IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 April 2025, accepted 7 May 2025, date of publication 12 May 2025, date of current version 21 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3569236

RESEARCH ARTICLE

New Approaches for Network Topology Optimization Using Deep Reinforcement Learning and Graph Neural Network

MOHAMMED ALI[1], (Member, IEEE), FLORENT DUCHESNE[1], GHASSAN DAHMAN[1,2], (Senior Member, IEEE), FRANÇOIS GAGNON[1], (Senior Member, IEEE), AND DIALA NABOULSI[1], (Member, IEEE)

[1]École de Technologie Supérieure, Montreal, QC H3C 1K3, Canada
[2]Ultra Communications and Intelligence Company, Mount Royal, QC H4T 1V7, Canada

Corresponding author: Mohammed Ali (Eng.m.a4@hotmail.com)

This work was supported by the Mitacs/Ultra Intelligence and Communications Project under Grant IT25839.

ABSTRACT The exponential growth in Internet-connected devices has escalated the demand for optimized network topologies to ensure high performance. Traditional optimization methods often fall short in scalability and adaptability when it comes to network topology planning. In this paper, we address the challenge of transforming mesh topologies into tree topologies for wireless networks, with the objective of maximizing throughput. We propose two new methods: Path Selection with Rejection Strategy (PSRS), which leverages Message-Passing Neural Networks (MPNN), and Dual-Agent Tree Topology Exploration (DATTE), which employs Graph Attention Networks (GAT). These schemes integrate Deep Reinforcement Learning (DRL) and Graph Neural Networks (GNNs) to construct efficient tree topologies with the goal of maximizing the minimum throughput of the wireless network. Experimental results validate the scalability and performance gains of the proposed approaches, highlighting their potential for real-world applications.

INDEX TERMS Deep reinforcement learning, graph neural networks, proximal policy optimization, tree topology, wireless network.

I. INTRODUCTION

The proliferation of communication technologies has led to an exponential increase in Internet-connected devices, resulting in a corresponding expansion of network scale and continuous upgrades to network infrastructure. Network topology optimization plays a pivotal role in ensuring a seamless user experience. Given that key network performance metrics such as link utilization, throughput, and latency are profoundly influenced by network structure, there is a heightened focus among network operators on the critical problem of topology optimization [1], [2], [3], [4], [5], [6].

The network planning problem presents formidable challenges. At its core, the topology problem is inherently combinatorial. Consequently, it exhibits high complexity, often growing exponentially with the number of links involved. When undertaking network topology optimization, technical constraints come into play. These constraints arise from the intricate management-specific requirements governing network topology. These requirements can be nonlinear and even nonconvex. Examples include stipulations related to the permissible fraction of altered links, overall modification costs, and post-optimization network performance metrics, such as link utilization and throughput. Researchers have proposed diverse models and approaches to tackle network topology planning scenarios. Some formulations treat the problem as a mixed integer linear programming challenge, while others delve into complex multi-objective optimization [7], [8], [9]. These optimization objectives often focus on minimizing costs and achieving multi-layer recovery. Existing topology optimization works employ various algorithms, such as minimum spanning tree [10], centralized connection [11], and approximation algorithms [12]. Additionally, heuristic methods [13], [14], [15] concentrate on explicit objective functions

The associate editor coordinating the review of this manuscript and approving it for publication was Hosam El-Ocla.

within limited connections. However, these algorithms lack guarantees of close-to-optimal performance and are of high computational complexity.

Deep reinforcement learning (DRL) has emerged as a promising approach for tackling the complex problem of network topology optimization [16]. By leveraging the power of neural networks and reinforcement learning algorithms, DRL offers a versatile framework for dynamically adjusting network configurations to meet performance objectives. In the context of network topology optimization, DRL algorithms learn to make decisions about which links to add, remove, or modify based on feedback from the network's performance metrics. Using DRL for network topology optimization offers adaptability to changing conditions, unlike traditional methods reliant on fixed models. DRL algorithms learn and improve decision-making by interacting with the network. Additionally, DRL can grasp intricate network relationships, enabling more comprehensive optimization strategies that traditional methods may overlook.

Graph Neural Networks (GNNs) offer advanced capabilities for network topology optimization by extending neural networks to graph-structured data. GNNs iteratively update node representations through message-passing phases, enabling the capture of local and global network patterns. This allows GNNs to effectively model relationships between network components, crucial for topology optimization [17], [18]. In practice, networks are represented as graphs with nodes and edges corresponding to devices and links. GNNs are trained to optimize performance metrics like link utilization and throughput, guided by a loss function to refine network configurations [19]. GNNs' primary advantage lies in their adaptability to various network scenarios, ensuring robust performance across different topologies and scales [20]. They also support real-time optimization, accommodating dynamic network changes [21].

The goal of network topology optimization is to find the optimal link structure between nodes to fulfill a given objective, such as getting the optimal throughput of a graph topology. Tree network topology is a mixture of star topology and bus topology. Tree topology allows for the expansion of an existing network and provides scalability, i.e., more and more users can be attached to one node with secondary nodes. Tree network topology is in play over a large global area [22]. Finding the optimal path for data transmission from the source node to the destination node is an important task. In this paper, the focus of the proposed approaches is to construct a graph tree for data transmission to maximize the throughput of wireless networks using DRL. The construction of a balanced tree involves a proper selection of the links to overcome the challenges of data transmission.

In this paper, we introduce two novel approaches that leverage the synergistic power of DRL and GNN for network topology optimization. The primary contribution lies in the development of two distinct schemes: the Path Selection with Rejection Strategy (PSRS) and the Dual-Agent Tree Topology Exploration (DATTE). The key contributions include:

1) Proposing a GNN-driven DRL agent framework for constructing tree topologies from mesh topologies, aimed at maximizing minimum throughput.
2) PSRS, which integrates proximal policy optimization (PPO) based DRL with Message-passing Neural Networks (MPNN) to select optimal paths, ensuring tree topology with high throughput. The rejection strategy ensures that actions leading to non-tree topologies are avoided, maintaining structural integrity and optimizing throughput.
3) DATTE, which employs dual agents using Graph Attention Network (GAT) to iteratively refine tree topologies by adding and removing links, enhancing overall network performance. By embedding prescribed constraints directly into the algorithm, we preclude the possibility of the agents converging to erroneous states, ensuring consistent tree topologies throughout the optimization process.

The remaining part of this paper is organized as follows. Section II covers the study of existing work related to tree graph construction approaches in wireless networks. Section III presents the problem statement and the general Architecture of the DRL and GNN schemes. The proposed approaches are presented in Sections IV and V. Section VI presents the simulation results and comparison of different performance metrics of the proposed approaches compared to the previous works. Finally, the work is concluded in Section VII.

II. RELATED WORK

DRL algorithms have surfaced as a means to discern the underlying patterns between inputs and optimal solutions [16]. Most DRL-based network optimization solutions adopt a centralized approach, where the network status (topology and link loads) and throughput matrix are assumed to be known. One example of a centralized DRL solution is ENERO [23], which uses a DRL agent powered by a GNN to compute network paths on top of the initial shortest paths. A Local Search algorithm is then used to further improve the DRL solution. Another DRL solution is CRF-RL [24], which aims to optimize the maximum link utilization (MLU) while minimizing the degradation of Quality of Service (QoS). This solution uses a two-step approach, where a DRL agent identifies critical flows to be re-routed in the first phase and a Linear Programming based approach is used to assess alternative paths for each critical flow in the second phase.

In another approach [25], a DRL agent is trained to optimize the Network Utility by computing splitting ratios for a set of communication flows. The agent determines the percentage of flow to be sent along each pre-computed path based on the throughput and delay of each flow. In addition, a notable challenge faced by these DRL-based solutions is their difficulty in effectively performing in new and unseen network scenarios. DATE approach [26] involves coordinated offline training of DRL agents that are installed in network edge nodes. These agents are able to compute ingress-egress

paths online based on network link loads. However, DATE has limitations in that it does not allow for fast reaction to internal link anomalies. Additionally, it requires synchronization among network nodes for the exchange of link load values, resulting in wasted time.

Recently, Graph Neural Networks (GNNs) have gained considerable attention for their ability to handle graph-structured data, capturing complex relationships efficiently. They have been successfully applied across diverse domains, ranging from multimodal sentiment analysis and emotion recognition in conversational graphs [27], to educational applications like course recommendation systems [28] and multimodal cross-modal retrieval [29]. Additionally, recent advances in hierarchical and quantum kernel-based approaches demonstrate GNN's power in effectively classifying graph-structured data [30]. In network optimization, GNNs integrated with DRL have shown promising results in capturing intricate dependencies within network topologies and significantly enhancing routing strategies [31].

