\} = \{\{U_{ci}(t), U_{Ri}(t), \dots\}, \{B_{ij}(t), L_{ij}(t), P_{ij}(t), J_{ij}(t), \dots\}\}. \quad (1)$$

where $i, j \in N$, and N denotes the set of nodes (routers or switches) in the SDN network.

3.3. Intelligent Prediction and Network State Awareness

The historical network state dataset collected by the SDN is analyzed, and the Dual-Graph Spatiotemporal Fusion Prediction Network (DG-SFN) is utilized to predict the network state at a future time step $t + \Delta t$. The DG-SFN model can be defined as:

$$S(t + \Delta t) = f(x_{t-T}, x_{t-T+1}, x_{t-T+2}, \ldots, x_t; \delta), \tag{2}$$

where $x_{t-T+i} \in X(t)$ for $0 \leq i \leq T$. f represents the neural network model corresponding to DG-SFN. δ denotes the set of parameters in the neural network. The output of the model, $S(t + \Delta t)$, represents the network state $Link(t + \Delta t)$ at the future time step $t + \Delta t$. This model integrates spatiotemporal data features to enable accurate prediction and awareness of future network states.

3.4. Dynamic Path Optimization

Based on the predicted network state information, the SDN intelligent control engine calculates the link weight coefficients and searches for the minimum weighted path between node pairs to solve the path optimization problem dynamically. The link weight coefficient function θ is defined as:

$$\begin{aligned} W_{ij}(t + \Delta t) &= \theta(S(t + \Delta t)) \\ &= \theta(B_{ij}(t + \Delta t), L_{ij}(t + \Delta t), \\ &\quad P_{ij}(t + \Delta t), J_{ij}(t + \Delta t), \ldots), \end{aligned} \tag{3}$$

where W_{ij} represents the link weight coefficient for the Link between adjacent nodes i and j. Based on the predicted results W_{ij}, the objective function for the minimum weighted path optimization problem is defined as:

$$F(t + \Delta t) = \min_{P \in \mathcal{P}(\mathcal{S},d)} \sum_{(i,j) \in P} W_{ij}(t + \Delta t), \tag{4}$$

where P denotes a path between two nodes, and $\mathcal{P}(\mathcal{S}, d)$ represents the set of all possible paths between the source node $\mathcal{S}$ and the destination node d.

(1) Prediction Target and Prediction-to-Weight Definition

Our routing optimization is performed on predicted link weights derived from the prediction target. Let $\hat{S}_{ij}(t + \Delta t) \in \mathbb{R}^m$ denote the predicted link-state vector of link (i, j) at time $t + \Delta t$, where $m = 4$ corresponds to *latency*, *jitter*, *packet loss rate* (*PLR*), and *available bandwidth* (*ABW*).

(2) Link-weight construction ($\theta(\cdot)$).

Given $\hat{S}_{ij}(t + \Delta t)$, we compute the corresponding link weight $w_{ij}(t + \Delta t)$ via the mapping function $\theta(\cdot)$ in Equation (3), i.e., $w_{ij}(t + \Delta t) = \theta\big(\hat{S}_{ij}(t + \Delta t)\big)$. Concretely, each QoS component is first normalized to a common scale to avoid a mismatch caused by heterogeneous units. For each component $k \in \{\text{lat}, \text{jit}, \text{plr}, \text{abw}\}$, we apply

$$\tilde{s}_{ij}^{(k)}(t + \Delta t) = \frac{s_{ij}^{(k)}(t + \Delta t) - \mu_k}{\sigma_k}, \tag{5}$$

where μ_k and σ_k are computed from the training set as described in Section 4.6.1. The normalized components are then combined into a scalar link cost as

$$w_{ij}(t + \Delta t) = \alpha_{\text{lat}} \tilde{s}_{ij}^{(\text{lat})}(t + \Delta t) + \alpha_{\text{jit}} \tilde{s}_{ij}^{(\text{jit})}(t + \Delta t) + \alpha_{\text{plr}} \tilde{s}_{ij}^{(\text{plr})}(t + \Delta t) + \alpha_{\text{bw}} \tilde{s}_{ij}^{(\text{abw})}(t + \Delta t), \tag{6}$$

where $\{\alpha_{\text{lat}}, \alpha_{\text{jit}}, \alpha_{\text{plr}}, \alpha_{\text{bw}}\}$ are non-negative weights determining the relative emphasis on each QoS objective.

(3) Relation to end-to-end QoS objectives.

We then route packets by minimizing the accumulated predicted path cost in Equation (4). For a path $p = (v_0, \ldots, v_L)$, the predicted path weight is $W_p(t + \Delta t) = \sum_{(i,j)\in p} w_{ij}(t + \Delta t)$. This corresponds to an end-to-end delay proxy because latency and jitter are modeled as link-level additive contributions over the path. PLR is represented by a link-level loss rate component that increases the path cost (via its contribution in $\theta(\cdot)$), and ABW is mapped to a cost term that penalizes low available bandwidth. Therefore, the shortest-path objective on W_p provides a direct mechanism to reduce delay/jitter/PLR while favoring routes with higher bandwidth.

(4) Optimization objective.

Finally, the Dijkstra-based decision procedure in Algorithm 1 selects the route with the smallest predicted path cost, ensuring that routing decisions are explicitly aligned with the end-to-end QoS objectives encoded in $w_{ij}(t + \Delta t)$.

Algorithm 1 Dynamic Shortest Path Decision for SDN Networks: DGSFN-DR(W, u, v)

Require: Initial priority queue Q, link weight array $W[]$, source node u, destination node v
Ensure: Shortest distance $dist[v]$, path information $prev[v]$

Initialization:
$Q \leftarrow \{\}$, $dist[v] \leftarrow \infty$, $prev[v] \leftarrow \emptyset$
$dist[u] \leftarrow 0$
while $Q \neq \emptyset$ **do**
  $u \leftarrow$ GetminDist(Q) {Select the node u with the minimum $dist[u]$ from Q}
  for all $v \in$ Neighbor(u) **do**
    $W'_{(u,v)} \leftarrow W_{(u,v)} + W_{(i(u\to v))}$ {Compute the adjusted weight for link (u, v)}
    $alt \leftarrow dist[u] + W'_{(u,v)}$ {Calculate the alternative distance through u}
    if $alt < dist[v]$ **then**
      $dist[v] \leftarrow alt$
      $prev[v] \leftarrow u$
    end if
  end for
end while
return $dist[], prev[]$

(5) End-to-end coupling via prediction-to-weight path objective.

Although the routing decision uses a Dijkstra-based shortest-path solver, the optimization mechanism is not limited to the classical path search. Our end-to-end design couples prediction to routing through an explicit prediction-to-weight construction and a routing-aligned path-cost objective: (i) the predicted link-state metrics at $t + \Delta t$ are transformed into link weights by the mapping function $\theta(\cdot)$ (Equations (3)–(6)), (ii) routing then selects the path by minimizing the accumulated predicted path cost $F(t + \Delta t)$ (Equation (4)) using the same predicted weights inside Algorithm 1. Therefore, the routing decision is aligned with the end-to-end QoS proxy encoded in the predicted link weights. To validate that the observed routing QoS gains are attributed to this tight prediction-to-routing coupling (rather than using a graph-based predictor alone), we conduct dual-graph ablations in Section 5.3. With the routing rule kept fixed, differences among GP-only/GL-only/Dual-w/o Fusion/Full Dual lead to consistent improvements in routing QoS (Tables 1 and 2) and path-level selection alignment (Figures 6 and 7). In particular, the gap between Full

Dual and Dual-w/o Fusion indicates that the proposed fusion/alignment is essential for producing prediction outputs that translate into path-relevant link-weight patterns.

Table 1. Routing QoS under Dual-Graph Ablation on the NSFNET Topology.

Traffic	Metric	DSP	GP-Only	GL-Only	Dual-w/o Fusion	Full Dual
LOW (20%)	Latency (ms)↓	13.720	13.128	13.386	13.001	**12.828**
	Jitter (ms)↓	4.910	4.801	4.846	4.703	**4.663**
	Packet loss (%)↓	0.0790	0.0760	0.0770	0.0740	**0.0730**
	Available bandwidth (Mbps)↑	80.329	81.762	81.002	83.125	**83.691**
MID (40%)	Latency (ms)↓	22.867	21.593	21.981	21.097	**20.852**
	Jitter (ms)↓	6.749	6.486	6.502	6.202	**6.125**
	Packet loss (%)↓	0.0980	0.0930	0.0950	0.0910	**0.0897**
	Available bandwidth (Mbps)↑	61.568	63.012	62.919	65.651	**65.974**
HIGH (60%)	Latency (ms)↓	31.575	30.725	31.021	29.155	**28.914**
	Jitter (ms)↓	9.757	9.475	9.502	9.104	**8.968**
	Packet loss (%)↓	0.1138	0.1102	0.1115	0.1088	**0.1068**
	Available bandwidth (Mbps)↑	35.385	37.223	36.922	38.868	**39.158**

Table 2. Routing QoS under Dual-Graph Ablation on the NYC Topology.

Traffic	Metric	DSP	GP-Only	GL-Only	Dual-w/o Fusion	Full Dual
LOW (20%)	Latency (ms)↓	14.827	14.298	14.336	13.954	**13.868**
	Jitter (ms)↓	5.156	5.013	5.097	4.876	**4.840**
	Packet loss (%)↓	0.0870	0.0860	0.0860	0.0840	**0.0830**
	Available bandwidth (Mbps)↑	79.547	81.865	81.235	83.323	**83.649**
MID (40%)	Latency (ms)↓	24.795	23.979	24.024	22.785	**22.564**
	Jitter (ms)↓	6.939	6.576	6.618	6.413	**6.329**
	Packet loss (%)↓	0.1060	0.1040	0.1040	0.0980	**0.0970**
	Available bandwidth (Mbps)↑	59.734	61.655	61.227	63.138	**63.894**
HIGH (60%)	Latency (ms)↓	35.781	33.452	33.549	31.992	**31.856**
	Jitter (ms)↓	11.894	11.022	11.039	10.355	**10.287**
	Packet loss (%)↓	0.1390	0.1276	0.1293	0.1201	**0.1189**
	Available bandwidth (Mbps)↑	29.146	31.725	31.663	33.465	**33.684**

3.5. Dynamic Adjustment and Control

The SDN controller updates the network control parameters and dynamically reconfigures the network settings according to the prediction and optimization results. This includes adjusting routing paths, bandwidth allocation, and prioritization to ensure optimal network performance and guarantee quality of service (QoS).

4. Methodology

4.1. Overview of Network State Prediction Framework

The Dual-Graph Spatiotemporal Fusion Prediction Network (DG-SFN) enables effective perception of the Software-Defined Networking (SDN) system state. Optimize dynamic routing strategies by leveraging spatiotemporal data fusion and graph-based modeling to improve the accuracy and efficiency of decision-making. Figure 2 illustrates the architecture of the DG-SFN framework, which integrates spatio-temporal features through a multilevel structural design. Using graph neural networks (GNNs) and recurrent neural networks (RNNs), the framework achieves accurate predictions of network states and facilitates dynamic adjustments to routing policies.

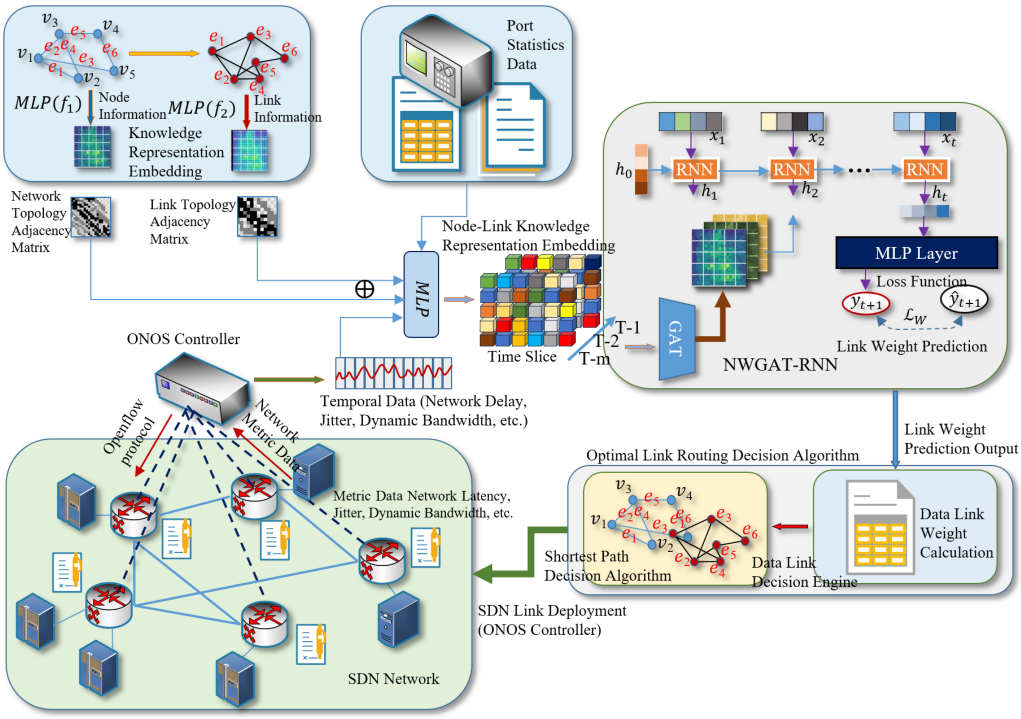


Figure 2. Overall Architecture of the Dual-Graph Spatiotemporal Fusion Network State Prediction Framework.

Although GATs for spatial feature extraction and RNN-family networks for temporal modeling are standard components in spatiotemporal learning, we emphasize that their selection in DGSFN-DR is aligned with our dual-graph formulation and the routing-aware prediction-to-weight pipeline. First, our spatial reasoning operates on two complementary graphs: the physical topology graph G_P and the line/edge adjacency graph G_L. Within each graph, attention-based message passing (GAT) is particularly suitable because it learns edge- (or node-) level importance from neighbor aggregation, which is consistent with our need to model shared-endpoint coupling and congestion-related propagation across traversed links. Second, our temporal module predicts link-state metrics at $t + \Delta t$ from a fixed historical window, and the prediction is subsequently mapped to link weights via $\theta(\cdot)$ and consumed by a deterministic Dijkstra-based routing rule (Equations (3) and (4) and Algorithm 1). In this closed-loop setting, modeling the temporal evolution of link states with an RNN provides stable sequence dependency capture without altering the downstream routing mechanism. Finally, we note that more advanced temporal/spatiotemporal alternatives (e.g., Transformer-based predictors) are included in the baseline comparisons under a unified protocol (Table 5). The experimental results (Tables 8–11) show that simply adopting a more complex backbone does not guarantee better prediction-to-routing effectiveness. Under the same routing pipeline constraint, the proposed GAT+RNN backbone, together with dual-graph spatiotemporal fusion, achieves consistent gains across prediction metrics and end-to-end routing QoS, supporting the adequacy of our design choice.

As illustrated in Figure 2, our framework introduces an innovative approach to spatial-temporal feature extraction and dynamic optimization strategies, offering new methodologies and technical pathways to efficiently manage software-defined network (SDN) systems. The primary components of the dual-graph spatiotemporal fusion network state prediction framework are as follows.

We first conduct a dual-graph embedding of node and link. The network topology is mapped into a dual representation consisting of a node adjacency graph and a link adjacency graph to capture the intrinsic spatio-temporal dependencies of complex network

structures. This dual-graph logical representation not only characterizes the physical connectivity of the network but also captures the complex interactions at the network level.

Using the ONOS platform, real-time multidimensional network metric data is collected, including latency, packet loss rate, bandwidth, and jitter. The network node and link features are embedded in a high-dimensional feature space using MLP(f_1) and MLP(f_2), respectively, providing a rich foundation for subsequent feature extraction.

Second, we extract spatiotemporal features of the nodes and links by graph attention network. We slice the spatiotemporal embeddings of nodes and links at each time step. Then, we employ GATs to capture the spatio-temporal features of two types of spatial propagation characteristics. The GAT effectively identifies and highlights critical connections and interactions by using the attention mechanism, providing high-quality spatio-temporal feature representations. The spatio-temporal embeddings of nodes in the dual-graph representation are subsequently transformed into time-series slices, reflecting the network state's dynamic evolution and propagation dependencies.

Third, we make network state predictions using a recurrent neural network. The DG-SFG (Dual-Graph Spatiotemporal Fusion GAT) generates the node propagation feature embeddings. Then, we construct spatio-temporal feature embeddings from such node propagation feature embeddings. The spatio-temporal feature embeddings are used as input features for the RNN to perform time-series prediction. The RNN predicts the future network state at Δt by effectively capturing long-term dependencies in the time-series data. After that, we minimize link weights to select the optimal path according to the RNN prediction results to perform the dynamic routing optimization. Thus, it significantly improves the efficiency of network resource utilization and the quality of service.

Last, we employ the network state to determine the data link. We use the prediction results to calculate the data link weight coefficients for adjacent node pairs through a weighted path optimization strategy. We determine the optimized node-to-node data links in the traffic matrix using the shortest data link search algorithm. We employ an optimized dynamic routing table for the SDN controller to enable real-time network adjustment and optimization, forming a closed-loop feedback mechanism. Our dynamic optimization mechanism allows the network to continuously adapt to changes in network state and maintain optimal performance.

4.2. Dual-Graph Embedding of Node and Link

The computer network topology and the links within the network topology can be represented as a graph. Graph neural networks (GNNs) [32], such as GATs [33], GCNs [34], and other variants, are effective in capturing the interconnections and transmission dependencies between nodes in the graph [35]. To capture the spatiotemporal dependencies between nodes and links in the network, we propose a dual graph-based node and link representation embedding method: (1) Node Adjacency Matrix. The network's physical topology is mapped into a node adjacency matrix representation. This matrix is processed using a multilayer perceptron (MLP(f_1)) to extract the feature embeddings of the nodes. (2) Link Adjacency Matrix. The network's logical link structure is represented as a link adjacency matrix (edge graph/Line graph). Similarly, this matrix is processed using a multilayer perceptron (MLP(f_2)) to generate the feature embeddings of the links. The dual-graph representation and link mapping method in the SDN network is shown in Figure 3.

