Achieving Load-Balanced Entanglement Distribution in Quantum Networks

***Abstract*—Entanglement distribution, which involves routing and swapping strategies, is essential for quantum communication in quantum networks. However, existing routing algorithms often overlook the constraints imposed by finite quantum memory at each node and tend to deplete the quantum memory of nodes along the shortest paths, thereby increasing the risk of end-to-end (E2E) entanglement failures. Additionally, suboptimal swapping strategies can further imbalance the quantum memory utilization across nodes. Therefore, effective load-balancing entanglement distribution is essential for improving the robustness and efficiency of quantum networks. To address this challenge, we first propose a *bottleneck node model* that considers shared memory at each node. Then, we present the *load-balanced entanglement distribution* problem, which aims to optimize routing decisions and swapping strategies to achieve balanced memory distribution. As finding the optimal solution is NP-hard, we decompose the problem into two sub-problems: "*path finding*" and "*quantum memory allocation*". We propose LBER (Load-Balanced Entanglement Routing) to optimize entanglement distribution across multiple paths and using our crafted adaptive swapping strategy, AdapSwap, to determine the optimal swapping order along each path. Extensive simulations demonstrate that our approach significantly improves load balance and increases the request satisfaction ratio by 7.73%∼31.70%.**

***Index Terms*—Quantum networks, entanglement routing, load balance, entanglement swapping**

I. Introduction

Quantum networks represent an emerging networking paradigm enabling secure communication [1], [2] and large-scale distributed quantum computing [3]. In such networks, quantum nodes, such as quantum processors, repeaters, switches, and routers, are interconnected via quantum links (e.g., optical fibers) to generate, store, exchange, and process quantum information. Unlike classical networks, quantum networks are restricted by the no-cloning theorem [4], which prohibits conventional storage and forwarding of quantum data. Instead, a quantum communication request between the source and destination node requires multiple end-to-end (E2E) entanglements.

To establish an E2E entanglement, nodes first generate elementary entanglement pairs with adjacent nodes and extend these short elementary entanglements to longer distances via entanglement swapping, which can be facilitated by quantum repeaters [5]. Once an elementary entanglement pair is generated, each of the entangled qubits is temporarily stored in the quantum memory unit at the corresponding end node until the communication request is fulfilled. This temporary storage imposes a memory "load" on the quantum nodes. The primary objective of quantum networks is to efficiently generate and distribute E2E entanglements while satisfying continuous requests under quantum memory constraints.

One key challenge lies in effectively routing entanglements across the network. When a request is received, the network must compute routing paths and determine the number of E2E entanglements to be generated along each path. Previous studies have proposed various entanglement routing protocols aimed at improving throughput [6], [7], robustness [8], resource efficiency [9], fidelity [10]–[12], and reducing total waiting time [13]. Despite these advances, most existing routing algorithms overlook the constraint imposed by quantum memory in quantum networks. Furthermore, current quantum routing algorithms often prioritize shortest-path routing, which can lead to the overloading of quantum memories along these shortest paths. Such resource exhaustion at a node creates single points of failure, undermining network resilience and blocking subsequent requests. Thus, designing a load-balanced entanglement routing protocol is essential for enhancing the robustness and reliability of quantum networks.

Another key challenge is optimizing the swapping strategy to ensure balanced memory usage across nodes along the selected path. A swapping strategy decides where entanglement swapping operations should occur along the path to extend entanglement, thereby influencing the order of entanglement swapping and how the load is distributed across nodes. Due to the probabilistic nature of successful entanglement swapping, some elementary entanglements may fail to achieve E2E entanglement and must be discarded. To ensure reliable transmissions, additional elementary entanglements must be generated in advance. Nodes experiencing higher entanglement loss during swapping require even more pre-established entanglements, which increases their memory load. Consequently, the choice of swapping strategy significantly impacts both the memory usage of individual quantum nodes and the overall load balance performance.

In this paper, we present a novel approach for load-balanced entanglement distribution in quantum networks. We note that previous studies on quantum entanglement distribution [7], [10]–[14] overlook critical limitations of the widely adopted bottleneck link model. This model assumes a fixed number of entangled pairs can be generated per time slot on each link and treats the link capacity as the bottleneck. However, real-world quantum devices typically share memory across multiple interfaces [15], leading to inherent interdependencies among connected links. To address such limitations, we propose a more realistic *bottleneck node model* that accounts for shared memory at each node. Then we formulate the load-balanced

entanglement distribution problem. Given its NP-hardness, we decompose it into two sub-problems: (1) path finding and (2) quantum memory allocation. To tackle these challenges, we propose the load-balanced entanglement routing (LBER) algorithm based on integer programming and linear relaxation techniques. The primary objective is to minimize the maximum load across all nodes in the quantum network. In addition, we address the entanglement swapping problem along each path by designing a dynamic programming-based algorithm, AdapSwap, to further balance node loads. Extensive simulation results demonstrate that our proposed approach achieves better load balance performance and a 7.73%∼31.70% higher request satisfaction ratio.

Our main contributions are summarized as follows: i) We formulate the load-balanced entanglement distribution problem and prove its NP-hardness. ii) We design a load-balanced entanglement routing algorithm, LBER, to optimize routing decisions efficiently and propose an adaptive swapping strategy, AdapSwap, to dynamically the optimal swapping strategies. iii) We conduct extensive simulations to evaluate LBER and AdapSwap under various network conditions. The results show that our approach improves load balance and achieves a 7.73%∼31.70% higher request satisfaction ratio.

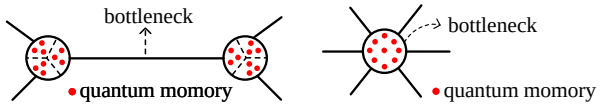
The rest of this paper is organized as follows. In Sec. II, we introduce the background of quantum network models and quantum operations. Then, we describe the load-balanced entanglement distribution problem in Sec. III. We present our routing algorithm and swapping strategy in Sec. IV. In Sec. V, we evaluate the performance of LBER and AdapSwap. Sec. VI reviews related works. Finally, Sec. VII concludes the paper.

