Segment Routing Header (SRH)-Aware Traffic Engineering in Hybrid IP/SRv6 Networks with Deep Reinforcement Learning

Shuyi Liu, Yuang Chen, Zhengze Li, Fangyu Zhang,
Hancheng Lu, *Senior Member, IEEE*, Xiaobo Guo, and Lizhe Liu

***Abstract*—Segment Routing over IPv6 (SRv6) gives operators explicit path control and alleviates network congestion, making it a compelling technique for traffic engineering (TE). Yet two practical hurdles slow adoption. First, a one-shot upgrade of every traditional device is prohibitively expensive, so operators must prioritize which devices to upgrade. Second, the Segment Routing Header (SRH) increases packet size; if TE algorithms ignore this overhead, they will underestimate link load and may cause congestion in practice. We address both challenges with DRL-TE, an algorithm that couples deep reinforcement learning (DRL) with a lightweight local search (LS) step to minimize the network's maximum link utilization (MLU). DRL-TE first identifies the smallest set of critical devices whose upgrade yields the largest drop in MLU, enabling hybrid IP/SRv6 networks to approach optimal performance with minimal investment. It then computes SRH-aware routes, and the DRL agent, augmented by a fast LS refinement, rapidly reduces MLU even under traffic variation. Experiments on an 11-node hardware testbed and three larger simulated topologies show that upgrading about 30% of devices allows DRL-TE to match fully upgraded networks and reduce MLU by up to 34% compared with existing algorithms. DRL-TE also maintains high performance under link failures and traffic variations, offering a cost-effective and robust path toward incremental SRv6 deployment.**

***Index Terms*—Traffic Engineering, Segment Routing over IPv6 (SRv6), SRv6 Deployment, Deep Reinforcement Learning, Graph Neural Networks.**

I. Introduction

As network traffic grows in volume and complexity, the demand for advanced network traffic management and optimization strategies has increased significantly. Traffic engineering (TE) [1] is widely employed by network operators to manage traffic demands, aiming to prevent link congestion and enhance the utilization of network resources. However, traditional IP control planes fall short of fully achieving the objectives of TE.

Segment Routing (SR) [2], an emerging technology, achieves TE objectives by customizing traffic paths through the establishment of tunnels between the source and destination addresses. When applied to TE, this technique-known as SR-TE-offers robust traffic control capabilities, facilitating the adjustment of traffic paths and significantly improving network performance. Current research primarily focuses on two SR paradigms, namely Segment Routing over Multiprotocol Label Switching (SR-MPLS) [3] and Segment Routing over IPv6 (SRv6) [4]. Compared to SR-MPLS, SRv6 not only inherits the advantages of SR-MPLS but also simplifies network architecture since it does not require all devices within the SR tunnel to support SRv6 functionality [5].

Despite the advantages of SRv6, upgrading existing IPv6 networks to full SRv6 networks presents both economic and technical challenges [6]. Network devices are typically heterogeneous, and upgrading to SRv6 devices often necessitates hardware replacements. Deploying SRv6 across all IPv6 devices simultaneously would impose prohibitive costs on operators. Furthermore, such widespread deployment could lead to technical issues, such as network downtime and configuration errors. Fortunately, SRv6 supports coexistence with IPv6, allowing for a more gradual transition [4]. Consequently, a partial deployment of SRv6, resulting in hybrid IP/SRv6 networks, emerges as a more cost-effective and practical approach. From a TE perspective, partial deployment of SRv6 that achieves better TE optimization results while minimizing deployment costs, i.e., the number of deployed SRv6 nodes, is a key concern for operators.

In hybrid IP/SRv6 networks, the SRv6 deployment determines the available paths for traffic, and the effectiveness of TE depends on how well these paths are utilized to optimize routing. This indicates a coupling between SRv6 deployment and routing optimization. Therefore, TE research in hybrid IP/SRv6 networks focuses on both the SRv6 deployment and subsequent routing optimization. Most TE studies deploy SRv6 based on the overall traffic distribution, upgrading devices that handle larger volumes of traffic [6]–[8]. However, not all traffic demands are transmitted using SRv6, and the overall traffic distribution is insufficient to identify critical nodes that are frequently used in routing optimization. Consequently, these approaches require upgrading a larger fraction of devices to SRv6, driving up deployment costs to attain performance comparable to full SRv6 networks. In addition to deployment challenges, routing optimization presents several obstacles. Firstly, when SRv6 is used to guide traffic paths, a Segment Routing Header (SRH) [9] is added to packet headers, consisting of a sequence of Segment IDs (SIDs) that define the transmission path. Each SID occupies a certain number of bytes, increasing the packet size and consuming more band-

Shuyi Liu, Yuang Chen, Zhengze Li, Fangyu Zhang, and Hancheng Lu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: lsy55@mail.ustc.edu.cn, yuangchen21@mail.ustc.edu.cn, hclu@ustc.edu.cn, lzzsa23006160@mail.ustc.edu.cn, fv215b@mail.ustc.edu.cn).

Xiao Guo and Lizhe Liu are with the National Key Laboratory of Advanced Communication Networks, Academy for Network & Communications of CETC (email: umbrac177@163.com, liu_lizhe@sina.com).

width. Most existing studies overlook the impact of SRH on transmission, leading to inaccurate traffic scheduling strategies [7], [10]. Secondly, SRv6 deployment is typically carried out progressively, as the SRv6 deployment ratio increases, the number of available paths expands, resulting in a vast solution space and longer execution time for routing optimization.[1] However, due to frequently changing traffic patterns, network operators must generate traffic scheduling strategies quickly, posing additional challenges. If routing decisions are made too slowly, they can be outdated by the time they are applied, resulting in poorer optimization. Moreover, existing routing optimization approaches have difficulty making full use of the additional available paths as the deployment ratio increases, and incur extra costs to re-adapt to the evolving network [11], [12].

To effectively overcome the aforementioned challenges, we propose a deep reinforcement learning (DRL)-based TE algorithm, integrated with a graph neural network (GNN), referred to as DRL-TE, to minimize the maximum link utilization (MLU) in the network.[2] The key contributions of this paper are as follows:

- To achieve more precise traffic control, we define the SRv6 deployment and routing optimization (SDRO) problem, taking into account the increased bandwidth consumption caused by SRH when applying SRv6. We formulate this problem as an integer linear programming (ILP) problem and note that it is NP-hard.
- We propose a TE algorithm based on deep reinforcement learning, DRL-TE, to address this SDRO problem. The additional bandwidth consumed by SRH is encoded in the state representation used by the DRL agent. During training, the algorithm treats the network as a fully deployed SRv6 network and identifies devices that are most likely to be utilized by the DRL agent for upgrade. Once deployment is complete, the DRL agent performs routing optimization, with a lightweight local search (LS) method refining the final strategy.
- To address the challenges posed by increasing SRv6 deployment ratios, we focus on optimizing only critical traffic on congested links, thereby reducing the size of the solution space. DRL-TE leverages a GNN to improve its adaptability to networks with varying SRv6 deployment ratios.
- We evaluate DRL-TE on an 11-machine testbed and three simulated topologies across multiple SRv6 deployment ratios. With 30% of devices upgraded, it achieves performance comparable to full SRv6 networks and reduces MLU by 3–34% versus representative heuristics; under link failures, it remains within 8.6% of the optimum and is stable under moderate traffic variations. Furthermore, DRL-TE adapts to different SRv6 deployment ratios without the need for retraining and is capable of delivering optimized routing strategies within a short time.

The rest of the paper is organized as follows: Section II reviews related work. In Section III, we present the problem statement, describing the hybrid IP/SRv6 scenario, formulating the problem, and indicating its NP-hardness. In Section IV, we introduce the proposed algorithm DRL-TE. We present the performance evaluation in Section V. Finally, Section VI concludes this paper.

[1]Deployment ratio means the ratio of SRv6 devices to the total number of devices in the network.

[2]Link utilization refers to the percentage of bandwidth used on a link.

II. Related Work

In this section, we review the current research on SRv6-TE in hybrid IP/SRv6 networks. SRv6-TE in hybrid IP/SRv6 networks can be grouped into two sequential phases: incremental deployment of SRv6 devices and routing optimization once deployment is in place.

Early SRv6 deployment methods prioritize nodes by topological centrality (e.g., degree, betweenness) [6]. Since these metrics reflect only static structure and ignore traffic distribution, meaningful TE gains typically arise only after upgrading a large share of devices. Subsequent studies [6]–[8] introduce traffic-aware criteria, yet aggregated traffic statistics still struggle to reveal which nodes are truly indispensable for optimized routes, meaning many SRv6 upgrades are still required to achieve satisfactory TE performance. Therefore, in this paper, we focus exclusively on the traffic demands routed via SRv6 paths (SRv6 traffic), which enables us to identify the devices that are truly pivotal for routing optimization and to prioritize the upgrade of these nodes, yielding a more cost-effective deployment.

After deployment, the routing optimization problem can be formulated as an ILP problem solved with tools such as Gurobi [13], [14]. While optimal, solver runtimes grow rapidly with network size and exceed the minute-level timescale on which traffic patterns evolve [15]. Constraint-programming approaches (e.g., DEFO [16]) shorten runtimes but need a priori convergence estimates, which are difficult to obtain in production. Human-designed heuristics such as Q-SR [17] deliver solutions in seconds, yet their hand-tuned rules degrade when traffic patterns deviate from those assumed at design time.

Machine learning offer greater adaptability [18], [19]. Saleh *et al.* [20] and Lovén *et al.* [21] emphasize the potential of incorporating machine learning to enhance network flexibility and performance. Supervised learning and reinforcement learning (RL), as two major branches of machine learning, have been actively explored for TE. Supervised learning requires labeled data, which is impractical to obtain at backbone scale, whereas RL learns by interacting with the network environment and thus avoids manual labeling. In large topologies, however, the state/action spaces are high-dimensional and require function approximation. Therefore, deep RL (DRL) has become the prevalent RL variant for TE, using neural networks to generalize across graph-structured observations. Tian *et al.* [7] couple a DRL agent with a linear programming (LP) solver to fine-tune link weights and flow-splitting ratios, which indeed lowers the MLU under variable traffic; however, the continual weight adjustments also disturb background traffic that is otherwise unaffected by the optimization. Ren *et al.* [10] cast routing optimization as a graph optimization problem and use DRL to select SRv6 paths, but their model ignores the extra bytes carried by the SRH. When traffic spikes, this oversight can push idle links into sudden congestion. Moreover, due to the incremental nature of SRv6 deployment, the DRL agent designed for lower deployment ratios experiences performance degradation, necessitating additional costs for adjustments, limiting scalability [22]. GNNs can enhance the scalability of the DRL model by capturing hierarchical relationships between nodes and links, yet none of the existing GNNs account for SRH overhead, rendering them unsuitable for SRv6 routing optimization [15], [22], [23]. In this paper, we fill that gap by designing a GNN that explicitly incorporates the additional bandwidth consumed by SRH, and by coupling it with a DRL agent that remains effective as the SRv6 deployment ratios grow.

