Interactive Path Generation and Selection Policy for Online Network Route Planning

Zhongxing Ai[†‡], Peng Wang[†‡*], Jing Sun[‡], Bo Wang[§]
[†]University of Chinese Academy of Sciences, Beijing, China
[‡]Institute of Software, Chinese Academy of Sciences, Beijing, China
[§]National Key Laboratory of Space Integrated Information System, China
aizhongxing2025@iscas.ac.cn, wangpeng@iscas.ac.cn, sunjing24@iscas.ac.cn, wangbo_nklsiis@163.com

***Abstract*—Dynamic multi-hop networks face fast link-level dynamics including topological changes, stochastic failures, time-varying residual capacities, etc., which can cause transmission timeout, persistent congestion and frequent route flapping, undermining network stability. Aiming at jointly optimizing route qualities including success rate, path length, congestion, and path-rerouting stability, etc., sophisticated planning algorithm is needed to flexibly adapt to the time-varying patterns of the whole network. In this paper, a novel interactive two-level reinforcement learning (RL) policy is proposed and decouples candidate path generation from online selection. The upper-level learns to propose a set of stable candidate paths for arbitrary source–destination data routing demands, guided by long-term per-path instability estimations aggregated from past collections. The lower-level performs online selection from the candidates, by employing a temporal graph attention encoder to capture state dynamics, and applying selection policy constrained by success rate, stability, low-congestion, tail-switching avoidance, etc. The two-level policy is interactively learned by bridging them with a flow of stability features from the lower- to upper-level. Experiments on public network datasets demonstrate that the proposed method consistently achieves high-quality routing, with the reduced congestion and path switching while maintaining high success rates and near-shortest paths.**

***Index Terms*—dynamic routing, traffic planning, reinforcement learning, temporal graph neural networks**

I. Introduction

Multi-hop network routing provides end-to-end connectivity via intermediate relays and supports distributed wireless systems such as Mobile Ad Hoc Networks (MANETs), sensor networks, Unmanned Aerial Vehicle (UAV) networks, and satellite swarms [1]–[3]. Millisecond-scale traffic bursts often cause tail latency and packet loss [4], motivating traffic engineering (TE) for better route planning and resource utilization [5]. Traditional TE, such as Open Shortest Path First (OSPF) weight tuning with Equal-Cost Multi-Path Routing (ECMP), works well under gradual topology and traffic changes [6], but becomes less effective in highly dynamic settings such as high-speed in-orbit satellite constellations. In practice, rapid topology changes, intermittent link failures, and time-varying residual capacities can cause route flapping, congestion, delay, and even transmission failures. These dynamics may also induce pronounced tail effects, i.e., rare but severe spikes [4] [7], which can trigger oscillatory rerouting and unstable data transmission.

While widely recognized as a major source of variability in highly dynamic networks [8] [9], the path-level temporal stability such as switching frequency and burstiness, is rarely treated as an explicit optimization objective. Classical methods including shortest paths [10], k-shortest paths (KSP) [11] [12], bio-inspired exploration [13] [14], etc., perform well under mild dynamics but become brittle under severe failures and capacity fluctuations. Recently, learning-based methods have been explored to improve scalability and generalization in route planning. In Software Defined Network (SDN) settings, RL-based policy learning under changing traffic [15] and Proximal Policy Optimization (PPO) based quality-oriented routing optimization [16] have both been studied. More generally, Deep Reinforcement Learning (DRL) has also been applied to higher-level network planning [17] and learned navigation [18] in communication networks. At the system level, TEAL combines learning with optimization for faster Wide Area Network (WAN) decision making [19], while DOTE revisits predictive TE under realistic uncertainty and dynamics [20]. However, these studies either address route stability only implicitly or ignore path-level switching as an important objective in route planning.

Graph Neural Networks (GNNs) further strengthen learning-based routing by encoding topologies and traffic states. For example, DRL can be integrated with GNN encoders for routing optimization [21], and graph convolutional networks (GCNs) can also be used in route exploration with DRL-based methods [22]. In SDN settings, PPO+GNN has also been applied to intelligent routing optimization [23]. Using GNNs as complementary tools, RouteNet [24] enables network performance modeling, optimization, and what-if reasoning in SDN, while RouteNet-Erlang [25] extends this idea to performance evaluation under queueing-related assumptions. Recent studies also improve the decentralization and generalization of routing methods, such as the GNN-based multi-agent TE system in [26] and the generalized routing optimization method combining GNN and DRL in [27]. Despite this progress, these policies still heavily depend on heuristic external candidate generation, failing to provide end-to-end planning solutions, while the suppression of path switching, especially tail bursts, remains under-addressed.

*Corresponding author: Peng Wang, Email: wangpeng@iscas.ac.cn.

ISBN 978-3-903176-82-9

Emerging dynamic multi-hop scenarios, such as UAV and satellite networks, further emphasize time-varying feasible connections and the need to react quickly to link disruptions. RL-based routing has been explored in UAV-aided networks [28] and air-to-air Ad-Hoc networks [29]. In dynamic space networks, SaTE [30] studies TE for satellite systems, while stability-oriented hierarchical routing [31] and resilient routing methods [32] have also been proposed for LEO networks and mega-constellations. For short-term dynamics, temporal graph learning offers natural representations through TGAT [33] and TGN [34]. However, tail-switching bursts in highly dynamic networks are still difficult to suppress effectively, even with CVaR- and quantile-based risk-sensitive RL methods [35]–[38].

To address these challenges, we propose an interactive two-level RL policy that decouples candidate path generation from online selection. The upper level learns a stability-aware proposer guided by long-term link-instability scores, while the lower level employs a TGAT-based selector to choose among the candidates using a policy learned from historical graph patterns. To the best of our knowledge, this is the first interactive policy-learning framework for high-stability routing in dynamic networks.

Our main contributions can be summarized as follows:

- An interactive two-level RL policy is proposed for high-quality route planning in dynamic networks.
- The upper-level candidate generation and lower-level online selection are bridged through alternating training with stability-aware feedback.
- A risk-sensitive lower-level reward is designed, combining Lagrangian-based success control with a CVaR-inspired tail penalty for congestion and switching suppression.
- Extensive experiments demonstrate the superior routing quality and stability trade-offs over the baselines.

II. Problem Formulation

A. Dynamic Network Model

As widely adopted for dynamic network problem solving, the connectivity graph $G = (V, E, C)$ is considered, where V is a fixed set of nodes in the network, E collects all links of nodes that may change over time, and C is the edge-related capacity, i.e., $c_e > 0$ for any $e \in E$. At each time step t, the network state can be represented as $S_t = \big(X_t, U_t\big)$, where X_t and U_t are current loads and connection states of E, each edge load $x_e(t) \geq 0$, and $u_e(t) \in \{0, 1\}$ indicates whether link e is up or not. Based on the above notations, the available edge set can be represented as $E_t = \{\forall e \in E, u_e(t) = 1\}$, and the residual capacity is $r_e(t) = c_e - x_e(t)$ for e at t.