II. Background

A. Quantum Network Model

We model the quantum network as a connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, C^{node})$, where $\mathcal{V}$ denotes the set of *quantum nodes*, $\mathcal{E}$ represents the set of *quantum links*, and C^{node} denotes the set of *node capacity* (i.e., the size of quantum memory at each node). Let $c_v \in C^{node}$ denote the quantum memory size of node v for $v \in \mathcal{V}$. It determines the maximum number of entangled pairs that node v can support. Denote $(u, v) \in \mathcal{E}$ as the quantum link between node $u \in \mathcal{V}$ and node $v \in \mathcal{V}$. When two nodes are connected by a quantum link, an *elementary entanglement* can be generated and each of the two entangled qubits is distributed to the corresponding end node. The probability of successful entanglement generation over a link depends on the link length and its physical properties, i.e., $p_{gen} = e^{-\eta\delta}$, where η is the attenuation coefficient of the quantum link, and δ is the link length. While entanglement generation is inherently probabilistic, repeated attempts can eventually succeed. Therefore, for simplicity, we assume that each link can reliably provide elementary entanglement.

The network needs to find a set of $\mathcal{K} = \{1, \cdots, K\}$ paths, from the source node s to the destination node d for generating E2E entanglements. Specifically, a path represents a series of quantum nodes that an E2E entanglement must traverse to connect the source and destination nodes. An *h-hop path* $k \in \mathcal{K}$ can be denoted as $\mathcal{P}_k = \{v_0, v_1, \cdots, v_{h-1}, v_h\}$, where $v_0 = s$, $v_h = d$, and $v_1, \cdots, v_{h-1}$ are the intermediate nodes. Moreover, we define the entanglement between two nodes v_i and v_j on path k as a *segment* $\sigma^k_{i,j}$. For example, the E2E entanglement between s and d is represented as segment $\sigma^k_{s,d}$.



(a) Bottleneck link model (b) Bottleneck node model

Fig. 1: Network bottleneck models

B. Quantum Network Bottleneck Model

Depending on where the bottleneck occurs, one can model a quantum network in two distinct ways:

Bottleneck link model. In most of the existing literature [7], [10]–[14], quantum networks are modeled as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, C^{link})$, where C^{link} corresponds to the set of link capacities. In this model, the *link capacity* is treated as the bottleneck since it assumes a fixed number of entangled pairs can be generated per time slot on each link. By modeling link capacity as the bottleneck, the bottleneck link model results in independent link capacities. As shown in Figure 1(a), the dotted lines inside the nodes indicate that each interface is assigned independent quantum memory. However, modern quantum devices typically share quantum memory across multiple interfaces [15] and create interdependencies between link capacities. By treating all links as independent entities, the *bottleneck link model* does not reflect real quantum devices.

Bottleneck node model. We propose *bottleneck node model* that considers the node capacity, C^{node}, as the bottleneck. In this model, the total capacity of all links connected to a node cannot exceed the node capacity. Specifically, the capacity of a link connecting node v_1 and node v_2 is no more than $\frac{1}{2}\min\{c_{v_1}, c_{v_2}\}$, where c_{v_1} and c_{v_2} are the memory sizes of nodes v_1 and v_2, respectively. To be more specific, if node v_1 and v_2 have quantum memory sizes of 10 and 6, respectively, the maximum capacity of the link between v_1 and v_2 would be 3. This is because each node requires one qubit to connect to the previous node and one qubit to connect to the next node. As illustrated in Figure 1(b), the quantum memory is shared across multiple interfaces on a node, meaning the capacity of a link depends not only on the link itself but also on the available memory of both connected nodes. This interdependency between link capacities results in correlated capacities, making our node capacity model more accurate for modern quantum devices where memory sharing significantly impacts network performance.

C. Quantum Operations

Entanglement generation: Quantum entanglement refers to a phenomenon where multiple qubit states are interconnected such that measuring one qubit instantly affects its entangled counterpart. In this paper, we consider EPR (Einstein-Podolsky-Rosen) pairs, specifically the Bell state $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. As shown in Figure 2(a), these entangled

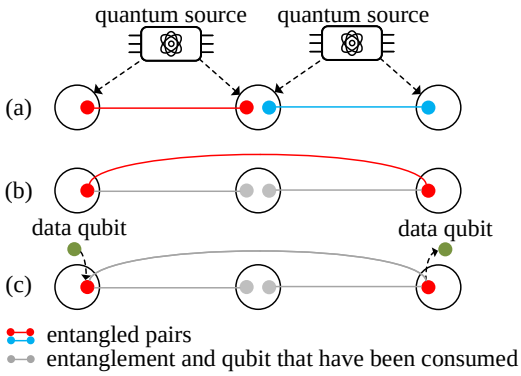


Fig. 2: Quantum operations

pairs are generated between connected nodes using a quantum source, and each of the two entangled qubits is distributed to the corresponding end via optical fibers. Once generated, the entangled pairs are stored in quantum memory units for subsequent use.

Entanglement swapping: Establishing long-distance entanglement directly is challenging due to channel noise and the fragility of quantum states. Entanglement swapping mitigates this by extending short-distance entanglements into long-distance entanglements with the assistance of quantum repeaters. As shown in Figure 2(b), a quantum repeater consumes two one-hop entangled pairs (i.e., elementary entanglements) to generate a new two-hop entangled pair. However, this operation is inherently probabilistic; if the swapping operation fails, the original entanglements are lost and cannot be recovered.

Quantum teleportation: Quantum teleportation [16] enables the secure transmission of quantum information without physically moving particles. As illustrated in Figure 2(c), the quantum state of a data qubit is recreated at the receiver using quantum entanglement and classical communication. This process ensures the secure transmission of quantum information as the original qubit remains unexposed during the process.

