2025 International Conference on Quantum Communications, Networking, and Computing (QCNC) | 979-8-3315-3159-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/QCNC64685.2025.00063

QPLAN: Deep Reinforcement Learning Assisted Quantum Network Planning

Yuhang Gan*, Shiyi Ling*, Ruilin Zhou, Lucinda Shen, and Chen Qian
University of California, Santa Cruz
{ygan11, rzhou39, lshen11, cqian12}@ucsc.edu
syling0913@gmail.com
*Equal contribution.

***Abstract*—Quantum networks attract increasing attention due to their unique advantages over classical networks. However, the challenge of efficiently and economically constructing reliable wide-area quantum networks remains unresolved. This problem, termed network planning, is typically modeled as an Integer Linear Programming (ILP) problem. Existing solver-based solutions for classical networks face scalability issues, while heuristic approaches that sacrifice optimality for scalability heavily rely on human expertise and lack generalizability. Recently, deep reinforcement learning (DRL) has demonstrated potential in adaptive decision-making for classical network planning. However, quantum networks introduce unique constraints due to their end-to-end entanglement distribution requirements. To address this, we propose a two-stage quantum network resource allocation framework, QPLAN, that combines multi-agent DRL with traditional optimization solvers. Our approach uses a novel multi-agent reinforcement learning strategy using graph neural networks (GNNs) and transformers for traffic flow and corresponding hardware resource allocation to minimize hardware cost while cooperatively satisfying problem constraints as much as possible. The output of the DRL algorithm is then used as an initial input to prune the solution space, "warm-starting" the ILP solver and further improving the solution. This hybrid approach effectively reduces manual effort and enhances scalability. Experimental results show that our method yields near-optimal solutions with better scalability than only using ILP solvers.**

***Index Terms*—Quantum Networks, Network Planning, Reinforcement Learning**

I. Introduction

Quantum networks are a critical component of next-generation communication technologies, enabling many quantum-unique applications [1]. Network planning—the process of optimizing resource allocation under physical constraints to meet service demands—has long been a fundamental problem in classical network design [2], [3]. Its core objective is efficiently configuring resources (e.g., fibers and switches in classical networks) to ensure reliable and high-performance data transmission. This task involves balancing multiple constraints, including fiber capacity, traffic load balancing, and reliability. While network planning is often formulated as an Integer Linear Programming (ILP) problem [2], the exponential growth of the solution space with network scale renders purely solver-based approaches impractical for real-world deployments. Consequently, practitioners often resort to expert-designed heuristics tailored to specific architectures—a strategy that sacrifices generalizability and optimality for computational tractability. Traditional heuristic-based methods suffer from several limitations: (1) High dependency on expert knowledge, requiring costly re-engineering for different network topologies; (2) Potential biases in heuristic rules that restrict solution optimality and adaptability; (3) Labor-intensive parameter tuning processes that hinder fast deployment.

Recent studies highlight the potential of Deep Reinforcement Learning (DRL) for adaptive network optimization [2], [3] Unlike heuristics, DRL autonomously learns topology-aware strategies through exploration in the search space of the problem, exhibiting strong generalizability across network configurations [4]. Although DRL cannot guarantee optimality, its bias-free exploration capability surpasses human-expert heuristics while drastically reducing manual effort. Furthermore, DRL-generated solutions effectively prune the search space. This motivates a natural hybrid paradigm: using DRL to "warm-start" solvers, where initial solutions provided by DRL guide ILP solvers to refine results within a constrained search space—a strategy proven powerful in recent research [2].

However, quantum network planning demands distinct modeling considerations due to its unique hardware characteristics. Unlike classical networks relying on switches and routers, quantum communication depends on specialized components such as entanglement sources, quantum memories, and quantum frequency converters (QFCs). These devices and unique features from the quantum world impose novel constraints, including wavelength compatibility requirements and end-to-end entanglement distribution fidelity degradation. For example, single-hop entanglement pair success rates decay significantly with link distance, and end-to-end fidelity depends on per-hop fidelity. To ensure viable entanglement distribution rates and quality, hardware allocation must account for distance-dependent resource provisioning and potential entanglement distillation operations—constraints absent in classical network planning. Despite its critical role in scalable quantum communication, systematic research on quantum network planning remains scarce, with existing studies focusing on theoretical models rather than practical resource allocation strategies [5].

To bridge this gap, we systematically model shared and unique characteristics of quantum and classical networks, introducing our two-stage optimization framework, QPLAN,

979-8-3315-3159-1/25/$31.00 ©2025 IEEE
DOI 10.1109/QCNC64685.2025.00063

that synergizes DRL with solver-based methods for scalable planning solutions. Specifically, We first model quantum network planning as a multi-agent reinforcement learning traffic allocation problem [6] and treat traffic demand from users as autonomous agents that perform problem feature extraction and make decisions. They share a policy network that coordinates the adjustment of the proportion of traffic passing through on pre-defined routing paths to reduce network construction costs. The policy network employs a Graph Neural Network (GNN) [7] to encode topological features and hardware states, followed by a Transformer-based [8] decision module to output allocation actions. This multi-agent model with shared policy networks significantly reduces the size of the policy network and, hence, the cost and difficulty of training compared to direct global optimization using a single agent with massive neural networks. Then, while neural networks outperform human-designed heuristics in unbiased distribution fitting and generalizability, they cannot guarantee constraint satisfaction or optimality. Instead of using the DRL algorithm to generate the final solution directly, we leverage DRL to prune the search space: DRL-generated solutions serve as high-quality initial inputs to "warm-start" the ILP solver. By constraining and rescaling the search space, this hybrid approach enables solvers to handle larger network instances while preserving solution quality, thereby enhancing scalability.