In summary, prior SRv6-TE studies break down along two fault lines. First, their deployment heuristics rely on static topology or coarse traffic aggregates, so they must upgrade a large share of devices before achieving adequate TE improvements. Second, existing routing optimizers tend to either ignore SRH overhead or falter as the action space expands during the incremental rollout of SRv6.

TABLE I
SUMMARY OF NOTATIONS

Symbol	Description
$G = (\mathcal{N}, \mathcal{L})$	The network graph, where $\mathcal{N}$ represents the set of nodes in the topology, and $\mathcal{L}$ represents the set of links.
$C(l)$	The capacity of link l.
TM	The traffic matrix, whose entry represents the bandwidth of traffic sent from source to destination.
$\mathcal{F}$	The set of all traffic demands of a TM.
$src(f)$, $dst(f)$, $d(f)$	The source node, destination node and the bandwidth of the traffic demand f.
$b(f_i)$, $m(f_i)$	The average packet size, packet transmission rate of service flow f_i.
$P(f)$	The set of available paths of f.
$\Phi_p(f)$	The set of subpaths of path p for demand f.
$r_{f,p}$	The SRv6 encapsulation header length when traffic demand f selects path p.
$d'_p(f)$	The additional bandwidth required to transmit the SRv6 encapsulation header for demand f when f selects path p.
α	The SRv6 deployment ratio.
$I(n)$, $I^*(n)$	Binary variables: $I(n) = 1$ if node n is selected for SRv6 upgrade in the current stage, and $I^*(n) = 1$ if node n is already SRv6-enabled.
$y_f(p)$	A binary variable represents whether demand f is routed through path p or not.
$z_\phi(l)$	A binary variable represents whether subpath $\phi_p(f)$ traverses link l or not.
$x(\phi)$	A binary variable represents whether $\phi_p(f)$ is subpath II or not.
U_{max}	The maximum link utilization.

III. PROBLEM STATEMENT

In this section, we first describe the hybrid IP/SRv6 network scenario and model the network environment accordingly. Based on this model, then we formulate the TE problem using LP methods.

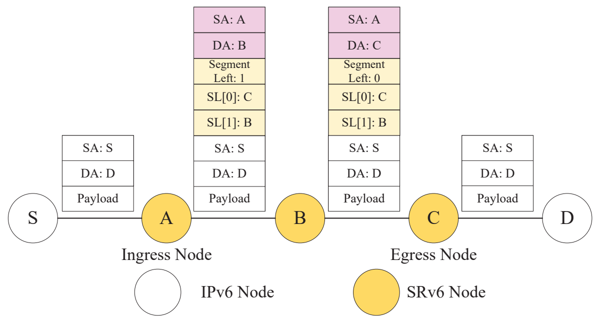


Fig. 1. An example of packet transmission within a hybrid IP/SRv6 network.

A. Hybrid IP/SRv6 Network Scenario

Table I summarizes the definitions of the symbols used in this paper. In the hybrid IP/SRv6 network scenario, we assume that, due to economic and technical constraints, operators have not fully deployed SRv6. The network topology is represented as a graph $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ denotes the set of nodes, and $\mathcal{L}$ indicates the set of links. We consider all nodes support IPv6, with Open Shortest Path First (OSPF) as the underlying default routing protocol. These nodes are interconnected by links, with link weights configured by the operator. The OSPFv3 protocol supports IPv6 nodes in forwarding SRv6 control messages. Each node maintains a forwarding table for the network, packets can be properly forwarded between IPv6 nodes and SRv6 nodes. Each link has a capacity $C(l)$, representing the maximum data it can carry per unit of time. The set of all traffic demands is denoted as $\mathcal{F}$, where each traffic demand is represented by f. The source and destination nodes for a traffic demand f are denoted as $src(f)$ and $dst(f)$, respectively, and $d(f)$ represents the bandwidth required for the traffic demand. We assume the presence of a logically centralized controller in the network, which can promptly obtain the current traffic demands, represented by a traffic matrix (TM). The TM is a table that quantifies the volume of traffic between each source–destination node pair in a network over a given time period.

Figure 1 illustrates an example of packet transmission within a hybrid IP/SRv6 network. Initially, the source node S generates a packet, which is forwarded to the ingress SRv6 node A. Node A identifies that the packet belongs to a specific traffic demand and encapsulates the SRv6 transmission path into the packet as an SRH. The path is encoded into the Segment List (SL) and placed within the SRH. In this paper, the packet, when using SRv6, includes an additional IPv6 header and the SRH. This encapsulation method avoids modifying the original data packet, ensuring compliance with the IPv6 protocol standard [24]. We refer to the combination of the SRH and the additional IPv6 header as the SRv6 encapsulation header. The figure shows the key components of the SRv6 encapsulation header, where the yellow blocks represent the SRH, and the red blocks indicate the additional IPv6 header. The SL consists of a series of Segment IDs (SIDs), and the Segments Left field denotes the number of SIDs remaining to be processed, helping the SRv6 node identify the next SID to execute. Each SID is a 16-byte IPv6 address that specifies the operations the packet must perform as it traverses the network. These operations may include forwarding the packet through intermediate nodes (i.e., node segments), across specific links (i.e., adjacency segments), or executing particular service functions (e.g., firewall services). In this paper, only node segments are considered. During transmission, the first SID (i.e., node B) is set as the destination IP of the outer IPv6 header, initiating the transmission. Nodes A, B, and C do not need to be directly connected; intermediate IPv6 nodes can be traversed, with transmission managed by the OSPF protocol. As the encapsulated packet reaches the intermediate SRv6 node B, node B processes the SL in the SRH, updates the next SID (i.e., node C) as the new destination address, and decrements the Segments Left field by one. Upon reaching the final egress SRv6 node C, which detects that the Segments Left field is 0 and its IP address matches the destination IP, the SRv6 encapsulation header is removed. The packet is reverted to its conventional IPv6 format. Finally, the packet is forwarded to the destination D.

In practice, fine-grained flow splitting is often unavailable or impractical: many devices offer only coarse per-demand controls or even disallow splitting due to hardware and operational constraints [25], and enforcing fractional allocations inflates control-plane state and scheduling overhead. Consequently, consistent with prior work [11], [26], [27], we assume single-path routing, i.e., each traffic demand is carried on exactly one path. In this work, we model each traffic demand $f \in \mathcal{F}$ as a composite flow, aggregating heterogeneous service flows (e.g., media streams, online transactions, file transfers) that share the same source–destination pair. This choice mirrors operational practice, where controllers manage traffic at the source–destination (SD) aggregate granularity rather than per micro-flow [28]. Since packet sizes and bandwidth needs differ markedly across service flows, and the SRv6 header adds a fixed number of bytes per packet, the header's bandwidth overhead varies from one flow type to another. Treating them as a composite flow, therefore, yields a more realistic accounting of SRH overhead while keeping the optimization problem tractable. We denote a service flow within a traffic demand f as f_i. $b(f_i)$ represents the average packet size of f_i, and $d(f_i)$ represents the required bandwidth. The total bandwidth required for a traffic demand f equals the sum of the bandwidth requirements of all service flows within f and is calculated as follows:

$$\sum_{f_i \in f} d(f_i) = d(f), \quad \forall f \in \mathcal{F}. \tag{1}$$

To ensure stable quality of service, we assume that the packet transmission rate remains constant along the end-to-end path in each service flow, with $m(f_i)$ packets sent per second. The relationship between $m(f_i)$, the packet size $b(f_i)$, and the required bandwidth $d(f_i)$ is expressed as

$$d(f_i) = m(f_i) \cdot b(f_i), \quad \forall f_i \in f, \forall f \in \mathcal{F}. \tag{2}$$

For each traffic demand f, the set of available paths is denoted as $P(f)$. As illustrated in Figure 1, when traffic demand is transmitted over a path $p \in P(f)$ using SRv6, the packets traverse three types of subpaths, denoted by $\Phi_p(f)$. The first subpath (subpath I) is the transmission from the source node S to the SRv6 ingress node A. The second subpath (subpath II) consists of the paths between intermediate SRv6 nodes specified by the SRH, such as the paths between A, B, and C. The third subpath (subpath III) is the transmission from the SRv6 egress node C to the destination node D. During subpath II, the packet size increases due to the SRv6 encapsulation header, which results in additional bandwidth requirements for the service flow f_i.

