symmetry

MDPI

Article

Congestion-Aware Adaptive Routing Based on Graph Attention Networks and Dynamic Cost Optimization

Jun Liu [1], Xinwei Li [1,*] and Lingyun Zhou [2]

[1] National Key Laboratory of Wireless Communications, University of Electronic Science and Technology of China, Chengdu 611731, China; liu_jun@uestc.edu.cn
[2] College of Electronics and Information Engineering, South-Central Minzu University, Wuhan 430074, China; lingyunzhou@scuec.edu.cn
* Correspondence: 202421220308@std.uestc.edu.cn; Tel.: +86-189-8318-3413

Abstract

To mitigate local congestion and address the adaptability limitations of traditional static routing under dynamic traffic, this paper proposes an end-to-end routing method based on a Graph Attention Network (GAT), termed Congestion-Aware Graph Attention Routing (CA-GAR). To alleviate the issue of local optima in traditional heuristic iterative optimization, we design a dynamic link cost optimization algorithm with multi-start parallel exploration. This algorithm employs a "penalty–reselection–reward" closed-loop feedback mechanism, performing global searches from multiple random initial states to generate a high-quality, empirically near-optimal cost matrix as supervised labels. Building on this, CA-GAR leverages a multi-head attention mechanism to adaptively aggregate high-order topological features of nodes and edges, and incorporates a staged hierarchical hyperparameter optimization strategy to map real-time network states to link costs. Simulation results demonstrate that CA-GAR outperforms traditional static routing under light, medium, and heavy loads. Under high-load burst conditions, the method exhibits effective congestion avoidance capability, reducing end-to-end delay by approximately 50% and lowering the packet loss rate to as low as 2%. Compared with QLRA, CA-GAR shows promising performance in multi-path traffic splitting and possesses robust fast rerouting capabilities during node failures, thereby achieving intelligent traffic distribution and global load balancing.

Keywords: dynamic routing; link cost optimization; graph attention network; congestion awareness; load balancing; multi-head attention

1. Introduction

With the rapid development of emerging services such as 5G/6G, industrial internet, and vehicular networks, network traffic exhibits complex characteristics including high dynamics, burstiness, and non-uniform spatial distribution, posing significant challenges to traditional routing mechanisms [1]. Traditional static shortest-path routing mechanisms, which lack real-time network state awareness and adaptive adjustment capabilities, may cause overload on local links or nodes when facing bursty traffic flows [2], resulting in a "hotspot" effect. Such local congestion further triggers surges in end-to-end latency, increased jitter, and higher packet loss rates [3], severely constraining quality-of-service (QoS) guarantees for critical applications. Therefore, researching dynamic routing mechanisms equipped with congestion awareness and adaptive regulation capabilities has become a core topic for enhancing next-generation network performance.

Check for updates

Academic Editor: Jie Yang

Received: 13 March 2026
Revised: 16 April 2026
Accepted: 21 April 2026
Published: 24 April 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.

Currently, academia has proposed various dynamic routing schemes. Early research mainly relied on heuristic rules or mathematical programming methods, periodically adjusting link weights to balance loads [4,5]. However, such approaches suffer from excessively high computational complexity in large-scale networks, failing to meet real-time requirements, and their search processes are prone to getting trapped in local optima. In recent years, network digital twin technology has provided new insights for routing optimization [6,7]. Some schemes utilize high-fidelity simulation environments to rapidly evolve routing configurations, forming a closed-loop strategy of "perception–evaluation–optimization". Although such methods can alleviate local congestion, they rely on iterative simulations, which introduce response delays that limit their applicability in rapidly changing dynamic network scenarios [8].

Meanwhile, data-driven deep learning methods have brought a paradigm shift to routing decisions. Existing studies attempt to use neural networks to establish mapping relationships between network states and performance metrics [9,10]. However, most methods simplify paths into sequential data, ignoring the inherent graph structural characteristics of networks, thus failing to depict the competitive relationships among multiple service flows on shared links and the laws of congestion propagation. Furthermore, existing graph neural networks still face challenges in routing applications. On the one hand, standard GNN architectures often exhibit limitations in generalization capability, making them sensitive to hyperparameter settings [11]. On the other hand, they struggle with the coupled modeling of multi-dimensional QoS indicators, often failing to balance optimality with robustness under dynamic network conditions [12].

To accelerate path allocation and perceive dynamic topologies, existing research primarily focuses on the fusion of spatiotemporal feature evolution and intelligent decision-making. In terms of temporal and heterogeneous feature extraction, Liu et al. [13] (2024) integrated a sequence-aware graph neural network into gated recurrent units to capture local temporal dependencies under non-fixed graph structures. Huang et al. [14] (2026) designed a heterogeneous perception joint contrastive learning mechanism, which extracts deep topological features by decoupling and reconstructing heterogeneous subgraphs along with hierarchical feature aggregation. Regarding the fusion of intelligent decision-making, Ding et al. [15] (2024) combined deep reinforcement learning with graph neural networks to achieve zero-shot policy generation under unseen topologies; Gómez-delaHiz et al. [16] (2024) and Xu et al. [17] (2026) introduced a local multi-agent architecture and the GraphSAGE-MAPPO joint algorithm, respectively, validating the effectiveness of collaborative optimization for routing and resources. For the deepened application of attention mechanisms, Simon et al. [18] (2025) utilized deep graph attention to enhance topology perception capabilities; Dhamala et al. [19] (2024) introduced an attention mechanism within the RouteNet framework, significantly improving the accuracy of network delay prediction through dynamic weighting of link states. Furthermore, while the adaptive diffusion routing protocol proposed by Hakim et al. [20] (2024) effectively reduced energy consumption in wireless sensor networks by constructing local gradient fields, it is limited by local information interaction and struggles to acquire a global topological view; moreover, pure reinforcement learning paradigms often face challenges of slow convergence. Specifically, studies on sliced ad hoc networks have investigated Q-learning-based routing for QoS optimization [21], and TDMA waveform designs coupled with GAT for decision-making [22]. In summary, when facing highly non-uniform bursty traffic, existing models often fall into a dilemma: complex graph neural structures improve the precision of multi-dimensional QoS modeling but lead to excessive online inference latency; conversely, lightweight operators or localized strategies, while reducing computational overhead, sacrifice global perspective, causing a sharp decline in generalization performance.

Addressing the problem that existing works struggle to balance model generalization capability and online inference overhead under highly non-uniform bursty traffic, this paper proposes the CA-GAR model. This model integrates labels generated from offline simulations, designs a dynamic link cost optimization algorithm with multi-start parallel exploration, and introduces a "penalty–reselection–reward" closed-loop feedback mechanism to mitigate the tendency of traditional heuristic optimization to fall into local optima. This process generates a high-quality, engineering near-optimal cost matrix as supervised labels. Based on this, CA-GAR utilizes a multi-head attention mechanism to adaptively aggregate high-order topological features and adopts a staged hierarchical hyperparameter optimization strategy to achieve robust mapping from real-time network states to optimal link costs. Finally, by computing routing for multiple service flows in multi-path network environments using the predicted end-to-end cost matrix, the method effectively enhances the network's congestion avoidance and load balancing capabilities.

2. Graph Attention Networks

To facilitate understanding of subsequent content, we first introduce the fundamental theory of Graph Attention Networks (GAT) in conjunction with network topology, including the operational principles of graph attention layers and the multi-head attention mechanism for enhancing feature representation.

2.1. Graph Attention Layer

The graph attention layer assigns varying importance weights to neighboring nodes via an attention mechanism, thereby achieving adaptive weighted aggregation of node features [23]. Given a graph $G = (V, E)$, for any node $v_i \in V$, its input feature is $\mathbf{h}_i \in \mathbb{R}^F$, where F denotes the input feature dimension. Through a shared learnable linear transformation matrix $\mathbf{W} \in \mathbb{R}^{F \times F'}$, node features are mapped to a new feature space:

$$\tilde{\mathbf{h}}_i = \mathbf{W}\mathbf{h}_i \tag{1}$$

where F' represents the output feature dimension. Using an attention vector $\mathbf{a} \in \mathbb{R}^{2F'}$ and the LeakyReLU (Leaky Rectified Linear Unit) activation function, the unnormalized attention score e_{ij} between node pairs (v_i, v_j) is calculated as:

$$e_{ij} = \text{LeakyReLU}\left(a^T \cdot [\tilde{\mathbf{h}}_i || \tilde{\mathbf{h}}_j]\right) \tag{2}$$

where $||$ denotes the vector concatenation operation. For node pairs without an edge $(v_i, v_j) \notin E$, their attention scores are set to extremely small values to ensure their weights approach zero during subsequent normalization. The softmax function is used to normalize the neighbors of node v_i, obtaining attention weights α_{ij}:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{v_k \in N(v_i)} \exp(e_{ik})} \tag{3}$$

Here, $N(v_i) = \{v_j | (v_i, v_j) \in E\}$ represents the set of neighbor nodes of v_i. The output feature $\mathbf{h}_i{}'$ of node v_i is obtained by performing a weighted sum of neighbor features using attention weights and introducing nonlinearity through the ELU (Exponential Linear Unit) activation function [24]:

$$\mathbf{h}_i{}' = \text{ELU}\left(\sum_{v_j \in N(v_i)} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right) \tag{4}$$

Figure 1 details the internal structure of the graph attention layer. This layer adaptively captures complex relationships within the graph structure through the attention mechanism, enhancing performance in tasks such as node classification and link prediction.

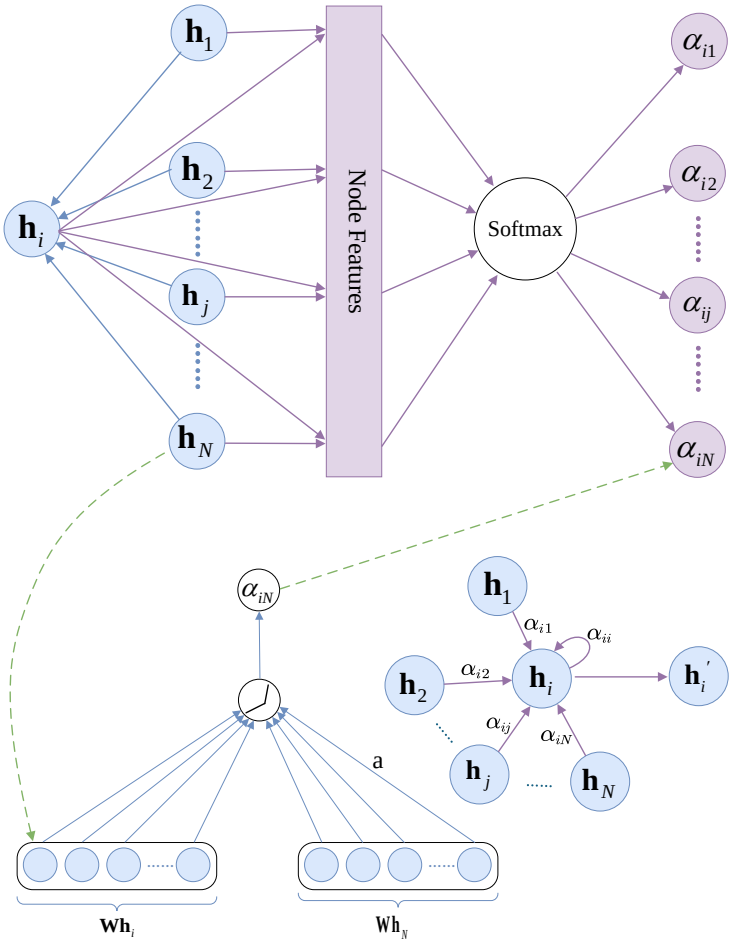


Figure 1. Structure of the Attention Layer.

Taking node v_i as an example. In the upper part of Figure 1, blue arrows represent the connections between nodes. Theoretically, it is necessary to compute the attention scores between this node and all other nodes, and then set the scores for non-neighbor nodes to an extremely small value through masking. Therefore, the nodes in the second column can be understood as all possible N nodes. Purple arrows indicate the process of linear transformation and attention score calculation. In the lower part, the content pointed to by green arrows on the left further details the calculation process from node features to attention weights, while the right side corresponds to the formula for the output features, showing the result of weighted aggregation of neighbor node features.

2.2. Multi-Head Attention Mechanism

A single-head graph attention layer aggregates neighbor information using only one set of attention weights, which may limit feature expressiveness by focusing on a single feature subspace. To enhance model stability and representational diversity, the multi-head attention mechanism runs K independent attention heads in parallel. Each head possesses independent parameter matrices $\mathbf{W}^{(k)}$ and attention vectors $\mathbf{a}^{(k)}(k = 1, \ldots, K)$, thereby capturing complementary information from neighbor nodes across different feature subspaces [25,26].

For the k-th attention head, the output node feature is denoted as ${\mathbf{h}_i}'^{(k)}$. Finally, the output features of all heads are concatenated to obtain the final representation of node v_i:

$$\mathbf{h}_i = ||_{k=1}^{K} \mathbf{h}'^{(k)}_i \in \mathbb{R}^{K \times F'} \tag{5}$$

where F' is the output dimension of a single attention head. If controlling the output dimension is required, an averaging operation can be employed instead of concatenation in specific layers, i.e., $\mathbf{h}_i = \frac{1}{K} \sum_{k=1}^{K} \mathbf{h}'^{(k)}_i$. The specific number of heads and layer strategies are typically adjusted according to task requirements.

3. Congestion-Aware Dynamic Link Cost Optimization Method

The dynamic link cost optimization problem studied in this paper essentially falls into the category of constrained multi-flow joint routing optimization and is theoretically NP-hard: even when considering only the end-to-end delay constraint of a single flow, finding the minimum-cost path that satisfies the constraint has been proven to be NP-hard [27]; when extended to multi-flow concurrent scenarios with global load balancing, the search space grows exponentially with network scale, making it computationally infeasible to obtain the globally optimal solution in polynomial time. Based on this theoretical understanding, we do not pursue mathematical global optimality. Instead, we design a heuristic framework that seeks high-quality feasible solutions within a finite iteration budget. Specifically, we propose an iterative, adaptive link cost adjustment framework, whose core consists of a congestion detection-driven reward–punishment mechanism as the backbone, supplemented by exploration mechanisms and soft restart mechanisms, forming a robust and efficient optimization closed loop [28]. The entire process is fully compatible with existing IGP (Interior Gateway Protocol) protocols (only modifying link weights) and can be directly deployed in network simulation environments such as NS-3.

3.1. Simulation-Driven Iterative Optimization Framework for Link Costs

Consider a directed network topology $G = (V, E)$, where $|V| = N$. The physical adjacency matrix $A^{\text{phy}} \in \{0,1\}^{N \times N}$ describes the topological structure; if $e_{ij} \in E$, then $A^{\text{phy}}_{ij} = 1$, otherwise it is 0.