Besides the continuous change of requested resources, the network dynamics is also characterized by the frequent capacity release of the occupied edges. When a request with stochastic demand λ ($\lambda > 0$) is accepted on a path p at time t, $x_e(t)$ needs to be increased by λ for all $e \in p$ and the release event needs to be monitored to schedule the capacity release on the corresponding links. Therefore, at each time step t, the environment applies both capacity demand and release events to update $x_e(t)$ and $u_e(t)$ to synchronize the exogenous link failures or recoveries.

B. Traffic and Routing Decisions

Aiming at providing real-time routing services, each routing decision needs to be made and aligned with the above dynamic network updating step. Traffic is a sequence of requests (s, d, λ) with s and d denote the route source and destination respectively and $s \neq d$. To provide high-quality traffic satisfactions, the two-stage planning paradigm is adopted with candidate generation followed by online selection, to improve scalability and generalization across topologies [21]. First, the potential route candidates are generated at decision step t, according to the current network state S_t and the required traffic (s_t, d_t, λ_t):

$$\mathcal{P}_{s_t,d_t} = \{p^1_{s_t,d_t}, p^2_{s_t,d_t}, \ldots, p^K_{s_t,d_t}\}, \tag{1}$$

which forms a discrete action space for choosing from these K candidates as the final route. Second, a decision policy is needed to select a final path $p_t = p^{a_t}_{s_t,d_t} \in \mathcal{P}_{s_t,d_t}$, $a_t \in \{1, \ldots, K\}$. Once p_t is chosen, its feasibility can be further verified by the indicator:

$$I_{p_t} = \mathbb{I}(u_e(t) = 1 \ \wedge \ r_e(t) \geq \lambda_t, \ \forall e \in p_t). \tag{2}$$

If $I_{p_t} = 1$, the traffic request is accepted and its capacity λ_t needs to be reserved; Otherwise, it is blocked as an infeasible route.

C. Multi-Objective Routing Requirements

Multiple metrics for data routing qualities are considered in synergy to avoid any quality indicator being violated, including congestion, success rate, path switching rate, tail-avoiding switching, long-term stability, etc.

Congestion. By defining the residual-capacity ratio, congestion can be reflected on a specific edge e as

$$\mathrm{res}_e(t) = \frac{c_e - x_e(t)}{c_e}. \tag{3}$$

For an attempted path p_t, let the minimum residual-capacity ratio through the whole path as $\mathrm{res}_{p_t}(t) = \min_{e \in p_t} \mathrm{res}_e(t)$. The congestion level can then be defined as the data routing bottleneck on p_t:

$$\mathrm{Cong}_{p_t} = 1 - \mathrm{res}_{p_t}(t). \tag{4}$$

Success Rate. To quantify the probability of paths being feasible, success rate is a statistic related to M successful ones and N total decisions, as $\mathrm{Succ} = M/N$.

Path Switching Rate. Path switching needs to be less frequent during data transmission, to keep the network paths stable and avoid unnecessary re-routing. For a traffic with pair (s, d), let $\tilde{p}_{s,d}(t)$ and $\tilde{p}_{s,d}(t+1)$ denote two feasible paths at consecutive time steps, the per-step switch indicator is

$$\mathrm{Sw}_{s,d}(t) = \mathbb{I}(I_{\tilde{p}_{s,d}(t)} \ \wedge \ I_{\tilde{p}_{s,d}(t+1)} \ \wedge \ \tilde{p}_{s,d}(t) \neq \tilde{p}_{s,d}(t+1)). \tag{5}$$

The path switching rate is computed as the path jitter rate over all feasible paths during a time period:

$$\mathrm{Jit} = \frac{\sum_{s,d} \sum_t \mathrm{Sw}_{s,d}(t)}{\sum_{s,d} \sum_t I_{\tilde{p}_{s,d}(t)}}. \tag{6}$$

Tail-Avoiding Switching Quantile. Even the average Jit value in Eq. (6) may remain low, fast link/capacity dynamics can induce pronounced *tail switching bursts*—rare but severe time periods when a flow repeatedly alternates among certain paths. Such tail events amplify the transient instability, and degrade performances by triggering oscillatory re-routing. We explicitly suppress these tail bursts using a quantile-based exceedance penalty [36]–[39]. By maintaining an FIFO buffer for a sliding window with length w, the path switching rate from time step $t-w+1$ to t for traffic pair (s_t, d_t) at t is

$$J_t = \frac{\sum_{t-w+1}^{t} \mathrm{Sw}_i}{\sum_{t-w+1}^{t} I_{\tilde{p}_{s_t,d_t}(i)}}, \tag{7}$$

where $I_{\tilde{p}_{s_t,d_t}(i)}$ indicates the feasibility of the chosen path for (s_t, d_t) at time step i in $[t-w+1, t]$. Then, for a longer time period w_q, the τ-quantile threshold [37], [38] can be computed by sorting all the J_t from $t-w_q+1$ to t to filter the serious switching as:

$$q_\tau(t) = \mathrm{Quantile}_\tau\big(\{J_i\}_{i=t-w_q+1}^{t}\big), \tag{8}$$

where $\mathrm{Quantile}_\tau(\cdot)$ denotes the empirical τ-quantile.

Long-Term Link Stability. For a link e, by aggregating the edge load and its link state, the link deficiency is defined as

$$z_e(t) = \lambda_1(1-u_e(t)) + \lambda_2 \frac{x_e(t)}{c_e}, \tag{9}$$

where λ_1 and λ_2 are weights to balance the two sources. We maintain an exponential moving average [40]–[42] as the long-term instability score by updating

$$\mathrm{fail\_ema}_e^{(t)} = (1-\beta)\,\mathrm{fail\_ema}_e^{(t-1)} + \beta\, z_e(t), \tag{10}$$

where $\beta \in (0,1)$ and $\mathrm{fail\_ema}_e^{(0)}$ is the initial instability value. For a path p, the stability is then defined over all its edges as

$$\mathrm{Stab}(p) = 1 - \frac{1}{|p|} \sum_{e \in p} \mathrm{fail\_ema}_e. \tag{11}$$

III. Interactive Two-Level Route Planning

The dynamic routing problem in Section II can be treated as a constrained Markov Decision Process (MDP), for which RL-based method is a natural fit. Motivated by the two-stage planning paradigm (candidate generation and selection), an interactive two-level RL framework is designed, to avoid intractable searching over the combinatorial space for all possible paths.