The author in [31] proposes a novel approach for network optimization by integrating GNN [32], [33] into DRL agents. The architecture focuses on solving routing optimization in optical networks and demonstrates impressive generalization capabilities over never-seen arbitrary topologies. Inspired by MPNN [34], the GNN captures crucial information about the relations between links and traffic in the network topologies. However, to generate an environment for large networks, taking the shortest paths using depth-first search [35] can result in very long runtimes and it can reach O(n!) in the complete graph of order n nodes. On the other hand, paths can have very similar features and can lead to selecting insufficient actions for the GNN model leading to a smaller reward. In this paper, we present two novel and comprehensive approaches that harness the strengths of both DRL and GNN technologies, aiming to construct a tree topology network achieving near-optimal performance and maximum throughput. Our proposed method addresses the trade-off between real-time performance and convergence speed, ensuring near-instantaneous and efficient topology optimization even in large-scale networks with complex topologies. The incorporation of GNNs, as a key technology enabling DRL to handle network changes, allows for exceptional generalization capabilities over diverse network configurations.

Recent studies highlight the growing importance of neural architecture optimization in solving complex network problems. The Single-Domain Generalized Predictor, recently introduced in [36], offers efficient strategies for neural architecture research. Additionally, innovative techniques have been proposed for network topology optimization using DRL as discussed in [37], underscoring the evolving interplay between neural network architectures and optimization challenges.

In this paper, we delve into the potential applications of an enhanced agent utilizing GNNs coupled with DRL techniques. Our paper explores the extraction of a tree topology graph from a complex mesh graph to achieve optimal capacity utilization. Establishing an optimized tree structure enhances network resilience, minimizes latency, and streamlines data flow, ultimately leading to improved overall network performance and robustness. We will provide a comprehensive explanation of the underlying principles and information aggregation mechanisms of the GNN architectures and DRL techniques used in each approach. Moreover, we will illustrate how these architectures handle graph-structured data and the advantages they offer in different scenarios. By providing detailed insights into each GNN and DRL technique, we aim to present a comprehensive understanding of their strengths and how they can complement each other when combined. This analysis will lay the foundation for our subsequent exploration of their synergy in real-world applications and empirical evaluations. To validate the effectiveness of our proposed approaches, we conduct extensive simulations on different network topologies and the results showcase significant advancements in network performance, scalability, and adaptability, making our approaches promising solutions for real-time network topology optimization.

III. PROBLEM STATEMENT AND GENERAL METHODOLOGIES

A. PROBLEM STATEMENT

The rapid advancement of communication technologies has significantly increased the number of Internet-connected devices, leading to the expansion of network scales and continuous upgrades to network infrastructure. This growth necessitates effective network topology optimization to maintain high network performance metrics such as link utilization, throughput, and latency. However, the inherent complexity of the network planning problem, which is combinatorial in nature, presents substantial challenges, especially as it involves high computational complexity and technical constraints like non-linear and non-convex management-specific requirements.

Taking a set of nodes that can communicate with each other, the objective is to determine the network topology in the shape of a tree. In this topology, each node must be able to reach all other nodes in the graph, and each node should be directly connected to only one other node. Traditional methods for network topology optimization, such as mixed integer linear programming and multi-objective optimization approaches, often fall short in terms of computational efficiency and near-optimal performance guarantees. Recently, DRL has emerged as a potential solution, offering dynamic adaptability and improved decision-making by learning from network interactions. Additionally, GNN has shown promise in handling graph-structured data, capturing intricate relationships between network components to optimize performance metrics.

Despite these advancements, existing DRL and GNN-based methods face limitations in real-time performance, generalization to unseen network scenarios, and the ability to maintain optimal network configurations dynamically.

These shortcomings highlight the need for more robust and efficient approaches to network topology optimization.

This paper addresses these challenges by introducing two novel approaches that leverage the combined strengths of DRL and GNNs: PSRS and DATTE. These approaches aim to construct tree topologies from mesh topologies to maximize throughput and ensure structural integrity. The proposed methods incorporate advanced mechanisms such as PPO-based DRL, MPNN, and multi-head attention in GAT to optimize network performance while adhering to predefined constraints.

B. TREE TOPOLOGY NETWORK CONSTRUCTION

In this paper, we present two novel approaches that harness the combination power of two distinct machine learning techniques, DRL and GNN. The inputs to each approach are the special distribution of the nodes to be connected as well as the parameters of their inter-links. The target of the approaches is to get the best tree topology that gives the maximum data streams (*minTH*). The establishment of a well-structured tree topology is equally crucial for network optimization, as seen in wireless communication systems. A tree topology facilitates efficient data dissemination, where nodes are organized hierarchically, minimizing the number of hops required for data transmission and enhancing the network scalability [38].

In Figure 1, we outline the DRL agent's process to construct the tree topology network. The agent interacts with an environment by observing its state, selecting actions from GNN based on the current state, receiving feedback in the form of rewards or penalties, and updating its strategy based on this feedback to achieve its goals. The agent aims to learn the best actions to take over time by balancing exploration and exploitation.

Taking the general framework of our approaches, the basic differences between the two proposed schemes are as follows:

1) The first scheme environment implementation depends on the selection of a path between a source and destination nodes as an action to construct the tree topology network, while the second scheme's environment is based on the idea of deleting and adding edges as an action for the same purpose.
2) The implementation of the GNN architecture in the first scheme consists of the MPNN model, while the second scheme uses GAT [39] to select the best link to delete from the mesh topology input.

In Sections IV and V, we will explore each scheme by introducing the implementation of their RL environment first, the detailed architecture of their GNN model, and finally, explain how the agent/agents of each scheme interact between the environment and GNN model.

IV. SCHEME 1: PATH SELECTION WITH REJECTION STRATEGY (PSRS)

In this study, we explore network optimization, building upon the groundwork established in [27]. Our attention is directed towards a GNN-driven DRL agent tasked with intricately

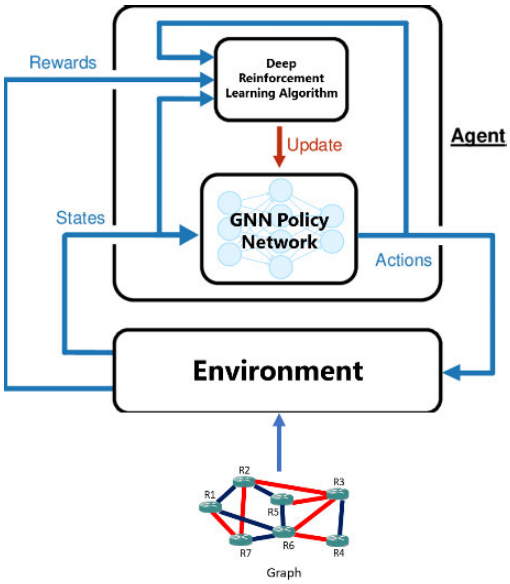


FIGURE 1. General Architecture of DRL and GNN Scheme. The red links in the graph represent the final selected links to construct the tree topology network.

orchestrating the construction of a tree topology from a mesh topology, aiming for optimal minimum throughput.

The agent's pivotal role involves making judicious decisions for incoming source-destination (src-*dst*) paths, taking into account the specific features of the links encompassing different potential paths. Each path selected for a src-*dst* pair must adhere to the imperative of generating a tree topology.

Within this section, we present the PSRS framework that comprises two key components, namely the GNN-based DRL agent and the optimization environment. The GNN-based DRL agent takes on the role of orchestrating actions within the network topology, specifically focusing on selecting potential paths that provide optimal minimum throughput. Our agent employs the PPO algorithm [40] to guide its actions and leverage a GNN model. Conversely, the optimization environment establishes the problem to be addressed. This environment houses critical information such as the network's topology and associated link features. Moreover, it is tasked with generating rewards following action execution, offering feedback to the agent regarding the efficacy of its choices. The following subsection illustrates the details of the DRL environment, GNN architecture, and the agent procedure for the construction of an optimal tree topology.

A. PSRS ENVIRONMENT

Within this section, we outline the contextual environment essential to the PSRS scheme. The environment in our scenario is structured for an input graph, where each path, defined as the links between a source node and a distention node, within the graph, represents a distinct state. The state applied to the GNN to get the action. The environment plays a crucial role in the agent's learning process by offering feedback based on the actions it takes. This feedback is typically

provided in the form of rewards for positive outcomes or penalties for unfavorable ones. The overarching objective of the agent within this environment is to develop a policy that maximizes the cumulative rewards obtained over time, ultimately guiding it from an initial starting state to a designated goal state.