Let the maximum number of iterations be $T_{\max} \in \mathbb{Z}_{>0}$. The algorithm starts with an initial link cost matrix $C^{(0)}$ and dynamically updates this matrix over subsequent $T_{\max}$ iterations. The elements of the link cost matrix $C^{(t)}$ are constrained by preset lower bound $C_{\min}$ and upper bound $C_{\max}$ (where $0 < C_{\min} < C_{\max} < \infty$), i.e., $C^{(t)} \in [C_{\min}, C_{\max}]^{N \times N} \bigcup \{\infty\}$, satisfying: for any physical link $e_{ij} \in E$, the initial cost is set to $C^{(0)}_{ij} = c_0$, where c_0 is an initial scalar cost value; for non-physical links $e_{ij} \notin E$, it always holds that $C^{(t)}_{ij} = \infty$, indicating unreachable. In the t-th iteration ($t = 1, 2, \ldots, T_{\max}$), based on the cost matrix $C^{(t-1)}$ from the previous round and the set of service flows $\mathcal{T} = \{(s_k, d_k)\}(s_k, d_k \in V)$, the standard Dijkstra algorithm calculates the shortest path for each source–destination pair, yielding the path set $\mathcal{P}^{(t)} = \{P^{(t)}_k\}$. Configuring $\mathcal{P}^{(t)}$ into the network simulator yields global performance metrics, with average end-to-end delay $D^{(t)}$ serving as the primary evaluation criterion. Depending on the current network state, the corresponding mechanisms described in Sections 3.2–3.4 are dynamically invoked to generate the next-round cost matrix $C^{(t)}$. After completing all $T_{\max}$ iterations, the cost matrix corresponding to the round with minimal delay is selected as the final output, denoted as:

$$C^* = \arg \min_{t \in \{1, \cdots, T_{\max}\}} D^{(t)} \tag{6}$$

Although this solution does not guarantee global optimality in the mathematical sense, under the given iteration budget, C^* can be regarded as an engineering near-optimal cost matrix. Considering the inherent NP-hard nature of the problem and the real-time decision-making requirements in dynamic network environments, pursuing theoretical global optimality is neither realistic nor necessary, and accepting such engineering "good enough" feasible solutions has clear practical value. From an engineering deployment perspective, the elements of C^* all fall within the preset cost interval and satisfy the format specifications of IGP protocols for link weights, thus can be directly deployed in actual networks supporting IGP. To support the above process, link load status must be quantified in each round. Let B denote the effective bandwidth matrix, where $B_{ij} > 0$ if and only if $e_{ij} \in E$. For each service flow $(s, d) \in \mathcal{T}$, its transmission rate r_{sd} is superimposed onto each hop link along the path $P_{sd}^{(t)}$ selected in the t-th round, forming the total load matrix $L^{(t)}$ for the t-th round. Correspondingly, the link utilization matrix is defined as:

$$U_{ij}^{(t)} = \begin{cases} \frac{L_{ij}^{(t)}}{B_{ij}}, & \text{if } B_{ij} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Simultaneously, the number of distinct service flows passing through each link e_{ij} in the t-th round is counted, yielding the flow count matrix $F^{(t)}$. These state variables will be used for subsequent congestion determination and mechanism triggering. The overall framework and iterative process of the congestion-aware dynamic link cost optimization method are shown in Figure 2.

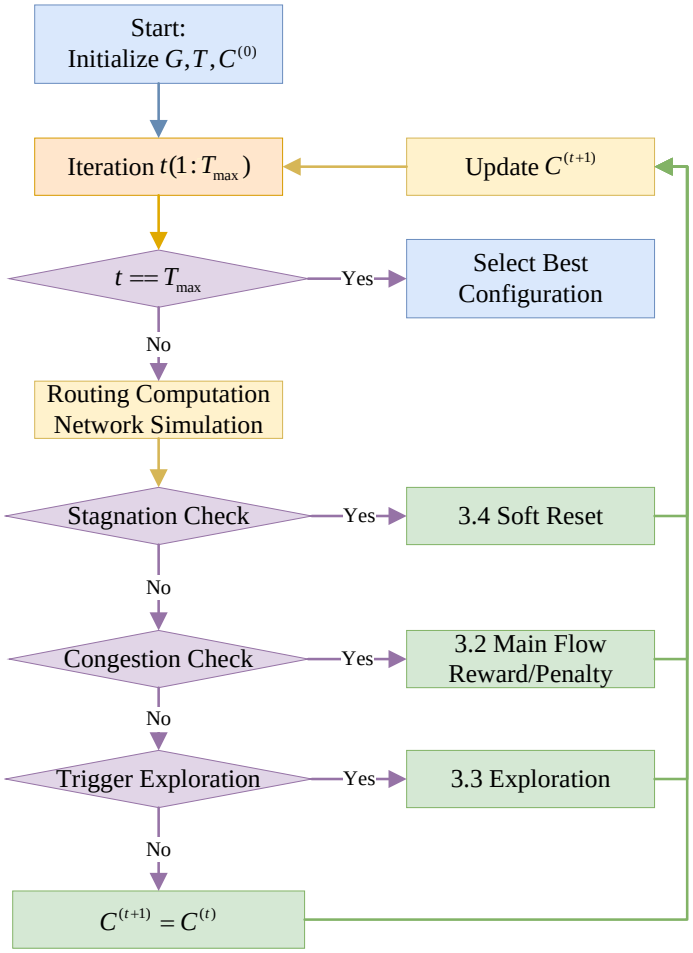


Figure 2. Flowchart of the Dynamic Link Cost Optimization Algorithm.

3.2. Congestion-Driven Reward-Punishment Evolution Mechanism

When significant congestion exists in the network, the algorithm prioritizes enabling the congestion-aware reward–punishment mechanism as the main optimization strategy. A high utilization threshold $\tau_{\text{cong}} \in (0.5, 1)$ is introduced. Physical links simultaneously satisfying $U_{ij}^{(t)} > \tau_{\text{cong}}$ and $F_{ij}^{(t)} \geq 2$ are defined as congested links $e_{ij} \in E$, i.e.,:

$$L_{\text{cong}}^{(t)} = \{e_{ij} \in E | U_{ij}^{(t)} \geq \tau_{\text{cong}} \wedge F_{ij}^{(t)} \geq 2\}. \tag{8}$$

If $L_{\text{cong}}^{(t)} \neq \varnothing$, the algorithm executes a penalty operation on each congested link $e_{ij} \in L_{\text{cong}}^{(t)}$, increasing its link cost from $C_{ij}^{(t)}$ to:

$$C_{ij}^{(t+1)} = \min(C_{\max}, C_{ij}^{(t)}(1 + \alpha)), \tag{9}$$

where $\alpha > 0$ is the penalty intensity coefficient. All service flows whose paths pass through at least one congested link are identified, constituting the affected flow set $\mathcal{T}_{\text{aff}}^{(t)}$. Under the condition of temporarily shielding the congested link set $L_{\text{cong}}^{(t)}$, the algorithm searches for up to k candidate alternative paths for each $(s, d) \in \mathcal{T}_{\text{aff}}^{(t)}$, filtering out healthy paths that satisfy the following conditions: (i) The path hop count h' satisfies $h' \leq \min(\eta \cdot h_{\text{orig}}, H_{\max})$, where h_{orig} is the original path hop count, $\eta > 1$ is the allowed hop expansion factor, and $H_{\max}$ is the absolute maximum hop count. (ii) All links e_{uv} on the path satisfy $U_{\text{uv}}^{(t)} < \tau_{\text{healthy}}$, where $\tau_{\text{healthy}} \in (0, 0.5)$ is the healthy utilization threshold, and $\tau_{\text{healthy}} < \tau_{\text{cong}}$. All physical links contained in the adopted healthy paths are included in the reward set $L_{\text{reward}}^{(t)}$, and a reward update is executed on these links:

$$C_{kl}^{(t+1)} = \max(C_{\min}, C_{kl}^{(t)}(1 - \beta)), \forall e_{kl} \in L_{\text{reward}}^{(t)} \tag{10}$$

where $\beta \in (0, 1)$ is the reward intensity coefficient. Conversely, if $L_{\text{cong}}^{(t)} = \varnothing$, indicating no obvious congestion in the current network, the algorithm ceases reward–punishment operations and transitions to the active exploration mechanism described in Section 3.3 to seek potentially superior routing solutions.

3.3. Active Probing Strategy Under Load Balancing Scenarios

When the main process detects $L_{\text{cong}}^{(t)} = \varnothing$, it indicates that the network is in a low-load or traffic-balanced state at round t, rendering the reward–punishment mechanism ineffective for providing optimization directions. To reduce the risk of the algorithm becoming trapped in local optima, an active exploration mechanism is introduced. Let $\tau_{\text{low}} \in (0, \tau_{\text{healthy}})$ be the low utilization threshold. For all physical links $e_{ij} \in E$ satisfying $U_{ij}^{(t)} < \tau_{\text{low}}$, an offset δ is randomly selected from a preset set of negative integer step sizes $\Delta_{\text{dec}} \subset \mathbb{Z}_{<0}$, and their costs are updated as:

$$C_{ij}^{(t+1)} = \max(C_{\min}, C_{ij}^{(t)} + \delta), \delta \in \Delta_{\text{dec}} \tag{11}$$

Among the remaining physical links not receiving rewards, a subset is randomly selected with proportion $\rho \in (0, 1)$ for random perturbation: for each selected link e_{kl}, an offset ε is randomly chosen from a symmetric integer perturbation set $\Delta_{\text{pert}} \subset \mathbb{Z} \backslash \{0\}$, and its cost undergoes truncated update:

$$C_{kl}^{(t+1)} = \text{clip}(C_{kl}^{(t)} + \varepsilon, C_{\min}, C_{\max}) \tag{12}$$

where $\text{clip}(x, a, b) = \min(b, \max(a, x))$ denotes restricting x within the interval $[a, b]$.

3.4. Perturbation and Soft Restart Mechanism

In cases where network congestion exists, if the optimization process fails to improve global delay over consecutive rounds, it may have become trapped in a locally suboptimal solution. To address this, an intelligent soft restart mechanism is designed. When performance metrics show no improvement over multiple rounds, partial link costs are randomly reset to escape the current search region.

Specifically, the algorithm maintains a stagnation counter $\kappa^{(t)}$. In the t-th iteration, if the current total delay $D^{(t)}$ is not better than the historical best value D^*, then $\kappa^{(t)} = \kappa^{(t-1)} + 1$; otherwise, $\kappa^{(t)}$ is reset to 0 and $D^* = D^{(t)}$ is updated. When $\kappa^{(t)} \geq \kappa_{\text{th}}$ (where $\kappa_{\text{th}} \in \mathbb{Z}_{>0}$ is the stagnation threshold), a soft restart operation is triggered with probability $p_{\text{reset}} \in (0, 1)$.

Upon triggering, a subset $\mathcal{E}_{\text{reset}}^{(t)} \subseteq E$ of physical links is randomly selected with proportion $\rho_{\text{reset}} \in (0, 1)$ (containing at least one link), and the costs of these links are independently reset to random integer values within the interval $[C_{\min}, C_{\max}]$, i.e.,:

$$C_{ij}^{(t+1)} = \xi_{ij}, \forall e_{ij} \in \mathcal{E}_{\text{reset}}^{(t)} \tag{13}$$

where $\xi_{ij} \in \{C_{\min}, C_{\min} + 1, \cdots, C_{\max}\}$ are uniformly randomly selected integer values, while other links remain unchanged. Integrating the aforementioned mechanisms, we consolidate them into unified Algorithm 1, which formally describes the complete closed loop from state perception to cost update.

Algorithm 1 Congestion-Aware Adaptive Link Cost Optimization Algorithm

Require: Network topology $G(V, E)$, traffic demand set $\mathcal{T}$, max iterations $T_{\max}$, initial cost c_0, coefficients α, β
Ensure: Optimal link cost matrix $\mathbf{C}^*$

```
Initialize t ← 0, D* ← ∞, κ ← 0
Set C_ij^(0) ← c_0 for all e_ij ∈ E
Compute initial paths P^(0) and delay D^(0); update D* ← D^(0), C* ← C^(0)
while t < T_max do
    t ← t + 1
    Calculate loads L_ij^(t), utilizations U_ij^(t), and identify congested set L_cong^(t) via Equation (8)
    if L_cong^(t) ≠ ∅ then
        Increase C_ij for e_ij ∈ L_cong^(t) via Equation (9)
        Reroute affected flows T_aff and reward adopted links via Equation (10)
    else
        Perform random perturbation on low-load links as described in Section 3.3
    end if
    Update paths P^(t) and measure delay D^(t)
    if D^(t) < D* then
        Update D* ← D^(t), C* ← C^(t), κ ← 0
    else
        κ ← κ + 1
        if κ ≥ κ_th then
            Execute soft restart strategy per Section 3.4 and reset κ ← 0
        end if
    end if
end while
return C*
```

4. End-to-End Cost Matrix Prediction

CA-GAR aims to learn the nonlinear mapping from the current network state to the engineering near-optimal link cost matrix C^*. The model's input features are refined based on the initial routing state, i.e., utilizing the initial path set $\mathcal{P}^{(0)}$ derived from the initial cost matrix $C^{(0)}$ and its load distribution; while the engineering near-optimal cost matrix C^* obtained in Section 2 serves as the supervised label for model training. It should be clarified that the learning objective of the model is to approximate the strategic logic embedded in the heuristic optimization process described in Section 2, rather than to pursue theoretical global optimality, thereby reproducing the performance of that strategy with extremely low computational overhead during online inference. Let $\mathcal{P}^{(0)} = \{p_1^{(0)}, p_2^{(0)}, \ldots, p_m^{(0)}\}$ be the initial path set calculated via the shortest path algorithm based on $C^{(0)}$, where m is the total number of service flows. Each path $p^{(0)}$ consists of an ordered node sequence $\{v_k\}$ and edge sequence $\{e_{kl}\}$. We will extract multi-dimensional features reflecting dynamic network load, congestion risk, and topological importance from $\mathcal{P}^{(0)}$ to drive the model to predict link costs approaching C^*. In feature calculation, define the indicator function $I(\cdot)$: it takes the value 1 if the condition holds, otherwise 0.

4.1. Node Feature Construction Based on Multi-Dimensional Load Perception

To describe the state of node $vi \in V$ under the current initial service flow distribution, we extract five features from $\mathcal{P}^{(0)}$ to form the vector $\mathbf{f}i$. These features reflect the node's forwarding activity, load pressure, and local congestion risk under specific traffic patterns. The node feature vector is defined as:

$$\mathbf{f}_i = \left[f_i^{(1)}, f_i^{(2)}, f_i^{(3)}, f_i^{(4)}, f_i^{(5)} \right] \tag{14}$$

The components sequentially represent: effective in-degree, effective out-degree, service load intensity, node path participation, and neighbor load variance. To simplify formulas, let $s_{p^{(0)}}$ denote the packet size of path $p^{(0)}$, $\tau_{p^{(0)}}$ the transmission interval, and K_i the node processing capacity. The specific physical interpretations and calculation formulas for these features are presented in Table 1.

Table 1. Network Node Features and Their Calculation Methods.