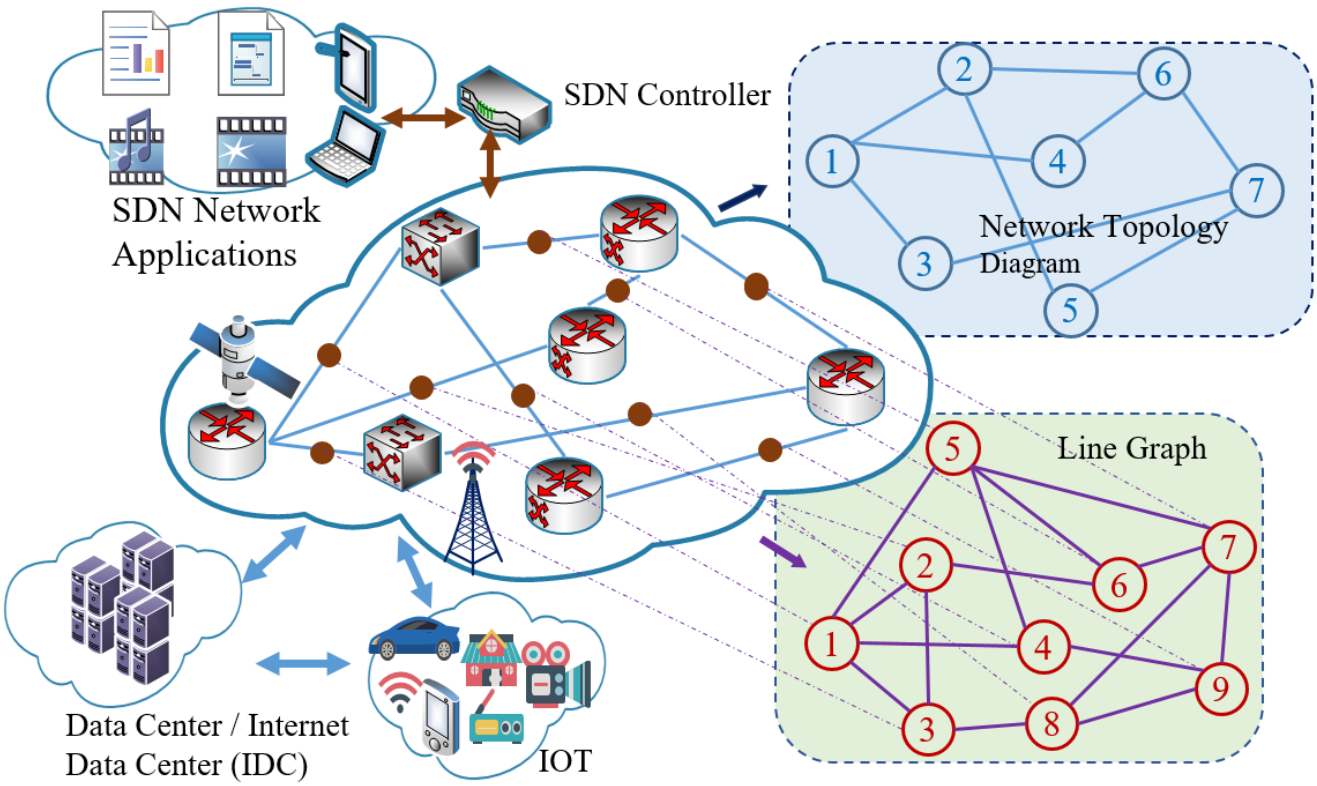


Figure 3. Illustration of the Dual-Graph Representation and Link Mapping Method in SDN Networks.

Figure 3 illustrates mapping the network topology into two graphs: the physical topology graph G_P and the link-associated vertex graph G_L. The dual-graph representation captures representation and propagation characteristics, providing a foundation for accurate network state perception and dynamic routing optimization. The corresponding definitions of the graphs are as follows:

- The physical topology graph of the network is defined as $G_{\mathcal{P}} = (V_{\mathcal{P}}, E_{\mathcal{P}})$, representing the actual physical connections within the network, where V_P is the set of nodes, representing switches or routers in the network; E_P is the set of edges, representing physical links. Each node $v_i \in V_{\mathcal{P}}$ and $v_j \in V_{\mathcal{P}}$ is connected by an edge $e_{ij} \in E_{\mathcal{P}}$, which denotes the direct physical connection between the nodes. The edge e_{ij} is also referred to as a link.
- The link graph is defined as $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}})$, representing the adjacency relationships of edges $E_{\mathcal{P}}$ in the original graph $G_{\mathcal{P}}$. Each vertex $v_e \in V_{\mathcal{L}}$ in the link graph $G_{\mathcal{L}}$ corresponds to an edge in the original graph $G_{\mathcal{P}}$. Specifically:
  - **Vertex set:** $V_{\mathcal{L}} = \{v_e | e \in E_{\mathcal{P}}\}$, where v_e is the vertex in $G_{\mathcal{L}}$ corresponding to the edge e in $G_{\mathcal{P}}$.
  - **Edge set:**

$$E_{\mathcal{L}} = \left\{ \begin{array}{c} e_i = (u,v), e_j = (u,w) \\ (v_{e_i}, v_{e_j}) \mid \text{or } e_i = (u,v), e_j = (v,w) \\ \text{and } u,v,w \in V_{\mathcal{L}}, e_i, e_j \in E_{\mathcal{P}} \end{array} \right\}$$

    In $G_{\mathcal{L}}$, there exists an edge between two vertices v_{e_i} and v_{e_j} if and only if their corresponding edges e_i and e_j in the original graph $G_{\mathcal{P}}$ share a common vertex u or v.

– **Edge mapping:** $G_{\mathcal{P}} \to G_{\mathcal{L}}$: If two edges e_i and e_j in the original graph $G_{\mathcal{P}}$ share a common vertex, then there exists an edge in $G_{\mathcal{L}}$ between the corresponding vertices v_{e_i} and v_{e_j}.

As illustrated in the example in Figure 3, the topology graph of the SDN network is defined as $G_{\mathcal{P}} = \{\{v_1, v_2, \ldots, v_7\}, \{e_{1,2}, e_{1,3}, e_{1,4}, \ldots, e_{6,7}\}\}$, where $|V_{\mathcal{P}}| = n_{\mathcal{P}} = 5$ and $|E_{\mathcal{P}}| = 9$. The corresponding link graph is defined as $G_{\mathcal{L}} = \{\{v_1, v_2, \ldots, v_9\}, \{e_{1,2}, e_{1,3}, e_{1,4}, \ldots, e_{8,9}\}\}$, where $|V_{\mathcal{L}}| = n_{\mathcal{L}} = 9$ and $|E_{\mathcal{L}}| = 15$.

Based on the port statistics data of the SDN network, the current time-slot network state information, represented by the graphs $G_{\mathcal{P}}$ and $G_{\mathcal{L}}$, is embedded through MLP(f_1) and MLP(f_2). Let $X_{\mathcal{P}}, X_{\mathcal{L}} \in \mathbb{R}^{n_0}$ denote the node data (network state information) of $G_{\mathcal{P}}$ and $G_{\mathcal{L}}$, respectively. The final embedding space has a dimensionality of d_1, and the mapping of node embeddings is defined as:

The embedding process is defined as follows:

$$\mathrm{MLP}(f_1) = \sigma(W_{\mathcal{P}} X_{\mathcal{P}} + b_{\mathcal{P}}), \tag{7}$$

$$z_0^{\mathcal{P}} = W_{\text{embed}}^{\mathcal{P}} \cdot \mathrm{MLP}(f_1) + b_{\text{embed}}^{\mathcal{P}}, \tag{8}$$

$$\mathrm{MLP}(f_2) = \sigma(W_{\mathcal{L}} X_{\mathcal{L}} + b_{\mathcal{L}}), \tag{9}$$

$$z_0^{\mathcal{L}} = W_{\text{embed}}^{\mathcal{L}} \cdot \mathrm{MLP}(f_2) + b_{\text{embed}}^{\mathcal{L}}, \tag{10}$$

where, $W_{\text{embed}}^{\mathcal{P}} \in \mathbb{R}^{d_1 \times n_{\mathcal{P}}}$ and $W_{\text{embed}}^{\mathcal{L}} \in \mathbb{R}^{d_1 \times n_{\mathcal{L}}}$ are learnable weight matrices for projecting to the embedding space, while $b_{\text{embed}}^{\mathcal{P}}, b_{\text{embed}}^{\mathcal{L}} \in \mathbb{R}^{d_1}$ are the corresponding bias terms. The activation function $\sigma(\cdot)$ is the Sigmoid function, and $z_0^{\mathcal{P}}, z_0^{\mathcal{L}} \in \mathbb{R}^{d_1}$ represent the resulting embedding representations.

4.3. Dynamic Collection of SDN Network State Data

In software-defined networks (SDNs), the ONOS controller can capture information about network nodes and links. First, the Link Layer Discovery Protocol (LLDP) is used to identify adjacent switches and links, thereby constructing the network topology. Subsequently, the network controller retrieves detailed information about devices, including flow tables and port states, through the OpenFlow protocol. Active probing, such as sending probe packets (e.g., ICMP Echo Requests), can be employed to measure latency and jitter along the path. Using the flow tables of SDN switches, packet counts can be periodically collected, and by comparing the number of transmitted and received packets, the packet loss rate can be calculated. Additionally, by monitoring the traffic load on ports, the current bandwidth utilization can be estimated, and the remaining bandwidth can be calculated.

Through these methods, the SDN controller can specify source and destination nodes to compute round-trip time and jitter. Monitoring port states and traffic statistics can assess the real-time connectivity and load status of links while also listening to network events to update the topology promptly. The collected information enables the controller to construct a global network view, facilitating more optimal decisions in traffic management and policy deployment. The corresponding algorithmic steps are shown in Algorithms 2 and 3.

Algorithm 2 SDN Network State Acquisition Based on ONOS Controller

Require: ONOS Controller instance
Ensure: Real-time network state data
Initialization:
    ConnectToONOSController();
    StateTable[] ← InitializeDataStructures();
for all host in network **do**
    sendPacketWithNeighborIPs(host);
end for
for all node in network **do**
    capturePacketAndStoreNeighborIPs(node);
end for
LJC(node *s*); // Latency and Jitter Calculation
while collectInterval **do**
    MC[] ← MetricCollection(Host *h*);
    BWS[] ← BandwidthSampling(Links *l*);
    StateTable[] ← PacketState(MC[], BWS[]);
end while
return StateTable[];

Algorithm 3 Active Probing-Based Metric Collection for SDN Network State Estimation

Require: Mininet topology and SDN controller
Ensure: Metrics: Delay, Jitter, etc.
Initialization:
StartSDNController();
InitializeMininetTopology();
StartThreads(ListenToHost, ListenToController);
while True **do**
    P_i ← ReceivePacket();
    h_s ← ProcessPacket(P_i);
    C_i ← ReceiveControllerPacket();
    HandleConfiguration(C_i);
    A ← GetNearbyHosts(C_i);
end while
for all h_s in A **do**
    $MC[]$ ← CalculateMetric(h_s);
end for
return StateTable[];

4.4. Spatiotemporal Feature Extraction

4.4.1. Node-GATs and Link-GATs

The vertex embeddings of $G_{\mathcal{P}}$ and $G_{\mathcal{L}}$, along with their corresponding adjacency matrices, are used as inputs to the GAT module in the DG-SFP framework to capture spatial dependencies among nodes and links.

Since the Number of nodes in the SDN network and the Number of nodes in the line graph representing the links differ, two separate GATs are employed to process node features and link features, respectively. Let the node features and link graph node information at time step t_i be represented as $Z_0^{\mathcal{P}}$ and $Z_0^{\mathcal{L}}$, respectively. These features are learned using the corresponding K-head GATs.

- The Node-GAT model takes the input feature matrix $H_{\mathcal{P}}^{(0)} = Z_0^{\mathcal{P}} \in \mathbb{R}^{n_0 \times d_1}$ as input. For the l-th layer and each attention head k, the attention mechanism is defined as:

$$e_{ij}^{(l,k)_{\mathcal{P}}} = \text{LeakyReLU}\left(\vec{a}^{(l,k)_{\mathcal{P}}^{\top}} \cdot \left[W^{(l,k)_{\mathcal{P}}} h_i^{(l)} \| W^{(l,k)_{\mathcal{P}}} h_j^{(l)}\right]\right), \tag{11}$$

where $\vec{a}^{(l,k)_{\mathcal{P}}^{\top}} \in \mathbb{R}^{1\times 2d_1}$ is the learnable self-attention vector for each node pair in the l-th layer and k-th head, $W^{(l,k)_{\mathcal{P}}} \in \mathbb{R}^{d_1\times d_1}$ is the learnable linear transformation matrix, and $h_i^{(l)}, h_j^{(l)} \in \mathbb{R}^{d_1}$ represent the features of nodes i and j, respectively. Node i and node j are adjacent in the graph.

- After obtaining the attention coefficients between node pairs, the normalized attention coefficient is computed as:

$$\alpha_{ij}^{(l,k)_{\mathcal{P}}} = \frac{\exp\left(e_{ij}^{(l,k)_{\mathcal{P}}}\right)}{\sum_{p\in N(i)} \exp\left(e_{ip}^{(l,k)_{\mathcal{P}}}\right)}, \tag{12}$$

where $N(i)$ represents the set of neighboring nodes of node i, derived from the adjacency matrix $A_{\mathcal{P}}^{(\text{in})}$ of the SDN network topology graph $G_{\mathcal{P}}$.

- The node features are updated as follows:

$$h_i^{(l+1,k)} = \sigma\left(\sum_{j\in N(i)} \alpha_{ij}^{(l,k)_{\mathcal{P}}} \cdot h_j^{(l,k)}\right), \tag{13}$$

where the attention coefficients $\alpha_{ij}^{(l,k)_{\mathcal{P}}}$ are used to aggregate the features of neighboring nodes.

- The multi-head aggregation and update are defined as:

$$h_i^{(l+1)} = \|_{k=1}^{K} h_i^{(l+1,k)}, \tag{14}$$

where $h_i^{(l+1)}$ represents the output feature of node i after applying the normalized attention weights to aggregate features from its neighbors. The nonlinear activation function $\sigma(\cdot)$ is implemented as the ReLU function.

- The Link-GAT model, which processes link features, takes the input feature matrix $H_{\mathcal{L}}^{(0)} = Z_0^{\mathcal{L}} \in \mathbb{R}^{n_{\mathcal{L}}\times d_1}$. For the l-th layer and each attention head k, the computation of attention coefficients is defined similarly to Node-GAT. The attention coefficient is denoted as $a_{ij}^{(l,k)_{\mathcal{L}}}$, and the linear transformation matrix is represented as $W^{(l,k)_{\mathcal{L}}}$. The input graph corresponds to the line graph of the SDN network, denoted as $G_{\mathcal{L}}$, with its adjacency matrix given by $A_{\mathcal{L}}^{(\text{in})}$.

4.4.2. Node and Link Feature Alignment

Due to the mismatch in the Number of features between nodes and links ($n_{\mathcal{L}} \neq n_{\mathcal{P}}$), feature alignment is required. Since DG-SFP predicts the future states of links, feature alignment is performed by aggregating features based on the nodes corresponding to the line graph of links. This process ultimately generates fused node features based on the line graph $G_{\mathcal{P}}$.

Let the output node features and link features after L layers of Node-GAT and Link-GAT be $h_i^{(L)} \in \mathbb{R}^{d_1}$ and $l_i^{(L)} \in \mathbb{R}^{d_1}$, respectively. DG-SFP aligns these features using a weighted averaging method applied to concatenated features. The alignment is defined as:

$$F_i = \left[l_i^{(L)} \| \text{mean}\left(h_{N(i)}^{(L)}\right)\right], \tag{15}$$

where $h_{N(i)}^{(L)}$ represents the features of all network topology nodes related to node i in the line graph (i.e., $e_i = (u, v), e_i \in G_{\mathcal{L}}, N(i) = \{u, v\} \in G_{\mathcal{P}}$, which are the features of nodes in $G_{\mathcal{P}}$). mean$\left(h_{N(i)}^{(L)}\right)$ denotes the average of the features of all nodes related to Link i. The operator $\|$ indicates the feature concatenation. After the alignment, the resulting features satisfy $F_i \in \mathbb{R}^{2d_1}$, and the feature matrix of the entire graph becomes $F \in \mathbb{R}^{n_{\mathcal{L}} \times 2d_1}$.

4.5. Spatiotemporal Fusion Based Link State Prediction

The node representations at each time step are concatenated into a historical time series of T steps, forming the input in the time-series format:

$$S = \{s_1, s_2, \ldots, s_T\} = \{F^{(t-T+1)}, F^{(t-T+2)}, \ldots, F^{(t)}\}, \tag{16}$$

where F^i represents the feature matrix of the entire graph at the i-th time step.

In DG-SFN, the RNN propagates and updates its hidden states across time steps as follows:

$$r_t = \theta(W_r \cdot s_t + U_r \cdot r_{t-1} + b_r), \tag{17}$$

where $r_0 = 0$ is the initial hidden state; $r_t \in \mathbb{R}^{d_r}$ represents the hidden state at the t-th step; $W_r \in \mathbb{R}^{n_{\mathcal{L}} \times d_r}$ is the weight matrix for input-to-hidden connections; $U_r \in \mathbb{R}^{d_r \times d_r}$ is the weight matrix for hidden-to-hidden connections; $b_r \in \mathbb{R}^{d_r}$ is the bias term; and $\theta(\cdot)$ is the activation function, implemented as ReLU.

The prediction model employs an MLP to forecast the future state of each Link, with the output representing the predicted future state for each individual Link. The output layer of the MLP prediction is defined as:

$$\hat{y}_{t+1} = \text{Sigmoid}(W_O \cdot r_t + b_O), \tag{18}$$

where $W_O \in \mathbb{R}^{(n_{\mathcal{L}} \cdot m) \times d_r}$ is the weight matrix of the output layer, responsible for transforming the RNN hidden state into the link state, and $b_O \in \mathbb{R}^{(n_{\mathcal{L}} \cdot m) \times 1}$ is the bias term.