We summarize our contributions as follows:

- We formalize the quantum network planning problem, discussing its similarities and differences with classical ones.
- we propose a generalized quantum network planning framework based on multi-agent reinforcement learning that avoids the cumbersome heuristic design process.
- The evaluation shows good results in the near-optimal solution case, demonstrating good scalability.

II. Background and Problem Formulation

A. Necessary Background about Quantum Network

We begin with a brief introduction to the basics of quantum networks, including network components, required hardware, and some possible operations to help formalize the quantum network planning task later.

First, the basic components in a quantum network are the quantum nodes and links on the edges between the nodes. A quantum node refers to a node with quantum computation and repeater functions. For this problem, we focus on the network functions of the nodes and, therefore, only consider the resource allocation with respect to the repeater capabilities of the nodes for entanglement distribution. In order to perform entanglement pair generation, distribution, and entanglement swapping operations between neighboring nodes, quantum nodes require various necessary hardware. Quantum networks are rapidly evolving with various hardware architectures for distributing entangled pairs between nodes. For example, it is possible to directly excite a quantum memory to emit photons and set up a measurement device (usually a Bell state analyzer for 2-qubit measurements) between two nodes to establish entanglement [9], [10]. Or set up an independent source of entangled photons between the two nodes and make measurements inside the repeater node to realize entanglement between quantum memories [11]. In general, we need to deploy the necessary entanglement sources and measurement equipment at the repeater node to realize the entanglement distribution between nodes. The ability of a node to generate entangled pairs with neighboring nodes is proportional to the number of devices it is equipped with. In addition, quantum frequency converters (QFC) are needed to convert photons to a specific wavelength suitable for transmission in optical fibers (e.g., 1550 nm) [12]. Because quantum memory, measurement devices, and QFCs need to be used in conjunction with each other and need to be quantitatively matched, we treat them together as a kind of entanglement-generating device whose function is to add *link capacities* over an edge and calculate the hardware cost together. There are some optical fibers between nodes for photon transmission. The optical fibers have certain *spectral capacities*. The spectral capacity of one edge is the sum of the capacities of all optical fibers. So, for an edge, its *link capacity* always can not be greater than its *spectral capacity*.

Besides, quantum networks have some unique limitations on transmission compared to traditional networks. First, the success rate of single photon transmission in optical fiber decreases exponentially with the transmission distance, which means the same number of entanglement-generating devices will not have the same link capacities over fibers with different lengths. Secondly, users have requirements for the fidelity of the end-to-end entangled pairs. If the fidelity of the final distributed entangled pairs is less than the minimum requirements, the successfully distributed entangled pairs can not be used, which is equivalent to a distribution failure. The fidelity of the end-to-end entangled pairs is related to the fidelity of each hop, roughly proportional to the product of all single-hop fidelities. Finally, quantum networks allow additional entanglement distillation operations, where multiple low-fidelity states are consumed to generate fewer but higher-fidelity entangled pairs. Notably, performing distillation on single-hop links for long-distance transmission incurs significantly lower operational overhead than end-to-end distillation. By incorporating fundamental constraints during network planning—such as deploying additional entanglement pair generators on long-distance links to improve single-hop success rates or equipping fidelity-sensitive paths with enhanced resources to facilitate distillation, we can establish a robust hardware foundation, thereby reducing the complexity of upper-layer software protocols.

B. Problem Formulation

For the quantum network planning problem, our overarching objective is to minimize the network construction cost while satisfying all constraints.

The construction cost primarily includes the one-time procurement cost of node-internal devices and inter-node fiber

TABLE I: Symbol Definitions for the Optimization Problem

Symbol	Meaning
E	Set of network edges
P	Set of routing paths
$P(e)$	Paths traversing edge e
P_d	Routing paths for demand d
x_p	Traffic flow on path p
y_e	Number of spectrum fibers on edge e
C_e	Capacity of a single fiber on edge e
$cost_e$	Unit link capacity cost on edge e
l_e	Length of edge e
$cost_f$	Cost per fiber per unit distance
D_d	Traffic demand for demand d
$F_d(p)$	E2e fidelity over path p for demand d
F_d	Required fidelity for demand d

infrastructure, as well as potential future operational and maintenance expenses. While fiber costs are relatively straightforward to quantify, estimating costs for entanglement generation devices (e.g., quantum memories, QFCs) remains challenging due to the rapid evolution of quantum hardware technologies. As agreed above, entanglement sources, quantum frequency converters, and measurement devices must be proportionally matched to avoid resource underutilization. To address this, we aggregate the costs of node-internal devices into an equivalent entanglement link capacity cost for each inter-node edge, analogous to the traffic forwarding capacity cost of switches or routers in classical networks. However, unlike classical networks, quantum link capacity varies with physical characteristics—particularly link length. Identical hardware allocations on links of different lengths yield distinct effective capacities due to varying entanglement distribution success rates. To achieve equivalent entanglement pair capacity, longer links require more hardware resources and thus incur higher costs, as mentioned above.