Feature Definition	Calculation Formula
$f_i^{(1)}$: Times node v_i is pointed to as a non-source in $\mathcal{P}^{(0)}$.	$f_i^{(1)} = \sum_{p^{(0)} \in \mathcal{P}^{(0)}} \sum_{(v_u, v_j) \in p^{(0)}} I(v_j = v_i, v_i \neq p_{\text{src}}^{(0)})$
$f_i^{(2)}$: Times node v_i points to others as a non-destination in $\mathcal{P}^{(0)}$.	$f_i^{(2)} = \sum_{p^{(0)} \in \mathcal{P}^{(0)}} \sum_{(v_j, v_w) \in p^{(0)}} I(v_j = v_i, v_i \neq p_{\text{dst}}^{(0)})$
$f_i^{(3)}$: Normalized traffic sum forwarded by v_i as an intermediate node.	$f_i^{(3)} = \sum_{p^{(0)} \in \mathcal{P}^{(0)}} \frac{s_{p^{(0)}}}{\tau_{p^{(0)}} K_i} I(v_i \in p^{(0)} \setminus \{p_{\text{src}}^{(0)}, p_{\text{dst}}^{(0)}\})$
$f_i^{(4)}$: Total occurrences of v_i across all paths in $\mathcal{P}^{(0)}$.	$f_i^{(4)} = \sum_{p^{(0)} \in \mathcal{P}^{(0)}} I(v_i \in p^{(0)})$
$f_i^{(5)}$: Variance of load intensity among neighbors of v_i.	$f_i^{(5)} = \text{Var}\left(\{f_j^{(3)} \mid e_{ij} \in E\}\right)$

Note: $p_{\text{src}}^{(0)}$ and $p_{\text{dst}}^{(0)}$ are the source and destination nodes of path $p^{(0)}$.

4.2. Link Feature Extraction Based on Utilization Perception

For each edge $e_{ij} = (v_i, v_j) \in E$, three key features are extracted from $\mathcal{P}^{(0)}$ to form the vector $\mathbf{g}_{ij}$, capturing the link's utilization, importance, and surrounding environmental status under the current scenario. The edge feature vector is defined as:

$$\mathbf{g}_{ij} = \left[g_{ij}^{(1)}, g_{ij}^{(2)}, g_{ij}^{(3)} \right] \tag{15}$$

The components sequentially represent edge utilization, link path participation, and average neighbor edge utilization. Here, B_{ij} is the effective bandwidth of the link. The specific physical interpretations and calculation formulas for these features are presented in Table 2.

Table 2. Network Edge Features and Their Calculation Methods.

Feature Definition	Calculation Formula
$g_{ij}^{(1)}$: Sum of normalized traffic flowing through e_{ij} in $\mathcal{P}^{(0)}$.	$g_{ij}^{(1)} = \sum_{p^{(0)} \in \mathcal{P}^{(0)}} \left(\frac{s_{p^{(0)}}}{\tau_{p^{(0)}} B_{ij}} \cdot I(e_{ij} \in p^{(0)}) \right)$
$g_{ij}^{(2)}$: Total occurrences of edge e_{ij} across all paths in $\mathcal{P}^{(0)}$.	$g_{ij}^{(2)} = \sum_{p^{(0)} \in \mathcal{P}^{(0)}} I(e_{ij} \in p^{(0)})$
$g_{ij}^{(3)}$: Average utilization of neighboring edges originating from endpoints of e_{ij}.	$g_{ij}^{(3)} = \frac{\sum_{e_{kl} \in N(e_{ij})} g_{kl}^{(1)}}{\lvert N(e_{ij}) \rvert}$

Note: $N(e_{ij}) = \{e_{kl} \mid (k = i \vee k = j) \wedge e_{kl} \neq e_{ij}\}$ denotes the set of neighboring edges, i.e., all edges starting from v_i or v_j excluding e_{ij} itself. $|N(e_{ij})|$ is the number of such neighbors.

4.3. Node-Edge Feature Fusion

To enable the model to simultaneously encode node attributes and edge dynamic relationships, we design a feature fusion scheme. For each node v_i, statistical aggregates (mean and maximum) of its incoming and outgoing edge features are computed, covering three dimensions: edge utilization, path participation, and average neighbor edge utilization.

This generates a 6-dimensional incoming edge aggregated feature $\mathbf{g}_{\text{in},i}$ and a 6-dimensional outgoing edge aggregated feature $\mathbf{g}_{\text{out},i}$. Concatenating the original node features with both yields a 17-dimensional fused feature vector $\mathbf{h}_i$. The features of all nodes in the network constitute the input matrix $\mathbf{H} \in \mathbb{R}^{N \times 17}$, where $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_N]^T$. The fused feature vector for a single node is expressed as:

$$\mathbf{h}_i = [\mathbf{f}_i, \mu(\mathbf{g}_{\text{in}}^{(1:3)}), \max(\mathbf{g}_{\text{in}}^{(1:3)}), \mu(\mathbf{g}_{\text{out}}^{(1:3)}), \max(\mathbf{g}_{\text{out}}^{(1:3)})] \tag{16}$$

Here, $\mu(\cdot)$ and $\max(\cdot)$ denote the mean and maximum operations applied to the specified dimensional feature sets, respectively. This fused vector $\mathbf{h}_i$ comprehensively preserves both the node's intrinsic load state and the congestion signals from its connected links.

4.4. CA-GAR Model

The CA-GAR model employs a depth-adjustable multi-head attention mechanism. By stacking multiple attention layers and integrating normalization and regularization techniques, it automatically learns the optimal spatial scale for feature aggregation, thereby balancing representation capability with generalization risk.

The model input consists of the node fused feature matrix $\mathbf{H}^{(0)} \in \mathbb{R}^{N\times 17}$ and the physical adjacency matrix $\mathbf{A}^{\text{phy}}$. Here, the superscript (0) denotes the initial input layer of the network, corresponding to the 17-dimensional fused feature vectors $\mathbf{h}_i$ generated in Section 4.3 (i.e., $d_{\text{in}} = 17$), where N represents the total number of nodes. Assuming the model comprises L graph attention layers, the computation process for the l-th layer ($l = 1, \ldots, L$) is described as follows.

First, neighbor information is aggregated via a multi-head attention mechanism. For the k-th attention head, the output features are computed similarly to Equation (5). The outputs of all K heads are concatenated to obtain the raw output of the l-th layer, denoted as $\mathbf{H}^{(l)}_{\text{raw}}$. To mitigate feature distribution shifts and gradient vanishing in deep networks caused by message passing, we sequentially introduce an ELU activation function, Layer Normalization (LayerNorm), and Dropout regularization after each attention operation. Specifically, ELU ensures non-linearity while maintaining zero-mean characteristics; LayerNorm enhances adaptability to varying topological scales; and Dropout breaks redundant coupling between features by randomly deactivating neurons, effectively alleviating the common "over-smoothing" problem in Graph Neural Networks (GNNs). The final output $\mathbf{H}^{(l)}$ of the l-th layer is defined as:

$$\mathbf{H}^{(l)} = \text{Dropout}\Big(\text{LayerNorm}\Big(\text{ELU}\Big(\mathbf{H}^{(l)}_{\text{raw}}\Big)\Big)\Big). \tag{17}$$

In this formulation, ELU compresses outlier feature values via non-linear transformation; LayerNorm normalizes node feature dimensions to accelerate convergence across graphs of different sizes; and Dropout randomly discards feature components with probability p to prevent over-smoothing and improve generalization. Intermediate GAT layers further fuse higher-order neighbor information based on the preceding layer's output, modeling global network structures and complex service interaction patterns.

For the final layer (the L-th layer), to preserve the linear correlation between node features and the downstream link cost prediction task—and to avoid excessive non-linear transformations weakening prediction accuracy—we omit the ELU activation function. Only LayerNorm and Dropout operations are retained to generate the final node embedding representation $\mathbf{Z} \in \mathbb{R}^{N\times d_{\text{out}}}$:

$$\mathbf{Z} = \text{Dropout}\Big(\text{LayerNorm}\Big(\|_{k=1}^{K'} \text{GATHead}^{(k)}(\mathbf{H}^{(L-1)})\Big)\Big) \tag{18}$$

where K' can be set to 1 or maintained as multi-head depending on task requirements.

To convert the node-level state representation $\mathbf{Z}$ into edge-level dynamic link cost predictions, the model incorporates an edge-feature-based decoder. For any potential edge $e_{ij} = (v_i, v_j)$, we first concatenate the source node feature $\mathbf{z}_i$ and the target node feature $\mathbf{z}_j$ to construct an edge-level interaction feature vector $\mathbf{z}_{ij} = [\mathbf{z}_i \| \mathbf{z}_j]$. Subsequently, a two-layer Multi-Layer Perceptron (MLP) directly outputs the predicted link cost $\hat{y}_{ij}$:

$$\hat{y}_{ij} = \mathbf{w}_2^T \cdot \text{ReLU}(\mathbf{W}_1 \mathbf{z}_{ij} + \mathbf{b}_1) + b_2, \tag{19}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{hidden}}\times 2d_{\text{out}}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{\text{hidden}}}$, $\mathbf{w}_2 \in \mathbb{R}^{d_{\text{hidden}}}$, and $b_2 \in \mathbb{R}$ are the learnable parameters for the first and second layers, respectively, with d_{hidden} denoting the hidden layer dimension.

Since link cost prediction is a regression task and the numerical range of the optimal cost matrix $\mathbf{C}^*$ depends on network scale and optimization results, the model's output layer utilizes a linear activation (implicitly via the final affine transformation without a bounding non-linearity like Sigmoid). This allows the predicted values $\hat{y}_{ij}$ to adaptively fit label data with arbitrary distributions. The predicted values $\hat{y}_{ij}$ for all edges constitute the

prediction matrix $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times N}$, which directly characterizes the dynamic congestion level or engineering near-optimal cost value of each link in the network. Its numerical range remains consistent with the engineering near-optimal matrix C^* generated in Chapter 3. The overall architecture and data flow of the model are illustrated in Figure 3.

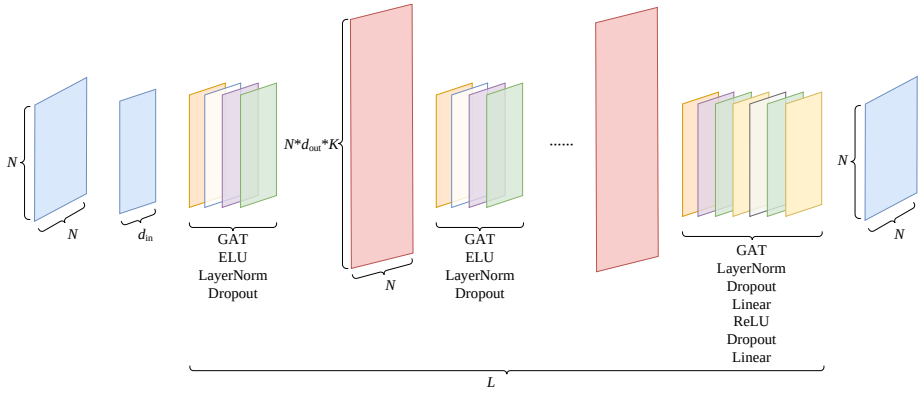


Figure 3. Architecture of the CA-GAR Model.

The blue sections in Figure 3 represent the input and output of the model. Specifically, the left blue module takes as input the node fusion feature matrix and the physical adjacency matrix, while the right blue module outputs the predicted link cost matrix via the decoder. The orange module is the multi-head graph attention layer. The white module is the ELU activation function. The purple module is the layer normalization operation. The green module is Dropout regularization. The pink modules denote the intermediate feature representations after stacking the GAT layer, ELU, layer normalization, and Dropout, which are then propagated into subsequent stacking modules. The yellow module is the linear layer. These modules are stacked sequentially to form the end-to-end mapping network of CA-GAR.

4.5. Adaptive Hyperparameter Optimization Strategy Based on Optuna

To address the strong coupling between structural and training parameters and the vast search space inherent in Graph Attention Networks, we introduce an adaptive hyperparameter optimization system based on the Optuna framework and propose a Phased Hierarchical Tuning Strategy [29].

This strategy defines three distinct parameter spaces: the structural parameter space $\mathcal{S}_{\text{struct}}$, the training dynamics parameter space $\mathcal{S}_{\text{train}}$, and the regularization parameter space $\mathcal{S}_{\text{reg}}$. The optimization process is divided into three sequential stages:

(i) Network Structure Optimization: Searching within $\mathcal{S}_{\text{struct}}$ for the optimal combination of hidden layer dimension $d_{\text{hidden}} \in [d_{\min}, d_{\max}] \cap \mathbb{Z}$, number of attention heads $K \in [K_{\min}, K_{\max}] \cap \mathbb{Z}$, and number of GAT layers $L \in [L_{\min}, L_{\max}] \cap \mathbb{Z}$, aiming to determine the optimal network topology.

(ii) Training Dynamics Optimization: Based on the optimal structure θ^*_{struct} determined in the first stage. In the following text, we use a superscript asterisk to mark the optimal parameter values obtained through phased optimization. On this basis, fine-tuning the learning rate $\eta \sim \mathcal{U}_{\log}(\eta_{\min}, \eta_{\max})$ (log-uniform distribution) and batch size $B_{\text{size}} \in \{2^k \mid k \in \mathbb{Z}, B_{\min} \leq 2^k \leq B_{\max}\}$ within $\mathcal{S}_{\text{train}}$ to adapt to the model's convergence characteristics.

(iii) Regularization Strategy Optimization: Building upon the first two stages, finely adjusting the Dropout ratio $p \in [p_{\min}, p_{\max}]$ within $\mathcal{S}_{\text{reg}}$ to optimize the model's generalization performance.

The framework of this phased symbolic hyperparameter optimization is illustrated in Figure 4.

The optimization algorithm employs Optuna's built-in TPE (Tree-structured Parzen Estimator), which dynamically constructs probability density functions based on historical trial results to guide sampling by maximizing the expected improvement [30]. The study defines the validation link cost prediction loss $\mathcal{L}_{\text{val}}(\theta)$ as the minimization objective function. A maximum number of training epochs $E_{\max}$ is set, and an early stopping mechanism is enabled: if the validation loss does not decrease significantly within E_{patience} consecutive epochs, the current trial is terminated. Furthermore, intermediate value pruning is utilized to automatically discard trial combinations that perform below the historical average during the early stages of training. Through these strategies, the model converges to a globally optimal or near-optimal hyperparameter configuration $\theta^* = \{\mathcal{S}^*_{\text{struct}}, \mathcal{S}^*_{\text{train}}, \mathcal{S}^*_{\text{reg}}\}$.