If predictions are made for multiple future time T steps, the total output dimension becomes $\mathbb{R}^{T \times (n_{\mathcal{L}} \cdot m) \times 1}$. Here, m denotes the Number of features for each link.

The loss function of the entire DG-SFP model is defined as:

$$L_{\text{data}} = \frac{1}{N} \sum_{i=1}^{n_{\mathcal{L}}} \sum_{t=1}^{T} \sum_{j=1}^{m} (\hat{y}_{i,t+1,j} - y_{i,t+1,j})^2, \tag{19}$$

$$L_{\text{Total}} = L_{\text{data}} + \lambda(\|W_O\|_1 + \|b_O\|_1), \tag{20}$$

where λ is the hyperparameter for the L1 regularization term, $L_{\text{L1}} = \|W_O\|_1 + \|b_O\|_1$, which controls the intensity of regularization. The L1 regularization penalizes all weights, forcing some unimportant weights to approach zero. The total loss function L_{Total} minimizes both the primary loss and the regularization loss simultaneously.

In this study, to improve prediction efficiency, we set $T = 6$ and $\lambda = 0$. In our loss formulation, the coefficient λ controls the strength of the L1 regularization term that encourages sparsity. We set $\lambda = 0$ in the final configuration because, in our prediction-to-routing framework, enforcing sparsity on the intermediate representations is not aligned to preserve spatiotemporal structure for accurate link-state forecasting. More importantly, during preliminary tuning, we observed that enabling the L1 term does not improve the prediction-to-routing performance under our metrics and may introduce additional optimization bias by over-penalizing non-sparse feature components. Therefore, we disable the L1 regularization by choosing $\lambda = 0$ for the reported results.

4.6. Dynamic Shortest Path Routing Decision

For the network link features predicted by DG-SFP, let the Number of features associated with each link be denoted as m. For example, if $m = 4$, the four features may include jitter, network latency, packet loss rate, and residual bandwidth. These four main features can be used to compute the weight of each link during SDN network configuration. The final routing strategy selects the optimal/shortest routing configuration based on these weights.

4.6.1. Link State Weight Coefficient Calculation

To comprehensively account for multiple network link features (e.g., jitter, network latency, packet loss rate, and residual bandwidth) and unify them into a single weight coefficient for each link, a weighted normalization method that adapts to different feature distributions is employed. This method can be defined as:

- Z-score data standardization is applied to each feature to eliminate the influence of different units, and it can be formulated as:

$$z_{ij} = \frac{L_{ij} - \mu_j}{\sigma_j}, \tag{21}$$

  where L_{ij} represents the j-th feature value of the i-th link ($1 \le i \le n_{\mathcal{L}}$, $1 \le j \le C$); μ_j and σ_j are the mean and standard deviation of the j-th feature, respectively.
- These features are normalized using inverse normalization or direct normalization, respectively. The corresponding formula is defined as:

$$P_{ij} = \begin{cases} \dfrac{z_{ij} - \min\limits_{1 \le k \le n_{\mathcal{L}}} (z_{kj})}{\epsilon + \sqrt{\sum\limits_{i=1}^{n_{\mathcal{L}}} \left(z_{ij} - \min\limits_{1 \le k \le n_{\mathcal{L}}} (z_{kj}) \right)^2}}, & j \in \mathcal{F}^+, \\ \dfrac{\max\limits_{1 \le k \le n_{\mathcal{L}}} (z_{kj}) - z_{ij}}{\epsilon + \sqrt{\sum\limits_{i=1}^{n_{\mathcal{L}}} \left(\max\limits_{1 \le k \le n_{\mathcal{L}}} (z_{kj}) - z_{ij} \right)^2}}, & j \in \mathcal{F}^-, \end{cases} \tag{22}$$

  where $\mathcal{F}^+$ represents the set of features where larger values are preferred (e.g., Bandwidth), while $\mathcal{F}^-$ denotes the set of features where smaller values are preferred (e.g., Latency and Jitter).
- The formula for calculating the entropy of each feature is defined as:

$$E_j = -\frac{1}{\ln(n_{\mathcal{L}})} \left[\sum_{i=1}^{n_{\mathcal{L}}} \left(P_{ij} \ln(P_{ij} + \epsilon) \right) \right], \tag{23}$$

  where $\epsilon = 1 \times 10^{-10}$ is introduced to prevent computational errors.
- The weight of each feature, which reflects the impact of the corresponding feature on link performance, is calculated as:

$$\beta_j = \frac{1 - E_j}{\sum_{j=1}^{C} (1 - E_j)} \cdot \gamma_j, \tag{24}$$

  where γ_j represents the optimization weight of the j-th feature for a link, and $\sum_{j=1}^{C} \gamma_j = 1$.
- The link weight coefficient, which integrates the normalized values of each feature with their corresponding weights, is calculated as follows:

$$W_i = \sum_{j=1}^{m} (\beta_j \cdot P_{ij}). \tag{25}$$

Based on the data distributions of each feature, the link weight coefficient is computed by comprehensively considering the influence of different features on network performance. Equations (22)–(24) not only account for the directionality of the features but also dynamically adjust their weights using the entropy weight method. This approach ensures that the path selection process is more rational and efficient.

4.6.2. Link Weight Coefficients Based Shortest Path Decision

Based on the computed link weight coefficients, the objective of the shortest path decision is to identify the path with the minimum total link weight coefficient W_i for a given link set $Link(u \to v)$ between node pair (u, v). This can be expressed as:

$$F_{\text{obj}}^{(u \to v)} = \min_{u,v \in V_P, i \in Link(u \to v)} \sum W_i, \tag{26}$$

where u and v ($u \neq v$) are two distinct nodes in the network topology. The objective is to find the shortest/optimal path between u and v among all possible paths.

Since Dijkstra's algorithm has been proven to possess universal optimality [29,36], the model employs a Dijkstra-based dynamic routing decision algorithm, referred to as DGSFN-DR, shown in Algorithm 1, for determining the dynamic shortest/optimal path.

5. Experiments

5.1. Experimental Environment and Configuration Description

We evaluate the DG-SFN prediction model and the DGSFN-DR dynamic routing optimization algorithm. We simulate network traffic on two classical topologies. We select NSFNet [37] and NYC [38] as the foundational network structures for our experiments. The objective is to evaluate the network performance through long-term time-series simulation. During the simulation, a combination of the gravity model and traffic classification methods (Mice flows and Elephant flows) [37] was employed. To obtain network state data, multiple traffic intensities set at 10%, 50%, 70%, and 90% were tested.

In our evaluation, we focus on the routing task defined by our framework: forecasting link-state metrics at $t + \Delta t$, mapping them to link weights via $\theta(\cdot)$, and then performing dynamic shortest-path routing using the induced predicted weights (Equations (3) and (4) and Algorithm 1). Therefore, rather than comparing against heterogeneous routing policies that operate under different decision interfaces, we conduct controlled comparisons at the predictor-to-routing interface by integrating various spatiotemporal predictors under the same routing pipeline. This design ensures that the reported improvements reflect the ability to produce routing-relevant link-weight predictions for QoS-driven path selection, which directly matches our SDN routing objective (Tables 1 and 2).

The descriptions of the network topologies are as follows.

1. NSFNet Topology [37]: This topology consists of 14 nodes and 21 links derived from the U.S. National Science Foundation Network's framework. It is commonly utilized to simulate efficient data transmission. The topology structure is shown in Figure 4.
2. NYC Topology [38,39]: This topology comprises 14 nodes and 28 links and is designed to emulate the high-connectivity structure commonly observed in enterprise networks and data centers. The topology structure is presented in Figure 5.

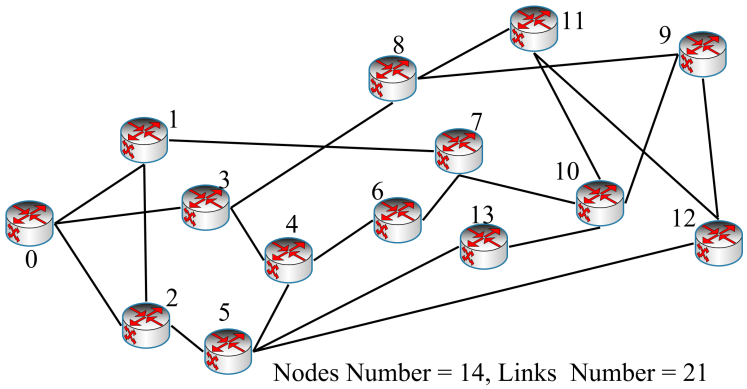


Figure 4. NSFNet Networks Topology.

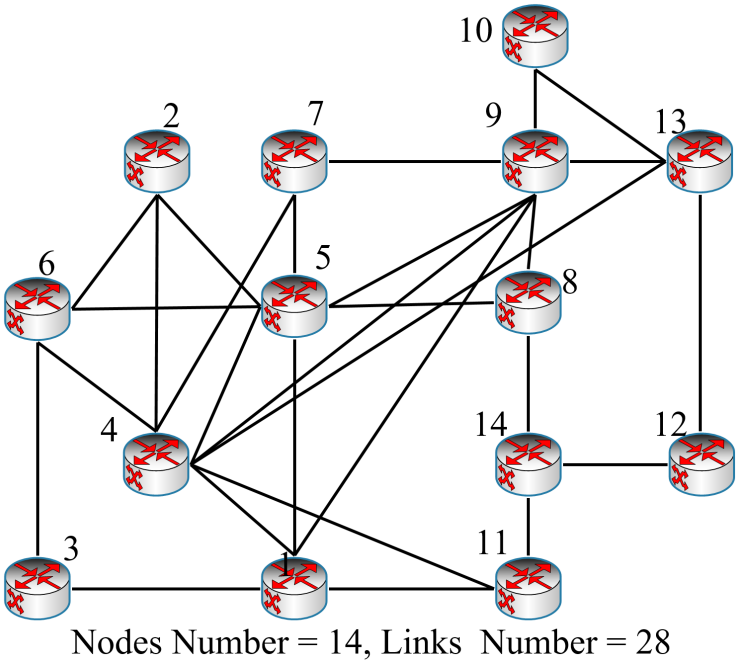


Figure 5. NYC Networks Topology.

The detailed experimental data and parameter settings are summarized in Table 3.

Table 3. Simulation Environment Parameter Settings.

Metric/Parameter	NYC	NSFNet
Number of nodes	14	14
Number of links	28	21
Maximum bandwidth (Mbps)	100	100
Minimum bandwidth (Mbps)	10	10
Minimum delay (ms)	10	10
Maximum delay (ms)	120	120
Mice flows	10	15
Elephant flows	8	10

Traffic Matrix Generation Rules: In the experiments, traffic matrices with varying traffic intensities (TI) were generated using a gravity model. The traffic intensities were set to occupy 10%, 50%, 70%, and 90% of the network bandwidth, respectively. For each TI level, multiple traffic matrices were generated. The gravity model is defined as:

$$TM_{s,d} = TI \times \left(\frac{C_s \times C_d}{N_e} \right) \times \mu(0.1, 1), \tag{27}$$

where $TM_{s,d}$ represents the traffic intensity between the node pair (s, d); $TI \in 10\%, 50\%, 70\%, 90\%$ denotes the selected traffic intensity level, randomly chosen among the four levels; C_s and C_d are the degrees of nodes s and d, respectively; $N_e = n_L$ is the total number of links in the network; and $\mu(0.1, 1)$ is a random perturbation factor uniformly distributed in the range $[0.1, 1]$.

The Google Trace dataset [40] was used as a basis for task arrival rates, serving as seeds for both temporal and task intensity simulations. This approach enables the generation of network traffic data that more closely approximates real-world network conditions.

Traffic Matrix Generation for Mice and Elephant Flows: When the node pair (s, d) corresponds to either Mice flows or Elephant flows, the traffic matrix generation formulas are defined as follows:

$$TM_{s,d} = \alpha \times BW_{\max}, \tag{28}$$

$$TM_{s,d} = \beta \times BW_{\max}, \tag{29}$$

where $BW_{\max}$ represents the maximum bandwidth of the link. The parameter $\alpha \in [0.01, 0.05]$ is the scaling factor for Mice flows, and the generated $TM_{s,d}$ is determined by a randomly selected value of $\alpha \times BW_{\max}$. Similarly, $\beta \in [0.1, 0.5]$ is the scaling factor for Elephant flows, and the generated $TM_{s,d}$ is determined by a randomly selected value of $\beta \times BW_{\max}$.

5.2. Datasets and Prediction Model Parameters

5.2.1. Datasets

The core of the DG-SFP framework is its prediction model, DG-SFN. To obtain the training data for DG-SFN, we utilized the Mininet simulator in an environment supported by the ONOS controller. The NSFNet and NYC network topologies, along with their associated data, were employed in the experiments. Following the experimental configurations in Table 3, network state data—including node states and link states—were collected over 34,560 consecutive time steps (with each time step lasting 5 s, corresponding to approximately 48 h of sequential data) for both network topologies.

The normalized data set was structured so that each time step corresponded to one instance. Based on the total number of time steps in each dataset, the samples were divided into training, testing, and validation sets in a ratio of 7:2:1. This resulted in 24,292 training cases, 6912 testing cases, and 3456 validation cases. Dataset split protocol: All datasets are split based on temporal order to avoid information leakage. Specifically, the earliest segments are used for training, the subsequent segments for validation, and the latest segments for testing.

5.2.2. Prediction Model Parameters

The experimental environment was configured as follows: the CPU used was an Intel Xeon(R) Platinum 8255C, and the GPU was an NVIDIA RTX V100 with 24 GB of memory. The operating system was Ubuntu. The software environment included Python 3.8, with the deep learning framework PyTorch 1.9.0 and TorchVision 0.10.0. GPU acceleration was enabled using CUDA 11.0.

The parameter settings for the DG-SFN model are summarized in Table 4.

Table 4. Model Parameter Configurations of DG-SFN.

Parameter	Description	Value
MLP(f_1/f_2)	Feature embedding layer for time slots	MLP(4, 128, 64/128)
K	Number of attention heads	4/8
d_1	Node representation embedding dimension	64/128
L_{GAT}	Number of stacked GAT layers	4
MLP(f_{out})	Prediction output layer	MLP(128, 128/256, 4)
m	Number of input features to the model	4
γ_j	Optimization weight for the j-th link feature	0.25
l_r	Learning rate of the model	0.001–0.01
t	Input time steps	12/24 (1 m/2 m)
T	Predicted future time steps	1/2 (5 s/10 s)
B_{size}	Batch size	16/32

Performance Evaluation Metrics: (1) Mean Absolute Error (MAE); (2) Root Mean Squared Error (RMSE); (3) R-squared (R^2); and (4) Pearson Coefficient (R_p). The corresponding formulas are defined as follows:

$$\text{MAE} = \frac{1}{N_S}\sum_{i=1}^{N_S}\left|Y_i - \hat{Y}_i\right|, \tag{30}$$

$$\text{RMSE} = \sqrt{\frac{1}{N_S}\sum_{i=1}^{N_S}\left(Y_i - \hat{Y}_i\right)^2}, \tag{31}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N_S}\left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{N_S}\left(Y_i - \frac{1}{N_S}\sum_{i=1}^{N_S} Y_i\right)^2}, \tag{32}$$

$$\begin{aligned} R_p \;=\; & \frac{N_S\sum_{i=1}^{N_S}\left(Y_i \cdot \hat{Y}_i\right) - \sum_{i=1}^{N_S} Y_i \sum_{i=1}^{N_S} \hat{Y}_i}{\sqrt{N_S\sum_{i=1}^{N_S} Y_i^2 - \left(\sum_{i=1}^{N_S} Y_i\right)^2}} \\ & \times \frac{1}{\sqrt{N_S\sum_{i=1}^{N_S} \hat{Y}_i^2 - \left(\sum_{i=1}^{N_S} \hat{Y}_i\right)^2}}. \end{aligned} \tag{33}$$

where N_S denotes the total number of observable samples, Y_i represents the true value of the i-th sample, and $\hat{Y}_i$ represents the predicted value of the i-th sample.

5.2.3. Fairness of Baseline Comparisons

We acknowledge that fair baseline comparison is critical for interpreting the percentage improvements reported in this paper. In this study, all compared baselines—FC-LSTM, GRU, Transformer, STFAGN, and GL-STGTN—are evaluated under a unified experimental protocol with respect to data split, feature set, prediction horizon, and routing-decision pipeline.

(1) Same data split.

For all methods, the dataset is partitioned into training, validation, and testing sets using the same ratio of 7:2:1, where each time step corresponds to one instance extracted from the collected network state time series. No method uses additional data beyond this shared split.

(2) Same input feature set and normalization.

We use the same underlying link-state variables for prediction (latency, jitter, packet loss rate (PLR), and available bandwidth (ABW)) and construct the input samples using identical temporal windows. All models receive inputs processed with the same normalization/standardization scheme as specified in Section 4.6.1. For graph-based baselines that require graph structure (e.g., STFAGN and GL-STGTN), the same topology graphs are employed to build their graph inputs; however, the subsequent routing stage remains identical for all methods (see below).

(3) Same prediction horizon.

The prediction horizon Δt is fixed across all methods. Specifically, we predict future link-state metrics at the same time offset as used by DG-SFN (e.g., $T = 1/2$ with the corresponding time-step duration in Table 4). Thus, differences in routing performance cannot be attributed to mismatched forecast horizons.

(4) Same routing-decision pipeline from predicted link states to paths.

Crucially, all methods—including DG-SFN-DR and the baselines—follow the same routing pipeline. Let the predicted future link-state metrics be denoted by $\hat{S}(t + \Delta t)$. We compute link weight coefficients using the same mapping function $\theta(\cdot)$ as in Equation (3) and then select routes by minimizing the accumulated predicted path weight using the same Dijkstra-based shortest-path decision procedure (Equation (4) and Algorithm 3). Therefore, the only difference across methods lies in the prediction model generating $\hat{S}(t + \Delta t)$;the routing rule and optimization objective are identical. The Fairness protocol for baseline comparisons is summarized in Table 5.