A. Method Overview

Fig. 1 illustrates the overview of the proposed interactive two-level method, composed of candidate proposer (upper-level, UL) and path selector (lower-level, LL). Though the Actor–Critic network in PPO is used for policy learning in both levels, the mechanisms in UL and LL are totally different including states, actions, rewards, etc. Because the role of the proposer is to generate a set of paths which are worth considering, the UL decision is made by a stability-aware policy based on the current network topology.

As shown in Fig. 1, historical state dynamics are learned in LL with TGAT for temporal abstraction, in order to decide how to correctly switch routes dynamically. Different with UL, PPO objective in LL explicitly penalizes route switching including tail bursts, encouraging the route to switch only when doing so improves feasibility or alleviates congestion. By aggregating episodic evidence of link deficiencies in LL, the long-term edge instability signal $\{\mathrm{fail\_ema}_e\}$ is fed into the states of UL and incorporated in UL's reward to penalize unstable path generation. By training UL and LL interactively, the UL preferentially avoids chronically unreliable or over-used links and outputs a set of stable candidates for LL's online routing.

B. Upper Level: Stability-Aware Candidate Path Proposer

To provide feasible candidates for LL's selection, UL takes the current snapshot of network at time step t and makes decisions on which edges to travel from s in order to reach d. This aims at generating a set of candidate paths $\mathcal{P}_{s,d}$ that avoid unreliable links while keeping short path lengths. PPO-based policy [43] is trained in this level due to its robust performances in stabilizing policy-gradient updates by optimizing a clipped surrogate objective.

State and Action. The path generation is a typical sequential decision problem, and for each pair (s, d), an episodic MDP is applied to builds a path from s to d. To avoid failures in LL's path selection caused by infeasible candidates, the current state is augmented with long-term instability scores $\{\mathrm{fail\_ema}_e\}$ derived from LL trajectories. At decision step k in generating the path, the states include (i) the current node v_k and the node-level features, e.g., the shortest distance to d; (ii) for each neighbor u of v_k, edge (v_k,u)'s features summarizing capacity, edge betweenness centrality [44], [45], and the current instability score $\mathrm{fail\_ema}_{(v_k,u)}$; (iii) a visited-node mask that prevents cycles. The action is to choose the next-hop node v_{k+1} from the unvisited neighbors of v_k.

Transition Dynamics. State transition in UL is deterministic given an action that selects a neighbor u of v_k, i.e., $v_{k+1} \leftarrow u$. Then append u to the current partial path, and update the visited-node mask accordingly. The next state is then rebuilt at v_{k+1} by recomputing node features and gathering the corresponding outgoing-edge features for its unvisited neighbors. The episode terminates when d or a maximum hop limit is reached, and the visited nodes form a candidate path p.

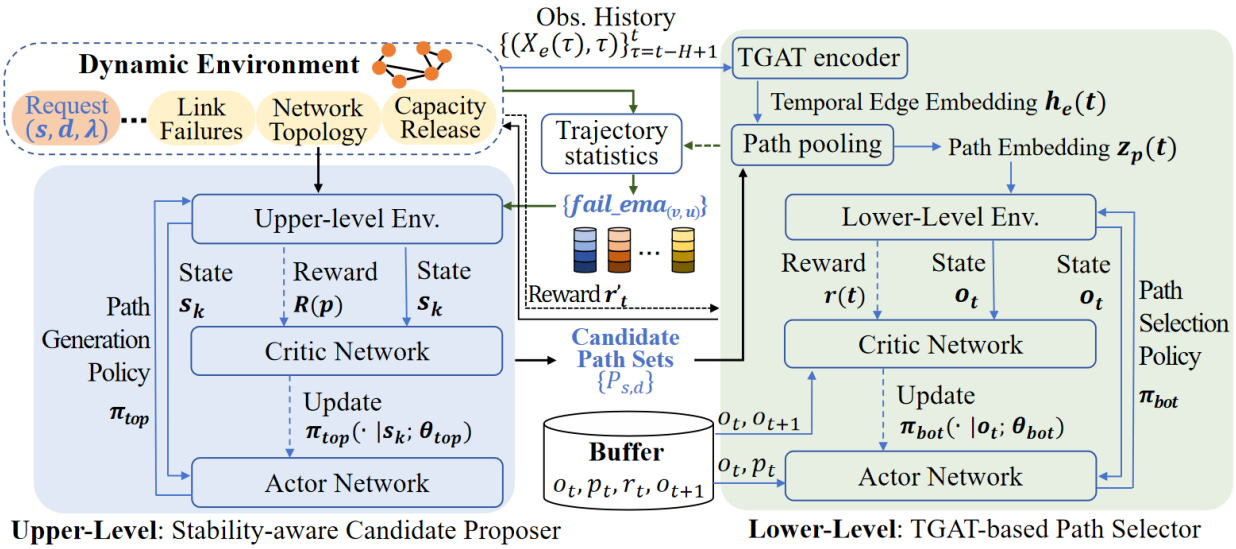

Fig. 1. Flowchart of the proposed interactive two-level policy for route planning.

Reward. For each completed path p, the path-level statistics can be calculated to form a weighted reward combination:

$$R(p) = w_{\text{stab}}\,\text{Stab}(p) - w_{\text{len}}|p| - w_{\text{bet}}\,\text{Bet}(p), \qquad (12)$$

where $\text{Stab}(p)$ is the stability in Eq. (11), $|p|$ is the hop count, and $\text{Bet}(p)$ is the average edge betweenness along p. With this reward, the stable, short paths avoiding structurally central "hot" edges can be favored. During training, we gradually increase w_{stab} relative to w_{len}, allowing the proposer to transition from a shortest-path–like heuristic to a stability-oriented policy as $\{\text{fail\_ema}_e\}$ becomes more accurate after some iterations in LL.

Training. The PPO proposer $\pi_{\text{top}}(\cdot;\theta_{\text{top}})$ in UL is trained on network topology at t for all ordered pairs $(s,d) \in \mathcal{S}$, as shown in Algorithm 1. For each episode of a given pair, the route starts from the source s and the proposer iteratively samples the next-hop node based on each step's state representation until termination. The path-level return can be computed according to Eq. (12) and the parameters are updated to minimize the Actor and Critic loss using Adam optimizer [41]. The trained proposer is used to generate a pool of paths for each (s,d) and retains at most K diverse candidates by filtering out invalid ones.

Algorithm 1 Upper-level candidate path proposer training

Input: Topology G; (s,d) set $\mathcal{S}$; stability feature $\{\text{fail\_ema}_e\}$; hop limit $H_{\max}$; PPO hyperparameters
Output: Trained path generation policy π_{top}

```
Initialize parameters θ_top for π_top
for all (s, d) ∈ S do
    for episode = 1 to N_ep do
        v ← s, p ← s, initialize visited mask m
        while v ≠ d and |p| < H_max do
            s_k ← BuildState(G, v, d, fail_ema, m)
            u ~ π_top(· | s_k; θ_top)
            Append u to p; mark u visited; v ← u
        end while
        Compute R(p) from stability, hop count, and betweenness
        Update θ_top with PPO via backpropagation (Adam)
    end for
end for
```

C. Lower Level: TGAT-Based Path Selection

To satisfy the multi-objective needs as described in Section II-C, the LL is responsible for fine-grained path selection according to the network dynamics which is modelled by TGAT. For this purpose, the LL policy π_{bot} selects one path from the candidates $\mathcal{P}_{s,d}$ to satisfy the success constraint while reducing congestion and switching rate.