The observation space is the graph links features of each path which embraces five integral features: link throughput, link path loss, link betweenness centrality, and the first and the second neighbor links. The action space is the selection of a path among K paths for each *src-dst* pair of nodes (or at each step) until generating a tree topology graph. Ensuring action space equivalence across various network topologies is essential for effective generalization. To establish paths between a given source and destination pair within the network graph, we employ the random paths algorithm (Algorithm 1). The method starts by generating a random spanning tree using a modified Kruskal's algorithm and stores the paths with the minimum throughput, and the capacity information for all *src-dst* pair links. The algorithm will iterate over each *src-dst* path for *IT* iterations and taking into account the paths with high minimum throughput. This offers increased flexibility and scalability, enabling a more comprehensive exploration of potential paths. The higher value of K ensures that the action space remains sufficiently diverse, especially in larger and more intricate networks comprising 50 nodes or more with high connectivity. Additionally, we find that the increased value of K does not result in a disproportionately high computational cost due to our strategic path update mechanism, which efficiently refines paths after each episode. The proposed algorithm for generating paths for *src-dst* pairs offers notable advantages over the traditional shortest path method [27] in terms of speed, efficiency, and scalability. This algorithm leads to a faster convergence to optimal or near-optimal solutions, as the algorithm explores a broader range of potential paths within the network. Consequently, the proposed algorithm can significantly reduce the computational time required to find suitable paths compared to the traditional shortest path method, especially in large-scale networks with complex topologies.

Our approach introduces a new network state representation s that encompasses two critical link-level attributes: the demand links (L_d) and the reserved links (L_r). L_d indicates the current demanded links that represent all the paths for a given *src-dst*, while L_r assigns a value of 1 to the links with respect to the selected path, chosen by the action, among all the paths for the given *src-dst* that will contribute to generating the tree topology. Thus, the state s at time step t can be represented as:

$$s_t = (L_d, L_r)_t \tag{1}$$

These attributes, in addition to the features of the observation space, enhance our ability to identify and prioritize key links that play a crucial role in generating the desired tree topology. By incorporating these attributes, our method captures the importance of each link more comprehensively, leading to better-informed path selections. Fig. 2 and Table 1 provide an overview of the attributes and features present in the link's hidden states. For instance, in Fig. 2, when a path request is made from node 4 to node 6, it is assigned to the path comprising nodes {4, 1, 6} due to adherence to the tree topology constraint.

The action at time step t is the selection of a path p from K possible paths for a given -*dst* pair:

$$a_t = p \in \{p_1, p_2, \ldots, p_K\}_{(\text{src}-dst)_t} \tag{2}$$

where $\{p_1, p_2, \ldots, p_K\}_{\text{src}-dst}$ is the K paths for a given *src-dst* based on the random path technique presented in Algorithm 1. The step function executes a step in the environment given a_t. It updates s_t based on the selected action to s_{t+1}. When an action a (path p_i) is selected, the state s=(L_d, L_r) transitions to a new state $s_{t+1} = s'$=(L'_d, L'_r), where L'_r is updated to include the links of the selected path p_i while L'_d reset to be ready for the next *src-dst*.

Algorithm 1 Random Path

Input: Data: graph, k, current_state_cap,Paths, Paths_Cap
Output: Result: *Paths'*, *Paths_Cap'*
if $i == 0$ **then**
  Paths' = {};
  Paths_Cap' = {};
else
  Paths' ← *Paths*;
  Paths_Cap' ← *Paths_Cap*;
for *j* ***in*** 0, . . . , 100 **do**
  R.G.T ← *generate_trec*(*graph*, *k*);
  for *src*, *dst* ***in*** *R_G_T.nodes*() **do**
    current_path = R.G.T (src,dst);
    cap = current_state_cap(src,dst);
    if *len*(*Paths*) < *k* & *current_path* ***not in*** *Paths* **then**
      Paths' {*src*, *d st*} ← *current_path*;
      Paths_Cap' {*src*, *d st*} ← *cap*;
    else
      if *cp* > *smallestcapacitypathinPaths_Cap* **then**
        Indx = Index of the smallest capacity in *Paths_Cap*;
        Paths' {*src*, *dst*}[*Indx*] ← *current_path* :
        Paths_Cap' {*src*, *dst*}[*Indx*] ← *cap*;

Formally, if p_i contains the links $\{l_1, l_2, \ldots, l_m\}$:

$$L'_r = L_r \cup \{l_1, l_2, \ldots, l_m\} \tag{3}$$

$$L'_d = ResetDemands((scr\_dst)_{t+1}) \tag{4}$$

The reward r for the agent is determined based on the connectivity and topology of the graph:

1) If the graph G_T is not connected and has no cycle connection, the reward is set to 0.5 to encourage exploration.
2) If the G_T forms a tree, the reward is computed based on *minTH* and the total data steams throughput (*totTH*)

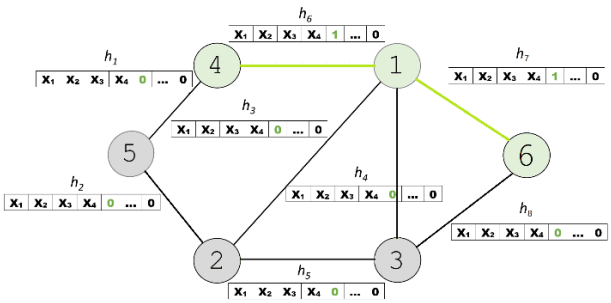


FIGURE 2. Representation in the link hidden states for tree topology network optimization.

TABLE 1. Input feature of the link hidden states. N corresponds to the size of the hidden state vector.

Notation	Description
x_1	Link capacity (Throughput)
x_2	Link path loss
x_3	Link betweenness
x_4	Link contributes to the tree topology (Reserved Link)
x_5	Action vector (Demand Link)
x_6 to x_N	Zero padding

of G_T, aiming to maximize both variables to achieve an optimal or near-optimal solution.

3) If the graph is connected but does not form a tree topology, the reward is set to 0, indicating a suboptimal or invalid state.

Mathematically, the reward r at a time step t can be represented as:

$$r_t = \begin{cases} 0.5, & \text{if } a_t \text{ contributes to an incomplete } G_T \\ 1 + minTH + (\alpha \cdot totTH), & \\ \quad \text{if } a_t \text{ completes a tree } G_T & \\ 0, & \text{if } a_t \text{ form a cycle in } G_T \end{cases} \tag{5}$$

where α is a constant less than 1. The episode will be over once it reaches either the second or the third option in r and a new episode is starting. The reset method resets the environment for a new episode. It returns the initial state along with randomly selected source and destination nodes to encourage exploration and learning across various starting points.

In the context of our proposed approach, the action reject strategy can be particularly advantageous for ensuring adherence to the tree topology extraction constraint. As our method aims to extract a tree topology network while maximizing its *minTH* and *totTH* values, rejecting certain actions can help the agent maintain the structural integrity of the tree throughout its decision-making process. In situations where a selected action might lead to a violation of the tree topology requirement, the agent can choose to reject that action and explore alternative options. This rejection mechanism acts as a safeguard against inadvertent violations and ensures that the resulting topology adheres to the desired tree structure.

B. PSRS GNN ARCHITECTURE

The GNN model is constructed as an MPNN model. In our context, emphasis is placed on the linked entity, and the message-passing process unfolds among all links. Opting for link entities, as opposed to node entities, stems from the fact that link features define the tree topology optimization problem. Algorithm 2 outlines the formal procedure of the message-passing process, taking links' features (x_l) as input and generating a q-value (q) as output.

The algorithm executes T message passing steps, as depicted in Fig. 3, iterating through all links in the network topology. For each link, its features are merged with those of neighboring links using a fully connected mechanism, denoted as M in Fig. 3. The outcomes of these operations are referred to as messages in GNN notation. Subsequently, the messages computed for each link with their neighbors are aggregated via an element-wise sum (line 5 in Algorithm 2). Following this, a Recurrent Neural Network (RNN) is employed to update the link hidden states (h_{LK}) with the newly aggregated information (line 6 in Algorithm 2). Upon completing the message passing phase, the resulting link states are aggregated using an element-wise sum (line 7 in Algorithm 2).