Table 5. Fairness protocol for baseline comparisons.

Method	Data Split	Feature Set	Prediction Horizon	Routing Pipeline
FC-LSTM	7:2:1	Latency/Jitter/PLR/ABW	$T = 1/2$	$\hat{S}(t+\Delta t) \xrightarrow{\theta(\cdot)} W_{ij}(t+\Delta t)$, Equation (3) + Equation (4) + Algorithm 3
GRU	7:2:1	Latency/Jitter/PLR/ABW	$T = 1/2$	$\hat{S}(t+\Delta t) \xrightarrow{\theta(\cdot)} W_{ij}(t+\Delta t)$, Equation (3) + Equation (4) + Algorithm 3
Transformer	7:2:1	Latency/Jitter/PLR/ABW	$T = 1/2$	$\hat{S}(t+\Delta t) \xrightarrow{\theta(\cdot)} W_{ij}(t+\Delta t)$, Equation (3) + Equation (4) + Algorithm 3
STFAGN	7:2:1	Latency/Jitter/PLR/ABW (same inputs)	$T = 1/2$	$\hat{S}(t+\Delta t) \xrightarrow{\theta(\cdot)} W_{ij}(t+\Delta t)$, Equation (3) + Equation (4) + Algorithm 3
GL-STGTN	7:2:1	Latency/Jitter/PLR/ABW (same inputs)	$T = 1/2$	$\hat{S}(t+\Delta t) \xrightarrow{\theta(\cdot)} W_{ij}(t+\Delta t)$, Equation (3) + Equation (4) + Algorithm 3
DG-SFN-DR	7:2:1	Latency/Jitter/PLR/ABW	$T = 1/2$	$\hat{S}(t+\Delta t) \xrightarrow{\theta(\cdot)} W_{ij}(t+\Delta t)$, Equation (3) + Equation (4) + Algorithm 3

With the above unified protocol, the reported relative gains reflect differences in prediction quality and its effect on routing under the same optimization objective, rather than being artifacts of inconsistent retraining settings or heterogeneous routing pipelines.

Because the compared baselines already include a set of representative time-series sequence modeling methods (FC-LSTM, GRU, and Transformer) as well as spatiotemporal graph/time-series variants (STFAGN and GL-STGTN), the effectiveness evaluation was originally designed to examine how different prediction modules translate into routing decisions. Under the unified evaluation protocol (same data split, feature set with $m = 4$, prediction horizon $T = 2$, and the same mapping function $\theta(\cdot)$ followed by the identical Equation (4) Dijkstra routing pipeline), the reported percentage gains reflect differences in predictive modeling capacity rather than discrepancies in experimental settings. Therefore, we do not introduce new ablation experiments solely for fairness verification; instead, we provide the explicit fairness constraints and pipeline alignment summarized in Table 5.

5.3. Ablation Studies on Dual-Graph Construction for Routing Optimization

To verify whether the reported gains are mainly attributable to the proposed dual-graph spatiotemporal fusion (rather than simply using a graph-based predictor), we perform an ablation that isolates the effects of graph construction and the feature fusion/alignment strategy. In all variants, the spatiotemporal forecasting backbone remains the same (a GAT+RNN), and the routing stage follows the same dynamic shortest-path formulation where link weights are computed from predicted link-state metrics at $t + \Delta t$. Accordingly, the only differences among variants lie in the graph construction and the feature alignment/fusion policy used for encoding.

We consider four variants: (i) GP-only: node-domain graph GP with a GAT encoder, (ii) GL-only: line-domain graph GL with a GAT encoder, (iii) Dual-w/o Fusion: dual graphs are constructed, but the proposed alignment/fusion strategy is removed, and (iv) Full Dual (DGSFN): the complete dual-graph spatiotemporal fusion model. The routing decision is made by minimizing the accumulated predicted path weight. In this setting, performance degradation can arise not only from marginal prediction errors but also from inconsistencies in the predicted link weights along a candidate path, which may lead to suboptimal path selection.

The dual-graph design provides complementary structural views of link-state evolution. GP captures node-centric spatial dependencies, while GL is constructed from edge adjacency and captures shared-endpoint coupling among links, which characterizes how congestion propagation and load/queue dynamics at intermediate switches may affect multiple traversed links. Therefore, improving routing performance requires the predictor to generate link-weight patterns that are more consistent with the path-level optimization objective, not only to reduce standalone prediction error.

The ablation results below quantify the contributions of (a) using dual graphs versus a single graph view, and (b) applying the fusion/alignment strategy versus naive dual encoding.

5.3.1. Prediction Accuracy and Correlation Effects from Dual-Graph Construction

Tables 6 and 7 report the prediction metrics on four link-state features (Latency, Jitter, Packet Loss Rate (PLR), and Available Bandwidth (ABW)) under the four variants. Across both topologies, the Full Dual model attains the strongest regression quality, indicating that the dual-graph fusion improves both error magnitude and predictive alignment with the true dynamics.

On NSFNET, Full Dual yields the best MAE/RMSE for all four features: Latency improves from Dual-w/o Fusion (2.8905 MAE, 5.1115 RMSE) to 2.7530 MAE and 4.9644 RMSE; Jitter improves from 1.2073 MAE and 2.2455 RMSE to 1.1537 MAE and 2.1219 RMSE; PLR decreases from 0.0012 MAE/0.0022 RMSE to 0.0011 MAE/0.0022 RMSE; ABW improves from 2.4665 MAE/4.7341 RMSE to 2.3571 MAE/4.6508 RMSE. The correlation indicators follow the same trend: Full Dual achieves Pearson $= 0.7840$ with $R^2 = 0.6037$ for Latency, and reaches Pearson $= 0.9232$ with $R^2 = 0.8811$ for ABW. Similar dominance holds on NYC: Full Dual reduces Latency MAE from 3.4495 (Dual-w/o Fusion) to 3.1990 while increasing correlation (Pearson 0.8037 vs. 0.7975), and improves ABW to 2.8225 MAE/5.1920 RMSE with Pearson 0.9329.

These results already suggest that the dual-graph fusion is doing more than aggregating information from a single view: it improves not only the magnitude of prediction errors but also the consistency between predicted trajectories and observed link-state evolution (reflected by R^2 and Pearson correlation).

Table 6. Statistical Metrics of Features under Dual-Graph Ablation on the NSFNET Topology.

Model	Feature	MAE↓	RMSE↓	R^2 ↑	Pearson↑
GP-only	Latency↓	3.2062	5.4075	0.5682	0.7507
	Jitter↓	1.3112	2.3428	0.5862	0.7544
	PLR↓	0.0013	0.0024	0.8563	0.8988
	ABW↑	2.5736	5.0791	0.8445	0.8992
GL-only	Latency↓	3.3081	5.4198	0.5601	0.7496
	Jitter↓	1.3254	2.3975	0.5707	0.7501
	PLR↓	0.0013	0.0024	0.8493	0.8972
	ABW↑	2.6022	5.1077	0.8335	0.8879
Dual-w/o Fusion	Latency↓	2.8905	5.1115	0.5839	0.7655
	Jitter↓	1.2073	2.2455	0.6085	0.7683
	PLR↓	0.0012	0.0022	0.8770	0.9098
	ABW↑	2.4665	4.7341	0.8713	0.9112
Full Dual	Latency↓	**2.7530**	**4.9644**	**0.6037**	**0.7840**
	Jitter↓	**1.1537**	**2.1219**	**0.6157**	**0.7768**
	PLR↓	**0.0011**	**0.0022**	**0.8887**	**0.9217**
	ABW↑	**2.3571**	**4.6508**	**0.8811**	**0.9232**

Table 7. Statistical Metrics of Features under Dual-Graph Ablation on the NYC Topology.

Model	Feature	MAE↓	RMSE↓	R^2 ↑	Pearson↑
GP-only	Latency↓	3.6002	6.3793	0.5808	0.7691
	Jitter↓	1.4855	2.6653	0.6005	0.7695
	PLR↓	0.0013	0.0024	0.8792	0.9105
	ABW↑	3.0255	5.5022	0.8669	0.9025
GL-only	Latency↓	3.6115	6.3902	0.5722	0.7608
	Jitter↓	1.4980	2.6827	0.5919	0.7603
	PLR↓	0.0013	0.0024	0.8710	0.9001
	ABW↑	3.0990	5.5767	0.8578	0.8913
Dual-w/o Fusion	Latency↓	3.4495	6.1720	0.6091	0.7975
	Jitter↓	1.4004	2.4109	0.6301	0.7865
	PLR↓	0.0013	0.0023	0.8837	0.9278
	ABW↑	2.9000	5.2816	0.8901	0.9300
Full Dual	Latency↓	**3.1990**	**5.9507**	**0.6274**	**0.8037**
	Jitter↓	**1.3326**	**2.3961**	**0.6434**	**0.8047**
	PLR↓	**0.0013**	**0.0023**	**0.8891**	**0.9314**
	ABW↑	**2.8225**	**5.1920**	**0.8934**	**0.9329**

5.3.2. Routing-Specific Benefit: Linking Dual-Graph Improvements to QoS

A key point is that the routing decision in our DGSFN-DR does not directly access future ground-truth link metrics. Instead, at each time step, DG-SFN predicts the future link-state: $S(t+\Delta t) = f(\cdot)$ (Equation (2)), and the routing layer deterministically derives the link weight coefficients from the predicted future states: $W_{ij}(t+\Delta t) = \theta(S(t+\Delta t))$ (Equation (3)). The shortest-path objective is then computed by minimizing the *accumulated predicted path weight* $\sum_{(i,j)\in P} W_{ij}(t+\Delta t)$ (Equation (4)).

Therefore, in the dual-graph ablations (GP-only/GL-only/Dual-w/o Fusion/Full Dual), the routing mechanism remains the same, and the only varying component is the quality/structure of the predicted link states, which induces different predicted link-weight patterns. This enables a clean separation in the end-to-end analysis: (1) Tables 6 and 7 quantify how dual-graph fusion improves prediction accuracy and correlation, (2) Tables 1 and 2) show that, under the same routing formulation that consumes predicted weights, the

downstream QoS improves accordingly, and (3) Figures 6 and 7 further validate that the predicted-weight-driven paths exhibit stronger alignment with the true optimal path tendency via path-level statistics (TPW/path selection accuracy).

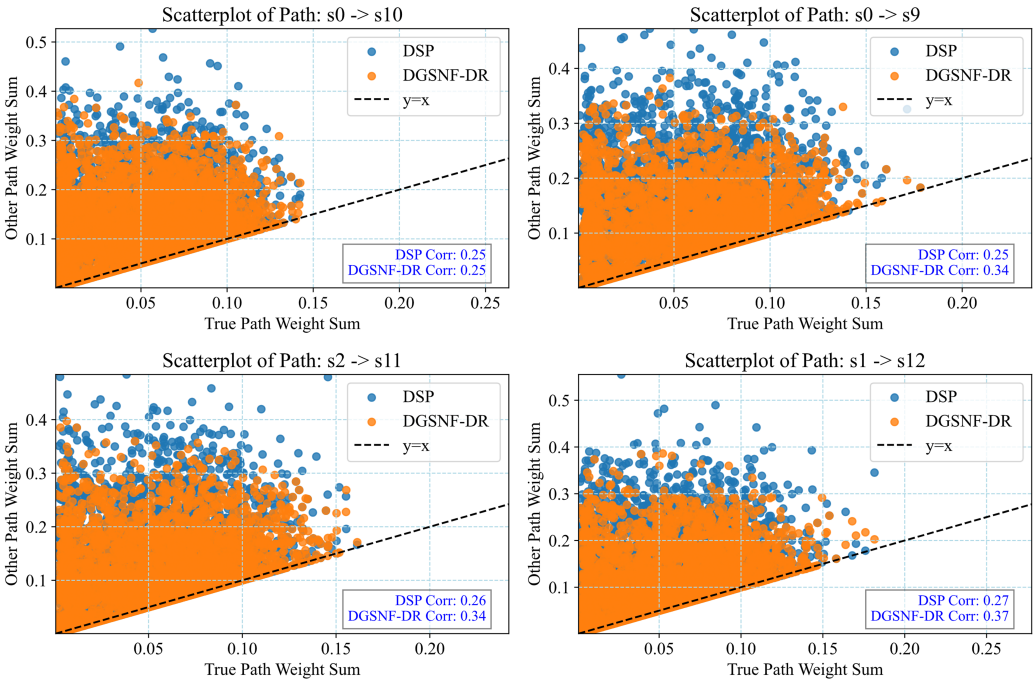


Figure 6. Scatter of Total Path Weight (TPW) Across Four Different Paths in NSFNET.

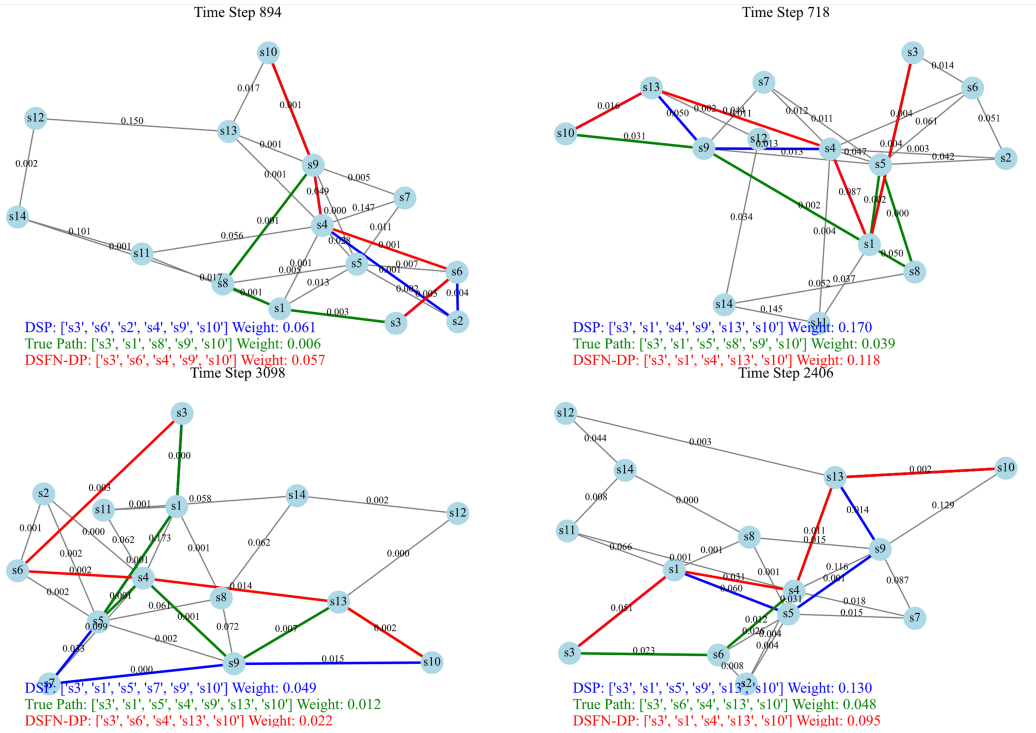


Figure 7. Specific Example of Shortest Path Routing for s3 → s10 Over Four Time Steps.

While improved prediction accuracy is encouraging, routing optimization requires that predicted link weights lead to better path selection. Tables 1 and 2 provide downstream evidence by reporting routing QoS under increasing traffic intensities (LOW/MID/HIGH). The results show a consistent advantage for Full Dual across all metrics and traffic levels,

with the largest relative improvements appearing under HIGH traffic, where the system is more sensitive to prediction-induced bias.

Importantly, the routing stage always computes paths by minimizing the predicted link weights $W_{ij}(t + \Delta t)$ from DG-SFN outputs (Equations (3) and (4)). Thus, the observed QoS gains in Tables 1 and 2 are not attributable to changing the routing rule, but to the improved prediction-induced weight patterns produced by dual-graph fusion. To further connect prediction quality with routing effectiveness, Figures 6 and 7 report path-level statistics showing that the selected paths under predicted weights are more correlated with the true optimal tendency than those under DSP.

On NSFNET under HIGH traffic (60%), Full Dual achieves Latency 28.914 ms, reducing it from DSP 31.575 ms and Dual-w/o Fusion 29.155 ms; it also reduces Jitter from 9.757 ms (DSP) and 9.104 ms (Dual-w/o Fusion) to 8.968 ms. The packet loss decreases from 0.1138% (DSP) and 0.1088% (Dual-w/o Fusion) to 0.1068%, while available bandwidth increases from 35.385 Mbps (DSP) and 38.868 Mbps (Dual-w/o Fusion) to 39.158 Mbps. A similar pattern is observed on NYC under HIGH traffic: Full Dual achieves Latency 31.856 ms (vs. 35.781 ms for DSP and 31.992 ms for Dual-w/o Fusion), Jitter 10.287 ms (vs. 11.894 ms for DSP and 10.355 ms for Dual-w/o Fusion), packet loss 0.1189% (vs. 0.1390% for DSP and 0.1201% for Dual-w/o Fusion), and available bandwidth 33.684 Mbps (vs. 29.146 Mbps for DSP and 33.465 Mbps for Dual-w/o Fusion).

Crucially, these downstream improvements are not reproduced by the single-graph baselines with comparable strength. For instance, on NSFNET under HIGH traffic, Full Dual improves latency relative to GP-only (from 30.725 ms to 28.914 ms) and relative to GL-only (from 31.021 ms to 28.914 ms). This supports the premise that routing gains are not simply an artifact of using graph learning, but depend on the structural properties of the encoded dependencies and the way the two views are fused.

5.3.3. Quantifying the Specific Contribution of Dual-Graph Fusion

To visualize where the dual-graph design translates into routing advantage, Figure 8 summarizes the gain patterns of Full Dual relative to DSP and the graph variants across traffic levels, topologies, and QoS metrics. The heatmap indicates that Full Dual consistently delivers positive gains and that the improvement remains robust when traffic grows, matching the numerical trend in Tables 1 and 2. More importantly, the gaps between Full Dual and Dual-w/o Fusion demonstrate that constructing two graphs alone is insufficient; the alignment/fusion strategy is what enables the predictor to produce link-weight patterns that better serve path-level optimization.