Temporal Graph Embedding with TGAT. For each edge e at time t, a feature vector $\mathbf{x}_e(t)$ is constructed, including the residual-capacity ratio (congestion) $\text{res}_e(t)$, up/down indicator $u_e(t)$, and their historical values. Within a sliding window of length H, the historical records $\{(\mathbf{x}_e(t'),t')\}_{t'=t-H+1}^{t}$ is fed into a standard TGAT encoder [33] which outputs an edge embedding $\mathbf{h}_e(t)$ that summarizes the recent temporal evolution of e and its neighbors. For an arbitrary candidate path $p \in \mathcal{P}_{s,d}$, the embeddings of the constituent edges can be aggregated into a path-level representation $\mathbf{z}_p(t)$, e.g., by averaging the edge embeddings within the path [46], [47].

State, Action, and Reward. In the LL path selection, the PPO model consumes the candidate-wise representations set $\{\mathbf{z}_p(t)\}$ and the request (s_t,d_t,λ_t) as states, and selects one path $p \in \mathcal{P}_{s_t,d_t}$ as the action to be determined. After a path p_t is selected, the environment attempts to route the request on p_t, according to which the network state is updated and a new candidate set's embeddings $\{\mathbf{z}_p(t+1)\}$ are yielded for

the next decision. The per-step reward is calculated as

$$r_t = r_t^{\text{succ}} + r_t^{\text{stab}} + r_t^{\text{cong}} + r_t^{\text{tail}} + r_t^{\text{fail}}, \tag{13}$$

$$r_t^{\text{succ}} = I_t - \eta_t \left(\rho_{\min} - \widehat{\text{Succ}}_t\right)_+,$$

$$r_t^{\text{stab}} = -\alpha_{\text{sw}} \, \text{Sw}_t, \tag{14}$$

$$r_t^{\text{cong}} = -\alpha_{\text{cong}} \, \text{Cong}_t,$$

$$r_t^{\text{tail}} = -\alpha_{\text{cvar}} \left(J_t - q_\tau(t)\right)_+, \tag{15}$$

$$r_t^{\text{fail}} = -\alpha_{\text{fail}}(1 - I_t), \tag{16}$$

where $\widehat{\text{Succ}}_t$ is an online success-rate estimate within the current episode, J_t and $q_\tau(t)$ are used for tail penalization, the coefficients $\alpha_{\text{sw}}, \alpha_{\text{cong}}, \alpha_{\text{cvar}}, \alpha_{\text{fail}} \geq 0$ are scalar weights to adjust the relative emphasis in the overall reward. $(x)_+ \triangleq \max(x, 0)$ is used to activate corresponding penalties only when the threshold is exceeded. As a Lagrangian multiplier, η_t is adaptively controlled as

$$\eta_{t+1} = \eta_t + \eta_{\text{lr}} \left(\rho_{\min} - \widehat{\text{Succ}}_t\right)_+ \tag{17}$$

where a Lagrangian relaxation [48] of the success-rate constraint is adopted, and $\eta_{\text{lr}} > 0$ is the multiplier learning rate for updating η_t. When $\widehat{\text{Succ}}_t < \rho_{\min}$, η_t is increased to strengthen the penalty in r_t^{succ}, thereby encouraging the policy to improve success rate.

Algorithm 2 Lower-level TGAT-based path selection training

Input: Dynamic network; candidate sets $\{\mathcal{P}_{s,d}\}$; TGAT encoder; PPO hyperparameters; Success-rate threshold $\rho_{\min}$
Output: Trained lower-level policy π_{bot}

```
Initialize TGAT and PPO parameters θ_bot
Initialize η_0 and tail-switching statistics q_τ(0)
for episode = 1 to N_ep do
    for time step t = 1, 2, ... do
        Updating dynamic network
        Sample request (s, d, λ) and its candidate set P_{s,d}
        Compute edge embeddings h_e(t) with TGAT
        Aggregate to path embeddings z_p(t) for all p ∈ P_{s,d}
        Build state o_t from {z_p(t)} and (s, d, λ)
        Sample action p_t ~ π_bot(· | o_t; θ_bot)
        Apply routing on p_t and compute reward r_t
        Observe the next state o_{t+1}
        Store transition (o_t, p_t, r_t, o_{t+1}) in PPO buffer
        if episode terminates then
            break
        end if
    end for
    Compute PPO loss; update θ_bot, Lagrangian multiplier
    η_t and tail quantile q_τ(t)
end for
```

Training. While the UL proposer operates on a snapshot of the network topology, the LL selector relies on a series of historical states and the candidate paths generated by the UL. The temporal embeddings from TGAT allow the policy to react to long-term patterns in link states rather than relying solely on a static snapshot. The training procedure is summarized in Algorithm 2. As shown in the pipeline, the aggregated path embeddings with the request forms the state o_t, based on which the policy selects one path for routing. The effects on environment including success, congestion, switching, etc., are then analyzed and combined into the reward computation. After each episode, the policy parameters are updated to minimize the loss through backpropagation similar to UL. Meanwhile, Lagrangian multiplier η_t is updated to enforce the expected success-rate, and tail-switching $q_\tau(t)$ is maintained to penalize bursty path switching. After some iterations, the long-term stability features $\{\text{fail\_ema}_e\}$ can be inferred in LL and fed into the training of UL to construct an interactive two-level optimization in Section III-D.

D. Interactive Optimization of Two-Level Policy

Rather than trained independently, the above described two-level policies including path candidate generation and selection are interactively trained and used to fulfill the online route planning in synergy. This is designed to train one level's policy by fixing the other level, and vice versa in an alternative manner. More importantly, the long-term stability feature $\{\text{fail\_ema}_e\}$ is used to bridge the two levels for a stability-oriented policy, which provides an interactive optimization as follows:

1) **Initialization.** Initialize the UL instability prior $\{\text{fail\_ema}_e\}$ for all $e \in E$; Initialize Algorithm 1 and Algorithm 2.
2) **Upper-level policy training.** Train the generation policy π_{top} on topology G according to Algorithm 1, using the current $\{\text{fail\_ema}_e\}$ until convergence; Apply the policy to generate candidate sets $\{\mathcal{P}_{s,d}\}$ for all (s, d).
3) **Lower-level policy training.** Fix $\{\mathcal{P}_{s,d}\}$ and train the TGAT-based selection policy π_{bot} in the dynamic environment, according to Algorithm 2 until convergence.
4) **Stability enhancement.** Apply the trained lower-level policy for a longer-term, and update the instability scores $\{\text{fail\_ema}_e\}$ from the trajectories buffer.
5) **Repeated alternations.** Repeat steps 2)–4) until the overall performance becomes stable in the final dynamic route planning.