III. Problem Formulation

In this section, we formulate the load-balanced entanglement distribution problem in quantum networks. Notations used throughout the paper are summarized in Table I.

A. Load-balanced Entanglement Distribution Problem

Let s represent the source node, d the destination node, and n_e the required number of E2E entanglement for a request $r = (s, d, n_e)$ from s to d. When a request r arrives, the objective of the quantum network is to establish E2E entanglement between s and d, while distributing the network load (i.e., quantum memory usage) as evenly as possible to enhance the network's robustness. We convert the undirected network $\mathcal{G}$ to a directed network flow graph by representing each undirected link as two directed links with opposite directions. The flow rate refers to the number of elementary entanglements established on each link. We begin

Table I: Notations

Notation	Description
c_v	Quantum memory size of node v
n_e	Number of E2E entanglement required for a request
τ_v	Quantum memory that has been used at node v
$f_{r,(uv)}$	Number of elementary entanglements established on link (u, v) for request r
L_v	Load on node v, i.e., the quantum memory utilization ratio
$\mathfrak{O}$	Swapping strategy that determines the swapping order for all paths
$\mathcal{O}_k$	Swapping order on path k
$\alpha_{uv}^{\mathfrak{O}}$	Ratio to quantify the proportion of E2E entanglements successfully generated after swapping with $\mathfrak{O}$ at link (u, v)
$\beta_{kv}^{\mathfrak{O}}$	Scaling factor of node v on path k under swapping strategy $\mathfrak{O}$
λ_k	Number of E2E entanglements to establish path k
$M(v, \lambda_k)$	Memory utilization function for node v with λ_k
$\mathbf{G}$	Matrix to describe the relationship between paths and nodes
$\mathcal{P}_k$	k-th routing path of request
p_u^{swap}	Entanglement swapping success probabilities at node u
q_u	Swapping cost at node u, which is the inverse of p_u^{swap}
$\mathbf{Z}$	Load matrix in *Adaptive Swapping* strategy
$\mathbf{E_m}$	Count matrix where each element represents the number of elementary entanglements established on link $(m-1, m)$

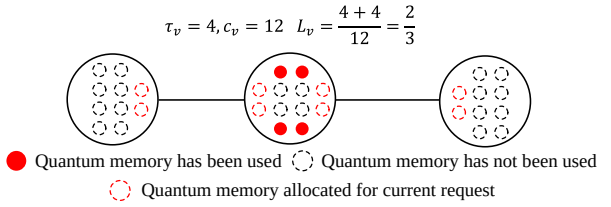
by considering the load-balancing objective of our problem as follows.

Load-balancing objective. Let $\tau_v \in \mathbb{N}$ denote the quantum memory that has been used at node v. Given a set of requests $\mathcal{R} = \{1, \cdots, R\}$, the network computes a *flow allocation matrix* $\mathbf{F} = [f_{r,(uv)}] \in \mathbb{N}^{R \times |\mathcal{E}|}$, where $f_{r,(uv)}$ denotes the number of elementary entanglements established on link (u, v) for request r. Once an entanglement pair is generated over link $(u, v) \in \mathcal{E}$, one entangled qubit is stored in the quantum memory of node u and also at the other node v's memory. The load on node v is defined as its quantum memory utilization ratio as follows:

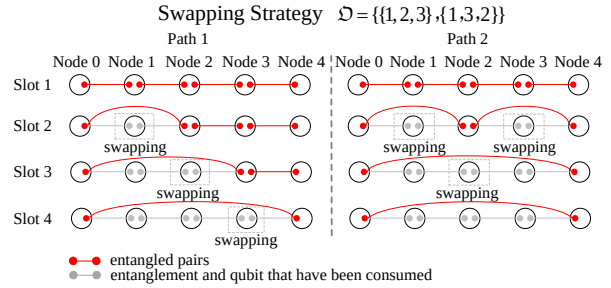
$$L_v = \frac{\tau_v + \sum_{r=1}^{R} \sum_{(u,v)\in\mathcal{E}} (f_{r,(uv)} + f_{r,(vu)})}{c_v}. \quad (1)$$

We use an example in Figure 3(a) to explain L_v in (1). In Figure 3(a), the red-filled circles indicate the quantum memory that has been used ($\tau_v = 4$), the red-dashed circles represent the quantum memory allocated for the current request ($\sum_{r=1}^{R} \sum_{(u,v)\in\mathcal{E}} (f_{r,(uv)} + f_{r,(vu)}) = 4$), and the black-dashed circles represent the unused quantum memory. The total of these three types equals the quantum memory size of node v ($c_v = 12$). Thus, the load on node v is $L_v = \frac{4+4}{12} = \frac{2}{3}$.

The network also needs to determine a swapping strategy $\mathfrak{O} = \{\mathcal{O}_1, \cdots, \mathcal{O}_K\}$ for all paths from s to d, where $\mathcal{O}_k$ represents the swapping order for path $k \in \mathcal{K}$, specifying the order of nodes that perform entanglement swapping. Specifically, $\mathcal{O}_k = \{u, \cdots, v\}$, with $u, v \in \mathcal{V}$, meaning that swapping starts at node u, progressively extends entanglement, and finally completed at node v. Figure 3(b) shows an example of the swapping strategy. Multiple elementary entanglements are extended to a long-distance entanglement by performing the entanglement swapping at nodes according to the given swapping strategy ($\mathfrak{O} = \{\{1, 2, 3\}, \{1, 3, 2\}\}$), which determines the swapping order for Path 1 and Path 2. The objective of the load-balanced entanglement distribution problem is to



(a) An example of node load



(b) An example of swapping strategy

Fig. 3: Illustration of node load and swapping strategy

balance the load across all nodes by minimizing the maximum node load, i.e., $\min_{\mathbf{F},\mathfrak{O}} \max_{v\in\mathcal{V}} L_v$ as shown in (2a).