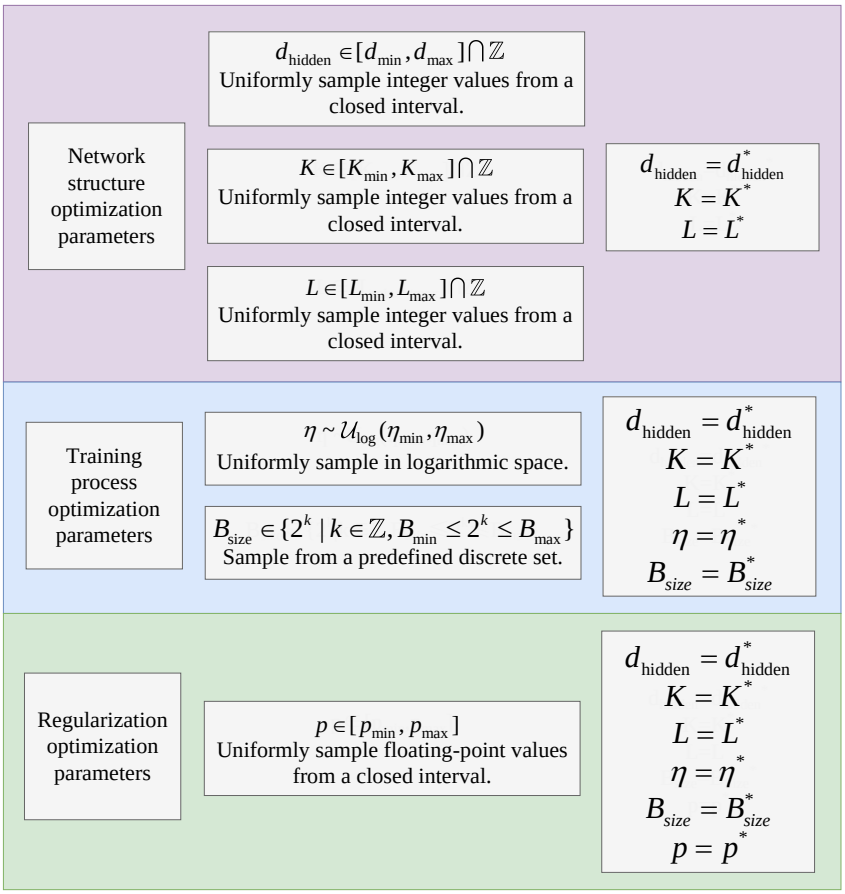


Figure 4. Framework of Phased Symbolic Hyperparameter Optimization.

5. Experimental Evaluation and Results Analysis

5.1. Experimental Setup and Computational Efficiency Analysis

High-fidelity network simulation scenarios were constructed using the NS-3 (Network Simulator 3) discrete-event network simulation platform (Version 3.37, available online: https://www.nsnam.org/ (accessed on 20 April 2026)). The deep learning models were implemented using the PyTorch 2.8.0 framework with CUDA 12.8, with training and inference performed on servers equipped with NVIDIA GPUs.

The simulation scenario covers an area of 150 × 150 km with N = 32 nodes. The physical layer configuration is defined as follows: carrier frequency f_c = 1.2 GHz, channel bandwidth B = 16 MHz, transmission power P_t = 40 dBm, and data rate of 16 Mbps. Nodes are positioned at an altitude of 1000 m, equipped with omnidirectional antennas (antenna

height 10 m, gain 10 dB). The channel propagation follows the Two-Ray ground reflection model. The receiver parameters include an SNR threshold of 10 dB, a noise figure of 10 dB, and a receiver sensitivity of -91 dBm. The MAC layer employs the Time Division Multiple Access (TDMA) protocol.

To evaluate the model's generalization performance, three distinct traffic load scenarios were constructed: light, medium, and heavy. Each scenario independently generated 5000 samples. Traffic flows were generated between random, non-repeating source–destination pairs. Packet sizes follow a uniform distribution within $[4000, 5000]$ bits, and inter-packet intervals are randomly generated within $[0.02, 0.2]$ s.

The number of concurrent flows, N_{flow}, is dynamically calculated based on the total number of links L and the average path length H. By setting specific ranges for the global network load factor (Light: 0.1–0.2, Medium: 0.2–0.35, Heavy: 0.35–0.55), N_{flow} is randomly sampled within the range: $N_{\text{flow}} \in \left[\frac{P \cdot U_{\min}}{H}, \frac{P \cdot U_{\max}}{H}\right]$ where $U_{\min}$ and $U_{\max}$ denote the lower and upper bounds of the load factor for the current scenario, respectively, and P represents a scaling factor related to network capacity. All generated flows undergo dual validation against link bandwidth and node capacity constraints; any flows requiring infeasible paths are discarded to ensure physical legitimacy.

Ground truth labels were generated using a phased heuristic optimization algorithm. A perturbation ratio of $\rho = 0.3$ was set, and a soft restart mechanism was introduced: if the total delay fails to improve over 10 consecutive iterations, there is a probability $p_{\text{reset}} = 0.6$ to reset 30% of the link cost values. The algorithm searches within the space $[1, 100]$ to find the optimal solution. The resulting optimal cost matrix is linearly normalized to the $[0, 1]$ interval to serve as the supervision labels for the CA-GAR model, thereby eliminating dimensional discrepancies and enhancing training stability.

For each load scenario, the 5000 independently generated samples were divided into training, validation, and test sets with a ratio of 7:1.5:1.5. The training set was used for model parameter updates, the validation set was used for hyperparameter tuning and early stopping determination, and the test set was used only for final performance evaluation after model training was completed. All samples were generated based on the same network topology, but the source–destination node pairs, number of flows, packet sizes, and transmission intervals of each sample were independently and randomly generated. The traffic patterns in the test set never appeared during the training process, and the test set did not participate in any parameter updates.

In the NS-3 simulation environment, the duration of a single simulation was set to 5 min, covering the complete cycle from traffic flow initiation to network stabilization. In the heuristic label generation phase, a total of 400 iterations were executed. In each iteration, an NS-3 simulation was run according to the current link cost configuration to obtain performance metrics such as end-to-end delay. Due to variations in the stability of path configurations across different iterations, not every iteration triggered an actual simulation: when changes in link costs did not cause any alteration in traffic flow paths, the simulation results remained identical to those of the previous iteration and could be reused directly. Statistics show that, out of 400 iterations, the number of iterations that actually triggered path changes and executed simulations ranged from approximately 80 to 180, depending on the load scenario. The runtime of a single simulation depended on the number of traffic flows and path complexity, averaging approximately 22 s in light-load scenarios, 34 s in medium-load scenarios, and 80 s in heavy-load scenarios.

After model training is completed, the system enters the online inference stage. The CA-GAR model directly outputs the link cost matrix through forward propagation based on the current network state, and then computes the shortest path for each source–destination pair. In the 32-node topology, a single inference takes approximately 15 to 25 ms, includ-

ing feature extraction, GAT forward propagation, and path computation. In contrast, if heuristic optimization were used for online decision-making, each decision would require iteratively running hundreds of simulations, with total time consumption reaching tens of minutes or even hours, making it difficult to meet the real-time requirements of dynamic networks. Therefore, by consolidating the heuristic optimization strategy into neural network parameters through offline training, the online inference stage can obtain near-optimal routing configurations with only millisecond-level forward propagation, offering significant practical value.

5.2. Near-Optimality Validation of the Congestion-Aware Dynamic Link Cost Optimization Method

Traditional static routing methods adopt a shortest-path-first strategy to independently select paths for each service flow. This local-perspective decision-making lacks awareness of the global network state. When multiple service flows concurrently compete for network resources, each flow tends to choose the same low-cost path, which can easily lead to congestion hotspots on certain critical links due to excessive traffic concentration. In fact, if some service flows appropriately detour through idle paths, although the hop count of individual flows increases, the pressure on core links can be effectively alleviated, achieving mutual benefits for all flows. As shown in Figure 5, this figure illustrates the variation of the network-wide total delay during the iterative process of the algorithm proposed in Section 3 under a typical scenario. The horizontal axis represents the iteration round, the vertical axis represents the total delay, and scatter points with different colors and shapes mark the optimization mechanisms triggered in each iteration.

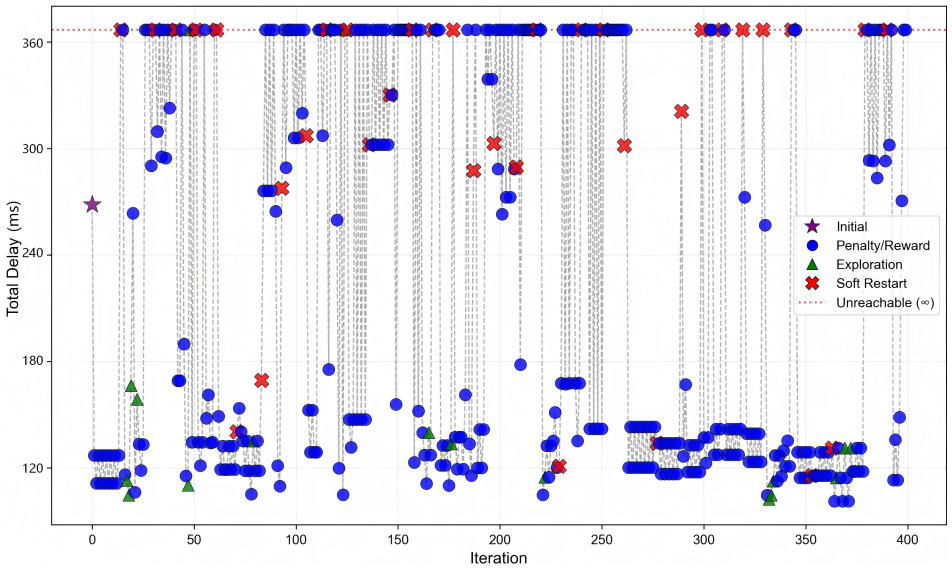


Figure 5. Delay variation and optimization mechanisms of the proposed algorithm.

It can be observed from the figure that the initial delay is at a relatively high level, reflecting the presence of severe local congestion in the network under the initial routing configuration. As the iteration proceeds, the algorithm continuously adjusts link costs under the dominance of the penalty-reward mechanism, and the total delay exhibits a downward trend. When the algorithm enters a performance plateau, the soft restart mechanism is triggered. By randomly resetting a portion of the link costs, it successfully releases the algorithm from local optima, further reducing the delay to a more optimal level. The exploration mechanism plays a role in the congestion-free phase, actively perturbing low-utilization links to create potential opportunities for subsequent optimization. Under the synergistic effect of the above mechanisms, the algorithm can find an engineering-

acceptable high-quality solution within a finite number of iterations, effectively avoiding local optima.

To verify that the congestion-aware dynamic link cost optimization method proposed in Section 3 can produce engineering near-optimal link cost configurations, this subsection compares it with the global optimal solution obtained via exhaustive search. Since exhaustive search requires enumerating all possible path combinations for all service flows, its search space grows exponentially with the number of nodes and the number of flows, making it computationally infeasible on large-scale topologies. Therefore, this subsection adopts a 6-node small-scale topology for validation, under which the exhaustive search can be completed within an acceptable time.

The core idea of the exhaustive search method is as follows. For each service flow, the top 5 shortest paths are precomputed to construct a candidate path set for each flow. The Cartesian product of all flows' candidate paths is then enumerated to generate all possible path combinations. Each combination is evaluated by performing high-fidelity simulations in the NS-3 network simulator to obtain the network-wide end-to-end total delay, and the path combination achieving the minimal delay is recorded as the global optimal solution. Although this method is computationally expensive, it guarantees optimality within the candidate path space and serves as a benchmark for evaluating the near-optimality of the heuristic algorithm.

The experiment covers three load scenarios: light, medium, and heavy. For each scenario, 15 independent random flow sets are generated. Figure 6 presents a scatter plot comparing the delays of the heuristic algorithm and the exhaustive search optimal solution on a 6-node topology, where the horizontal axis represents the global optimal delay obtained via exhaustive search, the vertical axis represents the delay output by the heuristic algorithm, and different colors represent different load scenarios.

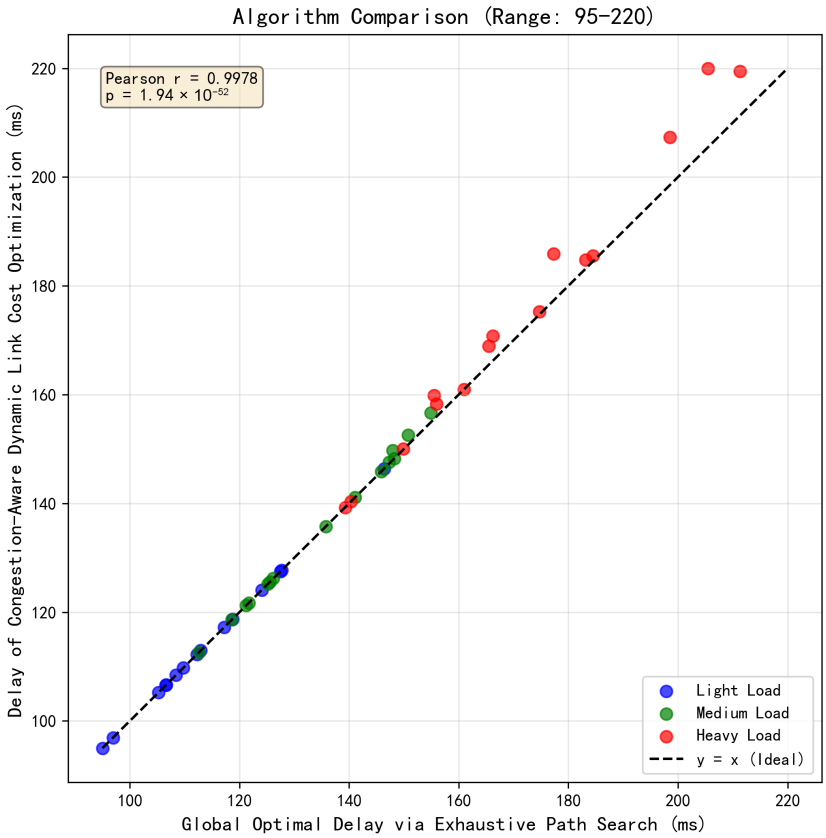


Figure 6. Comparison of heuristic algorithm and exhaustive search optimal delay under different load scenarios.

It can be observed from the figure that all scatter points are closely distributed around the $y = x$ diagonal line, indicating that the delay of the heuristic algorithm is highly consistent with the global optimal solution. Under the light load scenario, the average relative gap of the heuristic algorithm is $0.00\% \pm 0.00\%$, and all samples achieve the theoretical optimal solution. This is because network resources are abundant under light load, making it easy for the algorithm to find the optimal route. Under the medium load scenario, the average relative gap is $0.25\% \pm 0.49\%$, and the maximum gap does not exceed 1.24%. The algorithm can still stably approach the optimal solution. Under the heavy load scenario, the average relative gap is $2.08\% \pm 2.18\%$, and the maximum gap is 7.09%. Although the fluctuation increases slightly, the overall gap remains at a low level, verifying the robustness of the algorithm under congestion scenarios. Furthermore, the Pearson correlation coefficient is calculated to be 0.9978 with a p-value of 1.94×10^{-52}, indicating an extremely strong positive correlation between the heuristic algorithm and the global optimal solution with high statistical significance. To more comprehensively evaluate the near-optimality performance of the proposed algorithm, Table 3 summarizes the statistical results of the relative gaps between the heuristic algorithm and the exhaustive search optimal solution on the 6-node topology.

Table 3. Statistical comparison between heuristic algorithm and exhaustive search optimal solution.

Metric	Value
Total number of test cases	45
Mean relative gap	0.78%
Standard deviation	1.57%
Median	0.00%
Minimum gap	0.00%
Maximum gap	7.09%
Pearson correlation coefficient	0.9978
p-value	1.94×10^{-52}

From the table, it can be seen that the mean relative gap is 0.78% and the median is 0.00%, indicating that the heuristic algorithm directly finds the global optimal solution in more than half of the test scenarios. The standard deviation is 1.57% and the maximum gap is 7.09%, demonstrating that the algorithm performs stably across different load scenarios and maintains performance close to the optimal solution even in the worst case. The above results fully demonstrate that the heuristic algorithm can produce an engineering near-optimal link cost matrix within a finite iteration budget, and its output can serve as a high-quality supervised label for CA-GAR model training.