Put together, the dual-graph contribution has two distinct aspects supported by data: (1) using both node-domain and line-domain dependency structures yields better QoS than either single-graph view alone (GP-only or GL-only), and (2) applying the proposed fusion/alignment strategy further improves both prediction quality (Tables 6 and 7) and routing QoS (Tables 1 and 2), with the most pronounced benefits under HIGH traffic.

5.3.4. Module-Level Importance Within DGSFN-DR

The ablation study also clarifies why the overall architecture is necessary for routing optimization. The forecasting backbone (GAT+RNN) provides temporal prediction capacity, but the effectiveness of routing hinges on how predicted link states are transformed into a coherent set of link weights. The dual-graph spatiotemporal fusion module supplies the structural inductive bias required to model inter-link coupling around shared endpoints and nodes. Finally, the routing stage converts the predicted link-state metrics into a dynamic path decision by minimizing the accumulated predicted path weight. Therefore, the architecture components are complementary: the forecasting module alone improves link-state

estimates, while the dual-graph fusion specifically enhances the path-relevant consistency of predicted weights, which is reflected directly in downstream latency/jitter/packet loss and available bandwidth improvements.

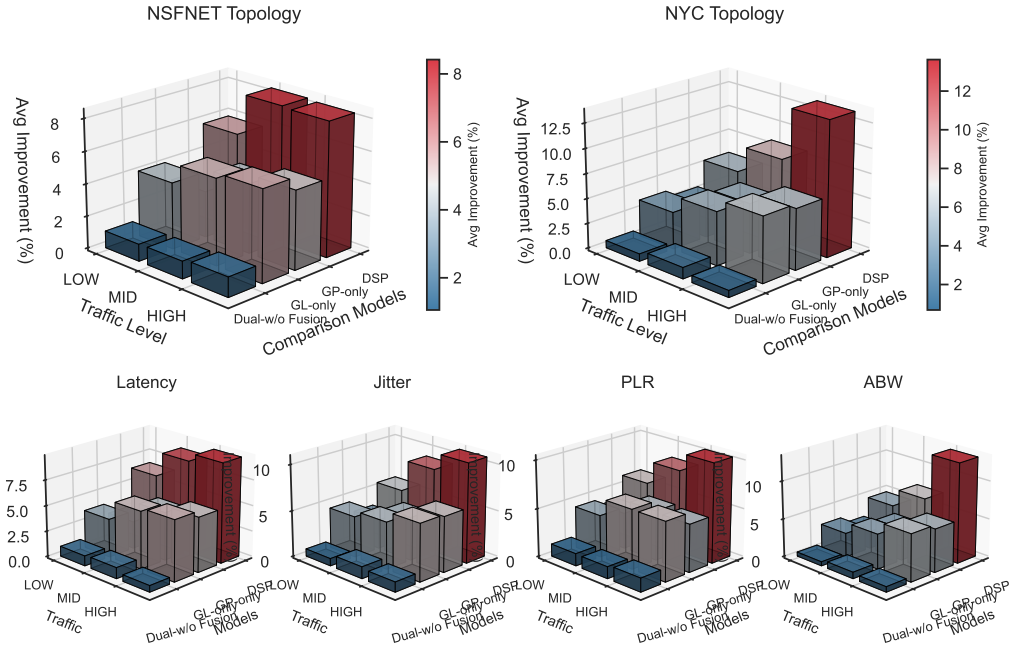


Figure 8. Routing Ablation Heatmap: Full Dual Gains (%) vs. DSP/Graph Variants.

The novelty of DGSFN-DR is not limited to combining commonly used components (GATs, RNNs, and Dijkstra). Instead, it lies in the dual-graph spatiotemporal fusion mechanism designed to improve prediction-to-weight coupling for routing optimization. Concretely, we construct two complementary structural views: the node-domain physical topology graph G_P and the line-domain edge adjacency (line graph) G_L. More importantly, we introduce an explicit feature alignment/fusion strategy (Equation (15)) that maps link-domain representations back to a coherent, fused node-centric space, enabling consistent modeling of inter-link coupling around shared endpoints. This design goal is distinct from single-graph predictors, where spatial dependencies are learned from only one structural view.

To convincingly demonstrate the benefit of dual-graph modeling over single-graph and other GNN-based time-series predictors, we perform dual-graph ablation with four variants (GP-only, GL-only, Dual-w/o Fusion, Full Dual). The routing decision procedure remains identical across variants and consumes only the predicted link states through the same mapping $\theta(\cdot)$ and Dijkstra-based shortest path objective (Equations (3) and (4) and Algorithm 1). Therefore, performance differences can be attributed to how the prediction module produces link-weight patterns. The results show that Full Dual consistently outperforms both single-graph variants and Dual-w/o Fusion, indicating that (i) using both G_P and G_L yields a stronger inductive bias for spatiotemporal link evolution, and (ii) the proposed alignment/fusion is essential to obtain path-relevant coherent link-weight trajectories.

5.4. Analysis of Experimental Data for the Prediction Model

The performance of the proposed link state prediction model, DG-SFN, was evaluated on three datasets. The results were compared with those obtained by FC-LSTM [41], GRU [42], Transformer [43], STFAGN [24], and GL-STGTN [44]. The detailed results are presented in Tables 8–11.

Table 8. Statistical Metrics of Features on the NSFNET Network Topology.

Model	Feature	MAE	RMSE	R^2	Pearson
LSTM	Latency	3.5110	6.2980	0.5333	0.7430
	Jitter	1.4481	2.6090	0.5459	0.7474
	PLR	0.0013	0.0026	0.8019	0.8976
	ABW	2.9362	5.7187	0.7848	0.8917
GRU	Latency	3.5111	6.2837	0.5354	0.7440
	Jitter	1.4490	2.5985	0.5496	0.7495
	PLR	0.0014	0.0026	0.8085	0.9009
	ABW	2.9339	5.5804	0.7950	0.8961
Transformer	Latency	3.7081	6.9574	0.4304	0.6690
	Jitter	1.5411	2.8682	0.4512	0.6795
	PLR	0.0017	0.0030	0.7328	0.8574
	ABW	5.0329	8.1283	0.5651	0.7972
STFAGN	Latency	3.0439	5.3954	0.5726	0.7656
	Jitter	1.2585	2.3116	0.5842	0.7609
	PLR	0.0012	0.0023	0.8579	0.9068
	ABW	2.5920	5.1441	0.8438	0.9015
GL-STGTN	Latency	2.9454	5.3237	0.5802	0.7553
	Jitter	1.2234	2.2977	0.6059	0.7664
	PLR	0.0012	0.0022	0.8606	0.9078
	ABW	2.5510	4.7730	0.8529	0.9073
DG-SFN	Latency	**2.7530**	**4.9644**	**0.6037**	**0.7840**
	Jitter	**1.1537**	**2.1219**	**0.6157**	**0.7768**
	PLR	**0.0011**	**0.0022**	**0.8887**	**0.9217**
	ABW	**2.3571**	**4.6508**	**0.8811**	**0.9232**

Table 9. Overall Performance Results of Different Models on the NSFNET Network Topology.

Model	MAE	RMSE	R^2	Pearson
LSTM	1.9742	3.6571	0.6665	0.8199
GRU	1.9738	3.6163	0.6721	0.8227
Transformer	2.5710	4.4892	0.5449	0.7508
STFAGN	1.7598	3.2462	0.7401	0.8596
GL-STGTN	1.7346	3.1844	0.7587	0.8615
DG-SFN	**1.6607**	**2.9911**	**0.8419**	**0.8830**

Table 10. Experimental Results on the NYC Network Topology.

Model	Feature	MAE	RMSE	R^2	Pearson
LSTM	Latency	4.0423	7.1883	0.5373	0.7493
	Jitter	1.6653	2.9625	0.5576	0.7589
	PLR	0.0015	0.0028	0.8200	0.9062
	ABW	3.3569	6.3048	0.8091	0.9026
GRU	Latency	4.0197	7.1050	0.5482	0.7528
	Jitter	1.6569	2.9255	0.5686	0.7631
	PLR	0.0015	0.0028	0.8252	0.9098
	ABW	3.3692	6.2638	0.8116	0.9062
Transformer	Latency	4.3070	8.2961	0.3840	0.6463
	Jitter	1.7818	3.4074	0.4148	0.6623
	PLR	0.0020	0.0035	0.7415	0.8637
	ABW	5.7530	10.819	0.6378	0.7659

Table 10. *Cont.*

Model	Feature	MAE	RMSE	R^2	Pearson
STFAGN	Latency	3.5945	6.3113	0.5861	0.7711
	Jitter	1.4652	2.6071	0.6126	0.7763
	PLR	0.0014	0.0024	0.8804	0.9220
	ABW	2.9841	5.4844	0.8716	0.9153
GL-STGTN	Latency	3.5371	6.2699	0.6058	0.7887
	Jitter	1.4073	2.4843	0.6254	0.7807
	PLR	0.0013	0.0023	0.8813	0.9203
	ABW	2.9249	5.3560	0.8853	0.9276
DG-SFN	Latency	**3.1990**	**5.9507**	**0.6274**	**0.8037**
	Jitter	**1.3326**	**2.3961**	**0.6434**	**0.8047**
	PLR	**0.0013**	**0.0023**	**0.8891**	**0.9314**
	ABW	**2.8225**	**5.1920**	**0.8934**	**0.9329**

Table 11. Overall Performance Results of Different Models on the NYC Network Topology.

Model	MAE	RMSE	R^2	Pearson
LSTM	2.2492	4.0890	0.6864	0.8319
GRU	2.2776	4.0728	0.6876	0.8324
Transformer	2.9544	5.5415	0.5064	0.7382
STFAGN	2.0396	3.6614	0.7636	0.8679
GL-STGTN	2.0334	3.7031	0.7792	0.8639
DG-SFN	**1.8739**	**3.4649**	**0.8301**	**0.8962**

Based on the results presented in Tables 8 and 10, the proposed DG-SFN demonstrates significant superiority in predicting the four features across both the NSFNET and NYC network topologies. These findings validate the effectiveness of DG-SFN compared to other models. From the statistical perspective, although the prediction results for each feature are relatively close due to the minimal variation in data packet loss rates, all models exhibit strong performance on the PLR and ABW features, particularly in terms of the R^2 and Pearson correlation coefficient metrics. This indicates that these models possess robust fitting capabilities for these two features, and the predicted results show a strong and significant correlation with the actual observations.

Compared to other models, the DG-SFN model demonstrates outstanding performance across all statistical metrics, with particularly significant advantages in MAE and RMSE. For example, in Latency prediction, the MAE of DG-SFN is 2.7530 on the NSFNET topology and 3.199 on the NYC topology. On the NSFNET topology, this represents a reduction of 21.59% compared to LSTM, 21.59% compared to GRU, 25.75% compared to Transformer, 9.55% compared to STFAGN (a model with a graph neural network architecture), and 6.53% compared to GL-STGTN.

In terms of R^2 and Pearson correlation coefficient, the DG-SFN model also achieves the best results among all models, with R^2 values exceeding 0.64 and reaching a maximum close to 0.9, and Pearson values exceeding 0.78 and reaching a maximum greater than 0.93. These metrics highlight the superior ability of DG-SFN to capture data variability and reflect network states.

Overall, these results indicate that the DG-SFN model effectively captures network state variations while demonstrating excellent stability and prediction accuracy.

The overall performance comparison of the models is presented in Tables 9 and 11, while the statistical improvement of DG-SFN across different features is detailed in Figures 9–12. Based on the results of the four statistical metrics, DG-SFN demonstrates the

most significant overall improvement compared to the Transformer model. Specifically, on the NSFNET topology, the improvements are 35.41%, 33.37%, 54.51%, and 17.61% for the respective metrics. On the NYC topology, the corresponding improvements are 36.57%, 37.47%, 63.92%, and 21.40%. Compared to STFAGN and GL-STGTN, the improvement margins of DG-SFN range from 3.5% to 8.7%, further validating its superior capability in capturing network state variations.

The comprehensive statistical results suggest that the Transformer model may fail to effectively model network topological structures when processing temporal data, leading to suboptimal prediction performance. While LSTM and GRU perform well in time-series prediction, STFAGN and GL-STGTN, which incorporate graph neural network architectures, may offer better performance in network state awareness. In contrast, the DG-SFN model more effectively captures variability and correlations within the network, providing robust decision support for state awareness and dynamic routing optimization in software-defined networking (SDN).

5.5. Experimental Analysis of Routing with DGSFN-DR

Simulation experiments offer significant advantages over physical network tests. They provide precise control over traffic loads and network configurations, ensure repeatability, and lower both cost and risk. Prior research validates simulation as a reliable approach for evaluating network protocols, and our experimental setup follows rigorous standards that closely mimic real-world dynamics.

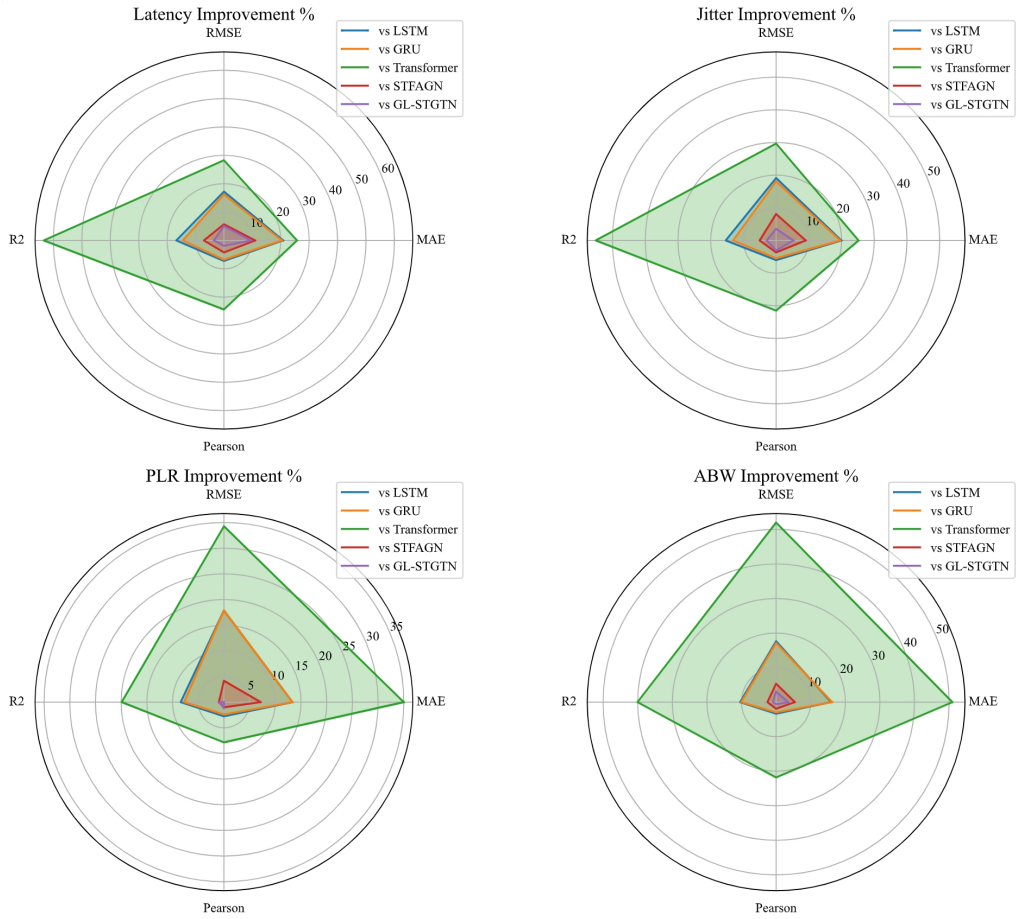


Figure 9. Performance Improvement Rates of DG-SFN Compared to Other Models for Each Feature on the NSFNET Network Topology.

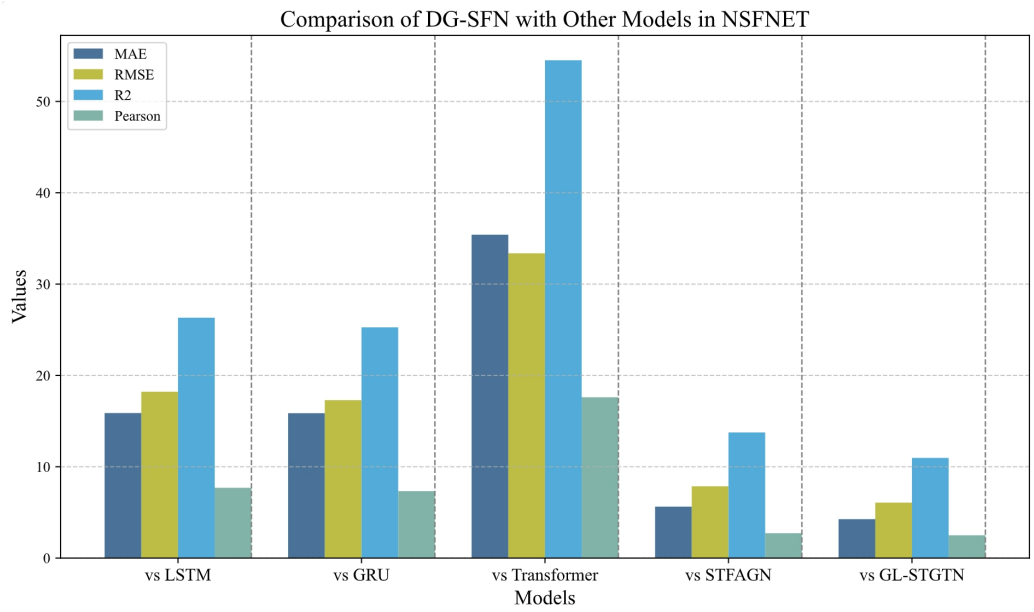


Figure 10. Overall Performance Improvement Rates of DG-SFN Compared to Other Models on the NSFNET Network Topology.