Constraints. Due to the probabilistic nature of entanglement swapping, the two shorter entangled pairs may fail to establish E2E entanglement, making them unusable for any request. To account for this, we introduce a ratio $\alpha_{uv}^{\mathfrak{O}} \in [0,1]$ at link (u,v) to quantify the proportion of E2E entanglements successfully generated under the swapping strategy $\mathfrak{O}$. Specifically, only $f_{r,(uv)} \cdot \alpha_{uv}^{\mathfrak{O}}$ E2E entanglements can be established when $f_{r,(uv)}$ elementary entanglements are prepared in advance. Combining this ratio with the flow conservation condition, we have the constraint in (2b), which ensures that the total flow rate entering a node equals the total flow rate exiting it for all intermediate nodes. For the source and destination nodes, the total flow rate entering and exiting the node must equal the number of requested entanglements, n_e. Additionally, the quantum memory usage at each node must not exceed its capacity as shown in constraint (2c), and the number of elementary entanglements on each link must be a non-negative integer as specified in constraint (2d).

In summary, the load-balanced entanglement distribution problem is formulated as follows:

$$\min_{\mathbf{F},\mathfrak{O}} \max_{v\in\mathcal{V}} L_v \tag{2a}$$

$$\text{s.t.} \sum_{(u,v)\in\mathcal{E}} \left(f_{r,(uv)} - f_{r,(vu)}\right) \alpha_{vu}^{\mathfrak{O}} = \begin{cases} -n_e, & \text{if } v = s, \\ n_e, & \text{if } v = d, \\ 0, & \text{otherwise,} \end{cases} \quad \forall r \in \mathcal{R}, \forall v \in \mathcal{V}, \tag{2b}$$

$$\sum_{r=1}^{R} \sum_{(u,v)\in\mathcal{E}} \left(f_{r,(uv)} + f_{r,(vu)}\right) \le c_v, \quad \forall v \in \mathcal{V}, \tag{2c}$$

$$f_{r,(uv)} \in \mathbb{N}, \quad \forall r \in \mathcal{R}, \forall u,v \in \mathcal{V}. \tag{2d}$$

However, finding an optimal solution to the problem in (2) is NP-hard as we will demonstrate in Theorem 1.

Theorem 1. *The load-balanced entanglement distribution problem in* (2) *is NP-hard.*

Proof. Due to space constraints, we refer interested readers to the Appendix in [17] for the detailed proof. ■

IV. Load-Balanced Entanglement Distribution

In this section, we present the load-balanced entanglement routing algorithm LBER and the adaptive swapping strategy AdapSwap.

A. Problem Decomposition

Since the load-balanced entanglement distribution problem is NP-hard, it is difficult to find the optimal solution in polynomial time. We decompose it into two subproblems: the *path finding problem* and the *quantum memory allocation problem* and solve them independently. By solving the two subproblems, we can obtain a feasible solution for the original problem efficiently.

Path finding problem. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, C^{node})$ and a request $r = (s, d, n_e)$, the goal is to find multiple paths from s to d for establishing E2E entanglements. In this paper, we adopt Yen's algorithm [18] to find the k-shortest loopless paths for load-balancing entanglement distribution.

Quantum memory allocation problem. After identifying the paths, the next step is to allocate quantum memory at each node across all paths while minimizing the maximum load at any node. In the following, we will describe the objective and the constraints of our quantum memory allocation problem.

Objective of quantum memory allocation. For each request, we compute the *flow rate vector* for the K paths, denoted as $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_K)$, where $\lambda_k \in \mathbb{N}$ represents the number of E2E entanglements established on path k. To achieve an E2E entanglement, each intermediate node must establish two elementary entanglements, storing a qubit from each entangled pair in their quantum memory. In contrast, the source and destination nodes require only one elementary entanglement. Accordingly, we define the memory utilization function $M(v, \lambda_k)$ for node v on a path $\mathcal{P}_k$ as follows:

$$M(v,\lambda_k) = \begin{cases} 2\lambda_k, & \text{if } v \in \mathcal{P}_k, \text{ and } v \notin \{s,d\}, \\ \lambda_k, & \text{if } v \in \{s,d\}. \end{cases} \tag{3}$$

To model the relationship between paths and nodes, we define the matrix $\mathbf{G} = [g_{kv}] \in \{0,1\}^{K\times|\mathcal{V}|}$, where $g_{kv} = 1$ if node v is on path k, and $g_{kv} = 0$ otherwise. Due to the probabilistic nature of entanglement swapping, additional elementary entanglements are required to compensate for potential failures. Here we introduce a scaling factor $\beta_{kv}^{\mathfrak{O}} \ge 1$ to represent the proportion of additional elementary entanglements required at node v on path k under the swapping strategy $\mathfrak{O}$, which depends on not only the paths but also the swapping strategies (discussed in Section IV-B). We denote the set of scaling factors at all the nodes on path k as $\beta_k^{\mathfrak{O}} = \{\beta_{k0}^{\mathfrak{O}}, \beta_{k1}^{\mathfrak{O}}, \cdots, \beta_{kh}^{\mathfrak{O}}\}$. Thus, the quantum memory allocated at node v for the current request is $\sum_{i=1}^{K} g_{kv}\mathcal{M}_v(\lambda_k\beta_{kv}^{\mathfrak{O}})$. Our objective is to minimize the maximum load among all

nodes, i.e., $\min_{\lambda_k \in \boldsymbol{\lambda}} \max_{v \in V} L_v(\boldsymbol{\lambda}) = \frac{\tau_v + \sum_{i=1}^{K} g_{kv} \mathcal{M}_v(\lambda_k \beta_{kv}^{\mathfrak{O}})}{c_v}$ as shown in (4a).

Constraints of quantum memory allocation. For each request, the total number of E2E entanglements established on all paths must equal the required number of requests n_e as defined in (4b), and the quantum memory usage at each node must not exceed its capacity as indicated in (4c).