5.3. Hyperparameter Optimization Results and Analysis

To determine the optimal configuration for the CA-GAR model, a phased hierarchical tuning strategy was implemented based on the Optuna framework. The search spaces for parameters in each phase were defined as follows: In the structural parameter space, the hidden dimension $d_{\text{hidden}} \in [32, 128] \cap \mathbb{Z}$, the number of attention heads $K \in [2, 8] \cap \mathbb{Z}$, and the number of GAT layers $L \in [2, 5] \cap \mathbb{Z}$. In the training dynamics parameter space, the learning rate $\eta \sim \mathcal{U}_{\log}(10^{-4}, 10^{-2})$, and the batch size $B_{\text{size}} \in \{16, 32, 64, 128\}$. In the regularization parameter space, the Dropout ratio $p \in [0.1, 0.5]$.

The optimization process consisted of three phases, with each phase executing up to 35 independent trials. A phase was terminated early if no significant performance improvement was observed over 10 consecutive trials. The maximum number of training epochs per trial was set to 30, with an early stopping mechanism enabled (patience $E_{\text{patience}} = 5$), meaning training was automatically halted if the validation loss did not

decrease for 5 consecutive epochs. The Tree-structured Parzen Estimator (TPE) algorithm was employed for sampling, combined with functional Analysis of Variance (fANOVA) to quantify the variance contribution of each hyperparameter to model performance.

In the first phase, the learning rate, batch size, and Dropout rate were fixed at default values, focusing on searching for the optimal combination of d_{hidden}, K, and L. Figure 7 illustrates the parameter importance distribution based on fANOVA analysis. The results indicate that the number of attention heads K accounts for an importance ratio of up to 0.62, significantly exceeding that of the hidden dimension and the number of layers. This suggests that, under the current network topology, the complementary feature subspace capture capability of the multi-head mechanism is a critical factor in enhancing model performance. Figure 8 further displays the optimization trajectory for this phase, where the red line represents the historical best validation loss, and the blue scatter points denote individual trial results. As iterations proceeded, the search space gradually converged to a low-loss region, ultimately determining the optimal combination of structural parameters.

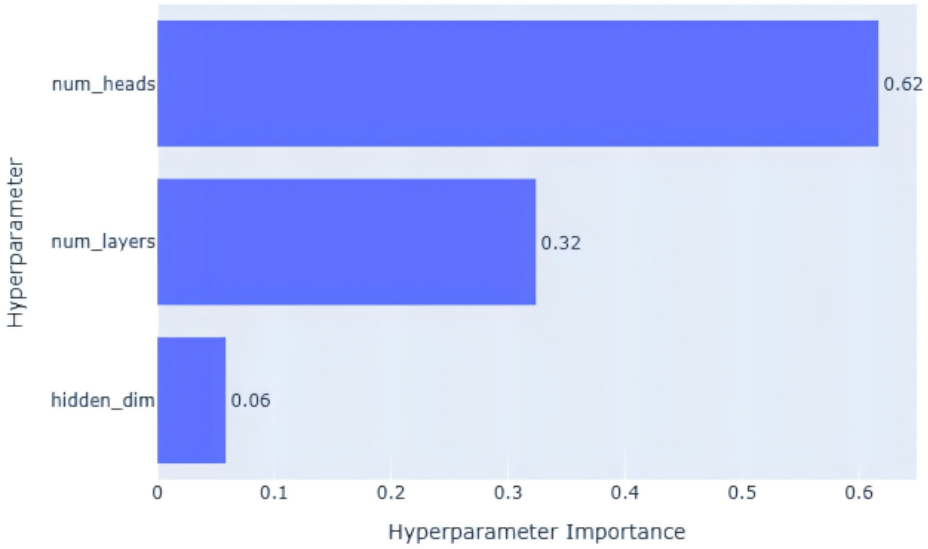


Figure 7. Sensitivity analysis of network structural parameters.

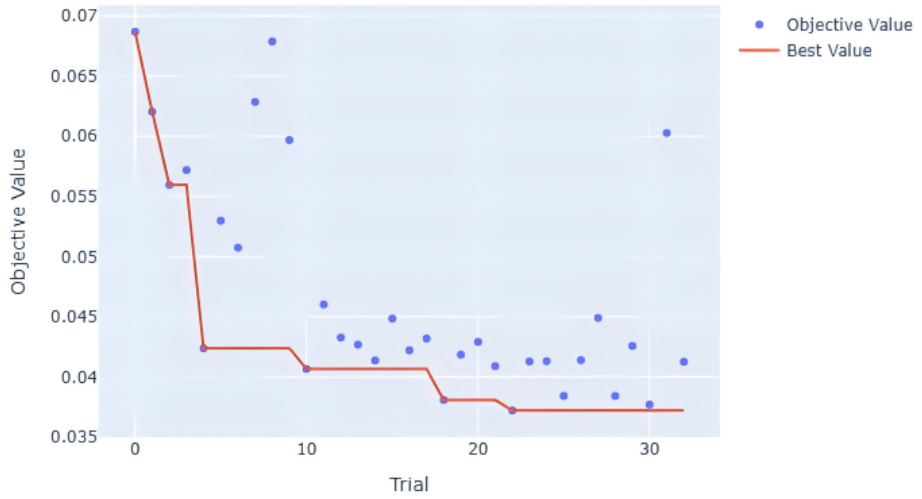


Figure 8. Convergence trajectory of hyperparameter optimization.

Based on the optimal structure determined in the first phase, subsequent phases focused on fine-tuning the learning rate, batch size, and Dropout rate. Figure 9 integrates the parameter importance and optimization trajectories from these latter two phases.

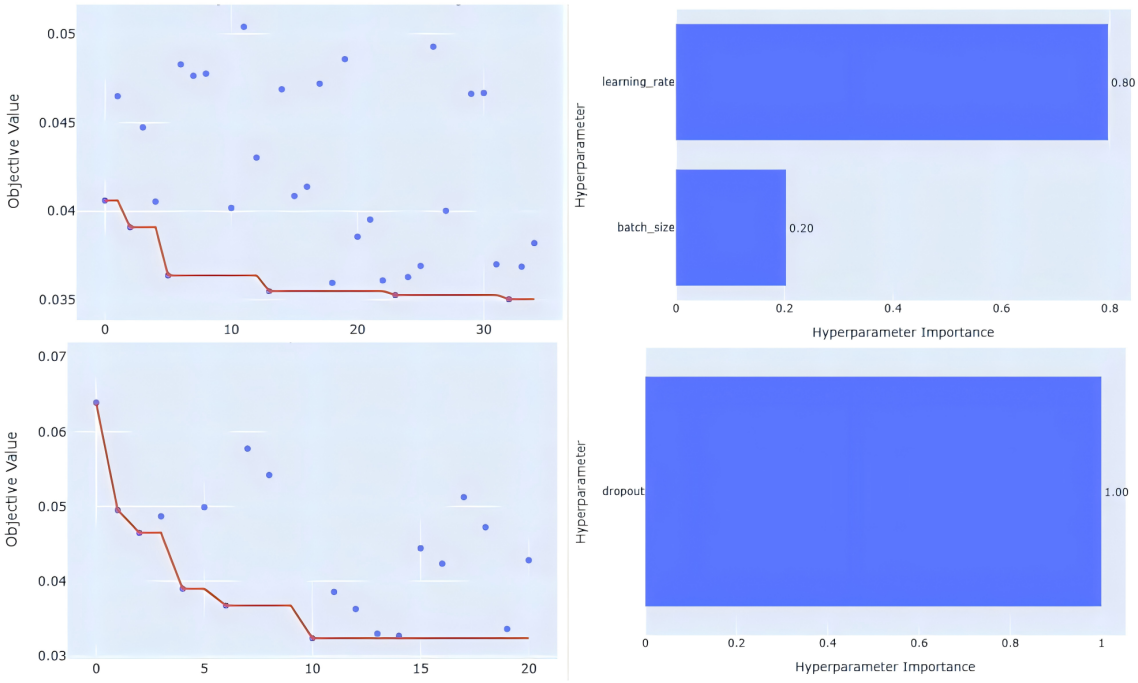


Figure 9. Comprehensive overview of the training process and regularization parameter tuning.

As illustrated in the figures, among the training dynamics parameters, the learning rate η accounts for an importance ratio of 0.80, significantly higher than that of the batch size B_{size} (0.20). This confirms the decisive role of the learning rate in determining the convergence path of gradient descent. The third phase focused exclusively on optimizing the Dropout rate p as a single parameter; consequently, its importance ratio is naturally 1.0. The optimization trajectory reveals that smaller Dropout values better balance the risks of underfitting and overfitting, which aligns with the inherent robustness observed in the dataset.

Synthesizing the results from all three phases, the optimal hyperparameter configuration for the CA-GAR model is presented in Table 4. This configuration will be fixed for all subsequent comparative and ablation experiments to ensure fairness and reproducibility of the evaluation results.

Table 4. Optimal Parameter Configuration.

Parameter	Value
d_{hidden}	71
K	8
L	2
η	0.001128
B_{size}	16
p	0.106786

5.4. Model Training Convergence and Generalization Performance Analysis

Based on the optimal hyperparameter configuration determined in Section 5.2, denoted as $\theta^* = \{d^*_{\text{hidden}}, K^*, L^*, \eta^*, B^*_{\text{size}}, p^*\}$, we constructed the CA-GAR model and performed end-to-end training. The specific implementation process is as follows: the model architecture was initialized using d^*_{hidden}, K^*, L^*, and p^*, while the Adam (Adaptive Moment Estimation) optimizer was employed with a learning rate set to η^*. The training process was configured with a maximum of $E_{\text{total}} = 100$ epochs and a batch size of B^*_{size}.

In each epoch e ($e = 1, \ldots, E_{\text{total}}$), the model first switched to training mode to traverse the training dataset, executing forward propagation, loss calculation, and backpropagation to update parameters and record the average training loss $\mathcal{L}_{\text{train}}^{(e)}(\theta)$. Subsequently, the model switched to evaluation mode to perform inference on the validation set and calculate the average validation loss $\mathcal{L}_{\text{val}}^{(e)}(\theta)$. To prevent overfitting and retain the best state, an early stopping mechanism was introduced: training was terminated if $\mathcal{L}_{\text{val}}^{(e)}(\theta)$ failed to decrease for $E_{\text{patience}} = 15$ consecutive epochs. Conversely, if the current validation loss outperformed the historical best value $\mathcal{L}_{\text{best}} = \min_{1 \le k \le e}\{\mathcal{L}_{\text{val}}^{(k)}\}$, the model weights were automatically saved to a checkpoint file. Figure 10 illustrates the trajectories of $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ as a function of the number of iterations during the training process.

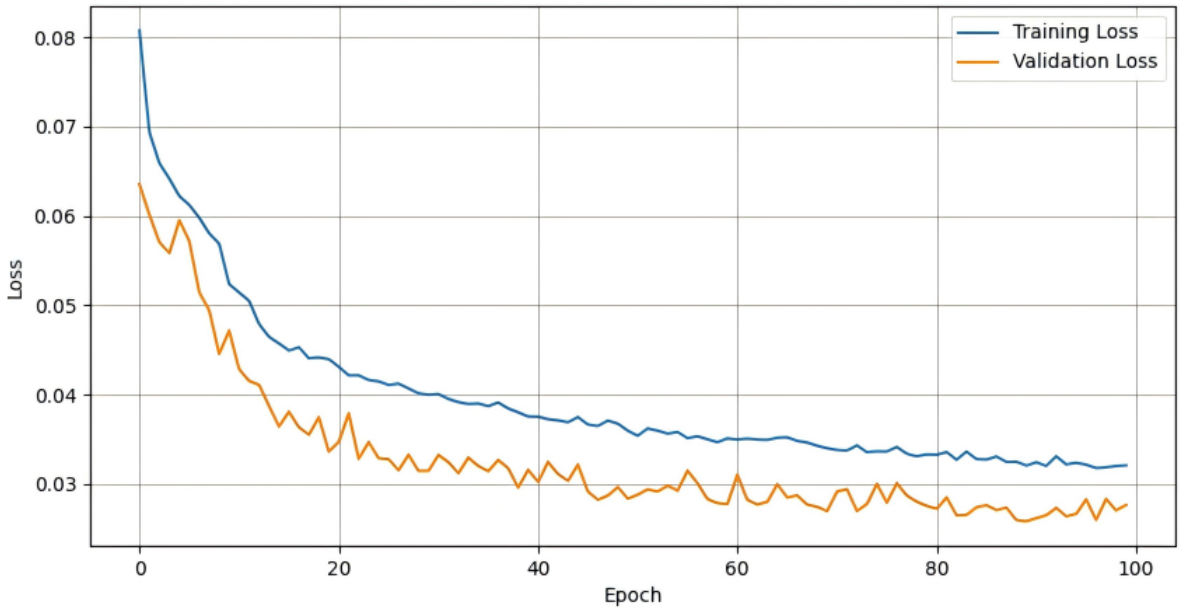


Figure 10. Convergence curves of training and validation loss.

The experimental results demonstrate that both training and validation losses exhibit a robust downward trend as the number of iterations increases, confirming the model's strong capability in extracting congestion features from network topologies. Although the validation curve shows slight oscillations, its overall convergence trend is clear without any rebound. This indicates that, through the phased optimization Dropout strategy and the multi-head mechanism, CA-GAR effectively suppresses overfitting to training samples and possesses generalization capabilities for unseen traffic scenarios.

5.5. Routing Optimization Performance Evaluation and Analysis

We selected typical network topologies under three different load intensities: light, medium, and heavy (as described in Section 5.1) to intuitively evaluate the dynamic scheduling performance of the CA-GAR model. By comparing the default equal-cost routing with the optimized routing based on CA-GAR predicted costs, we focused on analyzing the model's traffic splitting mechanism and congestion avoidance effectiveness in complex environments. Figures 11–13 present the visual comparisons of path distributions under these three load scenarios, respectively. The numbers on nodes indicate node IDs, and different colors represent different traffic flows. Due to the large number of flows, some flows share the same color, and lines may overlap when multiple flows pass through the same link.

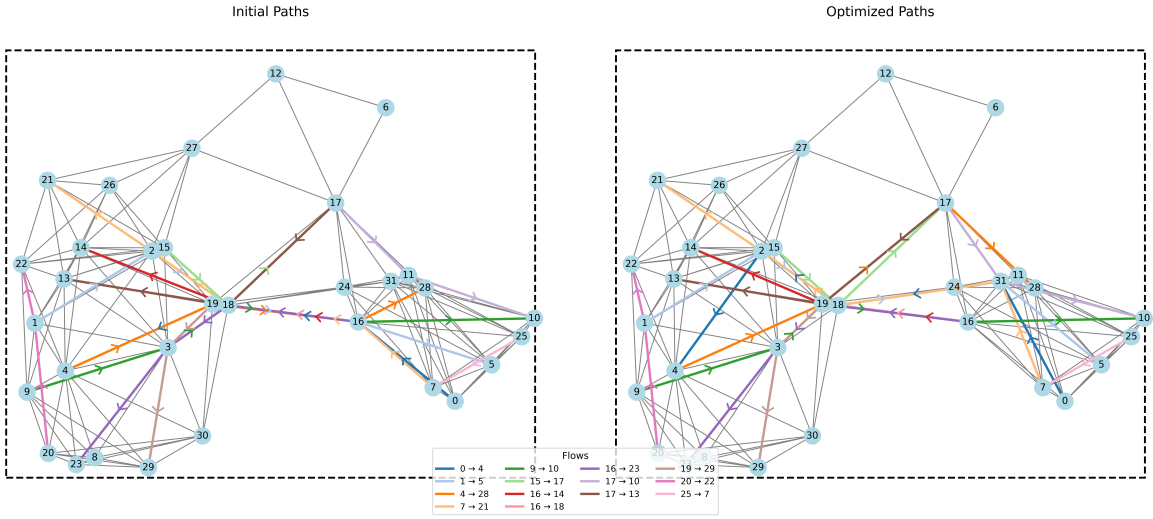


Figure 11. Comparison of initial and optimized path distributions under light-load scenarios.