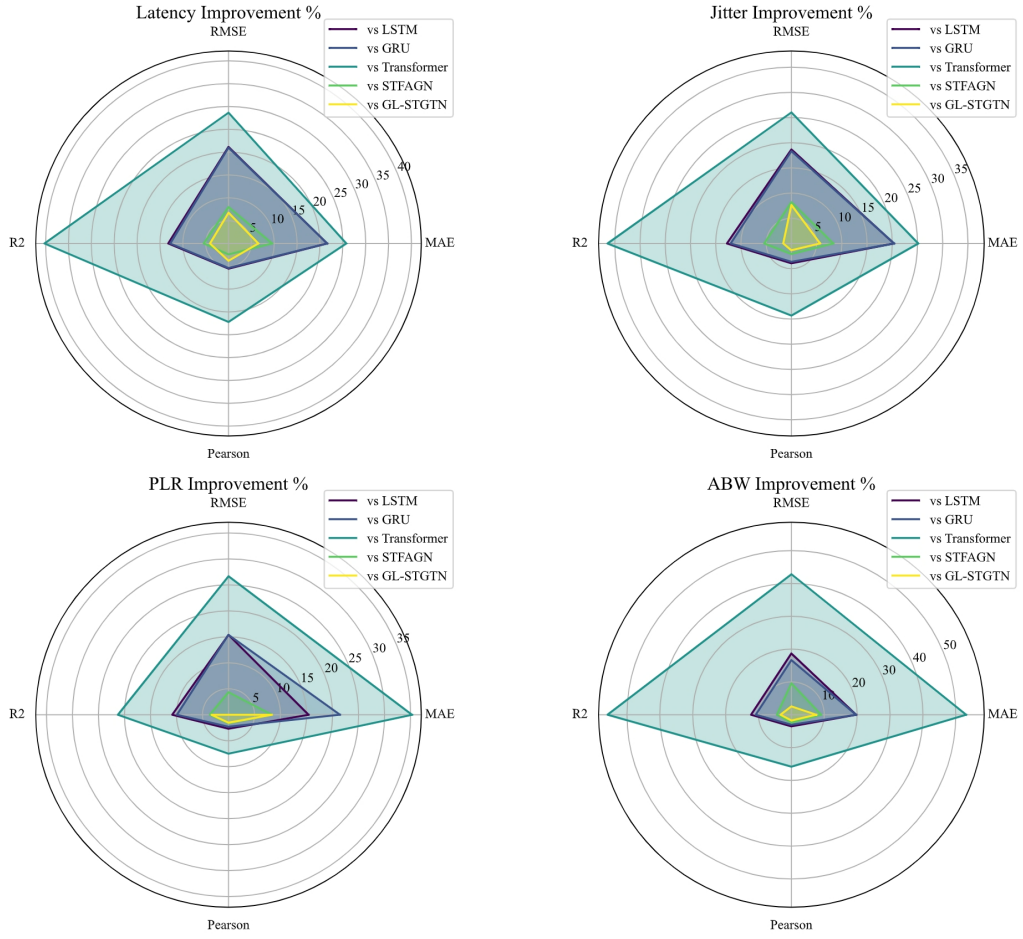


Figure 11. Performance Improvement Rates of DG-SFN Compared to Other Models for Each Feature on the NYC Network Topology.

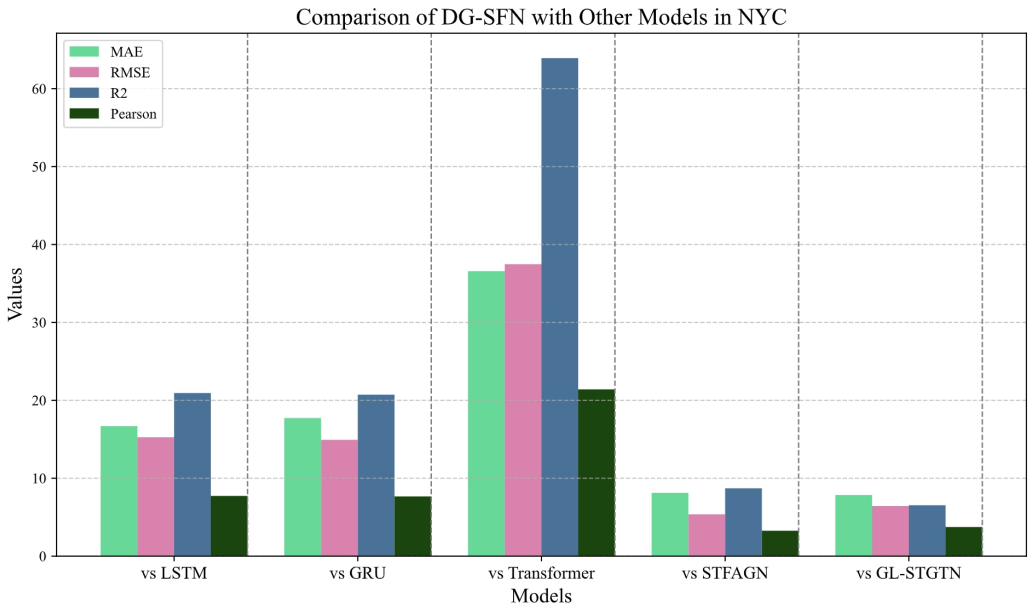


Figure 12. Overall Performance Improvement Rates of DG-SFN Compared to Other Models on the NYC Networks Topology.

We evaluate routing optimization performance of the DGSFN-DR algorithm in dynamic network environments. We perform comparative experiments on NSFNET and NYC network topologies with the Mininet simulation platform. We compare our approach with the conventional OSPF-based routing algorithm, referred to as DSP. DSP configures routing based solely on current link weights.

The DG-SFN model leverages deep learning to predict future link states by forecasting key metrics such as bandwidth, latency, and packet loss rate. Using these predictions, the DGSFN-DR algorithm constructs the shortest path between source and destination nodes. Following the experimental setup outlined in Section 4.1, we simulate two network topologies for 4 h (approximately 3450 time steps). At each time step, the DG-SFN model outputs predicted link states, and the DGSFN-DR algorithm computes the optimal path based on these predictions, while DSP calculates routes using current link weights. We acknowledge the inherent limitations of simulation experiments and plan to validate our findings in real network environments in future work.

In the comparative experiments, the performance of the algorithms was evaluated from the following perspectives:

1. Total Path Weight (TPW): This metric measures the overall cost of the path from the source to the destination.
2. Path Selection Accuracy: This evaluates the differences in path selection performance between DGSFN-DR and DSP under actual link states.

Figure 13 shows the average total weights of the shortest paths selected by DGSFN-DR for routes s0 → s10, s0 → s9, s2 → s11, and s1 → s12. We record average weights of 0.13, 0.11, 0.12, and 0.11, respectively. In contrast, DSP produces a weight of 0.14 for every path. DGSFN-DR outperforms DSP in terms of median values and standard deviations. Figure 6 shows the scatter distribution of total path weights for shortest path selection. DGSFN-DR achieves correlation coefficients of 0.25, 0.34, 0.34, and 0.37. These coefficients exceed those of DSP and indicate a stronger correlation with the true optimal paths. Figure 14 illustrates a specific example of shortest path routing for s2 → s11 over four time steps.

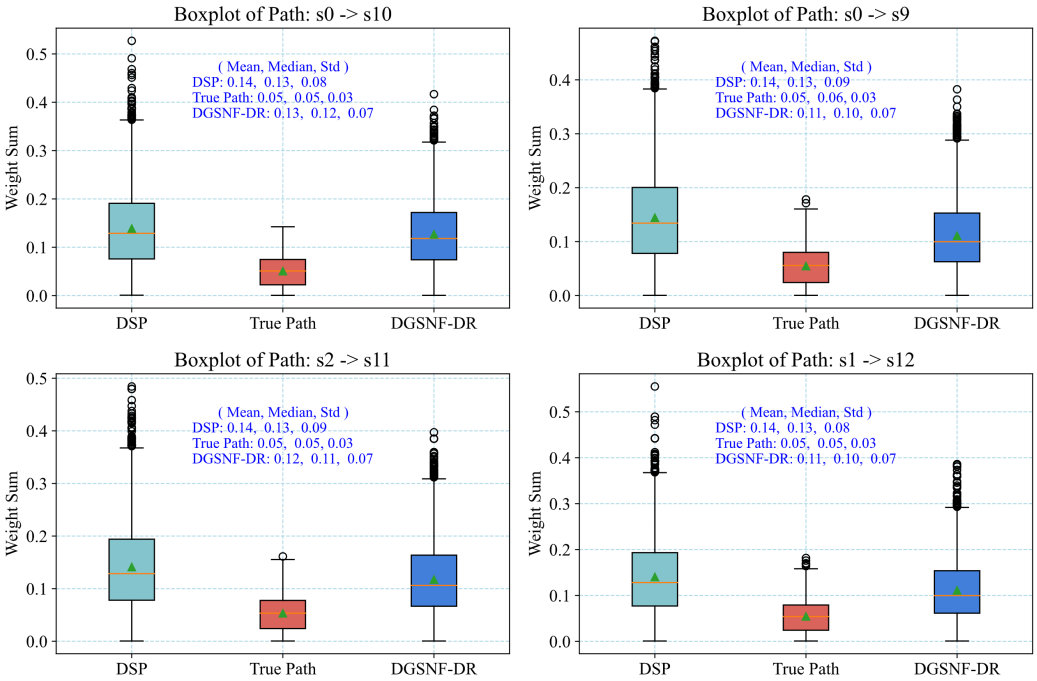


Figure 13. Boxplot of Total Path Weight (TPW) Distribution Across Four Different Paths in NSFNET.

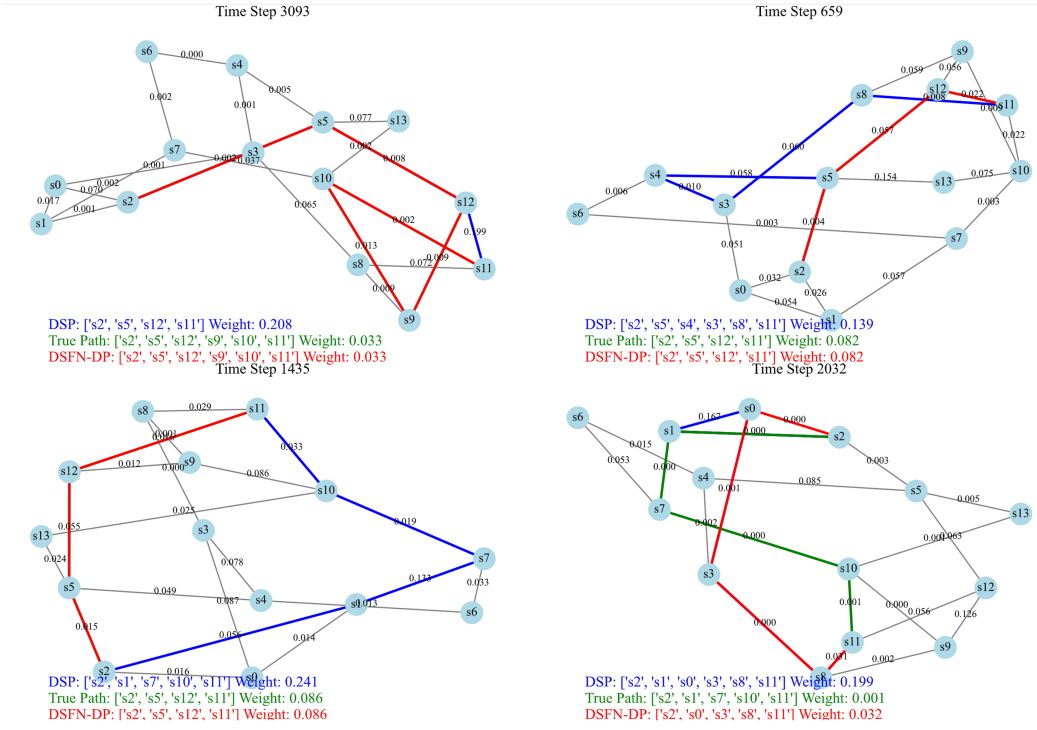


Figure 14. Specific Example of Shortest Path Routing for s2 → s11 Over Four Time Steps.

In the four-time-step routing selections presented in Figure 14, three of the time steps align with the optimal routing. Overall, on the NSFNET network topology, DGSFN-DR demonstrates better performance in path selection under actual link states compared to DSP. Similar results are observed on the NYC topology, as shown in Figures 7–15.

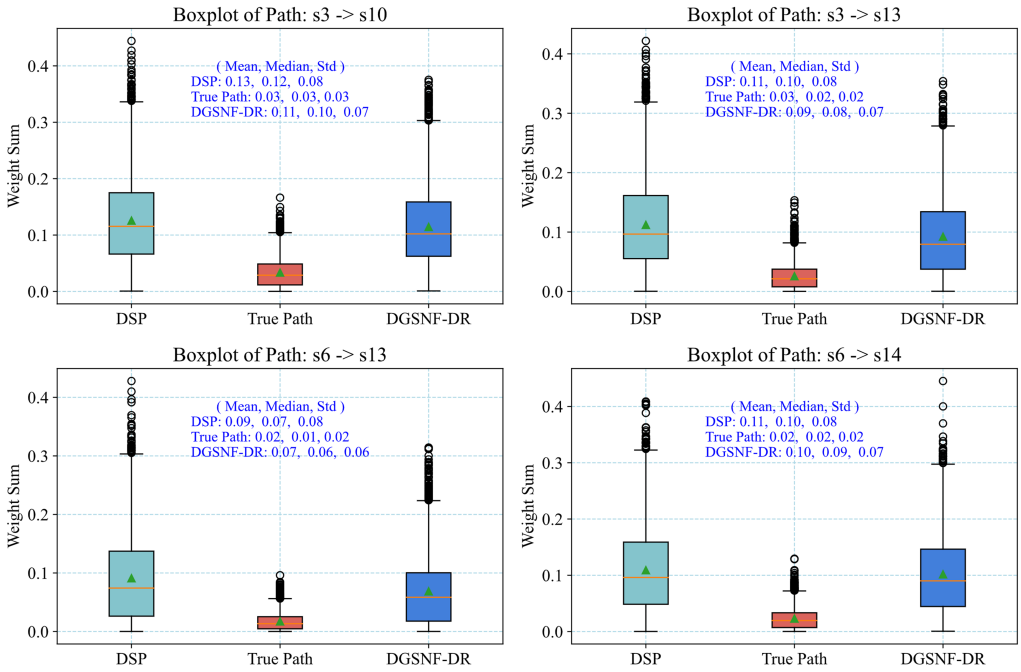


Figure 15. Boxplot of Total Path Weight (TPW) Distribution Across Four Different Paths in NYC.

The scatter plots in Figures 6 and 16 illustrate the distribution of total path weights and the corresponding correlation coefficients for several complex paths in the two network topologies. These results validate the high correlation between the DGSFN-DR algorithm and the actual optimal paths, demonstrating superior stability and accuracy.

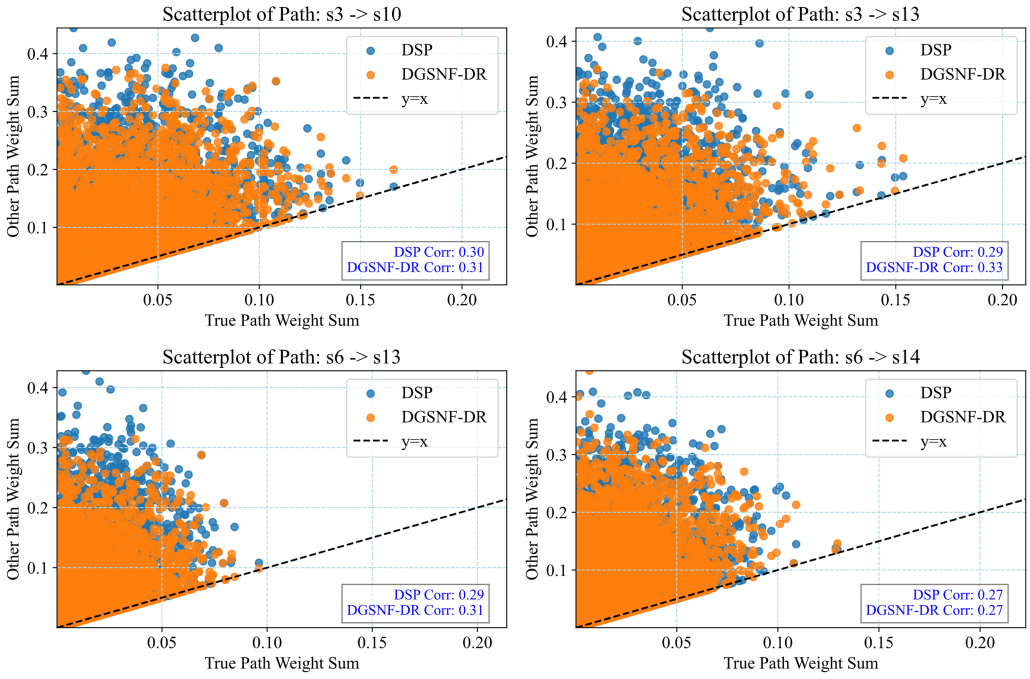


Figure 16. Scatter of Total Path Weight (TPW) Across Four Different Paths in NYC.

Figures 7 and 14 present examples of routing selections at specific time steps, providing an intuitive demonstration of the DGSFN-DR algorithm's path optimization capability in dynamic network environments. On both the NSFNET and NYC network topologies, the DGSFN-DR algorithm selects routes that align with the true optimal paths in most time steps, further confirming its practical value in dynamic routing optimization.

Figure 17 illustrates the link delays, the corresponding true link weights of the NSFNET network at a specific time step during the experiment, and the link weights utilized by the DSP and DGSFN-DR algorithms. Links with higher delays consistently exhibit larger weights, which decreases their likelihood of being selected. For example, links such as s10 → s11, s0 → s1, and s3 → s4 demonstrate higher delays and larger weights. In contrast, links such as s10 → s9 and s5 → s12 show smaller weights, indicating lower delays.