In summary, the quantum memory allocation problem is formulated as follows:

$$\min_{\lambda_k \in \boldsymbol{\lambda}} \max_{v \in V} L_v(\boldsymbol{\lambda}) \tag{4a}$$

$$\text{s.t.} \sum_{k=1}^{K} \lambda_k = n_e, \tag{4b}$$

$$\tau_v + \sum_{k=1}^{K} g_{kv} \mathcal{M}_v(\lambda_k \beta_{kv}^{\mathfrak{O}}) \leq c_v, \forall v \in \mathcal{V}. \tag{4c}$$

Note that the optimization problem in (4) is still an integer programming (IP) problem, as λ_k must be integers, i.e., $\lambda_k \in \mathbb{N}$. By relaxing this constraint to $\lambda_k \geq 0$, one can transform the problem in (4) into a linear programming (LP) problem, which can be solved efficiently. However, the LP solution may not be feasible as λ_k must be integers. To address this, we apply a rounding technique and convert the flow rates $\lambda_k \geq 0$ into integers $\tilde{\lambda}_k \in \mathbb{N}$, while ensuring $\mathbb{E}[\tilde{\lambda}_k] = \lambda_k, \forall \lambda_k \in \boldsymbol{\lambda}$. Details of the technique are provided in Sec. IV-C.

B. Swapping Strategy Optimization

The swapping order along a path affects the scaling factor at each node, making the selection of an appropriate swapping order critical for load-balancing entanglement distribution. Let the entanglement swapping success probabilities of all nodes on a h-hop path $\mathcal{P}_k$ be denoted as $P^{swap} = \{p_0^{swap}, p_1^{swap}, \cdots, p_h^{swap}\}$, where $p_u^{swap} \in [0, 1]$ represents the entanglement swapping success probability at node u for $0 \leq u \leq h$. To describe the impact of the imperfections in the entanglement swapping operations, we define the swapping cost incurred by entanglement swapping at node u as q_u, where $q_u = \frac{1}{p_u^{swap}} \geq 1$ for $0 \leq u \leq h$. Since the source and destination nodes do not require any entanglement swapping, we set their swapping cost to 1, i.e., $q_0 = q_h = 1$.

We first analyze the widely used *Hop-by-Hop Swapping* strategy [19]–[21] and derive its scaling factor. Then, we introduce the *Adaptive Swapping* strategy, AdapSwap, which adaptively adjusts the swapping order to balance node loads on each path. To simplify the description, we refer to the link $(u-1, u)$ in the path $\mathcal{P}_k$ as link u in the following for $1 \leq u \leq h$.

• **Hop-by-hop swapping strategy.** In this strategy, entanglement swapping is performed sequentially along the path from source node to destination, progressively extending elementary entanglements into E2E entanglements. We define the scaling factor at node v on path k as,

$$\beta_{kv}^{\mathfrak{O}} = \frac{\prod_{u=v-1}^{h} q_u + \prod_{u=v}^{h} q_u}{2}, \tag{5}$$

which represents the average number of elementary entanglements required on link v and link $v+1$. We use Figure 3(b) to illustrate how $\beta_{kv}^{\mathfrak{O}}$ is calculated on a 4-hop path ($k = 1$, $h = 4$, $q_0 = 1$, and $q_4 = 1$). The left side of Figure 3(b) shows a hop-by-hop swapping order ($\mathcal{O} = \{1, 2, 3\}$). To generate one E2E entanglement, $q_1 q_2 q_3$ elementary entanglements are required on link 1 due to entanglement swapping at Node 1, 2, and 3. Similarly, for links 2, 3, and 4, the number of elementary entanglements required are $q_1 q_2 q_3$, $q_2 q_3$ and q_3, respectively. Node 1 needs to store one of the entangled qubits from link 1 and link 2. Therefore, $\beta_{11}^{\mathfrak{O}} = \frac{1}{2}(q_1 q_2 q_3 + q_1 q_2 q_3)$. Similarly, $\beta_{12}^{\mathfrak{O}} = \frac{1}{2}(q_1 q_2 q_3 + q_2 q_3)$.

This example indicates that nodes further from the destination node typically have a larger scaling factor and require more elementary entanglements, resulting in load imbalances along the path. Thus, inappropriate swapping strategies exacerbate these imbalances, reducing the network reliability and fault tolerance. To address this issue, we propose a dynamic programming-based strategy, AdapSwap, to optimize the swapping order to minimize the maximum scaling factor across nodes for each path.

• **Adaptive swapping strategy.** The core idea is to minimize the maximum scaling factor along each path. We utilize dynamic programming to solve this problem. Let $\mathbf{Z} = [z_{ij}] \in \mathbb{R}_+^{(h+1)\times(h+1)}$ represent the load matrix, where z_{ij} denotes the minimum maximum load across all nodes when establishing an entanglement between node i and j on path k, i.e., segment $\sigma_{i,j}^k$. This entanglement is achieved by first establishing entanglements between node i and an intermediate node m and between node m and node j, followed by an entanglement swapping at node m . Thus, the maximum load of the segment $\sigma_{i,j}^k$ is given by $\max\{z_{im}, z_{mj}, L_m\}$, where L_m is the load at node m . Then we scale the final load to $q_m \max\{z_{im}, z_{mj}, L_m\}$ with swapping cost q_m to account for imperfections in the swapping operation at node m. Let $\mathbf{E_m} = [e_{ij}^m] \in \mathbb{R}_+^{(h+1)\times(h+1)}$ represent the elementary entanglement count matrix for link m, where e_{ij}^m denote the number of elementary entanglements established on link m for segment $\sigma_{i,j}^k$. The load at node m is the total number of elementary entanglements on the two links connected to node m, i.e., $L_m = e_{im}^m + e_{mj}^{m+1}$. Finally, we have the recursive equation for computing the minimum maximum load as follows:

$$z_{ij} = \min_{m \in [i+1, j-1]} \{q_m \max\{z_{im}, z_{mj}, L_m\}\}. \tag{6}$$

Using Eq. (6), we first compute the minimum maximum load for generating segment σ_{im}^k and σ_{mj}^k, then recursively compute z_{ij} by selecting the minimum of $q_m \max\{z_{im}, z_{mj}, L_m\}$ for $z \in [i+1, j-1]$, where $q_m \max\{z_{im}, z_{mj}, L_m\}$ is the maximum load with node m as the intermediate node.