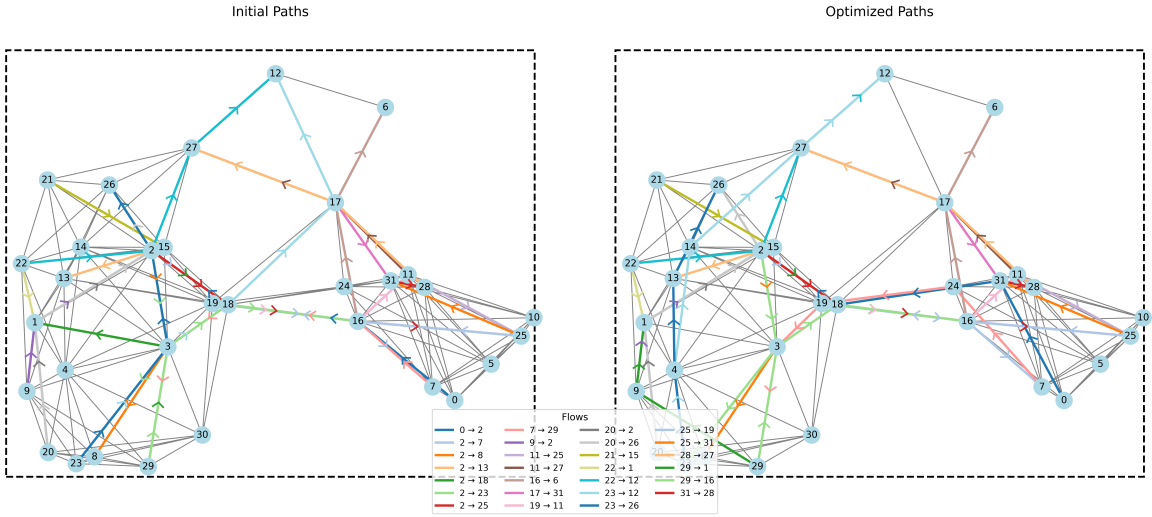


Figure 12. Comparison of initial and optimized path distributions under medium-load scenarios.

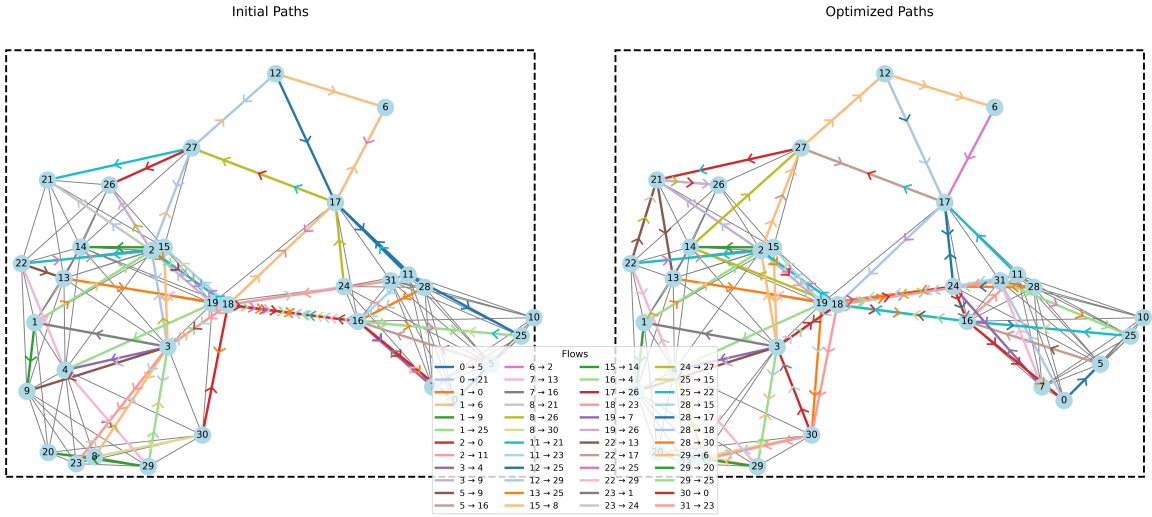


Figure 13. Comparison of initial and optimized path distributions under heavy-load scenarios.

Observing Figure 11 (light-load scenario), it is evident that although the initial paths do not cause severe congestion under low traffic volume, a trend of traffic concentration towards specific nodes is already emerging. In contrast, the optimized paths begin to divert

a portion of the traffic towards edge links, preliminarily demonstrating the capability for load balancing.

As the number of traffic flows increases, entering the medium-load scenario depicted in Figure 12, the deficiencies of the initial routing strategy gradually become apparent. Multiple traffic flows begin to superimpose on specific links, causing a rapid surge in local link utilization. In contrast, the optimized paths demonstrate a more pronounced traffic splitting effect; flows that originally traversed hotspot regions are rescheduled to relatively idle neighboring nodes, thereby alleviating pressure on the core areas.

In the extreme scenario of heavy load depicted in Figure 13, the initial paths exhibit severe "hotspot congestion". A large volume of traffic flows is forced to compete for the same physical path, resulting in extremely dense line concentrations in the core region, which significantly increases the risk of queue buildup and packet loss. In contrast, the optimized paths demonstrate that the CA-GAR model possesses robust global routing reconstruction capabilities.

To validate the effectiveness of the CA-GAR algorithm in dynamic load environments, this section selects two typical baseline algorithms for comparative analysis: initial static routing, as a representative of traditional shortest-path first strategies, which lacks dynamic adjustment capabilities; and QLRA, an intelligent routing method capable of performing path selection under multiple constraints through distributed learning and link state awareness. By comparing the performance of these three approaches on key metrics, this evaluation aims to comprehensively assess the advantages of the proposed algorithm in reducing delay, ensuring reliability, and improving network carrying capacity.

Figure 14 presents a comparison of the network-wide end-to-end delay across 30 different scenarios. The horizontal axis represents the test scenario index, while the vertical axis denotes the total network-wide end-to-end delay. Specifically, the solid red line with solid circles represents the initial static routing algorithm, the orange dashed line with solid triangles represents the QLRA algorithm, and the green dash-dotted line with solid squares represents the proposed CA-GAR algorithm. The analysis indicates that the CA-GAR algorithm achieves the lowest end-to-end delay in the vast majority of scenarios. However, the degree of its advantage over QLRA and the initial static routing is closely related to the network topology, the spatial distribution characteristics of traffic flows, and the load intensity. All subsequent path descriptions are based on the same network topology presented in Figures 11–13 of Section 5.5, which consists of 32 nodes with node indices and link connections as illustrated in the aforementioned path visualization comparison figures. Based on the differences in traffic patterns across the 30 test scenarios, the performance comparison can be summarized into the following four typical cases.

The first category corresponds to scenarios with highly dispersed spatial distribution of traffic flows, such as Scenarios 0, 23, and 28 in Figure 14. In these scenarios, the source–destination node pairs of each traffic flow are far apart from each other in the topology, and the paths exhibit little to no overlap in terms of nodes and links. For example, flow (21,0) is transmitted along path 21-27-17-11-0, flow (30,9) along path 30-3-9, and flow (7,2) along path 7-16-18-2, with no flow sharing any common link or node. Since there is no resource competition on shared links, each flow achieves local optimality by following its shortest path, and the superposition of these local optima exactly constitutes the global optimum. In this case, the routing results of the three algorithms are identical, resulting in no difference in end-to-end delay. This indicates that when the network load is naturally balanced, static routing is sufficiently efficient, leaving limited room for improvement by intelligent routing.

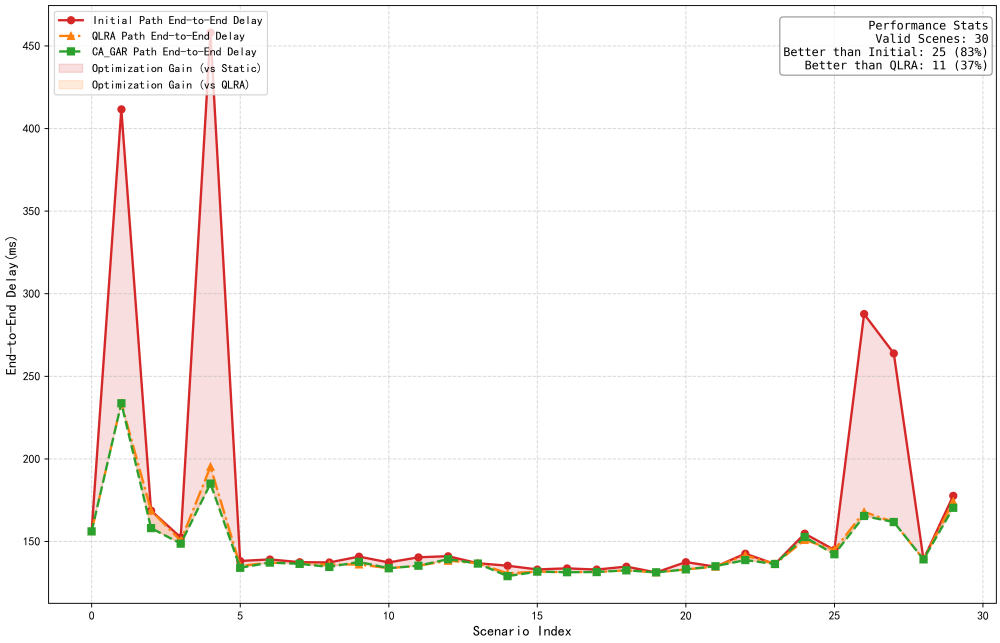


Figure 14. Performance Comparison of End-to-End Delay across 30 Scenarios.

The second category corresponds to light-load scenarios with a small amount of link overlap, such as Scenarios 6, 7, and 15 in Figure 14. In such scenarios, a small number of traffic flows intersect on certain links, but due to the relatively low overall load, the degree of congestion is limited. For example, flow (21,6) is transmitted along path 21-27-17-6, and flow (26,6) is transmitted along path 26-27-17-6, with both flows overlapping on link 27-17. Given the limited number of alternative paths, CA-GAR and QLRA exhibit comparable analytical capability: both algorithms can identify the queuing effect on the shared link and reroute flow (26,6) to path 26-27-12-6. Although this adjustment does not increase the hop count, it successfully achieves traffic offloading, effectively reducing the queuing delay on the shared link and lowering the overall delay compared to the initial static routing. In such scenarios, the performance of the two intelligent routing algorithms remains largely comparable.

The third category corresponds to medium-load or heavy-load scenarios with highly concentrated multi-flow traffic, such as Scenarios 1 and 27 in Figure 14. Consider the following traffic distribution: flows (21,6) and (26,6) converge toward node 6 via paths 21-27-17-6 and 26-27-17-6, respectively, while flows (15,17) and (14,17) converge toward node 17 via paths 15-27-17 and 14-27-17, respectively. All four flows traverse link 27-17, which becomes a north–south traffic bottleneck, with utilization approaching saturation and queuing delay rising sharply. Due to the lack of global awareness, the initial static routing algorithm independently selects the shortest path for each flow, causing a large volume of traffic to flood the same core link, resulting in a sharp surge in end-to-end delay with a peak exceeding 250 milliseconds. In such scenarios, both CA-GAR and QLRA can alleviate congestion through intelligent traffic offloading. Specifically, CA-GAR reroutes flow (26,6) to path 26-27-12-6, reroutes flow (15,17) to path 15-18-17, and reroutes flow (14,17) to path 14-19-17. This not only successfully avoids the congested link 27-17 but also bypasses the hotspot node 27, effectively distributing network traffic and suppressing end-to-end delay to a low level. QLRA also achieves a certain degree of traffic offloading, and the performance of the two algorithms remains comparable in such scenarios.

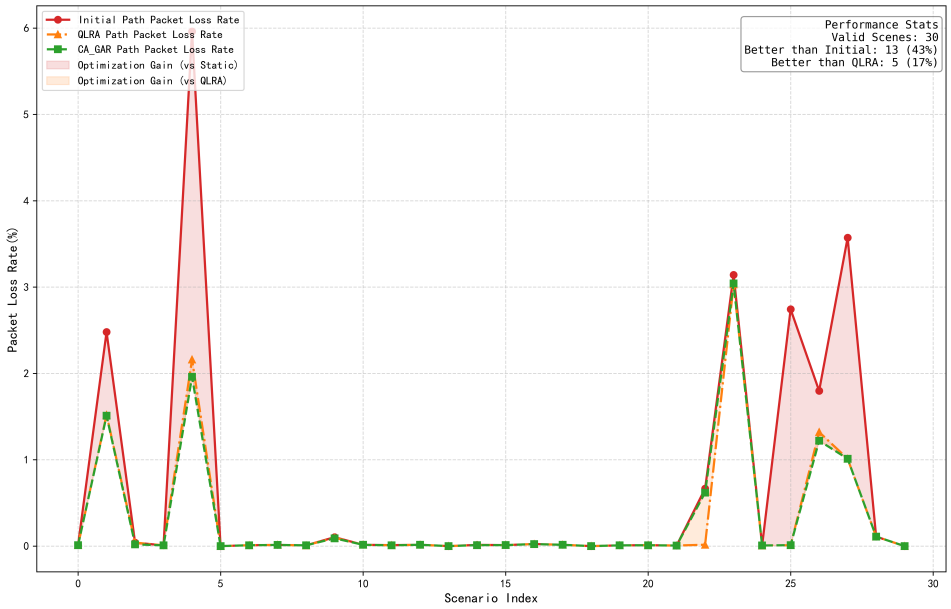
The fourth category also corresponds to medium-load or heavy-load scenarios with highly concentrated multi-flow traffic, such as Scenarios 4 and 26 in Figure 14. Similar to the third category, the initial static routing algorithm suffers from a sharp surge in delay due to

excessive traffic concentration at core nodes. However, in such scenarios, the delay optimization performance of CA-GAR and QLRA exhibits certain differences. CA-GAR, based on the SDN architecture, obtains a global view of network topology and load information, enabling it to coordinate multiple traffic flows from the perspective of global optimality. In contrast, QLRA adopts a distributed Q-learning architecture, where each node only maintains Q-value information of its local neighbors without requiring global information. Although this design reduces communication overhead and provides a certain capability for congestion avoidance, its local field of view limits the ability to cooperatively optimize global load balancing in complex scenarios with concurrent multi-flow traffic and tightly coupled congestion. Furthermore, CA-GAR's offline optimization and online mapping framework enables it to learn general congestion avoidance strategies from historical traffic patterns, allowing it to quickly generate high-quality link cost configurations even for unseen traffic distributions. In contrast, QLRA relies on online exploration to gradually adjust its strategy, requiring a certain convergence time to achieve stable performance under bursty traffic scenarios. These factors collectively contribute to CA-GAR's superior delay optimization performance over QLRA in high-load concentrated scenarios such as Scenarios 4 and 26. Taking Scenario 4 as an example, CA-GAR suppresses the end-to-end delay to approximately 50% of that of static routing, while the delay reduction achieved by QLRA is slightly lower than that of CA-GAR.