IV. Experiments

A. Experimental Setting

1) Network Topologies and traffic patterns: The experiments are carried out on four representative backbone topologies: NSFNet, GBN, and GEANT2 [49], and a 9-node "small" network, which are used as standardized multi-hop testbeds to enable reproducible comparisons. All networks are treated as undirected graphs, and each link is assigned with the base capacity of 200 units and a pre-computed edge betweenness value. Traffic demand (s, d, λ) is a random data transmission request with bandwidth requirement $\lambda \in \{16, 24, 32, 48, 64\}$. We consider two workload modes: (i) *moderate*, where each episode sample uniformly from all ordered node pairs with $s \neq d$; and (ii) *dense*10, where ten (s, d) pairs are fixed and

TABLE I
HYPERPARAMETERS IN THE TWO LEVELS.

UL PPO (path proposer)		LL TGAT-PPO (path selector)	
Name	Value	Name	Value
Hidden units	128	Buffer size	128
Batch size	256	Batch size	64
Epoch number E_{top}	80	Epoch number E_{bot}	150
Learning rate	3×10^{-4}	Learning rate	1×10^{-4}
Discount factor γ	0.99	Discount factor γ	0.99
GAE parameter	0.95	GAE parameter	0.95
Clip ratio	0.2	Clip ratio	0.2
Entropy coefficient	0.01	Entropy coefficient	0.03
Steps per epoch	4096	TGAT heads	4
Candidate set size K	4	Temporal depth T_{tgat}	2

drawn repeatedly as high-frequency requests. The dynamics are also injected through two mechanisms: the capacity-release mechanism described in Section II-A, with the occupied capacity of an accepted request released after 3–5 time steps randomly, and an independent stochastic link-failure process, under which each edge fails with 0.5% probability in a step and recovers after an average of 3 time steps.

2) Hyperparameters: The main hyperparameters are summarized in Table I, including PPO and TGAT-related configurations. The upper-level proposer emits at most $K = 4$ candidate paths per (s, d) pair; If fewer than K distinct paths are found for a given pair, all available candidates are retained for further selection. For temporal encoding, TGAT uses 4 attention heads to capture diverse aspects of link dynamics. The temporal depth is set to $T_{\text{tgat}} = 2$, which determines the number of stacked TGAT layers for hierarchical temporal abstraction. Standard PPO is adopted for both levels with discount factor $\gamma = 0.99$, GAE 0.95, and clip ratio 0.2. The learning rate and entropy coefficient are different in two levels, with 3×10^{-4}, 0.01 in UP and 1×10^{-4}, 0.03 in LL, respectively.

3) Baselines: The proposed method is compared to four baselines, including Dijkstra, Floyd, Ant Colony Optimization, GNN+DQN, etc.

Dijkstra. As a deterministic baseline, the minimum-hop path is computed for a pair (s, d) via Dijkstra [50], [51] on the current graph and attempts to route on it.

Floyd-Warshall. Similar to Dijkstra, the minimum-hop route is computed in a deterministic way by Floyd-Warshall algorithm [10], [52].

Ant Colony Optimization (ACO). ACO is a classical heuristic method widely used in routing, that constructs $s \to d$ paths using pheromone-guided neighbor sampling [13].

DQN+GNN routing. In [21], a learning-based method is proposed and a link-centric GNN and DQN are used to jointly select a path from a small set of shortest candidates, with the reward as the carried bandwidth if the chosen path can accommodate the demand.

For the first three methods, including Dijkstra, Floyd-Warshall and ACO, the most recently successful path is stored and reused whenever it remains feasible for the repeated (s, d) request. These methods are invoked to re-route only when the current path becomes infeasible. On the contrary, this conservative reuse heuristic is not applied to learning-based methods such as DQN+GNN and the proposed method in this paper, which are evaluated under their original per-request decision structure without combining extra reuse logic.

B. Experiments Evaluation

Extensive evaluations of the proposed interactive path generation and selection policy (InterPolicy) against the baselines are carried out under both the *moderate* all-pairs mode and the *dense*10 hotspot mode, as described in Section IV-A1. Because the training of each of the two levels is similar to normal PPO-based algorithms, the cost of interactive policy training depends on the number of repeated alternations and according to our experiments, dozens of alternations can achieve converged parameter training with satisfactory performances. The results reported in this section is analyzed based on the inference of trained model with the settings in Table I.

1) Evaluation on Moderate Mode: For the *moderate* mode, Fig. 2 demonstrates the performances of different methods measured by the metrics of success rate, hop count, congestion, switching rate, etc., averaged over five simulations, with each (s, d) pair tested for ten times per simulation. While most methods tend to maintain high feasibility as shown by high success rate in Fig. 2, the difference is obvious for temporal behavior as reflected by congestion and path switching rate. From the figure, InterPolicy achieves the lowest switching rate and improves the stability significantly by penalizing the high-frequency switching through tail-avoidance. Combined with the reuse policy for repeated requests in Dijkstra, Floyd-Warshall and ACO methods, these three methods also avoid high switching rate, but are still outperformed by InterPolicy. On the other hand, because the stability is not well modelled in DQN+GNN method [21], it suffers from high switching rate. Though the success rate, hop count, congestion are all considered in DQN+GNN, it is still outperformed by InterPolicy, in which the upper-level learns to generate feasible but short paths as candidates. By selecting from the high-quality candidates, the lower-level policy can determine the path with low congestion as the final route, as shown by the low congestion of InterPolicy in Fig. 2.

Note that the DRL+GNN method optimizes its original feasibility/bandwidth-oriented objective without including explicit regularization on switching or tail-avoiding stability. As a result, multiple candidate paths generated by DRL+GNN can have similar benefits such as high success rates and short hops under stochastic failures and time-varying residual capacities. However, the stability is still not guaranteed in this method because these paths are likely to induce frequent routing churns when their utilities change abruptly. Because DQN+GNN also generates a set of candidate paths, the switching rate is highly dependent on the number of candidates K. According to our experiments, the switching rate by DQN+GNN can be increased to around 0.8 if K is increased to $K = 5$.