We denote the additional header length introduced by SRv6 for traffic demand f on path p as $r_{f,p}$. When default routing is used, $r_{f,p}$ is 0. We assume the original packet size remains unchanged during transmission. We model the SRH as a fixed-length header that stays intact along the path, which reflects common behavior in current SRv6 devices and yields a conservative (worst-case) estimate of bandwidth overhead. If future deployments shorten the header by removing traversed SIDs, the actual overhead will be smaller, and our results will only improve; supporting such variable-length SRH is left for future work. Moreover, each packet undergoes encapsulation and decapsulation of the SRH at most once, implying at most one instance of subpath I and III in the end-to-end transmission. The number of subpath II segments equals the number of segments in the SRH. The additional bandwidth required to transmit the SRv6 encapsulation header for service flow f_i is denoted as $d'_p(f_i)$ and is calculated as

$$d'_p(f_i) = r_{f,p} \cdot m(f_i). \tag{3}$$

The total additional bandwidth required for traffic demand f on path p, denoted as $d'_p(f)$, is given by:

$$d'_p(f) = \sum_{f_i \in f} d'_p(f_i). \tag{4}$$

According to [4], SRH imposes both a base overhead, denoted as $r^1_{p,f}$, and additional overhead from SIDs, denoted as $r^2_{p,f}$. Disregarding the length of the optional Type-Length-Value (TLV) field, we assume $r^1_{p,f} = 8$ bytes. The length occupied by the SIDs, $r^2_{p,f}$, is calculated as

$$r^2_{p,f} = 16 * q_{p,f}, \tag{5}$$

where $q_{p,f}$ represents the number of SIDs in the SL for path p_f. According to [29], two segments are sufficient for achieving TE optimization goals, so we limit the number of SIDs to a maximum of two. Considering that excessively long headers may cause packet sizes to exceed the MTU, leading to fragmentation and unnecessary overhead, SRH compression techniques are commonly employed in practical deployments to prevent fragmentation [30]. If fragmentation cannot be avoided, the controller typically accounts for the bandwidth implications of packet fragmentation in advance. A detailed discussion of this topic is beyond the scope of this study; we intend to explore it in subsequent research. The length of the additional IPv6 header is fixed at 40 bytes, i.e., $r^3_{p,f} = 40$

bytes. Thus, the total length of the SRv6 encapsulation header for traffic demand f on path p is calculated as

$$r_{f,p} = r^1_{p,f} + r^2_{p,f} + r^3_{p,f}. \tag{6}$$

B. Problem Formulation

The goal of this paper is to partially deploy SRv6 and schedule traffic to minimize the MLU. Since SRv6 deployment determines the available paths for subsequent traffic, it is necessary to first decide which nodes will be upgraded to support SRv6. Additionally, SRv6 deployment is a gradual process, with α representing the deployment ratio at the current stage. We use a binary variable $I(n)$ to indicate whether node n is deployed with SRv6 at this stage, and a binary variable $I^*(n)$ to represent whether node n has already been deployed with SRv6. After this stage of deployment, the proportion of SRv6 nodes in the network should not exceed the deployment ratio α. This can be expressed as follows:

$$\sum_{n\in\mathcal{N}} I(n) + \sum_{n\in\mathcal{N}} I^*(n) \leq \alpha \cdot |\mathcal{N}|, \tag{7}$$

where $|\mathcal{N}|$ represents the total number of nodes in the network. To prevent redeploying SRv6 on nodes that have already been upgraded, we can obtain that

$$I(n) + I^*(n) \leq 1, \quad \forall n \in \mathcal{N}. \tag{8}$$

When optimizing transmission paths for traffic demands, we must ensure that the total bandwidth allocated on each link does not exceed the MLU. This constraint is expressed as follows:

$$\sum_{f\in\mathcal{F}} \sum_{p\in P(f)} \sum_{\phi\in\Phi_p} y_f(p) \cdot z_\phi(l) \cdot (d(f) \\ + x(\phi) \cdot d'_p(f)) \leq U_{max} \cdot C(l), \ \forall l \in \mathcal{L}, \tag{9}$$

where $x(\phi)$ is a binary variable indicating whether subpath $\phi_p(f)$ belongs to subpath II (set to 1 if true, 0 otherwise), and $y_f(p)$ is a binary variable representing whether traffic demand f is routed through path p. The variable $z_\phi(l)$ indicates whether subpath $\phi_p(f)$ traverses link l. As described in Section III-A, the additional bandwidth $d'(f)$ required for traffic demand f after SRv6 encapsulation depends primarily on the number of SIDs, denoted by $q_{p,f}$, which is derived based on the selected path p_f.

To ensure that each traffic demand reaches its destination, we impose the constraint as follows:

$$\sum_{p\in P(f)} y_f(p) \geq 1, \ \forall f \in \mathcal{F}. \tag{10}$$

With these constraints in place, we can formulate the SRv6 Deployment and Routing Optimization (SDRO) problem as an ILP model:

$$\min_{I_f(p)} \ U_{max} \tag{11a}$$

$$\text{s.t. } (1) - (10), \tag{11b}$$

Here, the variables to be solved include $I(n)$ and $y_f(p)$. We assume that α, $d(f_i)$, $m(f_i)$, and the TMs are known and that $I^*(n)$, $z_\phi(l)$, and $d(f)$ can be obtained based on the TMs and network topology.

When $\alpha = 1$, as each traffic demand is allocated only a single path, the problem reduces to a single-path multicommodity flow problem (SMCFP). It has been proven that SMCFP is NP-hard [31]. Our problem can be reduced to SMCFP in polynomial time because solving the SDRO problem with α set to 1 yields a solution to the corresponding SMCFP. Therefore, the complexity of the SDRO problem is NP-hard.

IV. SRv6 Deployment and Routing Optimization

In this section, we introduce the DRL-TE algorithm, a DRL-based approach integrated with a GNN to address the SDRO problem. DRL-TE optimizes SRv6 deployment with DRL algorithm proximal policy optimization (PPO) offline [32], and combines DRL-based decision making and LS method to optimize routing for TMs online.

A. DRL-TE Framework

We begin by outlining the DRL-TE framework. As discussed in the previous section, the SDRO problem not only pertains to routing optimization but also involves network design considerations. Once SRv6 deployment is completed, the infrastructure typically remains fixed for a period, whereas TMs frequently vary over time, necessitating adaptive routing optimization decisions. Consequently, DRL-TE operates in two distinct phases:

1) Offline SRv6 Deployment: The offline SRv6 deployment provides the set of SRv6 nodes necessary for subsequent routing optimization. Specifically, we employ the DRL training algorithm PPO to perform simulated routing optimization. After training, the deployment of SRv6 is carried out according to the SRv6 traffic distribution obtained from the routing optimization.

2) Online Routing Optimization: In the online routing optimization, we optimize routing decisions for varying TMs. To further enhance optimization performance, we employ a rapid LS method to refine the routing decisions output by DRL, aiming to minimize the MLU.

We now separately introduce the offline training of the DRL agent, the deployment of SRv6, and the online routing optimization.

B. Offline DRL Agent Training

Since the SDRO problem described in Section III-B is NP-hard, we resort to DRL. During training, the DRL agent interacts with the environment; in our setting, the environment is the network, and the agent is the proposed DRL-TE algorithm. When a new TM arrives, the agent first runs an OSPF-based simulation to derive the initial network state. It then identifies the set of critical traffic (see Section IV-B.4) and optimizes its routing. At each time step t, for the demand f currently being optimized, the agent temporarily removes f from the network, simulates its allocation over available paths, aggregates the resulting link load vector and related features into a state s_t, feeds s_t to its neural network, and outputs an action a_t, namely the selected available path for f. The reward r_t is computed from the change in the MLU $U_{\max}$ before and after applying a_t. The following are the specific settings of DRL:

1) *State* s_t: Since the focus is on the impact of traffic allocation on link bandwidth, the network state is determined by link characteristics. Each link $i \leq |\mathcal{L}|$ is represented by a vector $\boldsymbol{\nu}_i = [v_1^i, v_2^i, v_3^i, v_4^i, 0, ..., 0]$. Here, v_1^i represents the bandwidth capacity of link l, v_2^i represents the link utilization before the demand is assigned, v_3^i represents the bandwidth the currently allocated traffic will occupy, and v_4^i indicates the additional bandwidth required due to the SRv6 encapsulation header. To ensure consistent vector dimensions, we pad the vector with zeros to a fixed dimension Ω. When selecting the j-th available path, the network state is represented by the states of all links, $\mathbf{o}_j = [\boldsymbol{\nu}_1^{\mathrm{T}}, ..., \boldsymbol{\nu}_{|\mathcal{L}|}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{L}| \times \Omega}$. The overall state at time step t is expressed as $s_t = [\mathbf{o}_1, ..., \mathbf{o}_{|P(f)|}]$, where $|P(f)|$ is the number of available paths in $P(f)$.

2) *Action* a_t: The agent's task is to reconfigure the traffic demand path. The action a_t at time step t is the selection of a path from the available paths $P(f)$ for the demand f, i.e., $a_t = p, p \in P(f)$.

3) *Reward* r_t: The reward is calculated as shown in Equation (12). Specifically, r_t represents the change in $U_{\max}$ before and after reconfiguring the traffic path, multiplied by 10 to scale the reward appropriately:

$$r_t = 10(U_{\max}(s_{t-1}) - U_{\max}(s_t)). \tag{12}$$

4) *Critical traffic*: Given that backbone measurements consistently show most congestion arises from just a small subset of links and a handful of large traffic demands, concentrating optimization on these hotspots makes routing optimization more efficient while greatly reducing control-plane overhead [33]. We focus on optimizing only the most critical portion of the traffic, referred to as critical traffic. The agent sorts them by utilization, identifying the top $\mu\%$ most congested links, referred to as congested links. Since congested links often carry a large number of demands, optimizing all of them would not effectively reduce complexity. Therefore, we sort the traffic demands passing through congested links by bandwidth and select the top $\zeta\%$ of these demands as the critical traffic.

The agent focuses on optimizing the paths for the critical traffic. For each demand, the agent selects a path from the available paths. To achieve this, the agent first calculates the available paths, which are composed of three types of subpaths. Given an SRv6 deployment ratio α and a network with $|\mathcal{N}|$ nodes, $|\mathcal{L}|$ links, and $|\mathcal{F}|$ traffic demands, the Floyd-Warshall algorithm [34] is employed to compute the subpaths. This involves computing the shortest paths between all nodes based on OSPF, followed by calculating paths from the source node to the SRv6 ingress node, paths between SRv6 nodes, and paths from the SRv6 egress node to the destination node. The number of SRv6 nodes is represented as $\lceil \alpha \cdot |\mathcal{N}| \rceil$, and the computational complexity of this step is $\mathcal{O}(|\mathcal{N}|^3 + \alpha \cdot |\mathcal{F}| \cdot |\mathcal{N}| + (\alpha \cdot |\mathcal{N}|)^2)$. Subsequently, Depth-First Search (DFS) is used to assemble the subpaths into available paths for all demands, while eliminating redundant or looping paths. The computational complexity of DFS is $\mathcal{O}((\alpha \cdot |\mathcal{F}| \cdot |\mathcal{N}|)^2)$ [7]. Since these available paths can be computed and stored offline and are merely looked up during the online phase, this preprocessing cost is typically acceptable. After computing all possible paths, we use the K-shortest paths algorithm to select a subset of paths with minimal link overlap, thereby reducing the negative effects of potential link failures. Typically, $K = 4$ is adopted for general networks, while $K = 5$ is preferred for larger-scale networks [28].