Algorithm 1 outline our *Adaptive Swapping* strategy. The algorithm iteratively computes the minimum maximum load over all link lengths, $l \in [1, h]$, along diagonal lines as shown in Lines 2-3. For single-hop links, the maximum load and the number of elementary entanglements established at the

Algorithm 1 Adaptive swapping on path k (AdapSwap)

Input: k-th path $\mathcal{P}_k$, path length h, and swapping cost set $q = \{q_0, q_1, \cdots, q_h\}$
Output: scaling factor set $\beta_k^{\mathfrak{O}}$ and swapping order $\mathcal{O}_k$

1: **Initialize:** $\mathbf{Z} = [0]_{(h+1)\times(h+1)}$, $\mathbf{E_m} = [0]_{(h+1)\times(h+1)}$ for $m \in [1, h]$ and $\mathbf{S} = [0]_{(h+1)\times(h+1)}$
2: **for** $l = 1, \cdots, h$ **do**
3: $\quad i \leftarrow 0$, $j \leftarrow i + l$
4: $\quad$ **while** $j \leq h$ **do**
5: $\quad\quad$ **if** $l = 1$ **then** $z_{ij} \leftarrow 1$, $e_{ij}^j \leftarrow 1$
6: $\quad\quad$ **else**
7: $\quad\quad\quad L_m \leftarrow e_{im}^m + e_{mj}^{m+1}$ for $m \in [i+1, j-1]$
8: $\quad\quad\quad z_{ij}, m^\star \leftarrow \min_{m=i+1}^{j-1}\{q_m \max\{z_{im}, z_{mj}, L_m\}\}$
9: $\quad\quad\quad e_{ij}^m \leftarrow q_{m^\star} e_{im^\star}^m$ for $m \in [1, m^\star]$
10: $\quad\quad\quad e_{ij}^m \leftarrow q_{m^\star} e_{m^\star j}^m$ for $m \in [m^\star + 1, l]$
11: $\quad\quad\quad s_{ij} \leftarrow m^\star$
12: $\quad\quad i \leftarrow i + 1$, $j \leftarrow j + 1$
13: $\mathcal{T} \leftarrow$ ConstructTree$(\mathbf{S}, 0, h)$ ▷ *See Algorithm 2*
14: $\mathcal{O}_k \leftarrow$ ReverseLevelOrder $(\mathcal{T})$
15: $\beta_{k0}^{\mathfrak{O}} \leftarrow \frac{e_{0h}^1}{2}, \beta_{kh}^{\mathfrak{O}} \leftarrow \frac{e_{0h}^h}{2}$
16: **for** $m = 1, \cdots, h-1$ **do**
17: $\quad \beta_{km}^{\mathfrak{O}} = \frac{e_{0h}^m + e_{0h}^{m+1}}{2}$
18: **return** $\beta_k^{\mathfrak{O}} = \{\beta_{k0}^{\mathfrak{O}}, \beta_{k1}^{\mathfrak{O}}, \cdots, \beta_{kh}^{\mathfrak{O}}\}, \mathcal{O}_k$

link are set to 1 without considering the swapping cost, as shown in Line 5. For longer paths, additional elementary entanglements are required to compensate for the losses incurred by entanglement swapping. The minimum maximum load and the optimal intermediate node for entanglement swapping are computed recursively using Eq. (6) as shown in Line 7-8. This ensures the optimal swapping order by traversing intermediate nodes that balance node loads across all nodes. Let $\mathbf{S} = [s_{ij}] \in \mathbb{N}^{(h+1)\times(h+1)}$ represent the swapping matrix, where s_{ij} denotes the intermediate node that performs entanglement swapping for generating segment σ_{ij}^k. Then, the number of elementary entanglements required for each link is updated according to the chosen swapping strategy and the optimal intermediate node $m^\star$ is recorded in the swapping matrix, i.e., $s_{ij} \leftarrow m^\star$, as shown in Lines 9-11. Once iterations are completed, the optimal swapping tree that identifies the swapping order is constructed using Algorithm 2. A reverse level-order traversal generates the optimal swapping order, as described in Lines 13-14. The scaling factor for each node is calculated as the average number of elementary entanglements on its adjacent links as shown in Lines 16-17. In the end, the optimal swapping order and the corresponding scaling factor for each node are determined for the quantum memory allocation problem in (4).

Algorithm 2 describes how to construct the swapping tree from the swapping matrix $\mathbf{S}$. As shown in Line 1, the E2E entanglement is divided into two shorter segments at node m, where entanglement swapping is performed. Then, a tree node is created for node m. The algorithm recursively finds the left and right child nodes until the E2E entanglement is divided into elementary entanglements as shown in Lines 3-4.