$$\min \quad \sum_{e\in E}\left[\Big(\sum_{p\in P(e)} x_p\Big) cost_e \;+\; y_e \cdot cost_f \cdot l_e\right] \tag{1}$$

$$\text{s.t.} \quad \sum_{p\in P_d} x_p \geq D_d, \quad \forall\, d \in \{1,2,\ldots,|D|\}, \tag{2}$$

$$\sum_{p\in P(e)} x_p \leq y_e \cdot C_e, \quad \forall\, e \in E, \tag{3}$$

$$F_d(p) \geq F_d, \tag{4}$$

The remaining constraints ensure compliance with physical hardware limitations and traffic demand requirements:

- Demand satisfaction constraint: Each quantum node pair d requires a minimal end-to-end entanglement distribution rate as traffic demand D_d. The sum of the traffic on all routing paths needs to be greater than or equal to this.
- Link capacity constraint: The total quantum traffic allocated to an edge must not exceed its assigned spectral capacity. Each edge comprises several optical fibers, each with a fixed spectral capacity.

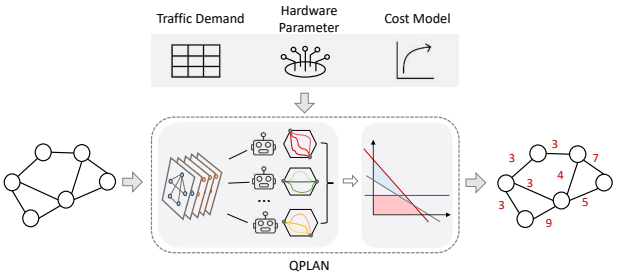

Fig. 1: Overview of QPLAN.

- Fidelity constraint: The distributed entangled pairs must satisfy user-defined minimum fidelity thresholds.

Table I summarizes the key variables for problem formulation, and Eq. 1 presents the complete optimization problem.

III. QPLAN Design

A. Overview

We describe the proposed quantum network planning framework, QPLAN, which combines a deep multi-agent reinforcement learning (MARL) algorithm based on graph neural networks (GNNs) and Transformers with solver-based planning methods to efficiently allocate quantum resources while maintaining scalability. More specifically, we adopt a two-stage approach as shown in Fig. 1.

First, a MARL algorithm takes topology information and network planning principles such as traffic demands, hardware parameters, and cost models as inputs to learn and infer an initial allocation that respects constraints as much as possible. At this stage, the obtained solution satisfies most constraints but may still violate some or allow further refinement. Subsequently, we use this learned solution to constrain the search space of variables for the integer linear programming (ILP) solver, effectively "warm-starting" the solver to complete the final resource allocation plan. This is equivalent to using the RL-generated solution as a starting point to search within a bounded region of the problem's solution space, further reducing network construction costs while satisfying all constraints. By limiting the search range, the optimization process is accelerated.

B. Equivalent Transformations for Link and Traffic

Compared to classical networks, quantum networks involve more complex hardware configuration requirements and traffic demand characteristics. First, the success rate of entanglement pair distribution on a single link—i.e., the expected link capacity—is highly correlated with its length. Longer links require more resources to achieve the same link capacity as shorter ones. This introduces heterogeneity in resource allocation: allocating a unit of hardware resources increases link capacity inconsistently across links of varying lengths. Additionally, the end-to-end fidelity of entangled pairs depends multiplicatively on the fidelity of each individual hop. Consequently, even for traffic from the same user pair, different routing paths impose distinct minimum fidelity thresholds for entanglement pairs on each single-hop link. These factors make quantum network

resource allocation far more complex than its classical counterpart. To address these, we apply equivalent transformations to hardware resources and user traffic demand.

To solve the first issue, we allocate *link capacity* mentioned above instead of directly using hardware counts as the basic allocation unit during resource allocation. After obtaining the final results, we convert this allocated capacity into actual hardware quantities based on link lengths. For the second problem, when verifying whether a generated solution satisfies traffic demand constraint (2) in Eq. 1 for traffic on different routing paths of the same user pair, we apply *fidelity-equivalent scaling* to the traffic on each path. Paths with more hops may require higher single-hop fidelity thresholds due to additional entanglement swapping operations, implying potential entanglement distillation operations on the links. Entanglement distillation uses multiple low-fidelity entangled pairs to generate fewer high-fidelity pairs. Thus, for long routing paths with more hops of the same user pair, even with the same allocated link capacity per hop, fewer usable entangled pairs will ultimately be generated compared to shorter paths. When testing the solution to determine whether it satisfies constraint (2), we take the entanglement distillation cost into the calculation. So, the resource allocation algorithm must allocate additional hardware resources on long paths to meet distillation requirements. The specific implementation of these equivalent transformations depends on hardware parameters and distillation protocols.