Algorithm 1: DRL-TE Training Process

```
   Input: Network topology $G$ =($\mathcal{N}$,$\mathcal{L}$), TMs.
   Output: Trained actor network parameters $\theta$.
1  Initialize actor network $\pi(s|\theta_0)$ and critic network
     $Q(s, a|\phi_0)$ with random $\theta_0$, $\phi_0$
2  for $e$ = 1 $\cdots$ $E$ do
3      Initialization operation
4      while replay buffer $RB$ is not full do
5          Collect and store experience data
             $\{s_t, a_t, r_t, s_{t+1}, q_t\}$ by running policy in the
             environment
6      end
7      Compute advantages $A_t$ and returns $R_t$
8      for $j$ = 1 $\cdots$ $J$ do
9          minibatch $RB^{'} = RB$.sample($N$)
10         for $\{s_t, a_t, r_t, s_{t+1}, q_t\} \in RB^{'}$ do
11             Compute the clipped surrogate objective:
                 $\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$
                 $L(t) = \min(\rho_t A_t, \mathrm{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A_t)$
12         end
13         Compute actor loss $L_\theta$:
             $L_\theta = -\frac{1}{N}\sum_{t=1}^{N} L(t) - \beta H(\pi(s_t|\theta))$
14         Compute critic loss $L_\phi$:
             $L_\phi = \frac{1}{N}\sum_{t=1}^{N}(Q(s_t|\phi) - R_t)^2$
15         Compute gradient and update actor and critic
             network parameters
16     end
17 end
18 return Trained actor network parameters $\theta$
```

To avoid retraining after deployment ratio changes, training is performed only in a full SRv6 network, where the network is treated as fully upgraded, and traffic allocation is simulated accordingly. The PPO algorithm is implemented to train the agent, which is an on-policy, actor-critic, policy-based algorithm. The training process of DRL-TE is outlined in **Algorithm** 1. The input includes the network topology G, which contains link weights, capacities, and TMs. The output is the trained actor network parameters θ. In **Line 1**, the actor and critic networks are initialized randomly. Both networks are built on GNNs, where θ_0 and ϕ_0 represent the initialized parameters, and s represents the input network state. The actor network outputs the probability distribution for selecting a path, while the critic network estimates the value of the current state. The critic network's output q_t helps assess the quality of actions taken by the actor. The agent is trained over E episodes, beginning with the initialization of the replay buffer

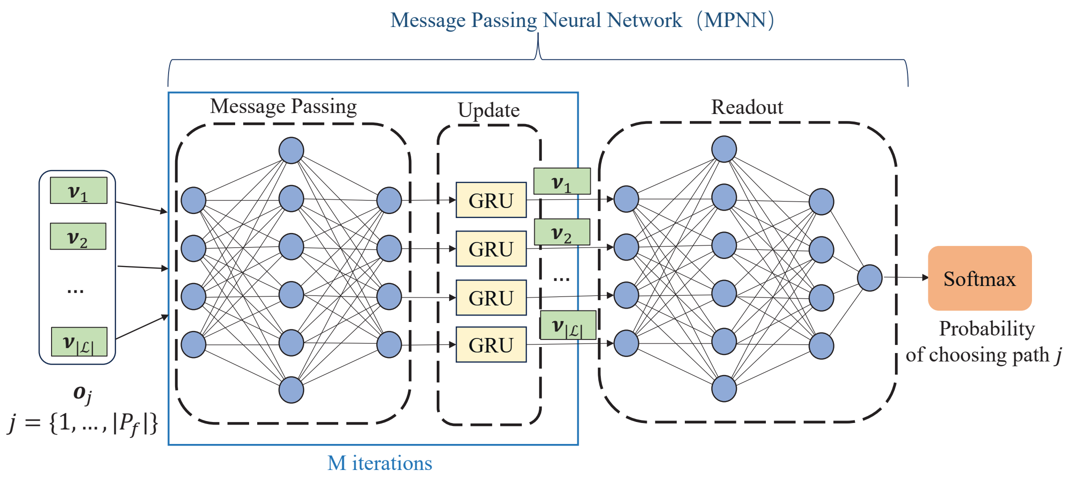

Message Passing Neural Network (MPNN)
Message Passing
Update
Readout
$\boldsymbol{\nu}_1$
$\boldsymbol{\nu}_2$
...
$\boldsymbol{\nu}_{|\mathcal{L}|}$
$\boldsymbol{o}_j$
$j = \{1, \ldots, |P_f|\}$
GRU
GRU
GRU
GRU
Softmax
Probability of choosing path j
M iterations

Fig. 2. The MPNN architecture consists of four key functions: Message Passing, Update, Readout, and Softmax.

RB and the random selection of a TM, for which demands are initially allocated using OSPF. Critical traffic is then identified (**Line 3**).

After initialization, the agent starts interacting with the environment. As mentioned earlier, the agent sequentially optimizes the paths for all traffic demands in the critical traffic, starting with the largest demand. For each demand's available paths, the network state s_t is input into the actor network, and a softmax function is applied to obtain the path selection probability distribution. An action a_t is then sampled based on this distribution. The agent allocates the traffic according to the selected action, receives the corresponding reward r_t and state evaluation q_t, and stores the experience data in the replay buffer [35], completing the routing optimization for one demand (**Lines 4-6**). Once the replay buffer is filled at time T, training of the actor and critic networks begins. Based on the experience data in the replay buffer, the advantage function A_t is estimated using generalized advantage estimation (GAE), as follows:

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{(T-1-t)}\delta_{T-1}, \tag{13}$$

where the temporal difference (TD) error is defined as $\delta_t = r_t + \gamma q_{t+1} - q_t$, γ is the discount factor, and λ is the GAE smoothing parameter. The return R_t is calculated as (**Line 7**)

$$R_t = A_t + q_t. \tag{14}$$

During each training loop, a minibatch RB' is selected from the replay buffer for each epoch (**Line 9**). For each experience in RB', the policy ratio ρ_t is calculated to represent the difference between the new and old policies, where π_{old} denotes the previously updated policy. The objective function for policy optimization is calculated using the clip form, with the clip parameter ϵ controlling the policy update range to ensure stability (**Lines 10-12**). The losses for both the actor and critic networks are then computed (**Lines 13-14**). The entropy term $H(\pi(s_t|\theta))$, which measures the randomness of the policy, is included in the loss function, with lower entropy indicating a more deterministic policy. The coefficient β controls the influence of the entropy term in the loss function. Finally, gradient descent is used to update the parameters of both networks based on the calculated losses (**Line 15**). After E episodes, the trained actor network parameters θ are returned.

To enable the neural network in the DRL framework to better adapt to a complex network environment where SRv6 nodes coexist with regular IPv6 nodes, we utilize DRL to train the message passing neural network (MPNN) [36], a type of GNN, understanding the network state through message passing between links. The structure of the MPNN, as shown in Figure 2, consists of four functions: Message Passing, Update, Readout, and Softmax. The state of each link in the network, $\boldsymbol{\nu}_i$, is input into the network and first processed through a fully connected network to enable message passing between the links. Next, a Gated Recurrent Unit (GRU) is used to learn state changes and update the link states. The message passing and update steps are iterated M times. After the iterations, the updated link states are passed through a Readout function, constructed with a fully connected network, to output the probability of selecting a particular allocation scheme. The time complexity of the MPNN can be approximated as $\mathcal{O}(M \cdot |\mathcal{L}| \cdot \Omega^2)$. In our setting, M is typically less than 10, and the feature dimension Ω is small (e.g., $\Omega = 20$). The forward-pass cost scales linearly with network size (through the number of links) and involves about 10^6 operations for a thousand-link graph. Under standard roofline reasoning, a single CPU core sustaining tens of GFLOP/s executes 10^6 FLOPs at the microsecond level [37], i.e., well below 1 ms; hence, the MPNN is practically lightweight. The Softmax function normalizes the values obtained from the Readout function. During DRL training, the parameters of the Message passing, Update, and Readout functions are optimized.

C. Offline SRv6 Deployment

To make the SRv6 deployment more strategic, we let the agent optimize routing for a specific TM and then determine which IPv6 nodes should be upgraded to SRv6 nodes based on their importance in the network after optimization. Inspired

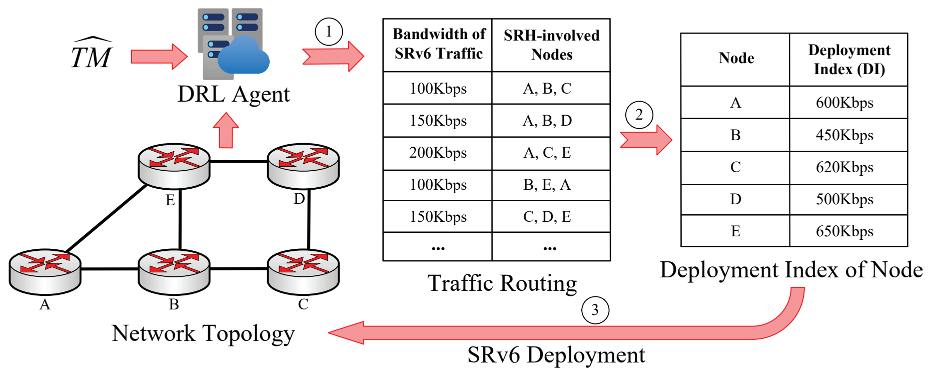


Fig. 3. SRv6 deployment is performed based on the deployment indexes of network nodes after DRL optimizes the routing for the representative traffic matrix $\widehat{TM}$.

by [7], rather than selecting an arbitrary TM, we calculate the barycenter of the TMs to obtain a representative $\widehat{TM}$.

Figure 3 shows an example of the SRv6 deployment. We treat the network as a full SRv6 network and perform simulated traffic allocation using $\widehat{TM}$. Initially, all nodes are assigned a deployment index (DI) of 0. SRv6 nodes are then deployed based on the DI values and the deployment ratio α. First, the agent allocates the demands in $\widehat{TM}$ using the default OSPF routing protocol. Next, the agent identifies the critical traffic in the network. For each demand in the critical traffic, the agent optimizes routing by selecting the path with the highest probability output by the actor network. After routing optimization, the influence of traffic still using default paths is eliminated, focusing only on SRv6 traffic demands, referred to as SRv6 traffic. By analyzing the distribution of SRv6 traffic, we identify which nodes play key roles in routing optimization.