Algorithm 2 Message Passing

Input: x_f
Output: $\mathbf{h}_l^T$, q
1: **for each** $l \in \mathcal{L}$ **do**
2: $\quad h_l^0 \leftarrow [\mathbf{x}_l, 0 \ldots, 0]$
3: **for** $t = 1$ to T **do**
4: $\quad$ **for each** $l \in \mathcal{L}$ **do**
5: $\quad\quad M_l^{t+1} = \sum_{i \in N(l)} m\left(h_l^t, h_i^t\right)$
6: $\quad\quad h_l^{t+1} = u\left(h_l^t, M_l^{t+1}\right)$
7: $rdt \leftarrow \sum_{\text{lec}} h_i$
8: $q \leftarrow R(ndt)$

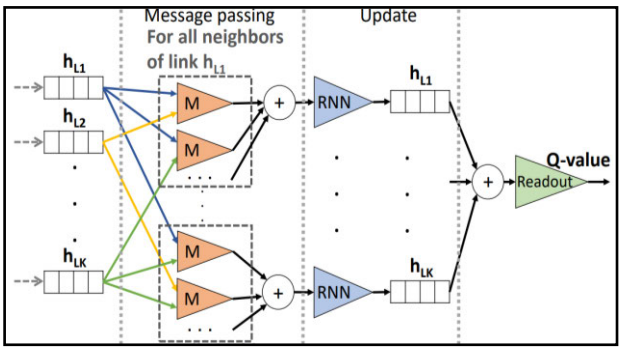


FIGURE 3. Message passing architecture [27].

The consolidated result undergoes processing through a fully connected Deep Neural Network (DNN), which models the readout function of the GNN. The output of this function is the estimated q-value corresponding to the input state and action. RNN's role is to learn how the link states evolve

during the message-passing phase. As link information permeates through the graph, each hidden state accumulates information from increasingly distant links, introducing the concept of time. RNNs, tailored for capturing sequential behavior in domains like text, video, and time-series data, are well-suited for this purpose. Additionally, certain RNN architectures, such as GRU, are designed to handle large sequences, mitigating issues like vanishing gradients, which is a common challenge with extensive sequences. These characteristics make RNNs suitable for learning how the links' states evolve during the message-passing phase, even for large values of T.

C. PSRS AGENT OPERATION FOR TREE TOPOLOGY CONSTRUCTION

On encountering a *src-dst* path, the agent creates a state representation that combines the current network state and the new path. This input guides its GNN model to determine a construction decision. The chosen action is converted into forwarding rules and implemented in network nodes. During training, the agent explores diverse path strategies, earning rewards tied to path contributions in constructing the tree topology.

PPO collects trajectories of actions and rewards from the environment, estimates advantages for actions, and updates the policy iteratively [40]. PPO uses a surrogate function to guide policy updates while preventing overly large changes through a clipping mechanism. It maintains a trust region to ensure controlled policy updates and can include entropy regularization for exploration. PPO focuses on improving the policy using a combination of optimization techniques and constraints to prevent drastic updates. The policy (π_θ) represents the probability of taking action a given state s and is parameterized by θ:

$$\pi_\theta(a|s) \tag{6}$$

In PPO, the policy is updated to maximize the expected r while ensuring that the updates do not change the policy too much, to maintain stability. The advantage Function $A(s, a)$ estimates the relative value of an action compared to the average action in a given state:

$$A(s, a) = Q(s, a) - V(s) \tag{7}$$

where $Q(s, a)$ is the action-value function, representing the expected return of taking action a in state s, and $V(s)$ is the value function, representing the expected return of being in state s. PPO uses a clipped surrogate objective function to limit the change in policy between updates. This prevents large updates that can destabilize training. The objective function $L(\theta)$ can be written as:

$$L(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right] \tag{8}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio, ϵ is a hyperparameter that controls the clipping range, and $\hat{A}_t$ is the estimated advantage at time step t.

In response to PPO, the GNN constructs a graph representation, with the links of the topology manifesting as the nodes of the graph. This representation initializes link hidden states by incorporating input link-level features and the path action under evaluation. Subsequently, a message-passing mechanism iterates amongst the hidden states of linked nodes, aligning with the graph's structure. The outcomes of this iterative process, namely the novel link hidden states, aggregate into a unified global hidden state that encapsulates essential topology insights. This collective hidden state is then channeled through RNN for further processing.

The DRL agent engages with the environment through a sequential process. The environment is initialized with link features. Simultaneously, the environment generates paths encapsulated as the tuple {*src*, *dst*}, and an initial environment state, denoted as Cumulative reward (s) is initialized to zero. DRL agent's operation focuses solely on the task of constructing a tree topology within the given mesh network. The DRL agent interacts with the environment to iteratively apply a series of actions aimed at achieving a tree topology graph. The primary objective is to form a tree topology with the highest minimum throughput and total graph throughput. The operational procedure is outlined as follows:

1) DRL agent persists in finding a proper path for each random *src-dst* until either the successful construction of a tree topology that complies with the throughput constraints and tree topology criteria or until it becomes apparent that a solution is unattainable given the current state and *src-dst*.
2) The reward calculation is designed to incentivize the agent to effectively construct the tree topology as shown in (5).
3) An Episode is considered completed if the agent has effectively established a tree topology graph. However, if the process encounters a violation of the tree graph criteria, the agent will apply an action-rejecting mechanism.

In the context of our proposed approach, the action-reject strategy can be particularly advantageous for ensuring adherence to tree topology construction. The action-reject strategy can provide a mechanism to address situations where certain configurations lead to undesirable outcomes. By incorporating action rejection, the RL agent can make more informed decisions by considering a wider range of possibilities, rejecting actions that are unlikely to lead to desirable results, and focusing on those with higher potential for optimization. The operation of the action rejection strategy within our proposed approach involves the following steps:

1) Action Evaluation: The RL agent evaluates a set of candidate paths using the established criteria based on the current state of the construction of the tree topology. These criteria include factors such as link throughput, link betweenness, and the links allocated for the tree topology graph.
2) Action Rejection Decision: A rejection mechanism is defined as representing a binary value for the quality

or potential of an action. Paths that do not help in constructing a tree topology graph based on the current state are deemed less favorable and are considered for rejection.

3) Path Update and Learning: If an action is rejected, the corresponding path is discarded from the set of K provided paths for the given *src-dst*, and the remaining paths will go through GNN to select a new action, and the agent updates its policy based on the outcomes of the selected actions. In the event that all provided *src-dst* paths violate the tree topology constraint, the present *src-dst* combination is dismissed. Subsequently, a new *src-dst* pair is assigned, while the state remains unaltered prior to the previous *src-dst* selection.

V. SCHEME 2: DUAL-AGENT TREE TOPOLOGY EXPLORATION (DATTE)

In this scheme, our aim is to embed the prescribed constraints directly into the algorithm itself, thus pre-empting any possibility for the agents to converge to an erroneous state, such as a non-tree topology. Employing a two-agent system, each agent complements the other by selectively activating and deactivating links within the initial topology, thereby seeking a configuration that maximizes the minimum throughput. Beginning with a tree topology and maintaining a consistent count of activated links while employing action masking, we ensure that, at every iteration, the topology remains a tree.

Herein, we introduce the DATTE framework, comprising three primary components: a creation agent, a deletion agent, and a shared environment facilitating their learning process.

A. COMMON ENVIRONMENT

Both agents share a common environment, encompassing their reward and reset functions, as well as states. In addition, both agents trained using the PPO algorithm [40] Their collective objective entails intelligently exploring tree topologies, endeavoring to iteratively refine the topology from its initial state. During evaluation, the agents collaborate within a shared environment, alternating their interactions. Episodes are defined as individual attempts to enhance the topology from its starting configuration, with agents allotted a fixed number of steps to enact changes. If, during a step, an improvement in minimum throughput is achieved, agents receive a positive reward contingent upon both the minimum and total throughput of the resulting tree topology graph (formatted as $minTH + (\alpha \cdot totTH)$. Conversely, if no improvement is observed within a step, agents typically receive a reward of zero. Moreover, upon exhausting the allocated steps without enhancing the topology, agents receive a reward of -1, denoting a failure to explore in the correct direction.

Notably, this framework diverges from traditional multi-agent reinforcement learning (MARL) paradigms [41]. Initially, the first agent undergoes solo training, with randomness simulating the actions of the second agent. Subsequently, the second agent is trained offline to cooperate with the first agent.

In this scheme, a state is defined as a connectivity graph *G(N, L)*, where N represents a set of n nodes and L represents the set of links that can be activated, along with G', a subgraph of G that connects all the nodes with *n-1* links L'. Implicitly, G' takes the form of a tree, and cannot contain any cycle. The agents' goal is to modify L' by removing and adding links to it until they find a solution that maximizes the minimum throughput.