C. Policy Network Design

After completing the transformations for links and traffic, we eliminate the operational complexity caused by quantum network characteristics and can start the following formal resource allocation. Since we are planning and allocating resources for a network, a graph structure, the network's topological properties must be thoroughly recognized and mastered by the reinforcement learning (RL) algorithm. Graph neural networks (GNNs), which have recently been proven effective for understanding and representing graph structures, have been widely applied to graph-related tasks such as molecular modeling and traffic prediction [7]. Therefore, we adopt GNNs as the encoding module of the RL algorithm's policy network for quantum network topologies.

GNNs inherently prioritize node features over edge features. The core principle of GNNs is topology-aware feature learning through message passing mechanisms [7]. Each node iteratively updates its representation by aggregating features from neighboring nodes and combining them with its own features. This hierarchical feature propagation naturally adapts to graph-structured data, effectively capturing spatial relationships between nodes. However, for quantum network planning, unlike traditional graph tasks such as social network analysis, edge features are equally critical. First, link costs (e.g., fiber count and length) constitute a major component of network costs, and the hardware resources within nodes can be equivalently mapped to link capacities. Second, user demands are specified as traffic flows. A user pair's demand for end-to-end entanglement pair distribution rates and fidelity manifests as traffic allocated across multiple routing paths between the pair. Finally, traffic from different user pairs sharing the same link directly competes for resources, impacting allocation decisions. To address this, following prior work that models classical networks using GNNs [2], [3], we encode network edges as primary entities in the GNN, representing them as GNN nodes.

Considering the graph structure of network topology and the reasoning capability needed for this task, we propose our policy network design for RL agents. The policy network comprises two main components: a GNN-based network encoding module and a Transformer-based decision making module. The GNN module processes network information through message passing, enabling each GNN node (representing an edge) to iteratively refine its representation based on local and neighboring contexts. Each GNN layer fully represents the entire topology. During layer updates, each node sends its feature information to adjacent nodes, receives aggregated messages from neighbors, and updates its own features. As information propagates across layers, messages reach distant nodes, allowing edge nodes to learn global network states. After obtaining the network encoding from the GNN module, the Transformer-based decision module performs further traffic allocation decisions. The Transformer module consists of three parts: input embedding, encoder, and decoder. First, we concatenate the GNN-encoded network features, routing path information, and traffic demand matrices into a joint input representation to form the input embedding. The encoder leverages multi-head attention mechanisms to capture local relationships between paths and network nodes in parallel. Finally, the decoder outputs allocation strategy is based on network and traffic conditions.

The Transformer architecture is particularly suited to this task due to the dual advantages of its attention mechanism. First, self-attention mechanisms dynamically establish association weights between any two network elements (edge nodes or paths), which is critical for coordinating resource competition across multiple paths. Second, multi-head attention allows the model to simultaneously focus on multiple dimensions, such as the physical topological constraints of routing paths (e.g., fiber length) and logical service requirements (e.g., traffic demands). Additionally, the autoregressive property of the decoder enables the model to allocate traffic resources incrementally in a sequential decision-making manner.

D. Multi-Agent Reinforcement Learning for Traffic Allocation

So far, we explained how to build a policy neural network to understand and learn the network planning problem for future resource allocation, but there are still a number of challenges on exactly how to train and utilize this network. An intuitive idea is to use regular supervised learning for training, but this requires that we have sufficient data to support training. On a small-scale network, we could use solver-based solution data for training, but this makes the policy network "at best as good as the solver" and does not allow us to go beyond the

distribution of the training data. For larger problems, we can't even obtain data to train the policy network, which has the same scalability dilemma as a directly solver-based approach. Reinforcement learning, on the other hand, is a powerful tool for neural network training when training data is difficult to obtain. This can be done by actively exploring the search space to obtain the features of the problem and train the policy network. The general steps of reinforcement learning are to model the search space of the problem as an environment that can be explored by a reinforcement learning agent, design a reward function based on the structure of the problem, and reward the reinforcement learning agent for searching in the right direction, e.g., the agent violates fewer constraints during the search process or finds a less costly solution to build the network, to help train the policy network. However, the number of parameters in the policy network required for successful single-agent reinforcement learning typically increases rapidly with the size of the problem, and huge neural networks tend to be costly and difficult to train. Considering that for a medium-sized network with about 1,000 nodes, there may be a traffic demand of million user pairs, which definitely increases the difficulty of training the policy network.

However, by looking at the characteristics of this problem, we can see that the main pain point of the problem is how to satisfy the user traffic demand while also minimizing the cost. We observe that the potential optimization space is to *reduce hardware waste as well as to improve the efficiency per unit hardware cost*. The cost of laying a single fiber and its spectral capacity are fixed, but the amount of traffic that passes through the fiber can be adjusted, and increasing the utilization of already assigned fibers may reduce the total number of fibers needed. Second, the cost per unit of desired link capacity added to a short link is significantly less than that of a long one.