The agent then inspects the SRH in the packets of these demands to identify the nodes included in the SRH. For example, if node A is included in the SRH of demand f, the deployment index (DI) of node A is incremented by the bandwidth of demand f. After updating the DI for all SRv6 traffic in $\widehat{TM}$, the nodes are ranked in descending order according to their DI values. Nodes already deployed as SRv6 nodes are excluded. Based on the DI ranking, the top ΔN IPv6 nodes are upgraded to SRv6 nodes, completing the deployment process.

Assuming the deployment ratio must reach α after this upgrade, with $|\mathcal{N}|$ representing the total number of nodes and α_0 being the deployment ratio before the upgrade, the number of new SRv6 nodes, ΔN, is calculated as follows:

$$\Delta N = \lceil (\alpha - \alpha_0) \cdot |\mathcal{N}| \rceil . \tag{15}$$

After deploying a certain number of SRv6 devices, if further deployments no longer yield significant improvements in TE performance, we consider the initial partial SRv6 deployment complete. However, if the traffic pattern later undergoes a sustained shift that degrades DRL-TE's performance, we can collect updated TMs and perform brief online retraining to restore accuracy. If performance remains unsatisfactory after retraining, our deployment approach can be repeated: obtaining a new representative TM based on the shifted traffic pattern and incrementally deploying additional SRv6 devices on top of the initial deployment, thus enhancing optimization performance.

D. Routing Optimization

During traffic routing optimization, SRv6 deployment is incremental, with the deployment ratio changing over time. As a result, DRL-TE encounters varying network scenarios. To avoid the need for retraining, we use the DRL agent trained under a full SRv6 deployment (deployment ratio of 1) to perform the initial traffic routing optimization. The agent first allocates all demands in the TM according to the OSPF protocol for initialization. It then calculates the critical traffic in the network and begins optimization. At this stage, the agent treats the network as if the SRv6 deployment ratio is 1, selects a path to optimize the routing for each traffic demand f, and optimizes the critical traffic sequentially.

Since the agent assumes the network operates with a full SRv6 deployment, there may be cases where some traffic demands are mistakenly routed through regular IPv6 nodes when the actual deployment ratio is less than 1. These nodes, not yet upgraded to SRv6, cannot parse the SRH and guide demands along specific paths. Additionally, data-driven DRL only iterates over the TM once, which can lead to performance degradation when faced with TMs of significantly different distributions. Therefore, we introduce a fast decision-making correction phase following the initial optimization to further refine traffic routing.

Specifically, after producing initial routing decisions, the agent checks the current SRv6 deployment to ensure the selected paths are feasible. Any infeasible routing decisions are filtered out, while only valid ones are retained. The valid routing decisions are not executed immediately in the real network. Instead, they are first simulated to help identify new critical traffic for further optimization in the subsequent correction phase.

In the correction phase, a hill-climbing-based LS method is then applied to iterate through the available paths for each demand in the critical traffic. The method can be stopped at any moment and still returns a feasible routing plan, or it can keep running until no further reduction in MLU is achievable. The resulting routing configuration remains valid throughout.

After the correction phase is completed, the optimized routing decisions are implemented in the real network.

V. Performance Evaluation

In this section, we present the performance evaluation of DRL-TE on a real-world testbed built with physical machines, as well as simulation assessments in larger-scale networks.

A. Testbed Evaluation Setting

TABLE II
Service Flow Parameter Settings

Service flow class	Parameter	Range of Values	Unit
Service flow 1	Packet size	[800, 1000]	Bytes
	Bandwidth	[400, 1000]	Kbps
Service flow 2	Packet size	[100, 500]	Bytes
	Bandwidth	[50, 100]	Kbps
Service flow 3	Packet size	[800, 1000]	Bytes
	Bandwidth	[100, 400]	Kbps

The TopologyZoo dataset [38] is widely used for analyzing TE, as it contains topologies of various real-world networks. We construct the testbed using 11 physical machines, following the Sprint topology (11 nodes, 36 edges, 110 demands). The physical machines are running Ubuntu 22.04, and we use the iperf [39] tool to transmit UDP traffic between the machines, with control over the packet sizes. We generate 200 TMs for the testbed, of which 100 TMs are used for training, while the remaining 100 are reserved for evaluation. For each traffic demand in the TM, we generate the packet size and bandwidth demand for the internal service flows. Three types of service flows are generated, as detailed in Table II. The packet sizes and bandwidth requirements for each service flow are randomly selected from the ranges provided in the table. Service flow 1 simulates video traffic with higher bandwidth demands, service flow 2 simulates online transaction traffic with lower bandwidth requirements and smaller packet sizes, and service flow 3 simulates file transfer traffic, which has a large total transmission volume but relatively lower bandwidth requirements [40], [41]. To better illustrate the experimental results, we scale the network capacities such that the MLU in the most congested case approaches 100%, and we achieve this using the traffic control (tc) tool [42].

The link selection percentage μ is 10%, and the demand selection percentage ζ is 15%. The reasoning behind these values will be explained in Section V-C. DRL-TE is implemented in Python, with the DRL and MPNN components developed using TensorFlow [43]. The training and evaluation of the algorithm are both conducted on the testbed. The relevant hyperparameter settings are shown in Table III. Hyperparameters that are widely recommended in the PPO literature, including clip range $\epsilon = 0.1$, discount factor $\gamma = 0.99$, and GAE $\lambda = 0.96$, are adopted because they consistently yield stable training in our preliminary tests and prior DRL-based TE studies [15], [22]. For the remaining hyperparameters (learning-rate, entropy coefficient, and the number of MPNN message-passing iterations), we perform a coarse grid search and select the setting that achieved the best balance between convergence speed and final MLU.

TABLE III
DRL-TE Hyperparameter Settings

Hyperparameter	Value
Training epochs J	6
Episodes E	2000
State number Ω	20
Result buffer RB size	3200
Mini-batch RB' size	64
Clip range ϵ	0.1
Decay rate, steps	0.96, 60
Entropy beta β (After 60 episodes)	0.01 (0.001)
Initial (Minimum) learning rate	0.0002, 1×10^{-5}
GAE γ, λ	0.99, 0.96
MPNN iteration M	4
Readout units, activation function	20, Selu

B. Simulation Evaluation Setting

To evaluate the performance of DRL-TE in larger topologies, we select three topologies from the TopologyZoo dataset for simulation: EliBackbone (20 nodes, 60 links, 380 demands), Biznet (29 nodes, 66 links, 812 demands), and Sanet (43 nodes, 90 links, 1806 demands). The link selection percentage μ is 5%, 10%, 15%, respectively, and the demand selection percentage ζ is 20%, 15%, 10%, respectively.

The training and evaluation environments of simulations are implemented using OpenAI Gym [44] and conducted on an Ubuntu 22.04 LTS machine equipped with an Intel Core i9-12900H CPU. The hyperparameter settings and traffic parameters settings are the same as those used in the testbed. Our implementation is based on the open-source Enero project [15] and extends it to support SRv6 deployment, SRH awareness, and available-path calculation. Our code is available [3].

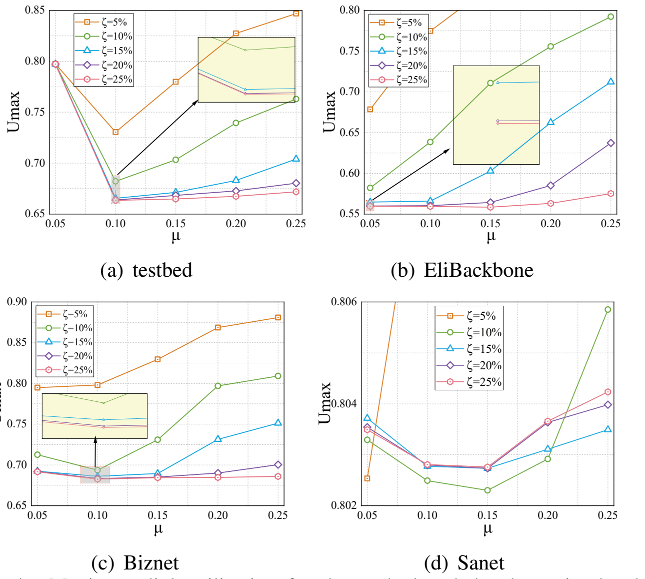
C. Ratio of Selected Links and Demands

To determine the most suitable percentage for link and traffic demand selection, we conduct experiments in the testbed and simulated topologies under varying selection ratios for congested links and critical traffic demands. Specifically, we vary the congested link selection ratio μ and the critical traffic demand selection ratio ζ from 5% to 25% [45]. Training is performed separately on the four topologies, with the SRv6 deployment ratio fixed at 1. Figure 4 presents the traffic optimization performance under different link and traffic demand selection parameters, where U_{max} represents the MLU in the network. For each topology, the trends of the curves corresponding to different ζ values are similar. Except for the largest network, Sanet, performance improves as ζ increases.

For Sanet, when $\zeta > 5\%$, the optimization results remain identical across different values of μ and ζ, with MLU stabilizing around 0.803. Given the large number of traffic demands

[3]https://github.com/lllsy00/DRL-TE

in Sanet, optimizing a smaller subset of demands proves sufficient. As the parameter μ increases, most curves initially rise and then decline. This is because, once a certain threshold is reached, the selected traffic demands are sufficient for the DRL to complete the routing optimization. Adding more demands beyond this threshold increases the solution space, reducing the exploration efficiency of DRL and ultimately degrading performance. To minimize the number of demands requiring optimization while maintaining DRL performance, both μ and ζ should be kept as low as possible. For example, in the case of our testbed, when μ is set to 10%, the curves reach their lowest point. The performance with ζ values of 15%, 20%, and 25% is very similar, so we select the lower ζ value of 15% to reduce the number of demands to be optimized. Consequently, the parameters (μ, ζ) are set as follows: (0.1, 0.15) for testbed, (0.05, 0.2) for EliBackbone, (0.1, 0.15) for Biznet, and (0.15, 0.1) for Sanet.