2) Evaluation on Dense10 Mode: The *dense*10 hotspot dataset is used to experiment on more extreme cases when

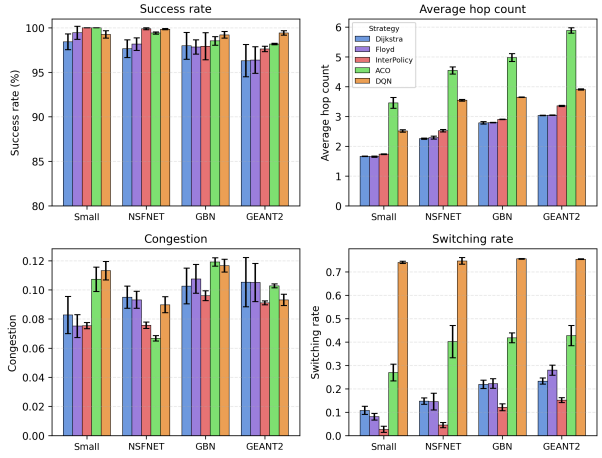

Fig. 2. Comparison using all (s, d) pairs under the *moderate* mode.

demands highly converge to certain (s, d) pairs, incurring the temporal correlation and load accumulation to be adapted by the algorithms. The bandwidth requirement is increased to higher values $\lambda \in \{64, 80, 96\}$ to impose more transmission burdens on the route planning algorithms. This challenges all the baselines as shown in Table II. However, the proposed InterPolicy method consistently yields higher success rate and more stable performances compared to the baselines. For example, the success rates by the baselines significantly decline in Table II compared to in Fig. 2, especially on larger topologies such as GBN and GEANT2 (75% vs. 97% by Dijkstra). Though success rates can be boosted to 84% by more sophisticated methods such as ACO and DQN+GNN, InterPolicy can still maintain high success rates comparable to in *moderate* mode (96% vs. 97%). Meanwhile the congestion and switching rate are both controlled at satisfactory levels. Because the data transmissions are concentrated to a small group of (s, d) pairs in *dense*10 mode, it makes sense that most methods suffer from much higher congestion without the ability to generate high-quality routes. While the baselines experience more than double their congestions, the lowest congestion by InterPolicy demonstrates its advantage in dynamic network adaption. Meanwhile, InterPolicy's best performance on switching rate in Table II indicates that the stability-aware online selection policy in the lower-level is beneficial when dealing with paths fluctuating sharply on repeatedly requested (s, d) pairs.

C. Ablation Studies

Unless otherwise stated, the ablation studies in this section are conducted on NSFNet as a representative medium-size backbone. To isolate the contribution of each component of our interactive routing policy, we consider three ablation dimensions: (i) *Path generation* (KSP vs. RLtop), contrasting K-shortest-path candidates with the generated candidates by the upper-level proposer in the interactive policy; (ii) *Path selection* (Random, Greedy, RLbot), which differs only in how a path is chosen from a fixed candidate set, using random

TABLE II
COMPARISON ON DENSE10 DATASET WITH MEAN ± STD.

Metric	Strategy	Small	NSFNet	GBN	GEANT2
Succ.↑	Dijkstra [50], [51]	96.20±2.25	84.50±3.00	79.17±1.27	75.10±1.30
	Floyd [10], [52]	95.47±0.71	83.33±4.16	78.03±1.89	76.00±2.66
	InterPolicy	**99.50±0.17**	**95.94±3.66**	**94.70±3.54**	**96.50±1.61**
	ACO [13]	97.47±1.89	91.17±1.40	86.10±0.75	84.53±0.90
	DQN+GNN [21]	98.43±0.64	89.40±1.71	87.83±1.30	84.73±0.80
Cong.↓	Dijkstra [50], [51]	0.2066±0.02	0.2315±0.06	0.2350±0.03	0.2093±0.03
	Floyd [10], [52]	0.1982±0.03	0.2551±0.05	0.2561±0.02	0.2180±0.02
	InterPolicy	**0.1175±0.01**	**0.1769±0.01**	**0.1343±0.02**	**0.1628±0.01**
	ACO [13]	0.2343±0.02	0.2933±0.02	0.2899±0.01	0.3317±0.01
	DQN+GNN [21]	0.1662±0.02	0.2913±0.01	0.2314±0.01	0.2745±0.01
Swit.↓	Dijkstra [50], [51]	0.1571±0.01	0.2464±0.06	0.2719±0.06	0.2910±0.00
	Floyd [10], [52]	0.1549±0.01	0.2523±0.05	0.2776±0.06	0.2819±0.01
	InterPolicy	**0.0757±0.01**	**0.1017±0.04**	**0.1091±0.02**	**0.1145±0.04**
	ACO [13]	0.3655±0.04	0.4405±0.05	0.4308±0.07	0.4555±0.04
	DQN+GNN [21]	0.7451±0.01	0.7492±0.03	0.7589±0.00	0.7401±0.01

TABLE III
COMPARISON WITH DIFFERENT UPPER/LOWER-LEVEL COMPONENTS.

Dataset	Upper	Lower	Succ. (%)↑	Hops↓	Congest.↓	Switch↓
Moderate	KSP	RLbot	99.20	2.36	0.1721	**0.0379**
		Greedy	**99.25**	**2.25**	0.1754	0.1355
		Random	81.70	3.50	0.1673	0.7338
	RLtop	RLbot	98.90	2.58	0.1451	0.0465
		Greedy	99.23	2.49	0.1514	0.1917
		Random	78.79	3.71	**0.1388**	0.7350
	RLtop_d	RLbot	96.50	2.51	0.1584	0.0592
Dense10	KSP	RLbot	92.70	2.81	0.2003	0.2051
		Greedy	93.30	**2.61**	0.2145	0.3010
		Random	48.70	3.62	0.2162	0.7452
	RLtop	RLbot	96.50	3.09	**0.1602**	**0.1061**
		Greedy	**96.80**	2.94	0.1793	0.1758
		Random	54.00	3.79	0.2105	0.7402
	RLtop_d	RLbot	90.70	3.22	0.1884	0.1162

method, greedy selection, or the lower-level selector in the interactive policy; and (iii) *Generation and selection Interaction*, which is analyzed by disconnecting the two layers by removing stability-related feedback of $\{\text{fail\_ema}_e\}$, denoted as RLtop_d to differentiate with RLtop. The combinations of these ablation dimensions are comprehensively compared in Table III.

1) Path Generation Ablation: The design of the upper-level proposer tries to improve the quality of the action (selection) space for the lower-level selector. As implied by Eq. (12), the upper-level proposer is a variant of KSP method implemented by RL because the path length is used in the reward to generate short paths. By fixing the lower-level method, we can find that, though the hops number by RLtop is not as short as KSP, RLtop can maintain path lengths with very close values, showing its capability in network structure learning. More importantly, the congestions can be significantly decreased by using RLtop in both *moderate* (0.1388 vs. 0.1673 for Random) and *dense*10 (0.1793 vs. 0.2145 for Greedy) mode. When data routing is highly demanded in *dense*10, the merits of RLtop are more obvious in enhancing the qualities of congestion and path switching. Taking Greedy as an example, both congestion (0.2145 vs. 0.1793) and switching (0.3010

vs. 0.1758) are decreased, by replacing KSP with RLtop. This demonstrates the advantage of RLtop in utilizing long-term instability priors to avoid persistently problematic or overused links by exploring "cooler" alternative paths.