All the above optimizations are related to traffic adjustment. Therefore, in the RL stage, we choose to use multi-agent reinforcement learning, where the traffic demand of one user pair is used as the basic RL agent that shares a policy network with others to adjust the proportion of its required traffic on different routing paths. Each flow agent accomplishes resource allocation on routing paths relatively independently with knowledge of other flow agents' policies and current resource allocation. Agents cooperate with each other to coordinate resource competition and increase hardware utilization efficiency. Since the action number of flow agents is only related to the number of routing paths, relatively simple and insensitive to the size of the network, we are able to significantly reduce the number of parameters required for the policy network and improve the training efficiency while ensuring the solution quality. In this step, when designing the reward function for agents, we consider the robustness of the constructed network and the potential pitfalls of the preference for short paths to reduce the cost. For each traffic demand, three routing paths are pre-computed (the path computation method depends on the routing protocol), and the traffic allocated on each path is required to be as close as possible. This makes the network more robust to single-link failures, as we have shown in the evaluation part, avoids potential bottleneck links caused by the extreme preference for short paths to reduce costs, and ensures load balancing to a certain extent. It should be emphasized that this design preference is not static but can be easily changed by modifying the reward function according to the actual network design needs, e.g., the quantum data center no longer needs to consider the length of the path [13], which ensures the generality of our approach.

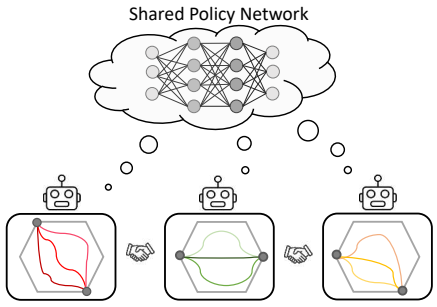

Fig. 2: Multi-agent Reinforcement Learning Traffic Allocation for different user traffic demands. Agents share one policy network and each agent is responsible for one demand.

E. Warm-Starting the solver using RL's solution

After the first stage, we obtained a solution using the reinforcement learning algorithm, but the solution based on the neural network output does not always satisfy all constraints and could potentially be further optimized. Our experiments demonstrate that, for the most part, this solution already satisfies the vast majority of traffic requirements, as shown in Fig. 6. Therefore, we choose to use the ILP solver for further optimization based on the solution obtained by the RL algorithm. More specifically, we kind of scale the output of RL using a scale factor α and set it as the upper bound of the variable to be optimized. This brings the significant benefit that the RL algorithm has fully explored the search space of the problem, and hence, the solution obtained is already relatively close to the desired optimal solution. The sub-search space tensioned with this solution as the origin will be significantly smaller than the original search space of the problem, and thus, the solver's solution process is accelerated, as we show in Section IV. Fig. 3 demonstrates this process.

IV. Performance Evaluation

A. Methodology

Implementation We ran our experiments on a Ubuntu 20.04 server equipped with two 64-Cores AMD Ryzen Threadripper PRO 5995WX and 1TB RAM. One additional Nvidia RTX A6000 GPU is available for the first stage of QPLAN. We use Graph Convolutional Network (GCN) [7] as our default GNN network architecture and implement it and Transformer architecture using PyTorch [14].

Topologies. Because there is no operational quantum network topology, we selected widely used classical topologies as candidates, leveraging their structural characteristics for

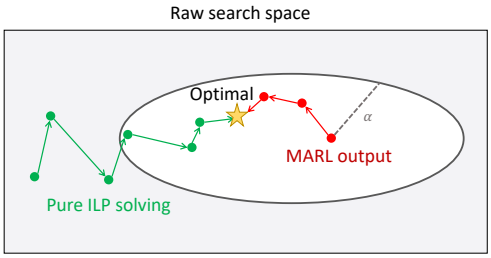

Fig. 3: Using multi-agent reinforcement learning (MARL) output as initial input to reduce search space, "warm-starting" ILP solver. The output of MARL is scaled appropriately using a scale factor α and set as the upper bound of the variable to be optimized.

our experiments. We use four network topologies in different scales. Abilene, Cogentco, and Kdl obtained from publicly dataset [15]. Due to the difficulty in obtaining network topologies with the complete information required from public datasets, we generated a large-scale topology using the Waxman model [16]. For ease of reference in subsequent descriptions, we have named the generated topology Waxman850. Table II summarizes their numbers of nodes and edges. Each edge can contain multiple optical fibers, and each optical fiber has a spectral capacity.

TABLE II: Network Topology Information

Topology Name	# of Nodes	# of Edges
Abilene	11	14
Cogentco	197	243
Kdl	754	1790
Waxman850	850	2240

Traffic data. Currently, there is no available quantum network traffic data. So, we generate them by sampling the traffic demand for each source-destination pair from a normal distribution. Of all generated traffic matrices, 80% are used as the training data set for MARL training in *QPLAN*, and 20% are used as the testing data set for all methods.

Baselines. We compare *QPLAN* with the following baselines.