(a) testbed (b) EliBackbone (c) Biznet (d) Sanet

Fig. 4. Maximum link utilization for the testbed and the three simulated topologies under different link and traffic demand selection parameters.

We select the corresponding DRL agents trained under the most suitable link and traffic selection parameters. Since the DRL convergence curves are similar across topologies, we present the training process for the testbed as an example. Figure 5 shows the training process. As training progresses, the reward obtained by DRL increases steadily, while MLU decreases. This indicates that DRL performance improves over the first 120 episodes. After this point, MLU and U_{max} stabilize, and the training eventually converges.

D. SRv6 Deployment Method

First, we evaluate the SRv6 deployment method. We evaluate the performance of DRL-TE and compare it with other methods in the testbed and three simulated topologies. We choose comparison baselines that cover the main design choices involved in partial SRv6 deployment. 1) OSPF-MLL [6], [7] is the most widely used heuristic: it upgrades nodes according to their most loaded links (MLL) computed from an OSPF-initialised TM, and it typically serves as the baseline for cost-aware incremental deployment in today's networks.

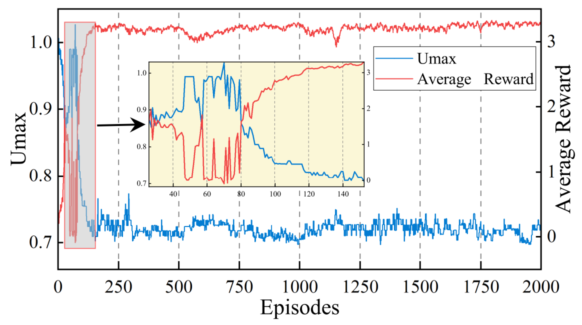
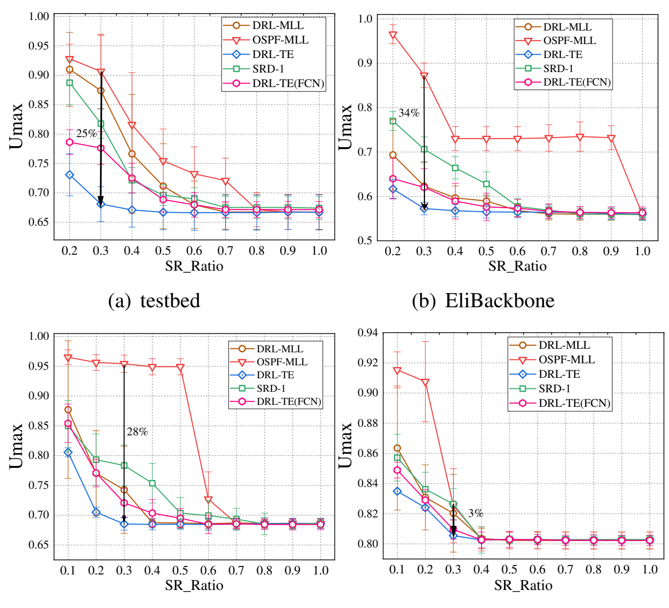


Fig. 5. The reward and U_{max} of the testbed during training process.

2) SRD [6], which is designed for SR-MPLS, is an LP-based solver that jointly decides deployment and routing but enforces contiguous upgrades; we keep it to contrast our flexible, SRv6-specific incremental strategy. 3) DRL-MLL [22] is included because it also uses DRL-assisted deployment, but its decisions are made on the overall traffic distribution and then upgrade nodes according to the MLL after DRL-based routing. In contrast, the deployment method of DRL-TE filters out non-SRv6 traffic and deploys only for SRv6 traffic, enabling finer-grained deployment. A comparison with DRL-MLL isolates the effect of filtering (or not filtering) non-SRv6 traffic on SRv6 deployment. Finally, 4) DRL-TE(FCN) replaces the MPNN with a fully connected neural network, providing an comparison that quantifies how much the graph-aware representation (MPNN) contributes compared with a topology-agnostic neural architecture. Deployment methods based on node degree and betweenness have been shown in the literature to be significantly less effective than OSPF-MLL due to their lack of consideration of traffic characteristics [6], [7], so they are not included in the comparison. The effect of SRv6 deployment is measured by the final optimized MLU, and the results are shown in Figure 6. Due to the limited number of nodes in the testbed and EliBackbone, the deployment ratio for these topologies starts from 0.2.

According to the evaluation results, DRL-TE can achieve TE performance comparable to full SRv6 deployment at lower deployment ratios, thereby effectively reducing deployment costs for network operators. In this case, their respective deployment ratios are all 0.3 for the four topologies, and DRL-TE reduces MLU by 3–34% relative to representative heuristics. This is because DRL-TE can upgrade nodes that schedule more traffic in routing optimization, while the OSPF-MLL and DRL-MLL methods base their deployment on the allocation of all traffic, including demands that do not use SRv6. As a result, these methods fail to accurately identify the nodes that are more critical to routing optimization. The DRL-MLL method performs better than OSPF-MLL because it relies on the MLL results after DRL allocation, which reflects the distribution of SRv6 traffic to some extent. SRD uses an ILP method, which requires SRv6 nodes to form a connected subgraph. Additionally, it overlooks the impact of the SRH, leading to suboptimal routing optimization results. These issues make it difficult to achieve good traffic optimization at lower deployment ratios. DRL-TE(FCN), which uses a traditional FCN instead of the MPNN proposed in this paper,



(a) testbed (b) EliBackbone

(c) Biznet (d) Sanet

Fig. 6. Maximum link utilization of different deployment methods for the testbed and the three simulated topologies. Each point in the figure represents the average value evaluated under 100 TMs at the corresponding SRv6 deployment ratio and the upper and lower limits showing the average value ± standard deviation.

struggles to capture the relationship between SRv6 deployment and TMs, resulting in poorer performance. Table IV details the nodes selected for SRv6 deployment by different methods at a deployment ratio of 0.3. It is notable that all methods consistently select nodes 0 and 4, suggesting their critical roles in routing optimization. However, DRL-TE uniquely selects node 10. Figure 7 shows the deployment of DRL-TE in the testbed, where each edge represents a bidirectional link.

TABLE IV
SRv6 DEPLOYMENT IN THE TESTBED AT 0.3 DEPLOYMENT RATIO

SRv6 deployment method	Node ID
DRL-TE	0, 4, 7, 10
DRL-TE(FCN)	0, 4, 5, 10
SRD-1	0, 2, 4, 6
DRL-MLL	0, 2, 4, 7
OSPF-MLL	0, 2, 4, 5

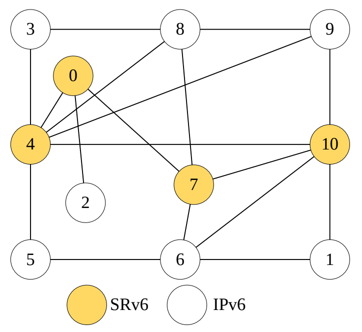
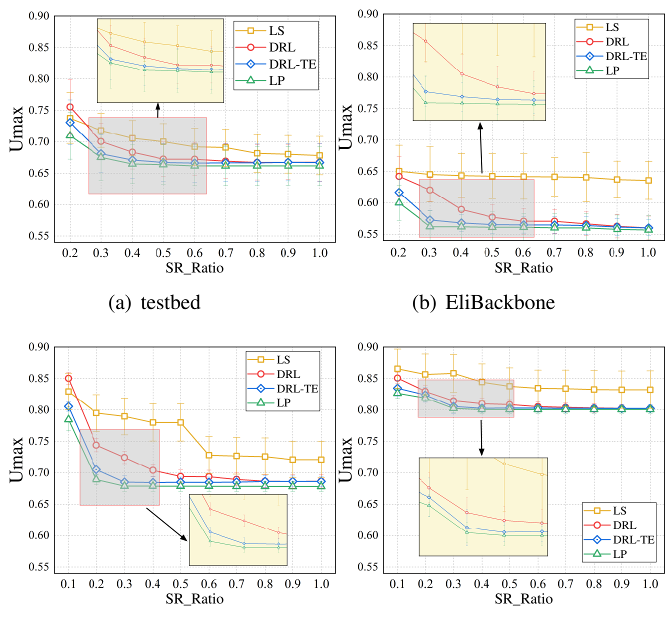


Fig. 7. SRv6 deployment of DRL-TE in the testbed at 0.3 deployment ratio.



(a) testbed (b) EliBackbone

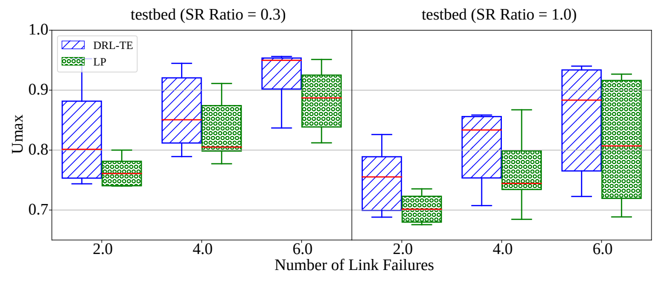
(c) Biznet (d) Sanet

Fig. 8. Maximum link utilization of different routing optimization methods for the testbed and the three simulated topologies. Each point in the figure represents the average value evaluated under 100 TMs at the corresponding SRv6 deployment ratio and the upper and lower limits showing the average value ± standard deviation.

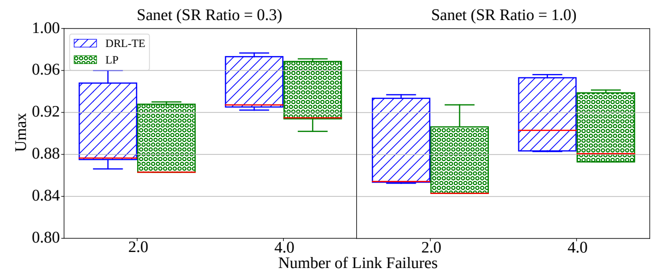
E. Traffic Routing Optimization Performance

Based on our deployment method, we evaluate various traffic routing optimization methods. DRL-TE first uses DRL to simulate traffic optimization in a network with the deployment ratio of 1, and then applies an LS method to adjust traffic routing at the actual deployment ratio. To verify the effectiveness of each phase of our algorithm, we compare it with several optimization algorithms, including 1) LS, 2) DRL, where the DRL agent performs routing optimization directly under the actual deployment ratio, eliminating the subsequent correction phase, while falling back to OSPF default routing when optimization of DRL agent is unattainable, and 3) LP, which uses the Gurobi solver [14] to solve the routing optimization problem, representing the optimal solution. We limit the solving time of LP to one hour to ensure efficiency. Figure 8 shows the traffic optimization results at different deployment ratios.