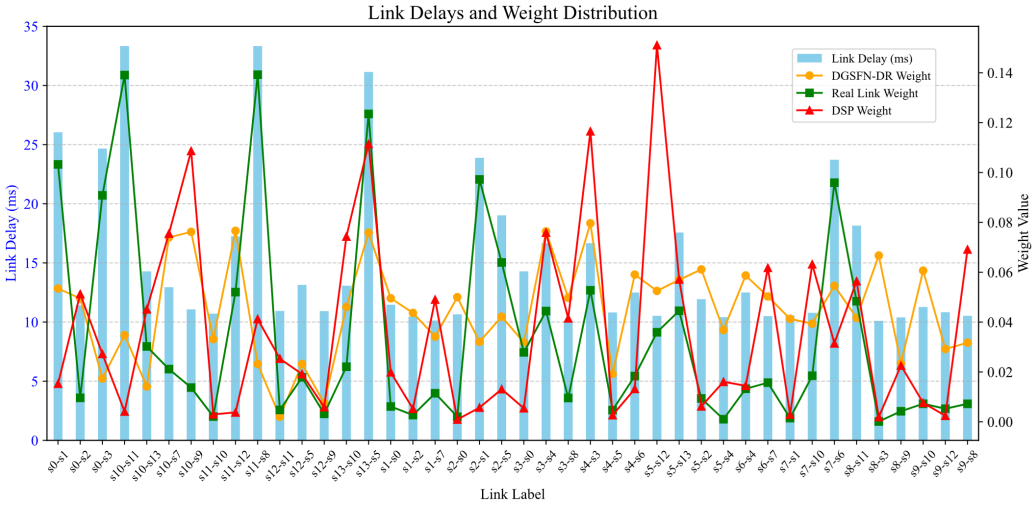


Figure 17. Link Delays and Corresponding Link Weights in the NSFNET Network.

However, regarding the selection probabilities of these links, DSP tends to select links with probabilities that are inconsistent with the actual optimal choices. Overall, the predicted link weights generated by DGSFN-DR align more closely with the trends of the true link weights, whereas the weights used by DSP deviate significantly from the true weights. As a result, DGSFN-DR is able to identify paths that are closer to the optimal ones.

In the experiment, the number of times each link was selected and its usage proportion in the shortest path was recorded for the DGSFN-DR algorithm when configuring the s0 → s9 route in the NSFNET network. The statistical results are shown in Figure 18.

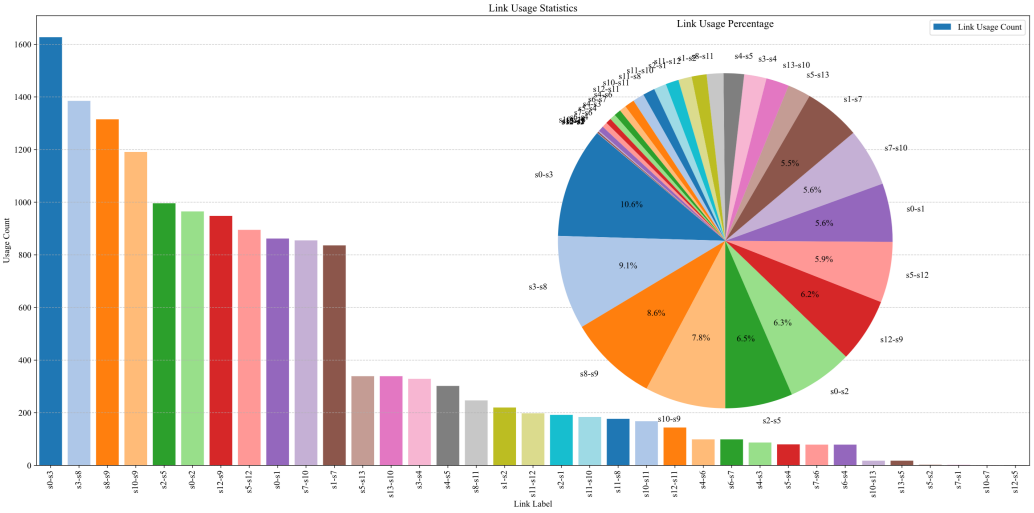


Figure 18. Link Utilization for the s0 → s9 Route Configured by the DGSFN-DR Algorithm in the NSFNET Network.

Based on Table 12 and Figure 18, it can be observed that in the computation of the s0 → s9 route using the DGSFN-DR algorithm, the most frequently utilized link is s0 → s3,

with the highest proportion of 10.6%, followed by s3 → s8 with a proportion of 9.1%. The most frequently used path is s0 → s3 → s8 → s9, accounting for approximately 30.78%.

Table 12. Statistical Results and Proportions of the Top 10 Most Frequently Used Paths for the s0 → s9 Route by the DGSFN-DR Algorithm.

Path for s0 → s9	Usage Count	Total Proportion (%)
s0 → s3 → s8 → s9	1063	30.78
s0 → s1 → s7 → s10 → s9	490	14.19
s0 → s2 → s5 → s12 → s9	416	12.04
s0 → s2 → s5 → s13 → s10 → s9	192	5.56
s0 → s3 → s4 → s5 → s12 → s9	170	4.92
s0 → s2 → s1 → s7 → s10 → s9	135	3.91
s0 → s3 → s8 → s11 → s12 → s9	118	3.42
s0 → s3 → s8 → s11 → s10 → s9	114	3.30
s0 → s1 → s2 → s5 → s12 → s9	112	3.24
s0 → s3 → s4 → s6 → s7 → s10 → s9	63	1.82

5.6. Analysis of the Application Results of Dynamic Routing with DGSFN-DR

Using the gravity model corresponding to Equation (23), traffic matrices of varying intensities were generated with TI = {20%, 40%, 60%}, representing 20%, 40%, and 60% of the network bandwidth, respectively. The DGSFN-DR and DSP algorithms were applied to these traffic matrices, and the overall network performance was evaluated. The statistical results are presented in the following Table 13.

Table 13. Network Performance Evaluation of NSFNET and NYC Under Different Traffic Loads Using DGSFN-DR.

Traffic Intensity	Metric	NSFNET (Average)			NYC (Average)		
		DSP	DGSFN-DR	Decr/Impro (%)	DSP	DGSFN-DR	Decr/Impro (%)
LOW (20%)	Latency (ms)	13.720	12.828	−6.496	14.827	13.868	−6.466
	Jitter (ms)	4.910	4.663	−5.021	5.156	4.840	−6.137
	Packet loss (%)	0.079	0.073	−6.645	0.087	0.083	−5.483
	Available bandwidth (Mbps)	80.329	83.691	4.186	79.547	83.649	5.157
MID (40%)	Latency (ms)	22.867	20.852	−8.813	24.795	22.564	−8.998
	Jitter (ms)	6.749	6.125	−9.245	6.939	6.329	−8.804
	Packet loss (%)	0.098	0.0897	−8.251	0.106	0.097	−8.422
	Available bandwidth (Mbps)	61.568	65.974	7.157	59.734	63.894	6.964
HIGH (60%)	Latency (ms)	31.575	28.914	−8.427	35.781	31.856	−10.969
	Jitter (ms)	9.757	8.968	−8.078	11.894	10.287	−13.516
	Packet loss (%)	0.1138	0.1068	−6.2024	0.1390	0.1189	−14.403
	Available bandwidth (Mbps)	35.385	39.158	10.663	29.146	33.684	15.569

Table 13 presents the experimental results comparing the prediction-based DGSFN-DR algorithm with the DSP algorithm across two network topologies. The DGSFN-DR algorithm consistently achieves better performance than DSP under all traffic intensities. For latency, DGSFN-DR reduces the average latency by over 6% across all traffic levels, with reductions of 8.427% and 10.969% observed under high-traffic conditions in the NSFNET and NYC topologies, respectively. For jitter, DGSFN-DR achieves significant improvements, reducing jitter by over 5% across all traffic intensities and achieving a 13.516% reduction under high-traffic conditions in the NYC topology. Packet loss and available bandwidth also show improvements, with performance gains ranging from 4% to 15%.

Under high-load conditions across various network topologies, the DGSFN-DR algorithm consistently achieves superior adaptability and optimization compared to DSP. In the smaller NSFNET topology, DGSFN-DR delivers moderate performance improve-

ments under low traffic conditions, with more pronounced gains observed at medium and high traffic levels. In contrast, the more complex NYC topology benefits significantly from DGSFN-DR, which achieves substantial reductions in latency, improved bandwidth utilization, and lower packet loss rates. These findings emphasize the algorithm's capability to handle complex, high-traffic scenarios and its potential for optimizing large-scale network deployments.

The DG-SFP framework incorporates the dual-graph spatiotemporal link state prediction model (DG-SFN) to enhance routing performance. By leveraging the adjacency matrices of both the network topology and its line graph representation, the GAT effectively captures the spatial distribution and propagation dependencies of network traffic across links. Simultaneously, the RNN models the temporal trends and magnitude variations of network traffic. This integration of spatiotemporal dependency features enables the DGSFN-DR algorithm to adapt to the spatial dependencies of diverse network topologies and account for the temporal delays introduced by routing algorithms, improving its flexibility and adaptability.

The experimental results further validate the routing optimization capabilities of the DGSFN-DR algorithm under high-traffic conditions. The algorithm dynamically selects optimal paths by leveraging its flow-awareness and adaptability, significantly reducing overall network costs. It achieves these improvements without requiring extensive modifications to the existing network infrastructure or control mechanisms, demonstrating its practicality for real-world deployments.

5.7. Simulation Realism and Practical Deployment Considerations

We recognize that our evaluation is conducted in simulated SDN environments on benchmark topologies (e.g., NSFNET and NYC). Consequently, our empirical results should be interpreted with respect to the modeling assumptions adopted in the simulation. To strengthen deployment relevance, we intentionally avoid a purely stationary synthetic workload. In particular, the traffic workload is generated using **Google trace** statistics, and routing performance is evaluated under **multiple pressure levels** (LOW/MID/HIGH), following the experimental protocol used throughout the manuscript. This setting allows us to examine the proposed prediction-guided routing framework across different congestion intensities, rather than reporting performance at a single operating point.

A central feature of our approach is the *prediction-to-weight optimization* pipeline: DG-SFN predicts future link-state metrics at time $t + \Delta t$, and DGSFN-DR derives routing decisions by minimizing the accumulated *predicted* path weight computed from the predicted link weights (Equations (3) and (4)). Accordingly, the key assumption for practical applicability is not that the controller/actuation delay is negligible, but that the information produced within the considered prediction horizon remains informative for the routing decision at the corresponding time scale. Under this time-scale consistency, improvements are meaningful because the routing computation is explicitly driven by prediction-derived weights, and the gains are observed consistently across different traffic regimes.

While the current simulation framework does not fully reproduce all system-level effects in real deployments (e.g., detailed controller/actuation delays, device-specific queueing behaviors, and rare failure/recovery dynamics), our results provide evidence under traffic patterns grounded in real-world measurements and under varying congestion pressures. In future work, we will further study implementation-specific factors such as controller sampling granularity and time misalignment between prediction and actuation, and evaluate the method on real SDN controllers and testbeds to quantify their impact at finer time scales.

Robustness under adversarial or malicious traffic (limitation): our current simulation framework validates the prediction-to-routing pipeline under realistic but non-adversarial workload generation (gravity model traffic matrices seeded by Google trace statistics). In real SDN deployments, the observed telemetry and derived link-state metrics may be affected by adversarial or malicious traffic patterns, leading to distribution shift and potentially biased predictions. Concretely, such interference may impact (i) the telemetry-derived feature statistics used by DG-SFN, (ii) the mapping from predicted link-state metrics to link weights $\theta(\cdot)$, and consequently (iii) the routing decision derived from Dijkstra on predicted weights. Therefore, the reported performance should be interpreted as evidence of effectiveness under normal traffic variability rather than robustness against adversarial manipulations. As future work, we plan to conduct robustness evaluations under explicit threat models, for example, by injecting controlled traffic bursts and distribution shifts (to emulate anomalous workloads), and by simulating telemetry corruption/noise or systematic metric bias before DG-SFN inference to measure how performance degrades and whether additional stabilization strategies (e.g., prediction smoothing or uncertainty-aware weighting) are needed.

Limitation regarding cybersecurity threats (e.g., web shells and jailbreak attacks): the current evaluation primarily focuses on prediction-to-routing performance under realistic network traffic and system operations, and does not include direct assessment against cybersecurity threats such as web shells or jailbreak attacks. These threats typically require threat-specific modeling and additional detection/mitigation mechanisms (e.g., application-layer compromise detection or model-behavior containment), which are beyond the scope of our current study on spatiotemporal fusion-based QoS prediction and dynamic routing optimization. Nevertheless, we note that certain cybersecurity incidents can indirectly affect the proposed pipeline by altering telemetry measurements and inducing distribution shift, potentially leading to biased predicted link states and suboptimal routing decisions. Therefore, the reported results should be interpreted as effectiveness under non-adversarial operational conditions. As future work, we plan to extend evaluation with explicit threat models, including telemetry-level perturbations (e.g., measurement noise/bias and controlled workload anomalies) to assess robustness of the prediction-to-routing pipeline under malicious interference, and—if additional LLM-based components are introduced for explanation—an evaluation of safety/robustness against prompt-manipulation and jailbreak-like behaviors.

Privacy and ethical evaluation scope: While our experiments primarily report performance-oriented metrics (e.g., latency and jitter) to validate the effectiveness of spatiotemporal prediction and prediction-guided routing, we acknowledge that modern edge-based AI systems must also consider privacy and ethical aspects when handling network telemetry and producing routing decisions. In this work, we treat privacy and ethical considerations as deployment constraints and provide a corresponding design rationale rather than conducting a comprehensive privacy/ethics benchmark. Specifically, our edge-integrated pipeline mitigates potential privacy exposure by processing telemetry-derived information as task-oriented features for prediction, and by restricting the decision outputs to the optimization-related artifacts (predicted link states and routing actions) instead of retaining unnecessary raw measurements. Moreover, ethical usability requires that routing decisions can be interpreted by operators when abnormal behaviors are detected, so that automated control remains accountable and aligned with operational intent. We leave a systematic privacy/ethics evaluation (e.g., quantifying information leakage from outputs and conducting operator-centric usability/interpretability studies) as future work, where concrete threat models and governance-oriented evaluation protocols can be defined for edge SDN deployments.

Data integrity and privacy protection in multi-tenant SDN: In multi-tenant SDN environments, multiple stakeholders share the same physical infrastructure while requiring isolation of their data and control decisions. To support this setting, our framework is designed such that privacy-preserving preprocessing is applied to telemetry measurements before they are consumed by the inference pipeline, thereby reducing the exposure of tenant-specific raw measurements during feature preparation and prediction. Moreover, the privacy wrapper parameterized by $\mathcal{K}$ can be instantiated with standard mechanisms (e.g., clipping/noise addition and/or encryption-oriented preprocessing), which helps prevent unauthorized inference from raw telemetry and limits information leakage across tenants. On the data integrity side, the predicted link-state outputs and derived routing decisions are generated from the transformed inputs within the edge decision cycle, which constrains the influence of cross-tenant interference to the modeled feature space rather than allowing direct manipulation of routing logic with untrusted raw telemetry. In practice, multi-tenant integrity can be further strengthened by enforcing tenant-scoped access control to telemetry streams, separating per-tenant control parameters, and validating the freshness and consistency of measurements used for prediction and routing. Together, these design choices support privacy-aware and integrity-preserving operation of the prediction-to-routing pipeline in multi-tenant SDN deployments, aligning with international expectations for responsible AI system data handling.

5.8. Scalability with Topology Size and Traffic Load

To evaluate practical scalability, we report a unified assessment that jointly considers (i) the *decision-cycle runtime decomposition* into QoS prediction and routing (Dijkstra on predicted edge weights), (ii) *peak controller memory* (peak RSS), and (iii) *controller overhead under dynamic traffic demand* using a throughput-based proxy. All numeric results are summarized in Table 14, and the combined visualization is given in Figure 19 (left: runtime, middle: memory, right: controller overhead).

Table 14. Scalability analysis on NYC-like topologies: runtime decomposition, peak memory, and controller throughput under load.

Topology	$\|V\|$	$\|E\|$	**Pred. Time (ms)**	**Routing Time (ms)**	**Peak RSS (MB)**	
Topo 0	8	9	9.999	0.042	1378.63	
Topo 1	10	13	9.815	0.042	1379.65	
Topo 2	12	21	9.742	0.040	1380.62	
Topo 3	14	28	9.738	0.030	1381.57	
Load level	1	2	4	8	12	16
Throughput (decisions/s)	97.95	98.97	96.57	98.91	99.94	98.59

(1) Runtime scaling (decision-cycle decomposition). As shown in the left panel of Figure 19, the decision-cycle time is dominated by the QoS prediction stage, while Dijkstra routing time is orders of magnitude smaller. As topology size increases from $(|V|, |E|) = (8, 9)$ to $(14, 28)$, the prediction time remains nearly constant at 9.999–9.738 ms (about 2.6% variation), whereas the routing time stays in a very small range of 0.042–0.030 ms. Therefore, within the evaluated scale, the overall decision-cycle latency does not exhibit meaningful degradation as the topology becomes larger and denser.

(2) Memory scaling (peak controller RSS). The middle panel of Figure 19 reports peak controller memory. Peak RSS increases slightly from 1378.63 MB to 1381.57 MB as $|E|$ grows from 9 to 28, staying within a narrow band of 1378.63–1381.57 MB. This indicates stable memory behavior and no observed memory bottleneck when scaling to denser topologies.

(3) Controller overhead under traffic load. The right panel of Figure 19 assesses controller overhead under dynamic demand. Throughput (decisions/s) remains high across load levels 1–16, ranging from 96.57 to 99.94 decisions/s (Table 14). Although there is a temporary dip at load level 4 (96.57 decisions/s), throughput recovers at higher loads and reaches the peak near load 12 (99.94 decisions/s), and only decreases slightly at load 16 (98.59 decisions/s). Overall, the small variation in throughput suggests that the controller does not experience throughput collapse under increased request pressure.

(4) Mechanism interpretability (QoS-to-weight contribution). To provide interpretability for the scalable routing step, we visualize the contribution of QoS components to the constructed link cost in Figure 20. This highlights how QoS-aware predicted weights are formed while keeping the routing stage computationally light.