- *Greedy*: For each traffic demand, *Greedy* chooses the path with the lowest cost for flow allocation.
- *ILP*: ILP solver solves the quantum network planning problem for all traffic demands by optimizing flow allocation on the candidate paths and the number of optical fibers allocated to each link. We choose the widely used commercial solver, Gurobi [17] v9.1.2, as the ILP solver.
- *First-stage*: The MARL model trained in the first stage of *QPLAN* outputs the flow allocation on each path, and then the number of optical fibers on each edge is determined based on the fiber capacity.

Objectives. Our objective is to minimize the total cost, which is the sum of the cost incurred by allocating flow on links and the cost generated by deploying optical fibers between links.

Metrics.

- *Total cost:* We measured the total cost of the network traffic plans generated by four different methods mentioned above. The flow allocation cost for each link is proportional to the corresponding distance, with the unit cost of optical fiber set as a constant value.
- *Computation time:* The computation time required for the network plans generated by methods.

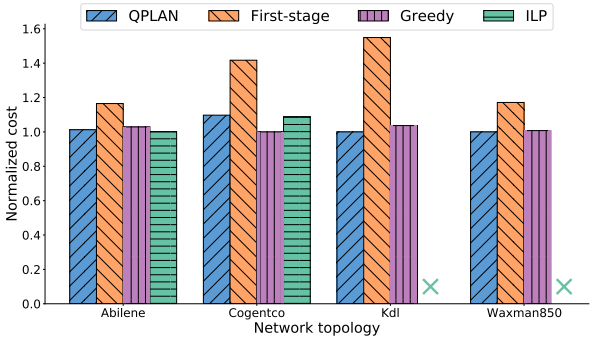

Fig. 4: *QPLAN* finds optimal solutions comparable to those of *ILP* in small topologies Abilene and Cogentco. *ILP* fails to scale to large topologies Kdl and ANS, while *QPLAN* attains scalable performance as the topology size grows. The results are normalized to the optimal cost of each topology.

Besides, we measured the satisfied demand of *First stage* to assess the quality of the initial solutions generated by the trained MARL model. We also evaluated the failure tolerance of the four methods in scenarios with link failures to examine the stability of the generated network plan under fault conditions.

B. Evaluation results

Optimality for small-scale topologies. Since *ILP* solver achieves optimal solutions on small-scale topologies, we evaluate the optimality of *QPLAN* by comparing it against *ILP* solver on these networks. Fig. 4 presents the cost results of all four methods across four topologies (Abilene, Cogentco, Kdl, and Waxman850), with the results normalized to the optimal cost of each topology. As we can see, in small-scale topologies Abilene and Cogentco, *QPLAN* is able to achieve the same optimal solutions as *ILP* solver.

Scalability for large-scale topologies. For large-scale topology, Kdl and Waxman850, *ILP* solver failed to solve in a reasonable time duration, shown in Fig. 4. In our experiment, the results of these two topologies can not be computed in two days. And *Greedy* fails to find the optimal solution. In contrast, *QPLAN* produces a high-quality plan for large-scale topologies Kdl and Waxman850. Benefiting from the high-quality solutions generated in the first stage, *QPLAN* effectively extends the solver's power, achieving a better trade-off between scalability and performance.

Computation time. Based on the cost results shown in Fig. 5, the *First-stage* method is very fast, even on large-scale topologies such as Kdl and Waxman850. Using the initial solution gained from *First-stage*, *QPLAN* can efficiently tackle large-scale problems that *ILP* solver cannot solve in a reasonable time duration. For small-scale topologies Abilene and Cogentco, *QPLAN* exhibits computation times quite similar to those of *ILP* solver. In large-scale topologies Kdl and

Waxman850, the computation time of *QPLAN* is comparable to that of *Greedy*; however, *QPLAN* is able to find better solutions than *Greedy*.

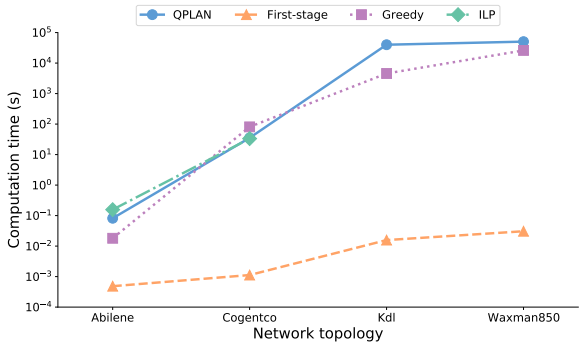

Fig. 5: Comparing computation time of different methods across different topologies.

High-quality traffic plan from the first stage of QPLAN. *First-stage*, which is the RL multi-agent model trained during the first stage of *QPLAN*, is capable of finding a plan for allocating traffic on candidate paths. Although this plan is not optimal, as shown in 6, it ensures a satisfied demand percentage of at least 80% across all tested topologies. Therefore, it is reasonable to use this solution as an upper bound for the optimization variables representing the flow on each path in the second stage of *QPLAN*, which employs ILP. This approach effectively narrows the search space of ILP, enabling *QPLAN* to find the optimal solution even for large-scale problems quickly.