B. ACTIVATING AGENT ENVIRONMENT

The primary objective of the activating agent is to augment the topology by adding links. Given an almost-tree topology as input, characterized by a single missing link for achieving tree structure, the agent selects from a pool of candidate links to produce a tree topology as output. At each step, a set of K links is proposed randomly to the agent, drawn from those capable of facilitating a tree topology. Utilizing action masking and padding, the agent is prevented from selecting invalid actions in scenarios where fewer than K eligible links are available. The selected link is ultimately added to L.

Observations provided to the agent comprise global statistics derived from potential link proposals, encompassing throughput, path loss, graph node count, and node degrees. The action space is structured as a set of K potential links, from which the agent selects one.

Algorithm 3 demonstrates the procedure mentioned in this section to generate the observation and rewards. For the neural network architecture, a basic multi-layer perceptron suffices, serving the purpose of classifying individual links through feature extraction.

C. DEACTIVATING AGENT ENVIRONMENT

Contrary to the activating agent, the goal of the deactivating agent is to selectively remove links from the topology. Given a tree topology as input, the agent aims to produce an output identical to the input, albeit with one link deactivated. At each step, the agent surveys the entire topology and deactivates a single active link i.e., removes a link from L'.

The observation space for this agent is more intricate, necessitating the employment of a GNN to classify links. Algorithm 4 illustrates the steps to extract the observation and the rewards using the deleting agent. Consequently, all features of both links and nodes within the current topology are required. The action space is denoted as $(N-1,)$, where N represents the number of nodes in the graph, with each possibility corresponding to a single link within the tree topology.

D. GNN ARCHITECTURE

The GNN utilized by the deleting agent comprises three GATv2Conv [42] layers followed by fully connected layers. The general framework of the GAT is shown in Fig. 4. It employs attention mechanisms to selectively weigh the

Algorithm 3 "Step" Function of the Activating Agent

Input: Input: Action
Output: Output: Obecrvations, Reward
G.addLink(action); /* Activate the selected link */
min *TH*, *tot TH* ← *evaluateTopology* (*G*); /∗ Calculate performance metrics */
reward ← 0; /* Calculate the reward */
if *minTH* > *previousBestMinTH* **then**
  reward ← min *TH* + tot *TH* /1000
  previousBestMinTH ← minTH
end
G.deleteRandomLink(); /* Randomly deactive one link from the
topology */
possibleLinks = []
for *each link li* ∈ *A* **do**
  G' ← *G.c opy*()
  G'*addLink*(*li*)
end
if G^r *isTree()* **then**
  possibleLinks.addl(li)
end
linkProposals = *random.choices*(*liensPossibles*, *K*)
observations ← *createObservations*(*linkProposals*); /* Create the observations used for the next step */
Return *reward*, *observations*

Algorithm 4 "Step" Function of the Deactivating Agent

Input: Action
Output: Observations, Reward
G.remoneLink ($\mathbf{a}_1$); /* Renove the selected link fron the topology */
creator.addLink(*G*); /* Use the proviously trained agent to add a link to the topology */
minTH, *totTH* ← *evaluateTopology* (*G*); /* Calculate performance metrica */
reward ← 0; /* Calculate the reward */
if *minTH* > *previousBestMinTH* **then**
  reward ← min *TH* + tot *TH* /1000 *previousBestMinTH* & *minTH*
end
observations ← *createObservations* (G^r) ; /* Create the observation used for the next step */
Return *reward*, *observations*

importance of neighboring nodes, enabling more flexible feature extraction. Given that GNN convolutions primarily extract features for nodes, the input data is structured such that each node embeds features of edges from the topology, while edges embed features of nodes. This arrangement enables the GNN to extract edge features from the topology, facilitating the selection of the link to be deleted based on the extracted feature vector. Table 2 shows the architecture of the proposed GNN for the deleting agent.

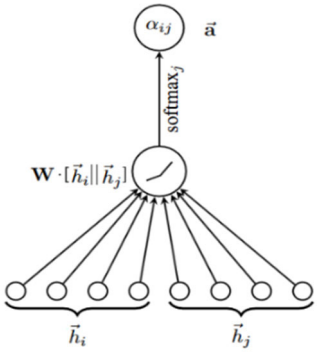


FIGURE 4. GATv2 convolution framework.

TABLE 2. GNN architecture of the Actor agent.

Layer	Input Shape	Output Shape	#Param
GNN_Extractor		[1]	3480
├─(conv1)GATv2Conv	[29, 3], [2, 78], [78, 2]	[29, 6]	72
├─(conv2)GATv2Conv	[29, 6], [2, 78], [78, 2]	[29, 12]	216
├─(conv3)GATv2Conv	[29, 12], [2, 78], [78, 2]	[29, 12]	360
├─(lin1)Linear	[29, 12]	[29, 16]	208
├─(lin2)Linear	[29, 16]	[29, 32]	512
├─(lin3)Linear	[29, 32]	[29, 64]	2,048
├─(value)Linear	[29, 64]	[29, 1]	64
├─(action)Argmax	[29, 1]	[1]	--

The GNN used in DATTE simply takes the implementation of GATv2Conv included in the Torch Geometric [40] library. In GATv2 networks, a weight matrix **W** is first applied to every pair of node's concatenation followed by a LeakyReLU activation. Then, a self-attention mechanism (a) is used to evaluate the importance of one node's features relative to other connected nodes. Finally, these attention coefficients are used to compute the next features of every node.

Graph Attention Networks can also be used in a "multi-heads" fashion, as multiple weight matrixes and self-attention mechanisms will be computed independently on the same data, and then concatenated or aggregated at the end. In our architecture, the first two layers of convolution have two heads while the third has only one.

VI. EXPERIMENTAL RESULTS

In this section, we present the evaluation of our proposed schemes designed for tree topology optimization for different networks as outlined in Sections IV and V. The primary focus of these experiments is to assess the performance and generalization capabilities of the two proposed schemes.

A. TRAINING SETUP

Our schemes, described in Sections IV and V, were implemented using Pytorch [43] and Torch Geometric [44] and subjected to evaluate the tree topology optimization, based on the OpenAI Gym framework [45]. For each simulated case, we assume having N nodes randomly distributed following a Poisson point process with an average node density of 1 node per 10 km^2. Then, we assume that b randomly selected links are fully blocked. For the remaining unblocked links,

we assume a simple path loss model, where a link's path loss is calculated as the summation of the free space path loss and a random excess loss representing the loss due to shadow fading and terrain diffraction, etc. Consequently, the throughput of each link is calculated based on the Shannon capacity formula, where we assume a bandwidth of 20 MB and an antenna gain of 8 dBi. In all cases, we assume symmetric links and no interference (due to careful frequency planning). In terms of traffic patterns, we assume saturated traffic, where each node is sending traffic to all other nodes. This assumption results in a total of $N \cdot (N-1)$ data streams. Consequently, each resulting tree topology is evaluated based on *minTH*. The best topology is the one that maximizes *minTH*. Different random wireless networks, including their adjacency matrix, path loss, and link throughput, are utilized for the evaluation. The achievement of a successful tree topology signifies the attainment of nearly optimal minimum throughputs of the tree topology graph.

Initial experiments were conducted to choose a suitable gradient-based optimization algorithm and fine-tune hyperparameters for both schemes. In the PSRS environment, link betweenness centrality is generated using the Brandes algorithm [46], which iteratively assesses individual links' contributions to shortest paths between node pairs, capitalizing on dependencies within the graph. In the PSRS GNN model, we defined link hidden states *hl* as 20-dimensional vectors populated with features from Table 1 for tree topology optimization, respectively. The size of these hidden states is linked to their potential information encoding. For PSRS GNN forward propagation, we performed $T = 4$ message passing steps using batches of 32 samples. As for DATTE, single passes were performed and batches of 64 samples were used. The chosen optimizer was Adam [47] with a learning rate of 10^{-4} and weight decay of 10^{-5} for both approaches. PSRS applied L2 regularization with a coefficient of 0.1 to the readout function. α was set to 0.01 for both algorithms. The critic discount factor γ was set to 0.8, while the entropy beta was set to 10^{-2}. Entropy beta corresponds to the strength of the entropy regularization, which makes the policy more random. This ensures that discrete action space agents are properly explored during training. The clipping value ε was set to 0.2 and it corresponds to the acceptable threshold of divergence between the old and new policies during gradient descent updating.