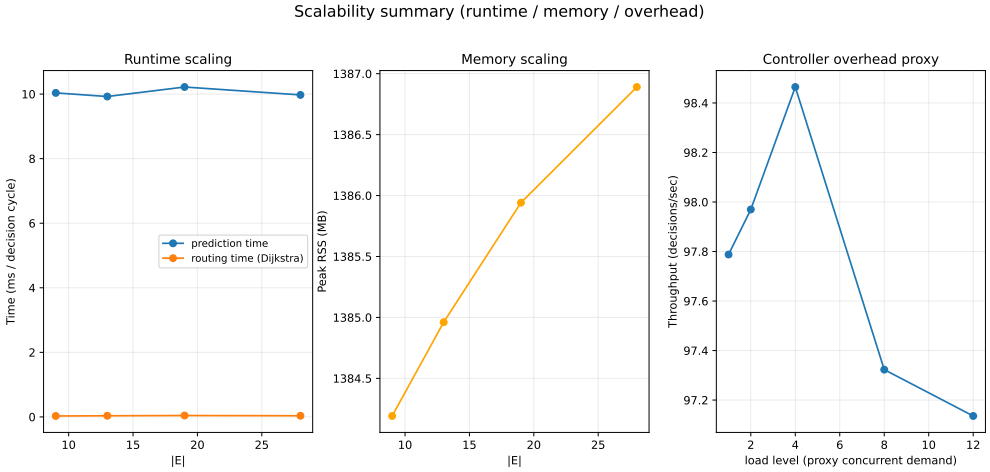


Figure 19. Scalability summary (runtime/peak memory/controller overhead).

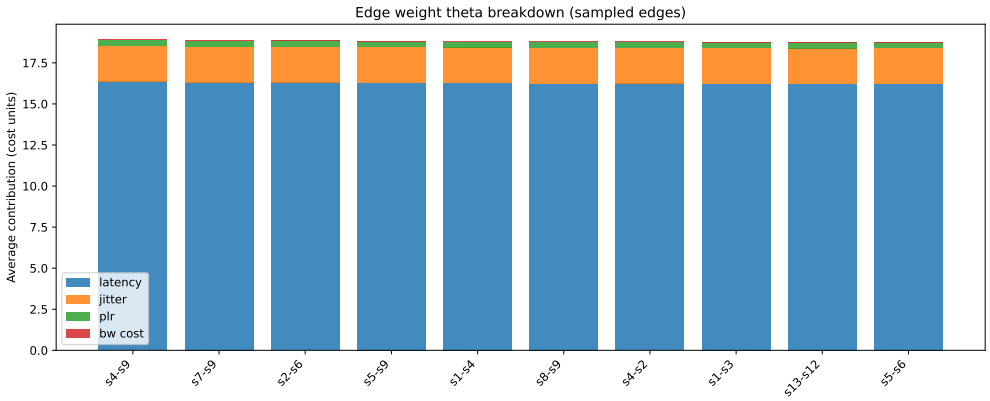


Figure 20. Edge weight theta breakdown (sampled edges): average contribution of latency, jitter, plr, and bw cost to the constructed link cost.

5.9. Privacy-Preserving and GDPR-Aware Real-World Deployment Feasibility

In edge-integrated SDN deployments, practical realization requires not only accurate spatiotemporal prediction and effective dynamic routing, but also that the end-to-end decision cycle remains feasible under data-handling requirements. In this regard, privacy-preserving mechanisms are commonly applied before the inference pipeline consumes network state measurements. In our prediction-to-routing architecture (DG-SFN followed by DGSFN-DR), we model the privacy effect at the runtime level as an additional preprocessing overhead within the edge decision cycle. Specifically, privacy-enabled inference can be expressed as

$$\hat{S}(t+\Delta t) = f_{\text{DG-SFN}}\big(\mathcal{P}(X(t), \mathcal{K})\big), \tag{34}$$

where $\mathcal{P}(\cdot)$ denotes a privacy-preserving transformation (e.g., clipping/noise addition and/or privacy-oriented preprocessing) with privacy-related parameters $\mathcal{K}$. Accordingly, the edge-side decision-cycle latency can be decomposed as

$$T_{\text{cycle}}^{\text{priv}} = t_{\text{priv}} + t_{\text{pred}} + t_{\text{route}}, \tag{35}$$

where t_{priv} is the additional runtime overhead induced by privacy-preserving preprocessing, t_{pred} is the DG-SFN inference latency, and t_{route} is the routing time (Dijkstra on predicted link weights).

(1) Effect on inference speed and real-time feasibility. Our scalability results indicate that the routing component is orders of magnitude smaller than the QoS prediction component (i.e., $t_{\text{route}} \ll t_{\text{pred}}$), meaning that privacy mainly affects the decision cycle through the additive term t_{priv} within the prediction stage budget. To maintain control-loop real-time constraints, edge-integrated SDN systems require $t_{\text{priv}} \ll t_{\text{pred}}$ so that $T_{\text{cycle}}^{\text{priv}}$ stays within the sampling interval. Under the tested topology scales, the prediction-time range remains approximately 9.999–9.738 ms, while the routing time stays in 0.042–0.030 ms. Therefore, even when privacy-preserving preprocessing is enabled, the overall decision-cycle latency remains dominated by DG-SFN inference and does not alter the routing-time bottleneck structure, supporting the practical applicability of the prediction-guided routing strategy.

(2) GDPR-oriented real-world deployment considerations. Beyond timing feasibility, real-world SDN operations must also align with regulatory requirements such as the General Data Protection Regulation (GDPR). Although our work primarily focuses on spatiotemporal fusion and dynamic routing optimization, the proposed edge-integrated design facilitates GDPR-aware deployment practices by clarifying where data is processed and how it is utilized. In our pipeline, the network state used by DG-SFN is treated as telemetry-derived *features* for predicting future link states, and the inference and routing decisions are executed in an edge-integrated SDN environment. This reduces reliance on centralized aggregation of raw telemetry and supports data minimization by using only the feature representations required for prediction and decision-making. Moreover, the predicted link-state outputs and routing decisions are used for the explicit purpose of network dynamic routing optimization and QoS improvement (latency/jitter/packet loss/bandwidth). Practical deployments can further follow storage limitation principles (e.g., short-term buffering aligned with the decision interval and avoiding unnecessary retention of raw measurements), which helps limit potential regulatory risk.

Overall, this section bridges the gap between deployment feasibility and regulatory-aware operation by (i) explaining how privacy-related preprocessing contributes additively to the edge decision latency without changing the routing bottleneck, and (ii) positioning the edge-integrated pipeline as a natural fit for GDPR-oriented data-handling practices.

6. Conclusions

The centralized control and programmability of SDNs enable flexible and optimized on-demand complex network control and management for load balancing and QoS improvement. This paper proposes a network state-aware dual graph spatiotemporal prediction framework (DG-SFP) for SDN dynamic routing optimization (DGSFN-DR). In our framework, we first represent the network topology as a node adjacency matrix and a link adjacency matrix corresponding to the line graph of the network topology to capture the physical and logical structures of the network. Then, we employ GATs to generate spatiotemporal feature embeddings for nodes and links according to real-time network metric data collected from the SDN, e.g., latency, packet loss rate, bandwidth, and jitter. Subsequently, we slice these embeddings into t-step time series and leverage a GNN to ex-

tract the spatial dependency features of traffic across links. These features are then fed into an RNN to capture temporal dependency features between links to predict future network states. The DGSFN-DR dynamically adjusts routing strategies in the SDN according to the DG-SFN prediction results to achieve optimal path selection by minimizing the link weights. We conduct extensive simulation experiments to evaluate the effectiveness of the DG-SFN. The DG-SFN achieves over 10% improvement of four performance metrics compared to the classical models of FC-LSTM, GRU, and Transformer, and achieves approximately 6% improvement compared to SOTA graph-based models, such as STFAGN and GL-STGTN. Furthermore, the DGSFN-DR demonstrates superior flow awareness on network links compared to the DSP algorithm under three different traffic intensities and two distinct network topologies. The DGSFN-DR selects optimal paths according to the predicted link weights more effectively than DSP, achieving a 4% to 10% improvement on NSFNET and a 5% to 15% improvement on NYC, with particularly significant optimization effects under high traffic intensity and complex network topologies.

Author Contributions: Conceptualization, J.Z. (Jiaxian Zhu) and J.Z. (Jialing Zhao); methodology, W.B.; software, Z.L.; validation, C.Z. and T.Z.; writing—original draft preparation, J.Z. (Jiaxian Zhu); writing—review and editing, T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the National Natural Science Foundation (No. 62462021), the Guangxi Natural Science Foundation (No. 2026GXNSFAA00641306), the Guangdong Province University Key Field (No. 2023ZDZX3041), the Philosophy and Social Sciences Planning Project of Zhejiang Province (No. 25JCXK006YB), the Hainan Provincial Natural Science Foundation of China (No. 625RC716), the Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515010197), Zhaoqing University Innovation Research Team Grant Project, and University-Level High-Caliber Training Project of Zhaoqing University (NO. gcc202607).

Data Availability Statement: The data and source code of the model presented in this paper can be obtained from the corresponding authors for non-commercial use.

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

References

1. Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 3–14. [CrossRef]
2. Hong, C.Y.; Kandula, S.; Mahajan, R.; Zhang, M.; Gill, V.; Nanduri, M.; Wattenhofer, R. Achieving high utilization with software driven WAN. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, Hong Kong, China, 12–16 August 2013; pp. 15–26.
3. Chen, Y.R.; Rezapour, A.; Tzeng, W.G.; Tsai, S.C. RL-routing: An SDN routing algorithm based on deep reinforcement learning. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 3185–3199. [CrossRef]
4. Masoudi, R.; Ghaffari, A. Software defined networks: A survey. *J. Netw. Comput. Appl.* **2016**, *67*, 1–25. [CrossRef]
5. Mao, B.; Tang, F.; Fadlullah, Z.M.; Kato, N. An intelligent route computation approach based on real-time deep learning strategy for software-defined communication systems. *IEEE Trans. Emerg. Top. Comput.* **2019**, *9*, 1554–1565. [CrossRef]
6. Ding, M.; Guo, Y.; Huang, Z.; Lin, B.; Luo, H. GROM: A generalized routing optimization method with graph neural network and deep reinforcement learning. *J. Netw. Comput. Appl.* **2024**, *229*, 103927. [CrossRef]
7. Rusek, K.; Suárez-Varela, J.; Mestres, A.; Barlet-Ros, P.; Cabellos-Aparicio, A. Unveiling the potential of graph neural networks for network modeling and optimization in SDN. In Proceedings of the 2019 ACM Symposium on SDN Research, San Jose, CA, USA, 3–4 April 2019; pp. 140–151.
8. Rusek, K.; Suárez-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]
9. Yang, S.; Liu, Y.; Cui, L.; Peng, Y.; Li, Z.; Leung, V.C. ER-OCN: Toward efficient network routing in ocean city based on deep reinforcement learning. *Comput. Commun.* **2024**, *222*, 97–107. [CrossRef]
10. Zhang, X.; Zhang, J.; Si, W.; Liu, K. Dynamic Weight Adjusting Deep Q-Networks for Real-Time Environmental Adaptation. *arXiv* **2024**, arXiv:2411.02559.

11. Huang, L.; Ye, M.; Xue, X.; Wang, Y.; Qiu, H.; Deng, X. Intelligent routing method based on Dueling DQN reinforcement learning and network traffic state prediction in SDN. *Wirel. Netw.* **2024**, *30*, 4507–4525. [CrossRef]
12. Tang, F.; Fadlullah, Z.M.; Mao, B.; Kato, N. An intelligent traffic load prediction-based adaptive channel assignment algorithm in SDN-IoT: A deep learning approach. *IEEE Internet Things J.* **2018**, *5*, 5141–5154. [CrossRef]
13. Ferriol-Galmés, M.; Paillisse, J.; Suárez-Varela, J.; Rusek, K.; Xiao, S.; Shi, X.; Cheng, X.; Barlet-Ros, P.; Cabellos-Aparicio, A. RouteNet-Fermi: Network modeling with graph neural networks. *IEEE/ACM Trans. Netw.* **2023**, *31*, 3080–3095. [CrossRef]
14. Almasan, P.; Suárez-Varela, J.; Rusek, K.; Barlet-Ros, P.; Cabellos-Aparicio, A. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Comput. Commun.* **2022**, *196*, 184–194. [CrossRef]
15. Mao, B.; Tang, F.; Kawamoto, Y.; Kato, N. AI models for green communications towards 6G. *IEEE Commun. Surv. Tutor.* **2021**, *24*, 210–247. [CrossRef]
16. Tang, F.; Kawamoto, Y.; Kato, N.; Liu, J. Future intelligent and secure vehicular network toward 6G: Machine-learning approaches. *Proc. IEEE* **2019**, *108*, 292–307. [CrossRef]
17. Noor-A-Rahim, M.; Liu, Z.; Lee, H.; Khyam, M.O.; He, J.; Pesch, D.; Moessner, K.; Saad, W.; Poor, H.V. 6G for vehicle-to-everything (V2X) communications: Enabling technologies, challenges, and opportunities. *Proc. IEEE* **2022**, *110*, 712–734. [CrossRef]
18. Asif, N.A.; Sarker, Y.; Chakrabortty, R.K.; Ryan, M.J.; Ahamed, M.H.; Saha, D.K.; Badal, F.R.; Das, S.K.; Ali, M.F.; Moyeen, S.I.; et al. Graph neural network: A comprehensive review on non-euclidean space. *IEEE Access* **2021**, *9*, 60588–60606. [CrossRef]
19. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
20. Wu, K.; Xu, C.; Yan, J.; Wang, F.; Lin, Z.; Zhou, T. Error-distribution-free kernel extreme learning machine for traffic flow forecasting. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106411. [CrossRef]
21. Yu, B.; Yin, H.; Zhu, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv* **2017**, arXiv:1709.04875.
22. Guo, S.; Lin, Y.; Feng, N.; Song, C.; Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 29–31 January 2019; Volume 33, pp. 922–929.
23. Li, M.; Zhu, Z. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; Volume 35, pp. 4189–4196.
24. Yang, S.; Li, H.; Luo, Y.; Li, J.; Song, Y.; Zhou, T. Spatiotemporal adaptive fusion graph network for short-term traffic flow forecasting. *Mathematics* **2022**, *10*, 1594. [CrossRef]
25. Zheng, C.; Fan, X.; Wang, C.; Qi, J. Gman: A graph multi-attention network for traffic prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 1234–1241.
26. Bai, L.; Yao, L.; Li, C.; Wang, X.; Wang, C. Adaptive graph convolutional recurrent network for traffic forecasting. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17804–17815.
27. Tumma, C.; Azmeera, R.; Ayyamgari, S.; Thumma, B.Y.R. Data security and privacy protection in artificial intelligence models: Challenges and defense mechanisms. *Int. J. Sci. Res. Eng. Manag.* **2025**, *9*, 1–9. [CrossRef]
28. BinSahaq, A.; Sheltami, T.; Mahmoud, A.; Nasser, N. Fast and efficient algorithm for delay-sensitive QoS provisioning in SDN networks. *Wirel. Netw.* **2024**, *30*, 4607–4628. [CrossRef]
29. Zhang, Y.; Chen, M. Performance evaluation of Software-Defined Network (SDN) controllers using Dijkstra's algorithm. *Wirel. Netw.* **2022**, *28*, 3787–3800. [CrossRef]
30. Mohammadi, R.; Javidan, R. EFSUTE: A novel efficient and survivable traffic engineering for software defined networks. *J. Reliab. Intell. Environ.* **2022**, *8*, 247–260.
31. Devarajulu, V.S.; Kanipakam, S.; Kavarakuntla, T. Explainable Anomaly Detection in Cloud-Native Systems via RAG-Enhanced Vector Retrieval. In *Proceedings of the 2025 6th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*; IEEE: New York, NY, USA, 2025; pp. 181–185.
32. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
33. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *Stat* **2017**, *1050*, 10-48550.
34. Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; Li, Y. Simple and deep graph convolutional networks. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 1725–1735.
35. Wang, X.; Bo, D.; Shi, C.; Fan, S.; Ye, Y.; Philip, S.Y. A survey on heterogeneous graph embedding: Methods, techniques, applications and sources. *IEEE Trans. Big Data* **2022**, *9*, 415–436. [CrossRef]
36. Haeupler, B.; Hladík, R.; Rozhoň, V.; Tarjan, R.E.; Tetěk, J. Universal optimality of dijkstra via beyond-worst-case heaps. In *Proceedings of the 2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*; IEEE: New York, NY, USA, 2024; pp. 2099–2130.

37. Al-Saadi, M.; Khan, A.; Kelefouras, V.; Walker, D.J.; Al-Saadi, B. SDN-based routing framework for elephant and mice flows using unsupervised machine learning. *Network* **2023**, *3*, 218–238. [CrossRef]
38. Sharma, P.; Bhatia, V.; Prakash, S. Impact of fragmentation in quantum signal channel of quantum key distribution enabled optical networks. *IET Quantum Commun.* **2024**, *5*, 164–172. [CrossRef]
39. Gunavathie, M.; Umamaheswari, S. Traffic-aware optimal routing in software defined networks by predicting traffic using neural network. *Expert Syst. Appl.* **2024**, *239*, 122415. [CrossRef]
40. Reiss, C.; Tumanov, A.; Ganger, G.R.; Katz, R.H.; Kozuch, M.A. Towards understanding heterogeneous clouds at scale: Google trace analysis. *Intel Sci. Technol. Cent. Cloud Comput.* **2012**, *84*, 1–21.
41. Ma, X.; Tao, Z.; Wang, Y.; Yu, H.; Wang, Y. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transp. Res. Part C Emerg. Technol.* **2015**, *54*, 187–197.
42. Dey, R.; Salem, F.M. Gate-variants of gated recurrent unit (GRU) neural networks. In *Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*; IEEE: New York, NY, USA, 2017; pp. 1597–1600.
43. Vaswani, A. Attention is all you need. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017.
44. Li, Z.; Zhou, J.; Lin, Z.; Zhou, T. Dynamic spatial aware graph transformer for spatiotemporal traffic flow forecasting. *Knowl.-Based Syst.* **2024**, *297*, 111946. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.