2) Path Selection Ablation: Table III also isolates the lower-level decision policy under a fixed upper-level candidate generation configuration. As shown in Table III, the Random selection yields the worst success rate and path switching because the path selection is made without informed policy. Instead, Greedy achieves the best feasibility with a myopic heuristics, i.e., sticking to a single "currently best" candidate. For example, the success rate is the highest at 99.25% for KSP+Greedy in $moderate$ mode and 96.80% for RLtop+Greedy in $dense10$ mode. When the shortest paths are generated in the upper-level, the Greedy is also superior in path length. For example, the hops number is also the best with KSP+Greedy in $moderate$ mode (2.25) and $dense10$ mode (2.61). However, because the Greedy overcommits to temporarily favorable routes, it becomes fragile once link intermittency and request dynamics accumulate. This can be highlighted by the less satisfactory performances on congestion and path switching qualities, compared with RLbot policy. As shown in Table III, the learned RLbot achieves the most balanced trade-off among feasibility, congestion, and stability, while keeping small hops number. For example, RLbot obtains the lowest path switching in $moderate$ mode, including KSP+RLbot (0.0379) and RLtop+RLbot (0.0465), which is much lower than Greedy, while feasibility and path length are kept comparable to Greedy. The advantage of RLbot is more prominent in more challenging $dense10$ mode. The proposed interactive policy of RLtop+RLbot outperforms the others achieving the best performance, including the lowest congestion (0.1602) and switching (0.1061), and the very close success rate (96.50% vs. 96.80%) and hops number (3.09 vs. 2.94) compared to Greedy. This demonstrates the advantage of the proposed method in stability-aware online inference and adaptiveness when high transmission demands evolve dramatically over time.

3) Ablation on Stability-Aware Interactive Policy: To evaluate the necessity of interactive design of path generation and selection policy, the two variants of the proposed method including interactive (RLtop+RLbot) and non-interactive (RLtop_d+RLbot) policy are both compared in Table III. Though the disconnected two-level policy of RLtop_d+RLbot is not outstanding in the overall performances, it still preserves competitive switching performance, showing the robustness of the proposed algorithm. In addition, it still has the advantage in small switching, compared to Greedy and Random-based methods. This benefits from the self-learning design of two RL-based policies. Even without the stability-aware connections, the two policies of path generation and selection can still fulfill their inferences respectively, based on their self-contained modular policies.

When the long-term stability information is fed into the upper-level path generation, this prior information plays an important role in searching for temporally stable paths according to exponential moving average (Eq. (10)) and per-path fusion (Eq. (11)). By bridging the two-level policy with stability-related features, including state cascade and reward design in the upper-level, the path generation policy can learn to propose paths with better quality for the lower-level to choose from. As shown in Table III, the interactive policy design can enhance the performance significantly in two datasets. For example, by including the interactive mechanism, the success rate, congestion and switching can be improved from RLtop_d+RLbot to RLtop+RLbot in $moderate$ mode. When it is more challenging to migrate away from temporarily tight routes, this interaction is more crucial as reflected by the $dense10$ dataset, in which both congestion (0.1602 vs. 0.1884) and switching (0.1061 vs. 0.1162) are significantly improved.

V. Conclusion

This paper proposes an interactive two-level RL policy for adaptive online routing in dynamic multi-hop networks, which decouples candidate path generation from online selection and bridges the two levels with long-term stability features. The upper level improves candidate quality through stability-aware path generation, while the lower level performs online selection under dynamic network conditions. By coupling stability-aware candidate generation with dynamic online selection, the framework better balances routing quality and path stability under time-varying network dynamics. Experiments on multiple public network topologies demonstrate that the proposed method achieves high-quality routing and favorable trade-offs among success rate, congestion, and path switching, showing clear advantages over the baselines. Future work can be extending the proposed policy learning method to larger and specific dynamic network scenarios such as satellite constellations and practical constraints need to be incorporated to improve its applicability in real-world dynamic routing settings.

Acknowledgments

This research work was part-funded by National Natural Science Foundation of China under Grant No. 61973313 and Key Project of Basic Research in Institute of Software under Grant NO. ISCAS-JCZD-202309.

References

[1] M. Abdollahi, F. Eshghi, M. Kelarestaghi, and M. Bag-Mohammadi, "Opportunistic routing metrics: A timely one-stop tutorial survey," *J. Netw. Comput. Appl.*, vol. 171, p. 102802, Dec. 2020.

[2] R. Hamidouche, Z. Aliouat, A. M. Gueroui, A. A. A. Ari, and L. Louail, "Classical and bio-inspired mobility in sensor networks for IoT applications," *J. Netw. Comput. Appl.*, vol. 121, pp. 70–88, Nov. 2018.

[3] M. Fogli, C. Giannelli, and C. Stefanelli, "Software-defined networking in wireless ad hoc scenarios: Objectives and control architectures," *J. Netw. Comput. Appl.*, vol. 203, p. 103387, Jul. 2022.

[4] F. Gui, S. Wang, D. Li, L. Chen, K. Gao, C. Min, and Y. Wang, "RedTE: Mitigating subsecond traffic bursts with real-time and distributed traffic engineering," in *Proc. ACM SIGCOMM Conf.*, 2024, pp. 71–85.

[5] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE Conf. Comput. Commun.*, vol. 2, Mar. 2000, pp. 519–528.

[6] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 779–792, Apr. 2017.

[7] K. Zhao, P. Goyal, M. Alizadeh, and T. E. Anderson, "Scalable tail latency estimation for data center networks," in *Proc. 20th USENIX Symp. Netw. Syst. Des. Implement. (NSDI '23)*, Apr. 2023, pp. 685–702.

[8] O. S. Oubbati, M. Atiquzzaman, P. Lorenz, M. H. Tareque, and M. S. Hossain, "Routing in flying ad hoc networks: Survey, constraints, and future challenge perspectives," *IEEE access*, vol. 7, pp. 81 057–81 105, 2019.

[9] A. Rovira-Sugranes, A. Razi, F. Afghah, and J. Chakareski, "A review of AI-enabled routing protocols for UAV networks: Trends, challenges, and future outlook," *Ad Hoc Netw.*, vol. 130, p. 102790, May 2022.

[10] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla, "Exploring the 'internet from space' with Hypatia," in *Proc. ACM Internet Meas. Conf.*, Oct. 2020, pp. 214–229.

[11] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Manage. Sci.*, vol. 17, no. 11, pp. 712–716, 1971.