B. EVALUATION PERFORMANCE FOR THE TREE TOPOLOGY OPTIMIZATION

Our schemes were evaluated for tree topology optimization in simulated wireless networks with 10 and 30 nodes with different path loss conditions. For training, both schemes' agents were employed on a 30-node mesh topology, with b set to 0.5. The optimization of wireless network topologies is a crucial task to enhance network efficiency, coverage, and overall performance. In this context, we conducted a comprehensive analysis of our two proposed schemes' capabilities in attaining near-optimal solutions while minimizing execution time.

We composed our performance into two experiments in terms of optimal minimum throughput and computational time, as demonstrated in Fig. 5 and Fig. 6. In Fig. 5, the proposed schemes were evaluated on 100 graphs of 10 nodes, and $b = 0.5$ for 60 and 600 seconds and compared with the brute-force solution in terms of cumulative distribution function (cdf) curve. We selected the durations of 60 and 600 seconds purely for analytical purposes to observe the impact of increased computational time on the performance of the two algorithms. These timeframes were chosen to provide a clear comparison and to understand how extending the computation time influences the results. This approach allows us to assess the scalability and efficiency of the proposed schemes in a controlled manner.

The comparative analysis of DATTE and PSRS across different timeframes reveals their respective strengths and weaknesses. To the best of the authors' knowledge, no existing works that address our exact problem statement (i.e., maximizing the minimum throughput in large-scale tree-topology wireless networks) are available for direct comparison with our proposed methods. Therefore, we evaluate the performance of our proposed methods, PSRS and DATTE, by comparing them with the brute-force approach, when possible, resulting in optimal solutions. In Fig. 5, the results show that DATTE achieves near-optimal *minTH* comparable to the brute-force method, with the same median value of 1.25 Mbps for both 60 and 600 seconds. This demonstrates DATTE's efficiency in producing high-quality solutions rapidly. In contrast, PSRS underperforms relative to DATTE when given only 60 seconds, showing a median difference of 0.16 Mbps from the brute-force result. However, the performance dynamics shift with an extended execution time. When allowed 600 seconds, PSRS surpasses DATTE, delivering solutions closer to the brute-force *minTH*. This suggests that while DATTE is advantageous for quick, near-optimal results, PSRS excels when more time is available, leveraging its computational resources to produce superior outcomes.

The effectiveness of relational GNN within DATTE plays a critical role in identifying and leveraging the structural distinctions of the tree-like topology, enhancing decision-making for optimal tree formation. Additionally, the reinforcement learning component enables agents to refine their strategies through iterative experiences, improving performance over time. Thus, DATTE is preferable for scenarios requiring rapid results, whereas PSRS is more suitable for situations where extended computation can be afforded to achieve higher precision.

Fig. 6 demonstrates the result of each proposed scheme to reach the near-optimal solution in 60 and 600 seconds for each of the 10,000 input mesh graphs of 30 nodes and three different probabilities of full blockage (0.2, 0.5, and 0.8). This outcome is essential in real-world deployment scenarios where timely decisions are crucial to maintain network

performance. The analysis of both algorithms reveals distinct advantages depending on the available computational time.

The computational complexity of performing brute-force optimization increases exponentially with the number of nodes in the network due to the rapid growth in the number of possible topologies. The number of distinct connectivity graphs, $G(N)$, and tree topologies, $T(N)$, for a fully connected graph with N nodes can be estimated using the following formulas:

$$G(N) = \prod_{i=0}^{N-1} 2^i \tag{9}$$

$$T(N) = N^{N-2} \tag{10}$$

The exponential growth of $G(N)$ is evident from the combinatorial nature of generating all possible topologies, while $T(N)$ follows Cayley's formula [48] for the enumeration of labeled trees. Table 3 summarizes these values for networks with up to 12 nodes.

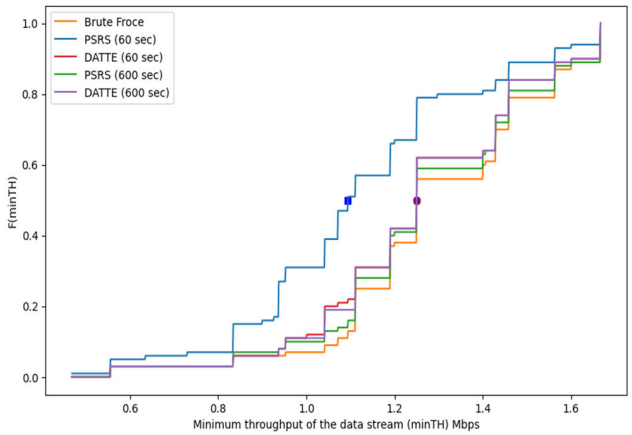


FIGURE 5. CDF comparison of the minimum throughput data stream between the two approaches against the brute-force solution for 100 graphs of 10 nodes for b = 0.5.

Moreover, the time required for brute-force optimization varies significantly with the level of network connectivity. Table 4 highlights the computational time for different network sizes and levels of connectivity (blockage). This emphasizes that the required computational time escalates rapidly with both the number of nodes and the level of connectivity blockage, making brute-force methods impractical for large-scale networks such as those with 30 nodes at 50% connectivity. Consequently, our proposed PSRS and DATTE methodologies offer a viable alternative by efficiently navigating the vast solution space to achieve near-optimal tree topologies without the prohibitive computational costs associated with exhaustive search techniques.

In Fig. 6, the DATTE algorithm demonstrates a significant speed advantage, generating near-optimal solutions more quickly. This is evident as DATTE consistently outperforms PSRS in shorter timeframes, such as 60 seconds, with minimal differences in reaching the optimal solution. Specifically, DATTE achieves better performance with median *minTH* differences of 0.024 Mbps, 0.041 Mbps, and 0.04 Mbps

TABLE 3. Number of possible tree topologies of a fully connected graph.

No. Nodes (N)	G(N) Number of topologies	T(N) Number of tree topologies
1	1	1
2	2	1
3	8	3
4	64	16
5	1,024	125
6	32,768	1,296
7	2,097,152	16,807
8	268,435,456	262,144
9	68,719,476,736	4,782,969
10	35,184,372,088,832	100,000,000
11	2^{55}	11^9
12	2^{66}	12^{10}

TABLE 4. The required time for brute-force optimization.

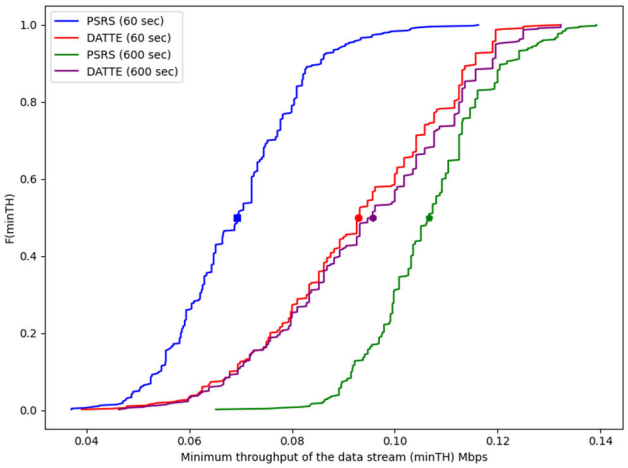
No. Nodes (N)	50% Blockage	20% Blockage	0% blockage
7	< 1 s	3 s	19 s
8	2 s	1 min 10 s	7 min 30 s
9	20 s	19 min	3 h
10	4 min	10 h	-
11	50 min	-	-

for blockage probabilities of 0.2, 0.5, and 0.8, respectively. However, when given more time resources, PSRS shows its strength and surpasses DATTE. With an extended execution time of 600 seconds, PSRS attains closer performance to the brute-force solution, exhibiting median differences of 0.011 Mbps, 0.004 Mbps, and 0.003 Mbps for the same blockage probabilities. This indicates that while DATTE is more efficient for quicker, near-optimal solutions, PSRS benefits significantly from additional computational time, ultimately providing superior results with extended execution.