Algorithm 2 ConstructTree

Input: the swapping matrix $\mathbf{S}$, the source node i and the destination node j
Output: the root node $\mathcal{T}$ of swapping tree

1: $m \leftarrow s_{ij}$, $\mathcal{T} \leftarrow TreeNode(m)$
2: **if** $m = 0$ **then return** None
3: $\mathcal{T}.left \leftarrow$ ConstructTree$(\mathbf{S}, i, m)$
4: $\mathcal{T}.right \leftarrow$ ConstructTree$(\mathbf{S}, m, j)$
5: **return** $\mathcal{T}$

To illustrate the process of AdapSwap, we provide an example of a 4-hop path ($h = 4$) in Figure 4. The swapping cost at each node is shown below the node. Initially, the optimal solutions for paths of different lengths, i.e., $l = 1, 2, 3, 4$, are updated along the diagonal as shown in Figure 4(a). The solutions for one-hop links ($l = 1$) are updated to 1, as elementary entanglements are generated without entanglement swapping. For lengths greater than one (from $l = 2$ to $l = 4$), the solutions are recursively computed using equation (6). The minimum maximum load for generating one entanglement between Node 0 and Node 4 is represented by the upper right element of matrix $\mathbf{Z}$, i.e., $z_{04} = 2q_1q_2$. As the matrix $\mathbf{Z}$ is updated, the corresponding elements of matrix $\mathbf{E_m}$ are adjusted according to Lines 9-10. The optimal swapping decisions are then recorded in matrix $\mathbf{S}$, i.e., $s_{ij} \leftarrow m^\star$. Based on matrix $\mathbf{S}$, the swapping tree is constructed using Algorithm 2, and the reverse level order traversal produces the optimal swapping order as depicted in Figure 4(b). Finally, the scaling factor for each node is computed from $\mathbf{E_m}$, which represents the average number of elementary entanglements required on its two adjacent links, as illustrated in Figure 4(c).

C. LBER Algorithm

Algorithm 3 details the routing decision and swapping strategy computation when a request r arrives. By using Algorithm 1, we obtain the scaling factors on all paths $\mathcal{B}^{\mathfrak{O}} = \{\beta_1^{\mathfrak{O}}, \beta_2^{\mathfrak{O}}, \cdots, \beta_K^{\mathfrak{O}}\}$ as the input. As shown in Line 1, the path set $\mathfrak{P} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_K\}$ is first computed using Yen's algorithm [18], where $\mathcal{P}_k$ is the set of nodes along path k. The fractional solution $\boldsymbol{\lambda}$ is obtained by solving the quantum memory allocation problem (4) via linear programming [22], [23] as shown in Line 2. In Line 3, a feasible integer solution $\tilde{\boldsymbol{\lambda}}$ is then obtained by rounding the fractional solution $\boldsymbol{\lambda}$ using a Modified Dependent Rounding technique modified from [24]. This involves dividing $\boldsymbol{\lambda}$ into integer and fractional parts, i.e., $\boldsymbol{\lambda_I} = \{\lambda_{Ik} | \lambda_{Ik} = \lfloor \lambda_k \rfloor, k \in \mathcal{K}\}$ and $\boldsymbol{\lambda_F} = \{\lambda_{Fk} | \lambda_{Fk} = \lambda_k - \lfloor \lambda_k \rfloor, k \in \mathcal{K}\}$, and iteratively rounding the fractional parts to 0 or 1 as shown in Lines 4-10. Finally, the integer solution is constructed by combining the integer and rounded fractional parts, i.e., $\tilde{\boldsymbol{\lambda}} = \{\lambda_k | \lambda_k = \lambda_{Ik} + \lambda_{Fk}, k \in \mathcal{K}\}$.

D. Complexity Analysis

Let $C_{\max}$ denote the maximum quantum memory size at any node $v \in \mathcal{V}$. The time complexity of Algorithm 1 is $O(|\mathcal{V}|^3)$. In the dynamic programming step (Lines 2-12), the load and count matrices are computed in $O(|\mathcal{V}|^3)$ time. Constructing

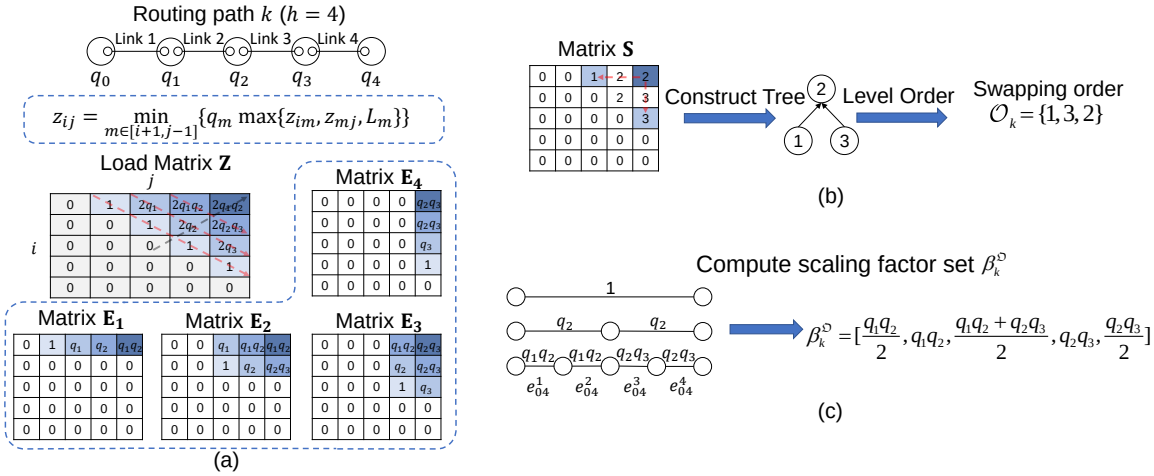


Fig. 4: An example of AdapSwap. (a) Compute the load matrix $\mathbf{Z}$ and count matrices $\mathbf{E_m}$ in diagonal order for a 4-hop path ($h = 4$), where the top-right element z_{04} is the optimal value; (b) Construct the swapping tree from the swapping matrix $\mathbf{S}$ and derive the swapping order via level order; (c) Calculate the scaling factor of each node based on $\mathbf{E_m}$.