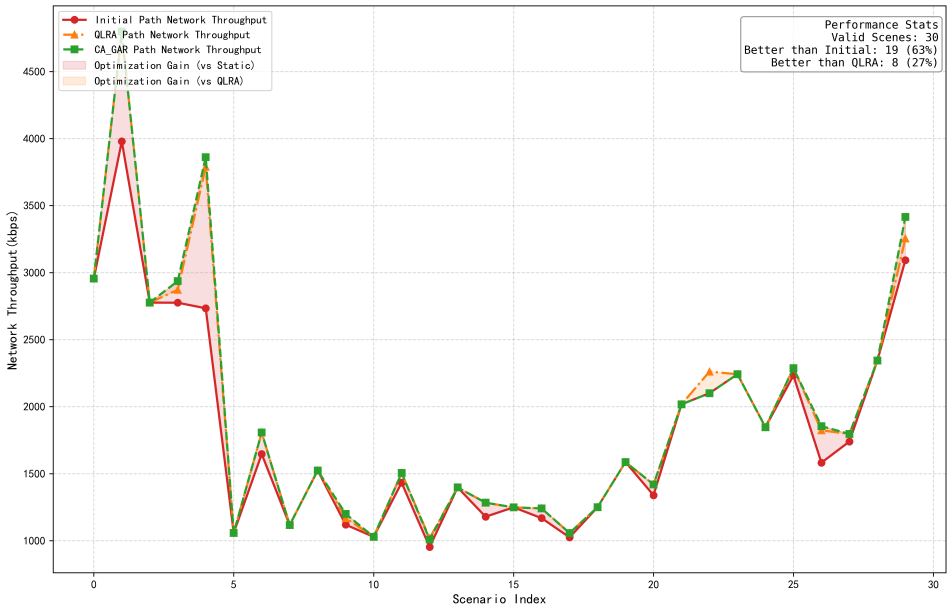
Based on the statistical results from all test data, CA-GAR achieves positive end-to-end delay optimization in 80.2% of the test scenarios compared to the initial static routing, demonstrating excellent congestion avoidance capability especially in high-load burst scenarios. Compared to QLRA, CA-GAR achieves positive delay optimization in 32.8% of the test scenarios, while the performance of the two algorithms remains comparable in the remaining scenarios. The above comparison indicates that CA-GAR possesses comprehensive competitiveness comparable to QLRA in multi-path offloading and congestion avoidance. Moreover, in complex scenarios with highly concentrated traffic flows and tightly coupled congestion, CA-GAR exhibits superior delay optimization capability by virtue of its globally centralized optimization architecture and offline learning mechanism.

To comprehensively evaluate the reliability and transmission efficiency of the network, Figure 15 presents a comparison of packet loss rate and network throughput. In Figure 15a, the packet loss rates of the three algorithms remain largely comparable across the majority of test scenarios, all maintained at relatively low levels. These occasional packet losses stem from the inherent characteristics of wireless channels and fall within a normal and controllable range. In scenarios with highly concentrated multi-flow traffic such as Scenarios 1 and 27, the initial static routing suffers from queue overflow due to excessive traffic concentration, causing the packet loss rate to rise to a relatively high level, while both CA-GAR and QLRA reduce it to approximately 1.2% through intelligent traffic offloading, with comparable performance. In extreme high-load scenarios such as Scenario 4, the difference in packet loss rate becomes more pronounced. Due to the lack of global awareness, the initial static routing directs a large volume of traffic into core links, causing buffers to overflow rapidly and the packet loss rate to exceed 6%. CA-GAR, leveraging its global view, coordinates multiple traffic flows from the perspective of global optimality, accurately identifies high packet loss risk links, and reroutes traffic accordingly, effectively suppressing the packet loss rate to around 2%. In contrast, although QLRA can perceive link states through local neighbor Q-value information, its distributed architecture lacks a grasp of the global congestion situation, making it difficult to achieve coordinated scheduling of multiple flows in complex scenarios with tightly coupled congestion, resulting in a smaller reduction in packet loss rate compared to CA-GAR. From a statistical perspective, CA-GAR

achieves positive optimization in packet loss rate in 39.2% of test scenarios compared to the initial static routing, and in 12.7% of scenarios compared to QLRA.



(**a**) Packet Loss Rate



(**b**) Network Throughput

Figure 15. Comprehensive Performance Comparison of Packet Loss Rate and Network Throughput.

In Figure 15b, the throughput of CA-GAR is higher than or comparable to the baseline algorithms in most scenarios. Notably, in Scenario 22, although CA-GAR outperforms QLRA in terms of end-to-end delay, its packet loss rate and throughput are inferior to those of QLRA. The underlying reason for this phenomenon lies in the difference in optimization objectives between CA-GAR and QLRA. CA-GAR's offline label generation is primarily oriented toward minimizing delay, with link cost adjustments prioritizing the reduction of end-to-end queuing and transmission delay, while the optimization of packet loss rate is an indirect consequence of congestion avoidance. In contrast, QLRA treats packet loss rate as a key metric in its multi-objective optimization, with its Q-value update mechanism directly incorporating a packet loss penalty term, causing the algorithm to actively avoid high packet loss risk links during path selection. Consequently, in scenarios where packet

loss rate is the primary concern and there exists a trade-off between delay and packet loss, QLRA can sacrifice some delay performance in exchange for lower packet loss rate and higher effective throughput, while CA-GAR tends to prioritize delay metrics, resulting in relatively inferior performance in terms of packet loss rate and throughput. Compared to the initial static routing, CA-GAR achieves positive optimization in throughput in 56.8% of test scenarios, with throughput improvement approaching 30% in high-load intervals such as Scenario 4. Compared to QLRA, CA-GAR achieves positive optimization in 30.4% of scenarios. These results demonstrate that the CA-GAR algorithm not only reduces delay and packet loss but also enhances the overall data carrying capacity of the network through more efficient link utilization.

To validate the adaptive capability of the CA-GAR algorithm under network failure scenarios, this section designs node failure experiments. By removing one or more nodes from the topology, we observe whether the affected service flows can be successfully rerouted and evaluate the quality of the rerouted paths. Figure 16 presents a comparison of the network-wide routing before and after the failure of Node 3, along with the path changes in the affected service flows. In the figure, red nodes represent failed nodes, blue nodes represent normal nodes, and dashed links represent links associated with the failed nodes.

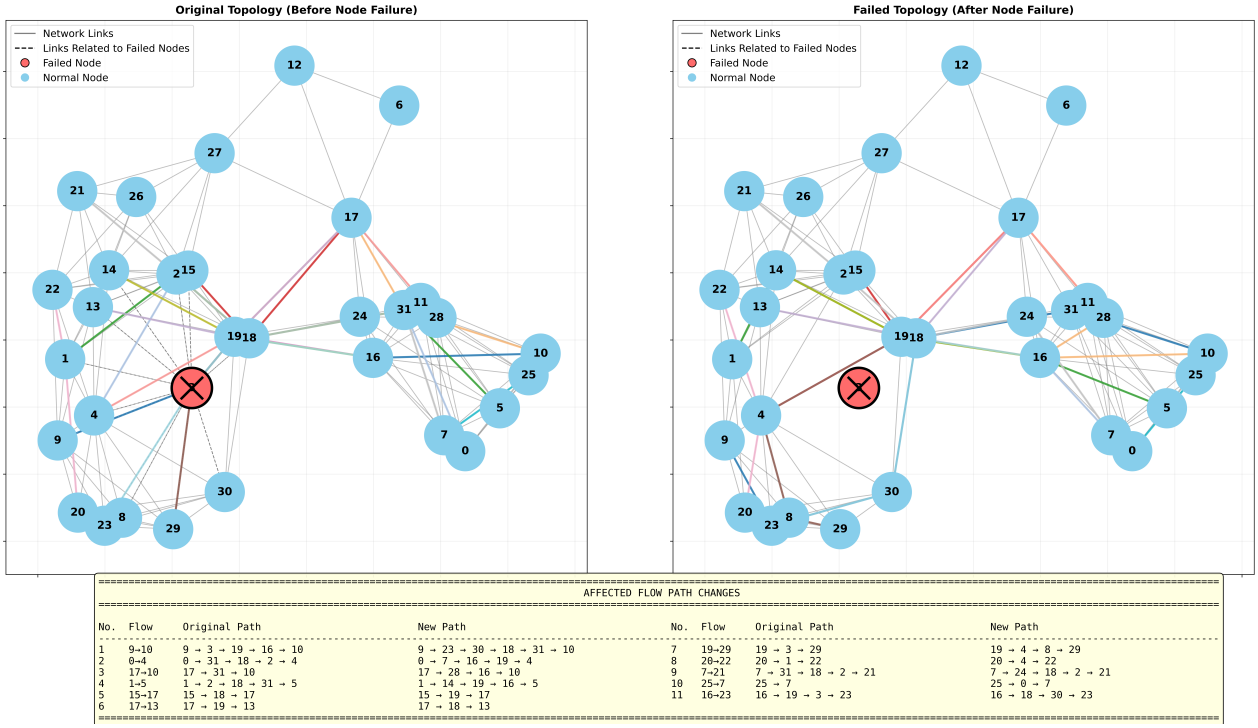


Figure 16. Routing comparison before and after single node failure (failed node: {3}).

As shown in Figure 16, when Node 3 fails, all links originally passing through this node are disrupted. The affected service flows can be divided into two categories: directly affected flows are those whose original paths pass through Node 3; indirectly affected flows are those that, although not passing through the failed node, experience changes in link congestion status due to network load redistribution, thereby triggering the CA-GAR algorithm to re-optimize the paths. Statistics show that a total of 11 service flows are affected by the failure of Node 3.

Figure 17 illustrates the path change in the directly affected flow (16,23). The original path of this flow was 16-19-3-23, which became unavailable after node 3 failed. The CA-GAR algorithm rerouted it to 16-18-30-23, a new path that completely bypasses the failed

node 3. Notably, the hop count of the rerouted path remains 3, identical to the original path, introducing no additional transmission hop overhead. This demonstrates that the CA-GAR algorithm can not only quickly find feasible alternative paths in response to node failures but also maintain transmission efficiency comparable to the original path when the topology permits.

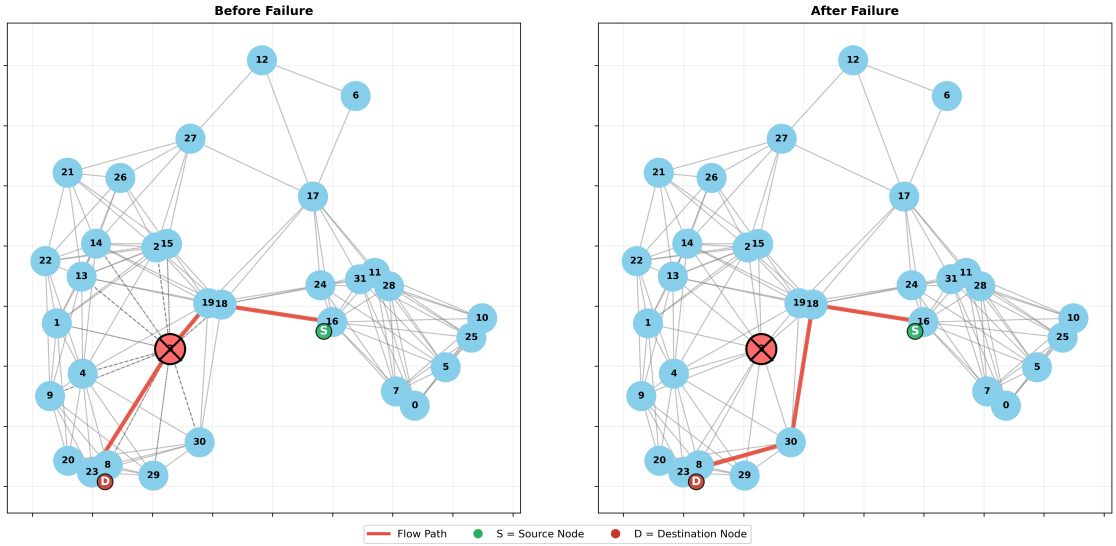


Figure 17. Rerouting of directly affected flow (16,23).

Figure 18 illustrates the path change in the indirectly affected flow (1,5). The original path of this flow was 1-2-18-31-5, which did not traverse the failed node 3. After node 3 failed, some traffic flows originally passing through node 3 were rerouted to other links, causing increased link utilization in regions such as nodes 18 and 30. The CA-GAR algorithm, sensing this load change, proactively optimized and adjusted the path of flow (1,5) to 1-14-19-16-5. Both paths have the same hop count of 5, but the new path effectively avoids potential new congestion hotspots that may arise from the load redistribution, thereby ensuring the end-to-end transmission performance of the flow. To further evaluate the robustness of the algorithm under more severe failures, Figure 19 presents the routing comparison after the simultaneous failure of nodes 3 and 31.

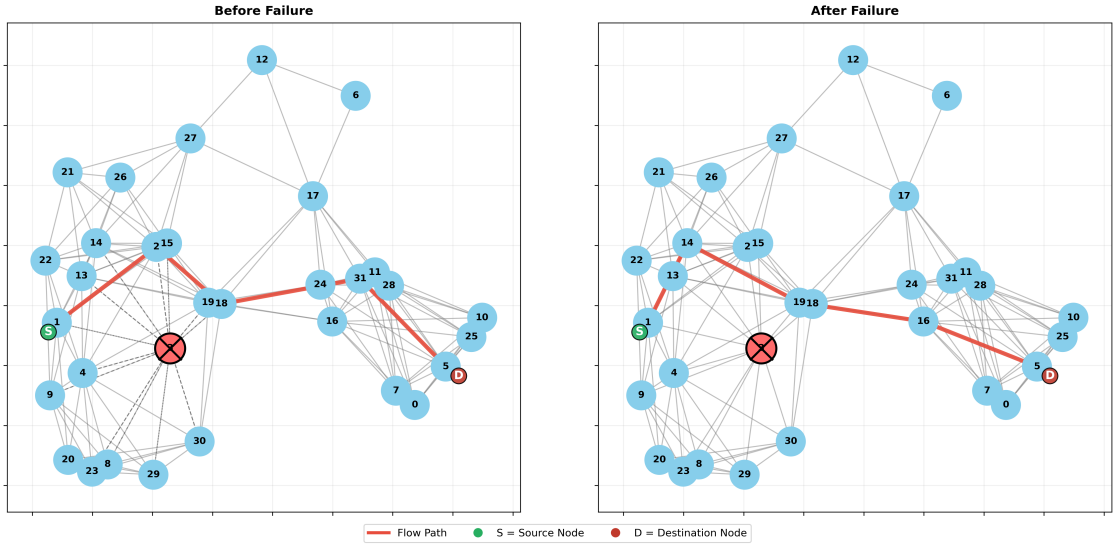


Figure 18. Rerouting of indirectly affected flow (1,5).