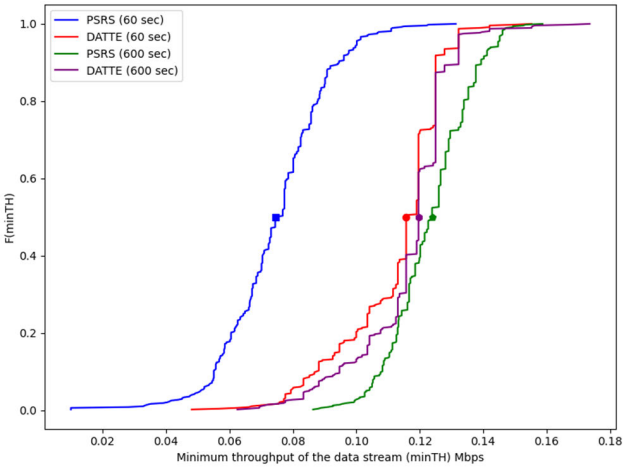
To further validate the scalability of our proposed methodologies, we extrapolate our experimental findings from networks comprising 10 and 30 nodes to larger-scale networks with up to 200 nodes. Note that the time complexity for generating all possible spanning trees in a graph with N nodes and different connectivity is known to grow exponentially with respect to the number of edges in the graph. Specifically, the time complexity is estimated as

$$O\left(2^{\left(c\frac{N(N-1)}{2}\right)}\right) \tag{11}$$

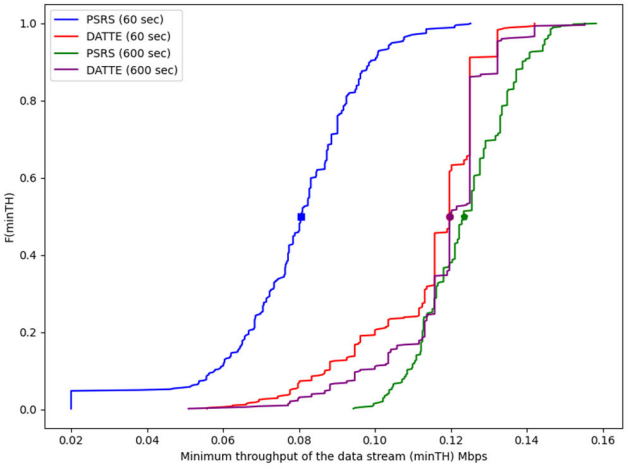
where c is the edge connectivity percentage of the graph. This leads to an exponential increase in computational time as the number of nodes increases. Figure 7 shows the relationship between the number of nodes and the time complexity for generating tree topologies at varying connectivity levels for PSRS and DATTE. PSRS and DATTE maintain high-performance efficiency as the network size increases and the



(a)



(b)



(c)

FIGURE 6. CDF of minimum throughput over 10,000 graphs of 30 nodes for b = 0.2 (a), 0.5 (b), and 0.8 (c) in 60 and 600 seconds.

computational complexity of both methods grows at a manageable rate, ensuring their applicability to larger and more complex network topologies. This scalability is attributed to the inherent design of PSRS and DATTE, which leverage

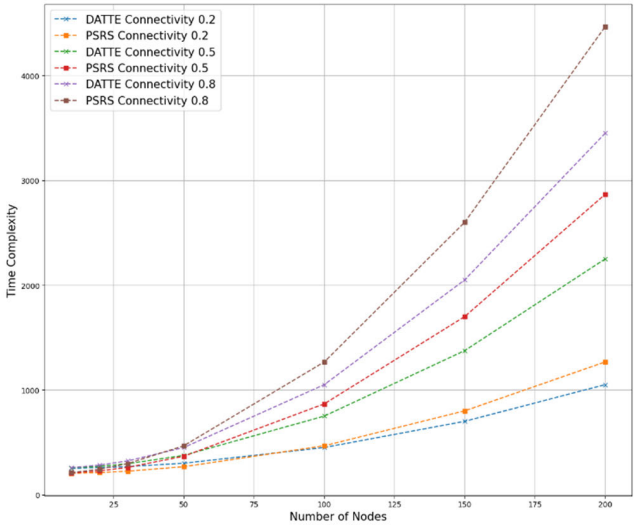


FIGURE 7. The time complexity for generating tree topologies with varying connectivity using PSRS and DATTE.

DRL and GNN to efficiently navigate and optimize the extensive solution space without exhaustive enumeration.

In practical deployment scenarios, particularly on resource-constrained edge devices, analyzing memory usage is essential. The memory footprint of PSRS and DATTE primarily stems from neural network parameters and intermediate activation states. PSRS, which utilizes Message-Passing Neural Networks (MPNN), demonstrates memory requirements that scale linearly with the number of nodes and hidden state dimensionality. In contrast, DATTE, based on

Graph Attention Networks (GAT), incurs greater memory consumption due to multi-head attention layers and per-edge computations. As shown in Table 5, DATTE consistently consumes more memory than PSRS, with the gap increasing from 23.3 MB at 10 nodes to over 60 MB at 50 nodes. This behavior reflects the additional memory overhead of storing attention coefficients and intermediate transformations. While DATTE yields faster convergence, its elevated memory demand may limit applicability in memory-constrained systems.

TABLE 5. Memory usage for PSRS and DATTE algorithms.

Network size (Nodes)	PSRS memory usage (MB)	DATTE memory usage (MB)
10	35.44	58.74
30	120.15	186.54
50	258.1	329.33
100	478.69	650.8

VII. CONCLUSION

This study delves into the realm of network optimization, expanding upon the groundwork laid in previous research. Focusing on a GNN-driven DRL agent, we endeavor to meticulously compose the transition from a mesh topology

to an optimal tree topology while maximizing the minimum throughput.

The proposed PSRS framework, comprised of a GNN-based DRL agent and an optimization environment, guided the agent's actions by using the PPO algorithm, leveraging the power of GNN models to navigate the network topology effectively. The optimization environment provides critical feedback to the agent, driving its learning process and shaping its decision-making capabilities. In parallel, the proposed DATTE framework was introduced, where two agents collaboratively manipulate the initial topology to maximize the minimum throughput. Employing a strategic combination of link activation and deactivation, the agents iteratively refine the topology toward the desired tree structure.

Through comprehensive experimentation and evaluation, we demonstrate the efficacy and efficiency of our proposed schemes. Our results showcase the remarkable potential of GNN-driven DRL agents in achieving near-optimal solutions for tree topology optimization, even in large and complex network environments. Furthermore, our schemes exhibit impressive efficiency in execution time, essential for real-world deployment scenarios where timely decisions are paramount.

Future studies could explore the impact of various neural network architectures on the effectiveness and reliability of network topology optimization algorithms. Such investigations could identify optimal architecture specifically tailored for different practical deployment scenarios.

REFERENCES

[1] X. Chen, B. Li, R. Proietti, H. Lu, Z. Zhu, and S. J. B. Yoo, "DeepRMSA: A deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks," *J. Lightw. Technol.*, vol. 37, no. 16, pp. 4155–4163, Aug. 15, 2019, doi: 10.1109/JLT.2019.2923615.

[2] G. Bernárdez, J. Suárez-Varela, A. López, B. Wu, S. Xiao, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Is machine learning ready for traffic engineering optimization?" 2021, *arXiv:2109.01445*.

[3] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 246–251, doi: 10.1109/INFCOMW.2019.8845132.

[4] N. Di Cicco, E. F. Mercan, O. Karandin, O. Ayoub, S. Troia, F. Musumeci, and M. Tornatore, "On deep reinforcement learning for static routing and wavelength assignment," *IEEE J. Sel. Topics Quantum Electron.*, vol. 28, no. 4, pp. 1–12, Jul. 2022, doi: 10.1109/JSTQE.2022.3151323.

[5] C. Natalino, C. Manso, L. Gifre, R. Muñoz, R. Vilalta, M. Furdek, and P. Monti, "Microservice-based unsupervised anomaly detection loop for optical networks," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Diego, CA, USA, Mar. 2022, pp. 1–3.

[6] D. Aureli, A. Cianfrani, M. Listanti, and M. Polverini, "Intelligent link load control in a segment routing network via deep reinforcement learning," in *Proc. 25th Conf. Innov. Clouds, Internet Netw. (ICIN)*, Paris, France, Mar. 2022, pp. 32–39, doi: 10.1109/icin53892.2022.9758091.

[7] O. Gerstel, C. Filsfils, T. Telkamp, M. Gunkel, M. Horneffer, V. Lopez, and A. Mayoral, "Multi-layer capacity planning for IP-optical networks," *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 44–51, Jan. 2014.

[8] O. Bondarenko, D. Ageyev, and O. Mohammed, "Optimization model for 5G network planning," in *Proc. IEEE 15th Int. Conf. Exper. Designing Appl. CAD Syst. (CADSM)*, Feb. 2019, pp. 1–4.

[9] B. B. Haile, E. Mutafungwa, and J. Hämäläinen, "A data-driven multi-objective optimization framework for hyperdense 5G network planning," *IEEE Access*, vol. 8, pp. 169423–169443, 2020.

[10] N. Li, J. C. Hou, and L. Sha, "Design and analysis of an MST-based topology control algorithm," *IEEE Trans. Wireless Commun.*, vol. 4, no. 3, pp. 1195–1206, May 2005.