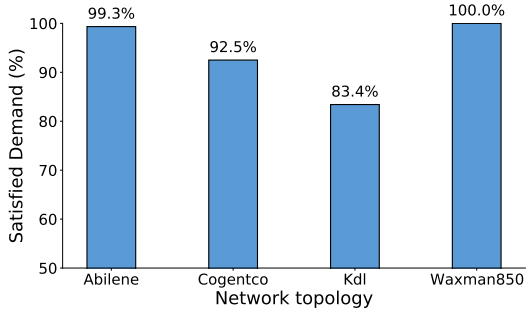

Fig. 6: Satisfied demand of solutions obtained by *First stage* across all topologies can achieve a percentage of at least 85%.

Failure tolerance. The solutions obtained by *QPLAN* can find the optimal results in a short time for both small-scale and large-scale problems. Additionally, it also demonstrates more substantial failure tolerance in situations where links may fail. Under such conditions, *QPLAN* achieves the highest percentage of satisfied demand compared to all other methods, as shown in Fig. 7.

C. Sensitivity Analysis

Finally, to better understand the impact of various hyperparameters of *QPLAN*, we conduct a sensitivity analysis on the number of GNN layers of the policy network and the scale factor α. We vary the number of GNN layers of the policy

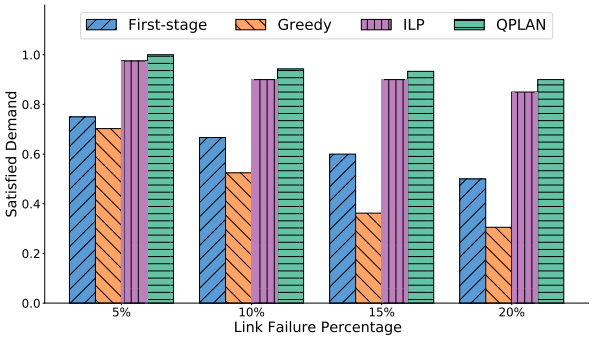

Fig. 7: Satisfied demand of solutions obtained by four methods under random link failures of 5%, 10%, 15%, and 20% on Cogentco network. *QPLAN* outperforms all other methods.

network to 1, 2, and 4, observing the corresponding changes in the cost results of the *First-stage* across these topologies. Fig. 8 displays the costs of the *First-stage* normalized against their optimal costs for each topology. The number of GNN layers significantly affects the output results of the *First-stage*. For different problems across various topologies, the optimal number of GNN layers may differ. For instance, in topologies Abilene and Cogentco, the *First-stage* performs best with 2 GNN layers, while in topologies Kdl and Waxman850, the optimal performance is achieved with 4 GNN layers.

Next, we assess the impact of the scale factor. The scale factor determines how large the sub-search space is generated with RL's solution as the initial input in the second stage. We evaluate *QPLAN* using different scale factors (1, 1.25, 1.5) for the four tested topologies. Fig. 9 presents the results of *QPLAN* normalized by the optimal cost for each topology. Our analysis reveals that a scale factor that is too small may hinder *QPLAN* from finding the optimal solution during the ILP solving stage due to a limited search space. When the scale factor is set to 1, *QPLAN* fails to identify the optimal solution for topologies Abilene, Cogentco, Kdl, and ANS. With a scale factor of 1.25, *QPLAN* still does not find the optimal solutions for topologies Cogentco, Kdl, and Waxman850. However, increasing the scale factor to 1.5 enables *QPLAN* to successfully find the optimal solution for all tested topologies.

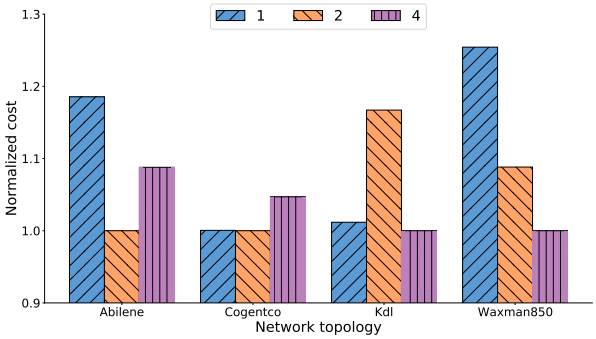

Fig. 8: Impact of GNN layers on the results of *First-stage*. The results are normalized to the optimal cost of each topology.