The results indicate that, without the initial traffic optimization by DRL, relying on LS alone performs reasonably well on the small testbed, but on larger simulated topologies, it more frequently gets trapped in local optima, yielding a noticeably higher MLU than LP. DRL also degrades when the deployment ratio at test time differs substantially from that used during training, reflecting limited generalization across markedly different SRv6 deployment ratios. Note that DRL optimizes only the identified critical demands (to shrink the search space and reduce online cost), which means a single DRL forward pass is challenging to approach the optimum; the subsequent LS phase is therefore essential to tighten the solution. The routing optimization of DRL-TE benefits from the correction phase, achieving optimization results that are close to the optimal routing performance. Overall, under the same SRv6 deployment ratios, DRL-TE achieves better routing optimization while avoiding the pronounced degradation observed for the individual components.



(a) testbed



(b) Sanet

Fig. 9. Comparison of DRL-TE and optimal LP performance under varying numbers of link failures for SRv6 deployment ratios of 0.3 and 1.0 in (a) testbed and (b) Sanet.

F. Robustness Analysis

We assess the robustness of DRL-TE by evaluating its performance under link failures and traffic demand fluctuations.

1) Link Failures: To assess robustness under link failures, we randomly remove k bidirectional links while preserving connectivity. For each $k \in \{2, 4, 6, 8\}$, we generate 10 independent failure scenarios. For every scenario, DRL-TE is evaluated on 50 TMs and multiple SRv6 deployment ratios. In the Sanet topology, removing $k \geq 6$ links leaves at most one feasible path for many SD pairs; therefore, we report only $k \in \{2, 4\}$. Figure 9 presents the results for deployment ratios 0.3 and 1.0 on the testbed and Sanet. Figure 9 presents the optimization performance of DRL-TE for deployment ratios of 0.3 and 1.0 on the testbed and Sanet. As illustrated, MLU increases steadily as the number of link failures rises. This increase is primarily attributed to the reduction of available bandwidth caused by link failures, which forces traffic to concentrate on a smaller set of remaining links.

Although link failures cause DRL-TE to perform slightly worse than the optimal solution, the performance gap remains small, with a maximum average difference of 8.6%. These results suggest that DRL-TE demonstrates certain robustness in the face of link failures. However, if even higher optimization performance under failure conditions is required, retraining the model specifically on failure topologies would be necessary.

Previous experiments demonstrated that an SRv6 deployment ratio of 0.3 can achieve near-optimal routing performance under no-failure conditions. However, Figure 9 shows that under link failures, both DRL-TE and the optimal solution achieve better performance at higher SRv6 deployment ratios compared to lower ones. Specifically, with full SRv6 deployment, DRL-TE can achieve up to a 9.2% average performance improvement compared to the case with a 0.3 deployment ratio. The reason is that, with fewer SRv6 devices, link failures involving SRv6 nodes substantially reduce available routing paths, impairing routing optimization capabilities. Conversely, a higher deployment ratio provides more redundancy and available routing paths, enhancing network resilience against link failures. We think that improving network robustness against link failures constitutes a critical driving factor for the incremental deployment of SRv6.

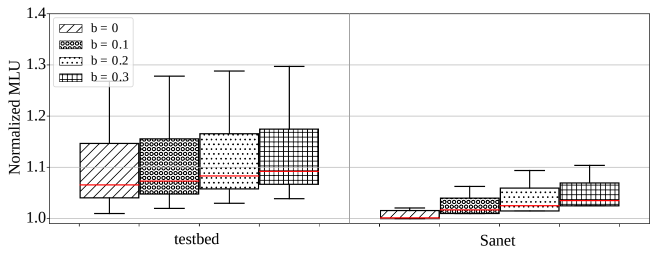


Fig. 10. Impact of traffic demand fluctuations on DRL-TE performance: normalized MLU under varying fluctuation levels for testbed and Sanet (SR Ratio = 0.3).

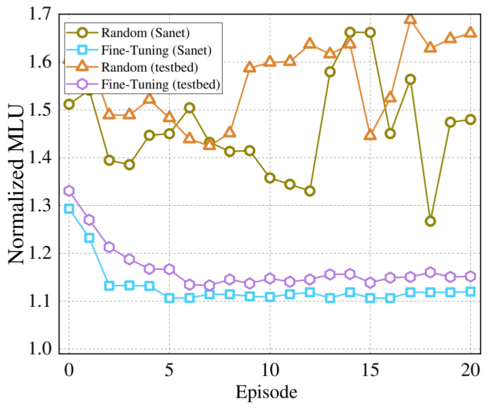


Fig. 11. Convergence performance of online training on testbed and Sanet: comparison between fine-tuning from a pre-trained model and training from scratch (random initialization), measured by normalized MLU over training episodes.

2) Traffic Demand Fluctuations: To evaluate the impact of traffic demand fluctuations on our algorithm, we perturbed the TMs by scaling each traffic demand's bandwidth by a random factor uniformly drawn from the interval $[1-b, 1+b]$, where $b \in \{0.1, 0.2, 0.3\}$. The results are shown in Figure 10. The SRv6 deployment ratio is set to 0.3, and the normalized MLU represents the ratio of DRL-TE's optimized result to the optimal value obtained by the LP solver. The results indicate that as the fluctuation level increases, DRL-TE performance degrades by only 0.9% to 2.7%.

As time goes by, traffic patterns gradually change. To enhance the performance of route optimization, DRL-TE needs to be retrained, and the training speed largely determines the practicality of DRL-TE. To evaluate the online training speed of DRL-TE, we retrained in the testbed and Sanet with an SRv6 deployment ratio of 0.3. We applied significant perturbations to the training TMs by setting $b = 0.5$. The training processes are shown in Figure 11. Fine-Tuning represents the rapid online training based on the previous model, while Random represents the random initialization of model parameters and training from scratch for comparison. It can be seen that our model has a fast online training speed and can converge in about 7 episodes. Compared to the case without retraining, the normalized MLU at convergence is reduced by 13% and 14%, respectively.

G. Algorithm Costs Analysis

Next, we assess the costs of DRL-TE. The experimental setup and data are the same as in Section V-E.

1) Reconfiguration Paths: When SRv6 is used to reconfigure demand paths, each change incurs a control-plane reconfiguration cost. We evaluate, over multiple TMs, the fraction of demands that are reconfigured and compare DRL-TE with the LP baseline; results for various deployment ratios are shown in Fig. 12. As the deployment ratio increases, the number of available paths for demands grows. However, since DRL-TE only optimizes a portion of the traffic, the increase in reconfigured traffic is limited, allowing DRL-TE to maintain a relatively low reconfiguration cost. In contrast, LP optimizes paths for all demands, leading to a higher reconfiguration cost. In short, DRL-TE reconfigures only a small subset of paths at each update, which limits control-plane overhead.

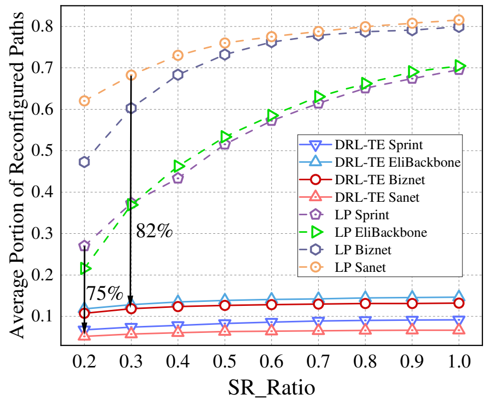


Fig. 12. Comparison of reconfiguration paths between DRL-TE and LP.

TABLE V
THE AVERAGE PATH STRETCH AFTER ROUTING OPTIMIZATION UNDER DIFFERENT SRV6 DEPLOYMENT RATIOS

Topology	SR Ratio=0.2	SR Ratio=0.6	SR Ratio=1.0
testbed	0.40%	0.50%	0.53%
EliBackbone	0.57%	0.68%	0.71%
Biznet	0.82%	0.94%	0.96%
Sanet	0.38%	0.47%	0.48%

2) Path Stretch: In addition, using SRv6 can result in traffic demands following longer paths, which may lead to excessive network bandwidth resource consumption. Table V summarizes the average path stretch, indicating how much longer paths become after routing optimization compared to original OSPF paths under different SRv6 deployment ratios. Notably, the path stretch values remain consistently low, even at higher deployment ratios, suggesting that DRL-TE efficiently avoids significantly lengthening paths. Therefore, we think that DRL-TE does not result in excessive unfairness and avoids rerouting traffic onto excessively long paths that would incur high latency.

TABLE VI
BANDWIDTH OVERHEAD INTRODUCED BY SRV6 HEADER UNDER VARIOUS DEPLOYMENT RATIOS

Topology	SR Ratio=0.2	SR Ratio=0.6	SR Ratio=1.0
testbed	0.40%	0.50%	0.53%
EliBackbone	0.57%	0.68%	0.71%
Biznet	0.82%	0.94%	0.96%
Sanet	0.38%	0.47%	0.48%

3) SRv6 Bandwidth Consumption: The SRv6 encapsulation header causes additional link bandwidth consumption. Table VI provides the average percentage of link bandwidth consumed specifically by the SRv6 encapsulation header under various deployment ratios. It is clear that although SRv6 headers introduce additional bandwidth overhead, this consumption remains modest even at higher deployment ratios. For example, at full SRv6 deployment, the consumed bandwidth remains below 1%. These results suggest the additional bandwidth overhead is acceptable compared to the benefits obtained from optimized routing flexibility.