[12] X. Pei, P. Sun, Y. Hu, D. Li, B. Chen, and L. Tian, "Enabling efficient routing for traffic engineering in SDN with Deep Reinforcement Learning," *Comput. Netw.*, vol. 241, p. 110220, 2024.

[13] G. Di Caro and M. Dorigo, "AntNet: Distributed stigmergetic control for communications networks," *J. Artif. Intell. Res.*, vol. 9, pp. 317–365, Dec. 1998.

[14] S. Yang, S. Wang, T. Li, T. Hu, Z. Xu, R. He, and B. Zhang, "Hybrid ant colony-based inter-cluster routing protocol for FANET," *Sci. Rep.*, vol. 14, p. 15632, 2024.

[15] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "CFR-RL: Traffic engineering with reinforcement learning in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2249–2259, Oct. 2020.

[16] W. Zhou, X. Jiang, B. Guo, and L. Meng, "PQROM: To optimize software defined network QoS-aware routing with proximal policy optimization," *J. Intelligent and Fuzzy Syst.*, vol. 42, no. 4, pp. 3605–3614, 2022.

[17] H. Zhu, V. Gupta, S. S. Ahuja, Y. Tian, Y. Zhang, and X. Jin, "Network planning with deep reinforcement learning," in *Proc. ACM SIGCOMM Conf.*, Aug. 2021, pp. 258–271.

[18] P. Krämer and A. Blenk, "Navigating communication networks with deep reinforcement learning," *ECEASST*, vol. 80, Sep. 2021.

[19] Z. Xu, F. Y. Yan, R. Singh, J. T. Chiu, A. M. Rush, and M. Yu, "Teal: Learning-accelerated optimization of WAN traffic engineering," in *Proc. ACM SIGCOMM Conf.*, Sep. 2023, pp. 378–393.

[20] Y. Perry, F. V. Frujeri, C. Hoch, S. Kandula, I. Menache, M. Schapira, and A. Tamar, "DOTE: Rethinking (predictive) WAN traffic engineering," in *Proc. 20th USENIX Symp. Netw. Syst. Des. Implement. (NSDI '23)*, Apr. 2023, pp. 1557–1581.

[21] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *Comput. Commun.*, vol. 196, pp. 184–194, Oct. 2022.

[22] S. S. Bhavanasi, L. Pappone, and F. Esposito, "Routing with graph convolutional networks and multi-agent deep reinforcement learning," in *Proc. IEEE Conf. NFV-SDN*, Nov. 2022, pp. 72–77.

[23] J. Wu and Z. Zhu, "Intelligent routing optimization for SDN based on PPO and GNN," *J. Netw. Comput. Appl.*, p. 104249, 2025.

[24] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2260–2270, Oct. 2020.

[25] M. Ferriol-Galmés, K. Rusek, J. Suárez-Varela, S. Xiao, X. Shi, X. Cheng, B. Wu, P. Barlet-Ros, and A. Cabellos-Aparicio, "RouteNet-erlang: A graph neural network for network performance evaluation," in *Proc. IEEE Conf. Comput. Commun.*, May 2022, pp. 2018–2027.

[26] G. Bernárdez, J. Suárez-Varela, A. López, X. Shi, S. Xiao, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "MAGNNETO: A graph neural network-based multi-agent system for traffic engineering," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 2, pp. 494–506, 2023.

[27] M. Ding, Y. Guo, Z. Huang, B. Lin, and H. Luo, "GROM: A generalized routing optimization method with graph neural network and deep reinforcement learning," *J. Netw. Comput. Appl.*, vol. 229, p. 103927, 2024.

[28] H. I. Minhas, R. Ahmad, W. Ahmed, M. Waheed, M. M. Alam, and S. T. Gul, "A reinforcement learning routing protocol for UAV aided public safety networks," *Sensors*, vol. 21, no. 12, p. 4121, 2021.

[29] Z. Wang, R. Han, H. Li, E. J. Knoblock, R. D. Apaza, and M. R. Gasper, "Deep reinforcement learning based routing in an air-to-air ad-hoc network," in *Proc. IEEE/AIAA DASC*, Sep. 2022, pp. 1–6.

[30] H. Wu, Y. Han, M. Rajpal, Q. Zhang, and J. Wang, "SaTE: Low-latency traffic engineering for satellite networks," in *Proc. ACM SIGCOMM Conf.*, Sep. 2025, pp. 896–916.

[31] Y. Li, L. Liu, H. Li, W. Liu, Y. Chen, W. Zhao, J. Wu, Q. Wu, J. Liu, and Z. Lai, "Stable hierarchical routing for operational LEO networks," in *Proc. 30th Annu. Int. Conf. Mobile Comput. Netw.*, 2024, pp. 296–311.

[32] L. Bai, H. Ma, Y. Jiang, Z. Yin, H. Wan, and H. Wang, "GRL-RR: A graph reinforcement learning-based resilient routing framework for software-defined LEO mega-constellations," *Comput. Netw.*, vol. 259, p. 111089, 2025.

[33] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," arXiv:2002.07962, 2020.

[34] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," arXiv:2006.10637, 2020.

[35] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *J. Risk*, vol. 2, no. 3, pp. 21–41, 2000.

[36] Y. Chow, A. Tamar, S. Mannor, and M. Pavone, "Risk-sensitive and robust decision-making: A CVaR optimization approach," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, Dec. 2015, pp. 1522–1530.

[37] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *J. Mach. Learn. Res.*, vol. 18, no. 167, pp. 1–51, 2018.

[38] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2892–2901.

[39] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "TEAVAR: Striking the right utilization-availability balance in WAN traffic engineering," in *Proc. ACM Special Interest Group Data Commun.*, Aug. 2019, pp. 29–43.

[40] S. W. Roberts, "Control chart tests based on geometric moving averages," *Technometrics*, vol. 1, no. 3, pp. 239–250, 1959.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2015.

[42] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. OTexts, 2018. [Online]. Available: https://otexts.com/fpp2/

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

[44] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, Jun. 2002.

[45] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, Mar. 1977.

[46] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. 31st NIPS*, 2017, pp. 3394–3404.

[47] Y. Sun, H. Deng, Y. Yang, C. Wang, J. Xu, R. Huang, L. Cao, Y. Wang, and L. Chen, "Beyond homophily: Structure-aware path aggregation graph neural network," in *Proc. 31st IJCAI*, 2022, pp. 2233–2240.

[48] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. ICML*, vol. 70, Jul. 2017, pp. 22–31.

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

[50] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[51] T. Pan, G. Ruan, Q. Fu, Z. Luo, J. Huang, X. Luo, and T. Huang, "StableRoute: When Dijkstra's algorithm meets topology-varying satellite networks," in *Proc. IEEE Conf. Comput. Commun.*, May 2025, pp. 1–10.

[52] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, pp. 344–348, Jun. 1962.