V. Related Work

Quantum network design. To our knowledge, there is no research on quantum network planning, especially resource

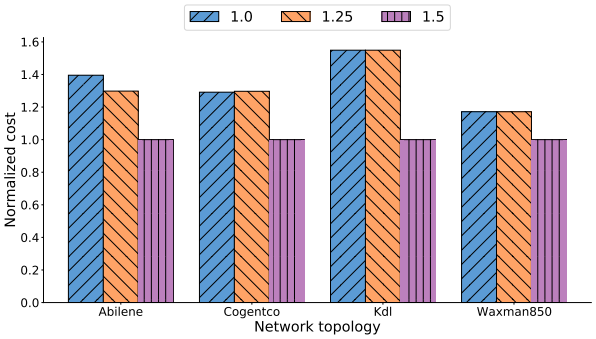

Fig. 9: Impact of scale factor on the results of *QPLAN*. The results are normalized to the optimal cost of each topology.

allocation. However, some related studies on quantum network design have been proposed. A related question is how to select quantum network repeater nodes based on the topology of existing Internet backbone nodes. Rabbie $et\ al.$ first introduced the concept of designing quantum networks using existing classical network infrastructure [18]. They formulated an optimization problem to satisfy user-specific rate and fidelity thresholds for aggregation while minimizing the number of repeater nodes required in the network and solved it using an ILP solver. Pouryousef $et\ al.$ [19] proposed a solver-based solution to this problem, but they attempted to maximize network utility. Islam and Arslan proposed a heuristic for this problem [20]. However, all these methods only consider the number of required repeater nodes and do not have fine-grained scheduling for hardware resources. Besides, Ref. [21] studied how to maximize network utility using existing network resources.

Network Planning for Classical Networks. As one of the fundamental problems of network design, network planning continues to receive attention in traditional networks. Zhu $et\ al.$ et al. proposed a DRL-based network planning algorithm to improve the scalability of the network planning problem, similar to the idea of this paper [2]. However, they used single-agent reinforcement learning, and the action number of agents increases by square with the increase of network size, which still has scalability limitations. Ref. [3] improves the operational efficiency of the network through traffic engineering. In particular, Ref. [3] also uses multi-agent reinforcement learning to model traffic, but neural network based traffic engineering does not guarantee that the traffic allocation is strictly legal, which is unacceptable for the task of resource allocation. All the above planning methods for classical networks cannot be directly used to solve quantum network planning because they do not consider the unique limitations of quantum networks.

VI. Conclusion

In this work, we proposed QPLAN, a DRL-assisted quantum network planning framework that integrates multi-agent reinforcement learning with solver-based optimization to address the unique constraints of quantum networks. By leveraging a shared-policy GNN-Transformer model, our approach efficiently extracts network features and optimizes resource allocation while using DRL-generated solutions to warm-start ILP solvers for improved scalability. Experimental results show that QPLAN achieves near-optimal performance with reduced computational overhead.

VII. Acknowledgment

The authors were partially supported by NSF Grants 2322919, 2420632, 2426031, 2426940, 2114113, and DoE Grant DE-SC0022069.

References

[1] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical computer science*, 2014.
[2] H. Zhu, V. Gupta, S. S. Ahuja, Y. Tian, Y. Zhang, and X. Jin, "Network planning with deep reinforcement learning," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021.
[3] 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 *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023.
[4] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!," *arXiv preprint arXiv:1803.08475*, 2018.
[5] S. Lloyd, "Capacity of the noisy quantum channel," *Physical Review A*, vol. 55, no. 3, p. 1613, 1997.
[6] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2008.
[7] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.
[8] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.
[9] R. Zhou, X. Lai, Y. Gan, K. Obraczka, S. Du, and C. Qian, "A simulator of atom-atom entanglement with atomic ensembles and quantum optics," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1, pp. 1271–1277, IEEE, 2023.
[10] Y. Gan, X. Zhang, R. Zhou, Y. Liu, and C. Qian, "A routing framework for quantum entanglements with heterogeneous duration," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1, pp. 1132–1142, IEEE, 2023.
[11] C. Jones, D. Kim, M. T. Rakher, P. G. Kwiat, and T. D. Ladd, "Design and analysis of communication protocols for quantum repeater networks," *New Journal of Physics*, vol. 18, no. 8, p. 083015, 2016.
[12] X. Wang, X. Jiao, B. Wang, Y. Liu, X.-P. Xie, M.-Y. Zheng, Q. Zhang, and J.-W. Pan, "Quantum frequency conversion and single-photon detection with lithium niobate nanophotonic chips," *npj Quantum Information*, vol. 9, no. 1, p. 38, 2023.
[13] R. Zhou, Y. Gan, and Y. Liu, "Towards flow scheduling in a quantum data center," in *Proceedings of the 1st Workshop on Quantum Networks and Distributed Quantum Computing*, pp. 45–50, 2023.
[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.
[15] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.
[16] B. M. Waxman, "Routing of multipoint connections," *IEEE journal on selected areas in communications*, 1988.
[17] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2021.
[18] J. Rabbie, K. Chakraborty, G. Avis, and S. Wehner, "Designing quantum networks using preexisting infrastructure," *Quantum Information*, 2022.
[19] S. Pouryousef, H. Shapourian, A. Shabani, and D. Towsley, "Quantum network planning for utility maximization," in *Proceedings of the 1st Workshop on Quantum Networks and Distributed Quantum Computing*, pp. 13–18, 2023.
[20] T. Islam and E. Arslan, "A heuristic approach for scalable quantum repeater deployment modeling," in *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, pp. 1–9, IEEE, 2023.
[21] G. Vardoyan and S. Wehner, "Quantum network utility maximization," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1, pp. 1238–1248, IEEE, 2023.