[11] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *Proc. IEEE INFOCOM Conf. Comput. Commun. 19th Annu. Joint Conf. IEEE Comput. Commun. Societies*, vol. 2, Mar. 2000, pp. 404–413, doi: 10.1109/INFCOM.2000.832213.

[12] D. P. Williamson and D. B. Shmoys, *The Design approximation Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[13] A. H. Halim and I. Ismail, "Combinatorial optimization: Comparison of heuristic algorithms in travelling salesman problem," *Arch. Comput. Methods Eng.*, vol. 26, no. 2, pp. 367–380, Apr. 2019.

[14] A. Rezoug, M. Bader-El-Den, and D. Boughaci, "Guided genetic algorithm for the multidimensional knapsack problem," *Memetic Comput.*, vol. 10, no. 1, pp. 29–42, Mar. 2018.

[15] D. Santos, A. de Sousa, F. Alvelos, M. Dzida, and M. Pióro, "Optimization of link load balancing in multiple spanning tree routing networks," *Telecommun. Syst.*, vol. 48, nos. 1–2, pp. 109–124, Oct. 2011.

[16] P. Almasan, J. Suárez-Varela, B. Wu, S. Xiao, P. Barlet-Ros, and A. Cabellos-Aparicio, "Towards real-time routing optimization with deep reinforcement learning: Open challenges," in *Proc. IEEE 22nd Int. Conf. High Perform. Switching Routing (HPSR)*, Jun. 2021, pp. 1–6, doi: 10.1109/HPSR52026.2021.9481864.

[17] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020, doi: 10.1016/j.aiopen.2021.01.001.

[18] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.

[19] S. He, S. Xiong, Y. Ou, J. Zhang, J. Wang, Y. Huang, and Y. Zhang, "An overview on the application of graph neural networks in wireless networks," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 2547–2565, 2021, doi: 10.1109/OJCOMS.2021.3128637.

[20] H. Cai, V. W. Zheng, and K. C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[22] M. Gerla and L. Fratta, "Tree structured fiber optics MANs," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 6, pp. 934–943, Jul. 1988.

[23] 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, Sep. 2022, Art. no. 109166, doi: 10.1016/j.comnet.2022.109166.

[24] 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, doi: 10.1109/JSAC.2020.3000371.

[25] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," 2018, *arXiv:1801.05757*.

[26] N. Geng, M. Xu, Y. Yang, C. Liu, J. Yang, Q. Li, and S. Zhang, "Distributed and adaptive traffic engineering with deep reinforcement learning," in *Proc. IEEE/ACM 29th Int. Symp. Quality Service (IWQOS)*, Tokyo, Japan, Jun. 2021, pp. 1–10, doi: 10.1109/IWQOS52092.2021.9521303.

[27] M. Li, J. Shi, L. Bai, C. Huang, Y. Jiang, K. Lu, S. Wang, and E. R. Hancock, "FrameERC: Framelet transform based multimodal graph neural networks for emotion recognition in conversation," *Pattern Recognit.*, vol. 161, May 2025, Art. no. 111340.

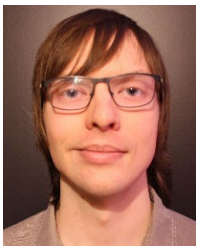
[28] M. Li, Z. Li, C. Huang, Y. Jiang, and X. Wu, "EduGraph: Learning path-based hypergraph neural networks for MOOC course recommendation," *IEEE Trans. Big Data*, vol. 10, no. 6, pp. 706–719, Dec. 2024.

[29] L. Bai, L. Cui, Y. Wang, M. Li, J. Li, P. S. Yu, and E. R. Hancock, "HAQJSK: hierarchical-aligned quantum Jensen–Shannon kernels for graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6370–6384, Nov. 2024.

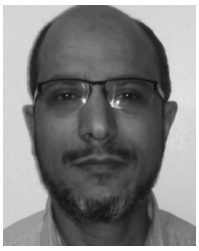
[30] M. Li, S. Zhou, Y. Chen, C. Huang, and Y. Jiang, "EduCross: Dual adversarial bipartite hypergraph learning for cross-modal retrieval in multimodal educational slides," *Inf. Fusion*, vol. 109, Sep. 2024, Art. no. 102428.

[31] 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, Dec. 2022, doi: 10.1016/j.comcom.2022.09.029.
[32] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009, doi: 10.1109/TNN.2008.2005605.
[33] P. W. Battaglia, "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*.
[34] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," 2017, *arXiv:1704.01212*.
[35] R. Sedgewick, *Algorithms C, Part 5: Graph Algorithms*, 3rd ed., Reading, MA, USA: Addison-Wesley, 2001.
[36] L. Ma, H. Kang, G. Yu, Q. Li, and Q. He, "Single-domain generalized predictor for neural architecture search system," *IEEE Trans. Comput.*, vol. 73, no. 5, pp. 1400–1413, Feb. 2024, doi: 10.1109/TC.2024.3365949.
[37] Z. Li, X. Wang, L. Pan, L. Zhu, Z. Wang, J. Feng, C. Deng, and L. Huang, "Network topology optimization via deep reinforcement learning," *IEEE Trans. Commun.*, vol. 71, no. 5, pp. 2847–2859, May 2023, doi: 10.1109/TCOMM.2023.3244239.
[38] Z. Zhao, G. Verma, C. Rao, A. Swami, and S. Segarra, "Link scheduling using graph neural networks," *IEEE Trans. Wireless Commun.*, vol. 22, no. 6, pp. 3997–4012, Jun. 2023, doi: 10.1109/TWC.2022.3222781.
[39] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[41] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," 2019, *arXiv:1911.10635*.
[42] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" 2021, *arXiv:2105.14491*.
[43] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," 2019, *arXiv:1912.01703*.
[44] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," 2019, *arXiv:1903.02428*.
[45] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*.
[46] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Netw.*, vol. 30, no. 2, pp. 136–145, May 2008, doi: 10.1016/j.socnet.2007.11.001.
[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[48] L. Takács, "On Cayley's formula for counting forests," *J. Combinat. Theory A*, vol. 53, no. 2, pp. 321–323, Mar. 1990, doi: 10.1016/0097-3165(90)90064-4.

MOHAMMED ALI (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Concordia University, Montreal, QC, Canada, in 2024. He has authored in the area of CNC, power amplifiers, 5G transceivers, signal and image processing, fingerprint recognition, wireless networks using CNN, and graph neural networks; and holds a patent. His research interests include machine and deep learning, signal processing, biometrics, computer vision, and communication systems.



FLORENT DUCHESNE received the bachelor's degree in software engineering from the École de Technologie Supérieure (ÉTS), University of Québec, Canada, in 2023. He is currently pursuing the M.A.Sc. degree with ÉTS. His research interests include wireless communications, machine learning, and computer vision.



GHASSAN DAHMAN (Senior Member, IEEE) received the Ph.D. degree from Carleton University, Ottawa, ON, Canada, in 2010. From 2010 to 2012, he was an Assistant Professor with Umm Al-Qura University, Makkah, Saudi Arabia. He held research positions with Lund University, Lund, Sweden, from 2012 to 2016; and with the NSERC-Ultra Electronics TCS Industrial Chair, École de Technologie Supérieure (ÉTS), Montreal, QC, Canada, from 2016 to 2018. He is currently with ULTRA-TCS, Montreal. His main research interests include radio propagation and channel modeling, including massive multiple-input–multiple-output (MIMO) systems, distributed antenna systems, overwater point-to-point communications, and anomalous propagation of microwaves in the troposphere.

FRANÇOIS GAGNON (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in electrical engineering from the École Polytechnique de Montreal. He has been a Professor with the Department of Electrical Engineering, École de Technologie Supérieure (ÉTS), since 1991, where he worked as the Director, from 1999 to 2001, and has held the industrial research chair position, since 2001. He is also the NSERC-Ultra Electronics Chair in Wireless Emergency and Tactical Communication, the most prestigious industrial chair program in Canada. He also founded the Communications and Microelectronic Integration Laboratory, ÉTS, and was it's the first Director. Most recently, he was appointed as the Director General of ÉTS, from June 2019 to June 2024. His research interests include wireless communications, modulation, coding, microelectronics, signal processing, equalization, software-defined radio, mobile communication, and fading channels.

DIALA NABOULSI (Member, IEEE) received the Ph.D. degree in computer science from INSA Lyon, Villeurbanne, France, in 2015. She is currently an Associate Professor with ÉTS, Montreal, Canada, with more than 12 years of experience in research. She has been involved in many Canadian and European projects, with a strong record of industrial collaborations. Her research interests include mobile networks, virtualized networks, and wireless networks.