TABLE VII
EXECUTION TIME COMPARISON OF ROUTING OPTIMIZATION METHODS

Topology	SR Ratio=0.2	SR Ratio=0.6	SR Ratio=1.0
testbed	0.22s	0.24s	0.26s
EliBackbone	0.29s	0.53s	0.62s
Biznet	0.58s	0.76s	0.78s
Sanet	1.04s	1.62s	1.80s

4) Computation Time: Finally, we evaluate the computation time required for DRL-TE to make routing optimization decisions while processing TMs. Table VII shows the average computation time at deployment ratios of 0.2, 0.6, and 1.0. The results show that although DRL-TE applies LS, it only optimizes a portion of the demands, which ensures that it does not consume excessive time. Additionally, increasing deployment ratios does not significantly impact the computation time for DRL-TE. Despite the larger scale of Sanet, DRL-TE can still complete the calculation in 2 seconds. With more powerful computers, the computation time would be significantly reduced.

H. Limitation

While the results above to some extent demonstrate the effectiveness of DRL-TE, several limitations of the present study should be acknowledged.

Firstly, bursty traffic occurs frequently in operational networks and is unpredictable and short-lived [28]. Our centralized workflow relies on prior prediction or timely inference of the TM; as a result, its reaction time may be inadequate for traffic bursts. In such cases, DRL-TE may not reconfigure routes quickly enough to fully mitigate transient congestion. We plan to investigate distributed, event-driven sensing and faster control loops that trigger local TE decisions on demand, thereby improving responsiveness to sudden traffic bursts.

Secondly, link failures are common in practice [46]. Selecting the lowest SRv6 deployment ratio under failure-free assumptions is therefore not fully realistic. Because different links exhibit heterogeneous failure probabilities [47], a more practical approach is to derive failure-aware, robust SRv6 deployments that explicitly account for per-link (or shared risk link group) risk, e.g., via chance-constrained objectives or scenario-based optimization. Exploring such robustness-oriented formulations is part of our future work.

VI. Conclusion

In this paper, we tackle two practical obstacles to SRv6 adoption: (i) how to plan partial upgrades under budget constraints, and (ii) how to perform routing optimization while explicitly accounting for SRH bandwidth overhead. We formulated these coupled decisions within a single framework and proposed DRL-TE algorithm, which couples a DRL policy with a lightweight LS refinement. The algorithm targets upgrades and routing on the most impactful nodes and demands, and uses a GNN to remain effective across different SRv6 deployment ratios. Across an 11-machine testbed and three simulated networks, DRL-TE achieves optimization performance comparable to full SRv6 networks: with only 30% of devices upgraded, it reduces MLU by 3–34% relative to representative heuristics. DRL-TE stays within 8.6% of the LP optimum under link failures and maintains stable performance across traffic variations. The algorithm typically reconfigures only a small fraction of demands per update. Taken together, these results indicate that DRL-TE offers a practical, cost-aware path to incremental SRv6 deployment while preserving strong TE efficiency in the scenarios we evaluated.

References

[1] B. Fortz, J. Rexford, and M. Thorup, "Traffic Engineering with Traditional IP Routing Protocols," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 118–124, 2002.

[2] L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," *RFC 8402*, 2018.

[3] A. Bashandy, C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing with the MPLS Data Plane," *RFC 8660*, 2019.

[4] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming," RFC 8986, Feb. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8986

[5] X. Yan, N. L. S. d. Fonseca, and Z. Zhu, "Self-Adaptive SRv6-INT-Driven System Adjustment in Runtime for Reliable Service Function Chaining," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 5, pp. 4962–4973, 2024.

[6] A. Cianfrani, M. Listanti, and M. Polverini, "Incremental Deployment of Segment Routing Into an ISP Network: a Traffic Engineering Perspective," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3146–3160, 2017.

[7] Y. Tian, Z. Wang, X. Yin, X. Shi, Y. Guo, H. Geng, and J. Yang, "Traffic Engineering in Partially Deployed Segment Routing Over IPv6 Network with Deep Reinforcement Learning," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1573–1586, 2020.

[8] L. Wang, L. Lu, M. Wang, Z. Li, H. Yang, S. Zhu, and Y. Zhang, "SNS: Smart Node Selection for Scalable Traffic Engineering in Segment Routing Networks," *IEEE Trans. Netw. Service Manag.*, pp. 1–1, 2024.

[9] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer, "IPv6 Segment Routing Header (SRH)," RFC 8754, Mar. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8754

[10] B. Ren, D. Guo, Y. Yuan, G. Tang, W. Wang, and X. Fu, "Optimal Deployment of SRv6 to Enable Network Interconnection Service," *IEEE/ACM Trans. Netw.*, vol. 30, no. 1, pp. 120–133, 2022.

[11] A. Brundiers, T. Sch, N. Aschenbruck *et al.*, "Fast Reoptimization with Only a Few Changes–Enhancing Tactical Traffic Engineering with Segment Routing Midpoint Optimization," *IEEE J. Sel. Areas Commun.*, 2025.

[12] Y. Liu, N. Geng, M. Xu, Y. Yang, E. Dong, C. Liu, Q. Gan, Q. Li, and J. Wu, "Adaptive and Low-cost Traffic Engineering: A Traffic Matrix Clustering Perspective," *IEEE J. Sel. Areas Commun.*, 2025.

[13] X. Li and K. L. Yeung, "Traffic Engineering in Segment Routing Networks Using MILP," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1941–1953, 2020.

[14] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022. [Online]. Available: https://www.gurobi.com

[15] P. Almasan, S. Xiao, X. Cheng, X. Shi, P. Barlet-Ros, and A. Cabellos-Aparicio, "ENERO: Efficient Real-time WAN Routing Optimization with Deep Reinforcement Learning," *Comput. Netw.*, vol. 214, p. 109166, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622002717

[16] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks," in *Proc. 2015 ACM Conf. Spec. Interest Group Data Commun.*, ser. SIGCOMM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 15–28. [Online]. Available: https://doi.org/10.1145/2785956.2787495

[17] J. Zhang and C. Zhao, "Q-SR: An Extensible Optimization Framework for Segment Routing," *Comput. Netw.*, vol. 200, p. 108517, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621004497

[18] 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*, 2023, pp. 378–393.

[19] X. Liu, S. Zhao, Y. Cui, and X. Wang, "FIGRET: Fine-Grained Robustness-Enhanced Traffic Engineering," in *Proc. ACM SIGCOMM*, 2024, pp. 117–135.

[20] A. Saleh, R. Morabito, S. Dustdar, S. Tarkoma, S. Pirttikangas, and L. Lovén, "Towards Message Brokers for Generative AI: Survey, Challenges, and Opportunities," *ACM Comput. Surv.*, 2024.

[21] L. Lovén, M. Bordallo López, R. Morabito, J. Sauvola, S. Tarkoma *et al.*, "Large Language Models in the 6G-Enabled Computing Continuum: a White Paper," 2025.

[22] S. Liu, H. Lu, Y. Chen, B. Chong, and T. Luo, "Partial SRv6 Deployment and Routing Optimization: A Deep Reinforcement Learning Approach," in *GLOBECOM 2023 - 2023 IEEE Glob. Commun. Conf.*, 2023, pp. 7133–7138.

[23] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "Path-Based Graph Neural Network for Robust and Resilient Routing in Distributed Traffic Engineering," *IEEE J. Sel. Areas Commun.*, 2025.

[24] D. S. E. Deering and B. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 8200, Jul. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8200

[25] J. Networks, *Junos OS: Technical Documentation*, Juniper Networks, 2024, available at: https://www.juniper.net/documentation/en_US/junos/.

[26] A. Brundiers, T. Schüller, and N. Aschenbruck, "Midpoint Optimization for Segment Routing," in *IEEE INFOCOM 2022 - IEEE Conf. Comput. Commun.*, 2022, pp. 1579–1588.

[27] A. Brundiers, T. Schüller, and N. Aschenbruck, "An Extended Look at Midpoint Optimization for Segment Routing," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 1447–1468, 2024.

[28] 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*, 2024, pp. 71–85.

[29] D. Wu and L. Cui, "A Comprehensive Survey on Segment Routing Traffic Engineering," *Digit. Commun. Netw.*, vol. 9, no. 4, pp. 990–1008, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864822000189

[30] W. Cheng, C. Filsfils, Z. Li, B. Decraene, and F. Clad, "RFC 9800 Compressed SRv6 Segment List Encoding," 2025.

[31] H. Masri, S. Krichen, and A. Guitouni, "A Multi-start Variable Neighborhood Search for Solving the Single Path Multicommodity Flow Problem," *Appl. Math. Comput.*, vol. 251, pp. 132–142, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300314014982

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[33] 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, 2020.

[34] R. W. Floyd, "Algorithm 97: Shortest Path," *Commun. ACM.*, vol. 5, no. 6, pp. 345–345, 1962.

[35] P. K. Donta, S. N. Srirama, T. Amgoth, and C. S. R. Annavarapu, "iCoCoA: Intelligent Congestion Control Algorithm for CoAP Using Deep Reinforcement Learning," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, no. 3, pp. 2951–2966, 2023.

[36] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," in *Int. Conf. Mach. Learn.* PMLR, 2017, pp. 1263–1272.

[37] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 65–76, Apr. 2009. [Online]. Available: https://doi.org/10.1145/1498765.1498785

[38] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, 2011.

[39] T. iPerf Development Team, "iPerf - The Ultimate Speed Test Tool for TCP, UDP and SCTP," https://iperf.fr/, accessed: 2024-10-17.

[40] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, and S. Guo, "A Survey of Intelligent Network Slicing Management for Industrial IoT: Integrated Approaches for Smart Transportation, Smart Energy, and Smart Factory," *IEEE Commun. Surv. Tutor.*, vol. 24, no. 2, pp. 1175–1211, 2022.

[41] F. Mason, G. Nencioni, and A. Zanella, "Using Distributed Reinforcement Learning for Resource Orchestration in a Network Slicing Scenario," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 88–102, 2023.

[42] L. Foundation, *Linux Traffic Control - tc*, https://man7.org/linux/man-pages/man8/tc.8.html, 2024, accessed: 2024-10-17.

[43] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "TensorFlow: Large-scale Machine Larning on Heterogeneous Systems," 2015.

[44] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[45] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624000525

[46] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of Failures in An Operational IP Backbone Network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, 2008.

[47] 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 SIGCOMM*, 2019, pp. 29–43.