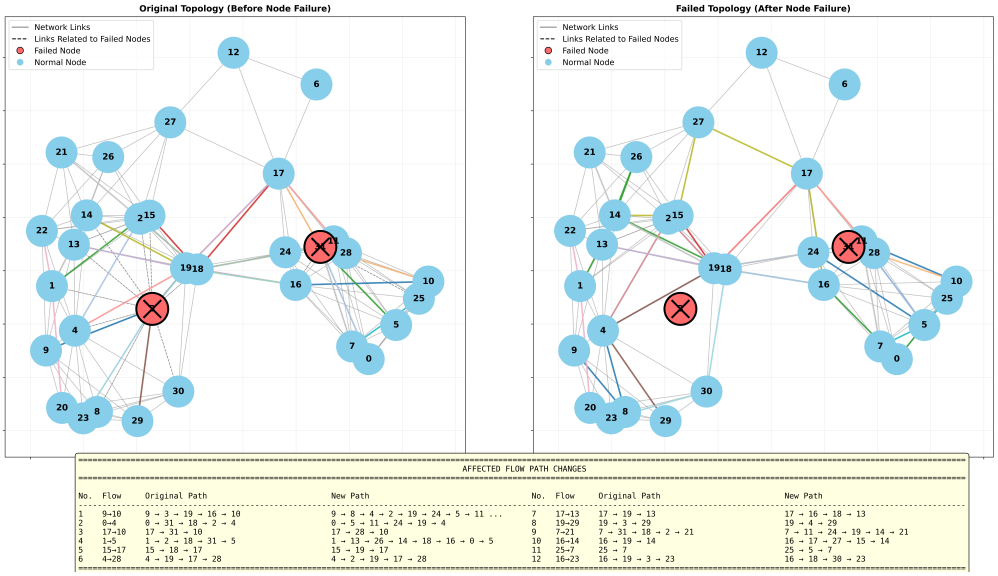


Figure 19. Routing comparison before and after double-node failure (failed nodes: {3, 31}).

As shown in Figure 19, when Node 3 and Node 31 fail simultaneously, the network topology undergoes significant changes, and multiple links are disrupted. Statistics show that a total of 12 service flows are affected by this double-node failure. In this scenario, the CA-GAR algorithm is still able to successfully plan feasible alternative paths for all affected service flows.

Based on the experimental results of both single-node and double-node failure scenarios, the CA-GAR algorithm demonstrates its ability to promptly perceive topology changes under network failure conditions and replan feasible paths for affected service flows. Although the rerouted paths exhibit a certain degree of increase in hop count and delay, these increases remain within an acceptable range, and no service interruption or severe performance degradation occurs. This indicates that the algorithm possesses a certain degree of fault adaptability and is capable of handling common node failure scenarios in networks.

6. Limitations and Future Work

Despite the relatively stable performance improvements demonstrated by the CA-GAR algorithm in simulation experiments, it is necessary to acknowledge several limitations in the current study, which also point to directions for future improvement. This research was conducted entirely on the NS-3 network simulation platform, with all experimental data derived from a discrete-event simulation environment. Although the simulation model strives to approximate real physical layer characteristics in parameter settings, inherent differences remain between the simulation environment and real-world wireless networks. Furthermore, there is room for improvement in the computational efficiency of the CA-GAR model. The Graph Attention Network employed in this paper captures high-order topological features through a multi-head attention mechanism, which meets the requirements for offline training and online inference in a simulated network with 32 nodes. However, as the network scale expands to hundreds of nodes, the parallel computational overhead of multi-head attention and the inference latency of the multi-layer stacked structure will increase significantly. Meanwhile, the model's training relies on supervisory labels generated by a heuristic algorithm; while this algorithm can effectively mitigate local optima, it theoretically cannot guarantee that the labels represent a global optimum, which to some extent limits the upper bound of the model's performance.

Future work will focus on enhancing the practicality and scalability of the system, centering on three key areas: (1) leveraging inductive graph learning mechanisms to enhance cross-scenario transferability, enabling the model to learn generic routing strategies from historical topologies and rapidly adapt to unseen network topologies without retraining; (2) constructing a multi-objective joint optimization framework to address the inadequacy of coupled modeling for multi-dimensional Quality of Service (QoS) metrics, thereby simultaneously satisfying the differentiated requirements of diverse traffic flows regarding latency, jitter, and packet loss rate [31]; and (3) exploring distributed deployment schemes for lightweight models within a Cloud–Edge–End collaborative architecture. This approach aims to reduce the computational burden on central nodes through hierarchical collaborative inference, thereby significantly improving the system's response speed and scalability in complex dynamic environments.

7. Conclusions

To address the problem of local network congestion induced by traffic burstiness in 5G/6G and industrial Internet scenarios, this paper proposes a Congestion-Aware Adaptive Routing mechanism (CA-GAR). The proposed mechanism addresses the limitations of traditional static routing by constructing a collaborative framework based on dynamic link cost optimization and graph attention networks. By offline generating engineering near-optimal cost labels that integrate a "penalty–reselection–reward" mechanism, and leveraging a multi-head attention mechanism to achieve a robust mapping from network states to link costs, CA-GAR effectively addresses the problems of congestion avoidance and load balancing under complex traffic patterns.

Experimental evaluations conducted on the NS-3 platform demonstrate the effectiveness of the proposed mechanism. In comprehensive tests covering light, medium, and heavy loads, CA-GAR exhibits strong generalization capability. Specifically, it achieves lower end-to-end delay than traditional static routing in 80.2% of the scenarios, and obtains positive gains in network throughput in 56.8% of the scenarios. Particularly under high-load burst scenarios, the model exhibits robust congestion avoidance performance: end-to-end delay is reduced by approximately 50% compared with static routing, network throughput is improved by nearly 30%, and the packet loss rate is effectively reduced from over 6% (at the congestion peak) to a low level of 2%. Furthermore, in comparison with the intelligent routing algorithm QLRA, CA-GAR shows competitive performance in multipath distribution and congestion avoidance, validating its effectiveness in coupled multi-dimensional QoS metrics modeling.

This study confirms that CA-GAR, through its closed-loop architecture of "offline optimization–online mapping", facilitates a shift from static shortest-path routing to dynamic congestion-aware routing, thereby providing a feasible technical pathway for adaptive scheduling in next-generation networks.

Author Contributions: Conceptualization, J.L. and L.Z.; Methodology, J.L. and L.Z.; Software, X.L.; Validation, X.L.; Formal analysis, J.L.; Investigation, L.Z.; Resources, L.Z.; Data curation, X.L.; Writing—original draft, J.L. and X.L.; Writing—review & editing, J.L. and X.L.; Visualization, X.L.; Supervision, J.L.; Project administration, J.L. and L.Z.; Funding acquisition, J.L. and L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Laboratory of Wireless Communications, grant number 2024-KGR-JJ-05, and the National Natural Science Foundation of China, grant numbers 61271168 and 62501650. The APC was funded by the National Key Laboratory of Wireless Communications.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Yang, H.; Pu, C.; Wu, J.; Wu, Y.; Xia, Y. Enhancing OLSR Protocol in VANETs with Multi-Objective Particle Swarm Optimization. *Phys. A Stat. Mech. Its Appl.* **2023**, *614*, 128570. [CrossRef]
2. Robinson, Y.H.; Julie, E.G.; Saravanan, K.; Hoang Son, L.; Kumar, R.; Abdel-Basset, M.; Thong, P.H. Link-Disjoint Multipath Routing for Network Traffic Overload Handling in Mobile Ad-hoc Networks. *IEEE Access* **2019**, *7*, 143312–143323. [CrossRef]
3. Draz, U.; Yasin, S.; Ali, A.; Khan, M.A.; Nawaz, A. Traffic Agents-Based Analysis of Hotspot Effect in IoT-Enabled Wireless Sensor Network. In *Proceedings of the 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*; IEEE: New York, NY, USA, 2021; pp. 1029–1034. [CrossRef]
4. Huang, F.; Zhang, J.; Xu, J.; Shao, Y.; Pu, L. An SDN-Based QoS Guaranteed Mechanism for Geospatial Flows. In *Proceedings of the 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*; IEEE: New York, NY, USA, 2019; pp. 1394–1401. [CrossRef]
5. Ma, Y.; Zhao, R.; Yin, N. Application of an Improved Link Prediction Algorithm Based on Complex Network in Industrial Structure Adjustment. *Processes* **2023**, *11*, 1689. [CrossRef]
6. 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]
7. Rakshit, S.K.; Sundararajan, M. Digital Twin Based Supply Chain Routing. U.S. Patent US20220083976 A1, 17 March 2022. Available online: https://patents.google.com/patent/US20220083976A1 (accessed on 20 April 2026).
8. Liao, H.; Kara, L.B. Reinforcement Learning for Routing. In *Machine Learning Applications in Electronic Design Automation*; Ren, H., Hu, J., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 277–306. [CrossRef]
9. Kiruthiga, R.; Nithya, B. Machine Learning-Driven Packet Loss Classification via TCP Jersey and Multi-Layer Perceptron. In *Proceedings of the 2023 International Conference on Computational Intelligence, Networks and Security (ICCINS)*; IEEE: New York, NY, USA, 2023; pp. 1–6. [CrossRef]
10. Wang, Y.; Wang, F.; Tian, Q.; Sun, D.; Hu, P.; Wang, R.; Tang, X. Dynamic Bandwidth Allocation Algorithm Based on Traffic Classification with the Aid of LSTM and GRU for Industrial Passive Optical Networks. In *Proceedings of the 2023 21st International Conference on Optical Communications and Networks (ICOCN)*; IEEE: New York, NY, USA, 2023; pp. 1–3. [CrossRef]
11. Zhang, S.; Liu, X.; Qi, Z.; Yan, X.; Yang, W. GI-Graph: A Generative Invariant Graph Learning Scheme Towards Out-of-Distribution Generalization. *IEEE Trans. Knowl. Data Eng.* **2025**, *37*, 5934–5947. [CrossRef]
12. Yu, Z.; Guo, Y.; Chen, Y. Learning Trajectory Routing with Graph Neural Networks. In *Proceedings of the 5th International Conference on Big Data and Computing*; ICBDC '20; ACM: New York, NY, USA, 2020; pp. 121–126. [CrossRef]
13. Liu, H.; Zhu, C.; Zhang, D. Global-Aware Enhanced Spatial-Temporal Graph Recurrent Networks: A New Framework For Traffic Flow Prediction. *arXiv* **2024**, arXiv:2401.04135. [CrossRef]
14. Huang, R.; Wu, X. Research on Graph Reasoning Mechanism Based on Heterogeneous Perception and Contrastive Learning. *J. East China Univ. Sci. Technol.* **2026**, *52*, 129–141. [CrossRef]
15. 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]
16. Gómez-delaHiz, J.; Galán-Jiménez, J. Improving the Traffic Engineering of SDN Networks by Using Local Multi-Agent Deep Reinforcement Learning. In *Proceedings of the NOMS 2024-2024 IEEE Network Operations and Management Symposium*; IEEE: New York, NY, USA, 2024; pp. 1–5. [CrossRef]
17. Xu, L.; Han, S.; Fu, W.; Zhu, Z.; Wu, J.; Zhu, X. GNN-DRL-Based Intelligent Routing and Resource Allocation Algorithms for Multi-Layer Wireless Mesh Network. *Sensors* **2026**, *26*, 1170. [CrossRef] [PubMed]
18. Simon, J.; Kapileswar, N. Deep Graph Neural Networks for Intelligent Traffic Routing in Software-Defined Networks. In *Proceedings of the 2025 6th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*; IEEE: New York, NY, USA, 2025; pp. 1843–1848. [CrossRef]
19. Dhamala, B.K.; Dawadi, B.R.; Manzoni, P.; Acharya, B.K. Performance Evaluation of Graph Neural Network-Based RouteNet Model with Attention Mechanism. *Future Internet* **2024**, *16*, 116. [CrossRef]
20. Hakim, G.; Braun, R.; Lipman, J. Adapted Diffusion for Energy-Efficient Routing in Wireless Sensor Networks. *Electronics* **2024**, *13*, 2072. [CrossRef]

21. Chen, B. Research on Intelligent Routing and Embedding of Sliced Ad-Hoc Network. Master's Thesis, University of Electronic Science and Technology of China, Chengdu, China, 2021. [CrossRef]
22. Wang, Y. TDMA Waveform Design and Intelligent Routing Decision for Ad-Hoc Networks. Master's Thesis, University of Electronic Science and Technology of China, Chengdu, China, 2025. [CrossRef]
23. Liu, X.; Zhang, Z.; Meng, F.; Zhang, Y. Fault Diagnosis of Wind Turbine Bearings Based on CNN and SSA–ELM. *J. Vib. Eng. Technol.* **2023**, *11*, 3929–3945. [CrossRef]
24. Han, L.; Wang, Z.; Guo, J.; Li, Y.; Luo, H.; Liu, L. LSTM-GAT Networks Based on ResNet Structure for Prediction of Complex Multivariable Systems. In *Proceedings of the 2024 36th Chinese Control and Decision Conference (CCDC)*; IEEE: New York, NY, USA, 2024; pp. 3104–3109. [CrossRef]
25. Wang, H.; Tu, M. Enhancing Attention Models via Multi-head Collaboration. In *Proceedings of the 2020 International Conference on Asian Language Processing (IALP)*; IEEE: New York, NY, USA, 2020; pp. 19–23. [CrossRef]
26. Dong, X.; Dong, L.; Luo, Z.; Liu, S.; Liao, Z. Study on Graph Structure Construction Method Based on Transformer Multi-Head Attention Mechanism. In *Proceedings of the 2025 5th International Conference on Mechanical, Electronics and Electrical and Automation Control (METMS)*; IEEE: New York, NY, USA, 2025; pp. 720–723. Available online: https://ieeexplore.ieee.org/abstract/document/11047618 (accessed on 20 April 2026).
27. Komech, S.; Kupavskii, A.; Vezolainen, A. Choosing optimal parameters for a distributed multi-constrained QoS routing. *arXiv* **2023**, arXiv:2310.07350. [CrossRef]
28. Kim, S.Y.; Gong, S.L.; Lee, J.W. Joint Optimization of Link Weight and Link Capacity Expansion in Communication Networks. In *Proceedings of the 2010 International Conference on Information and Communication Technology Convergence (ICTC)*; IEEE: New York, NY, USA, 2010; pp. 125–126. [CrossRef]
29. Lai, L.H.; Lin, Y.L.; Liu, Y.H.; Lai, J.P.; Yang, W.C.; Hou, H.P.; Pai, P.F. The Use of Machine Learning Models with Optuna in Disease Prediction. *Electronics* **2024**, *13*, 4775. [CrossRef]
30. Shen, K.; Qin, H.; Zhou, J.; Liu, G. Runoff Probability Prediction Model Based on Natural Gradient Boosting with Tree-Structured Parzen Estimator Optimization. *Water* **2022**, *14*, 545. [CrossRef]
31. Fu, Y.; Jia, Y.; Huang, B.; Zhou, X.; Qin, X. The RapidIO Routing Strategy Based on the Double-Antibody Group Multi-Objective Artificial Immunity Algorithm. *Sensors* **2022**, *22*, 914. [CrossRef] [PubMed]

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.