Algorithm 3 Load-Balanced Entanglement Routing (LBER)

Input: the quantum network $\mathcal{G}$, the request r, and the scaling factor $\mathcal{B} = \{\beta_k^{\mathcal{O}}\}_{k\in\mathcal{K}}$
Output: the flow rate vector $\tilde{\boldsymbol{\lambda}}$

1: Find path set $\mathfrak{P}$ using Yen's algorithm [18] for r
2: Find $\boldsymbol{\lambda}$ by solving the relaxed LP problem in (4) using $\mathcal{B}$
3: $\tilde{\boldsymbol{\lambda}} \leftarrow$ MODDEPROUNDING($\boldsymbol{\lambda}$)
4: **function** MODDEPROUNDING($\boldsymbol{\lambda}$)
5: $\boldsymbol{\lambda_I}, \boldsymbol{\lambda_F} \leftarrow$ the integer and fractional parts of $\boldsymbol{\lambda}$
6: **while** exists k such that $0 < \lambda_{Fk} < 1$ **do**
7: Find distinct k, j, such that $0 < \lambda_{Fk}, \lambda_{Fj} < 1$
8: $x \leftarrow \min\{1-\lambda_{Fk}, \lambda_{Fj}\}$, $y \leftarrow \min\{\lambda_{Fk}, 1-\lambda_{Fj}\}$
9: $(\lambda_{Fk}, \lambda_{Fj}) = \begin{cases} (\lambda_{Fk}+x, \lambda_{Fj}-x), \text{ w.p.} \frac{y}{x+y}; \\ (\lambda_{Fk}-y, \lambda_{Fj}+y), \text{ w.p.} \frac{x}{x+y}; \end{cases}$
10: **return** $\tilde{\boldsymbol{\lambda}} = \{\lambda_k | \lambda_k = \lambda_{Ik} + \lambda_{Fk}, k \in \mathcal{K}\}$

the swapping tree and performing the level-order traversal take $O(|\mathcal{V}|)$ time. Similarly, computing the scaling factors (Line 16 to Line 17) takes $O(|\mathcal{V}|)$ time. Thus, the overall time complexity of Algorithm 1 is $O(|\mathcal{V}|^3)$.

For Algorithm 3, the time complexity is analyzed as follows. First, the path set with K paths is obtained using Yen's algorithm [18], with a complexity of $O(K|\mathcal{V}|(|\mathcal{E}| + |\mathcal{V}|\log|\mathcal{V}|))$. For each path, Algorithm 1 computes the optimal swapping order and scaling factors set $\beta_k^{\mathcal{O}}$, with a complexity of $O(K|\mathcal{V}|^3)$. Solving the relaxed linear programming problem in (4) using the interior point method takes $O(K^{3.5})$ time [23]. Finally, the MODDEPRPUNDING function rounds the fractional solution to an integer solution in $O(K)$ time. Therefore, the overall time complexity of Algorithm 3 is $O(K|\mathcal{V}|^3)$, compared to a brute-force approach with a complexity of $O((C_{\max}+1)^{|\mathcal{E}|})$. In terms of space complexity, Algorithm 3 requires $O(|\mathcal{V}|^3)$ space, as the sizes of the matrices $\mathbf{Z}$ and $\mathbf{E_m}$ in Algorithm 1 are both $O(|\mathcal{V}|^3)$.

Remark. *Algorithm 1 and Algorithm 3 are polynomial algorithms for the relaxed load-balanced entanglement distribution problem. However, the original problem in* (2) *is still NP-hard.*

V. EVALUATION

A. Evaluation Setup

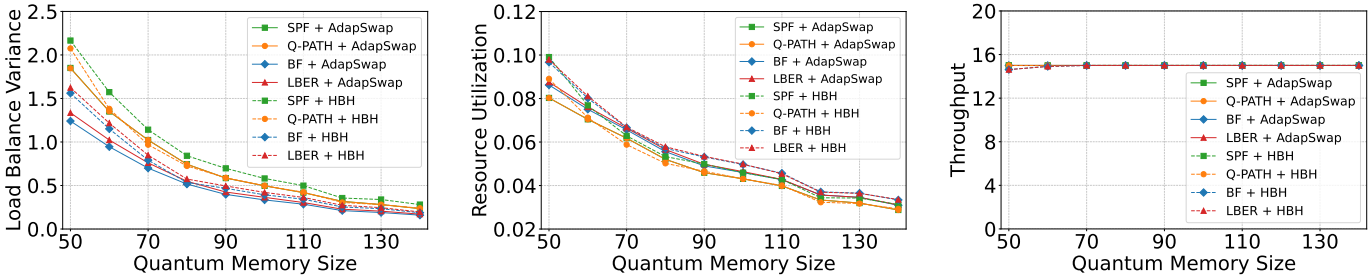
In this section, we conduct extensive experiments to evaluate the performance of LBER and AdapSwap in quantum networks. Simulations are run on a platform with a 12th-gen Intel Core i7-12700 2.10 GHz CPU, 16 GB RAM, and Windows 11 64-bit OS. According to [15], the state-of-the-art quantum processors can store up to 72 optical qubits, so we set the quantum memory of nodes following a Gaussian distribution $N(100, 3^2)$. Each experiment result is averaged over 500 runs. We compare LBER with the following routing algorithms:

- **SPF (Shortest Path First)**, which repeatedly selects the shortest path for E2E entanglements until the request is satisfied;
- **Q-PATH**, which is adapted from [12] and designed for fidelity-constraint entanglement routing for quantum networks under the *link capacity model*; We modified it to fit the *node capacity model* and exclude fidelity constraints for fair comparison;
- **BF (Brute Force)**, which is impractical for larger networks due to the high complexity of problem in (2) (e.g., with 8 links and 100 quantum memory, the solution space exceeds 10^{16}); We only apply it to problem in (4) for comparison.

In addition to AdapSwap, we evaluate the hop-by-hop (HBH) swapping strategy, resulting in 8 entanglement distribution schemes (4 routing algorithms with 2 swapping strategies). For example, LBER-AdapSwap represents AdapSwap applied to LBER, while LBER-HBH applies HBH to LBER.

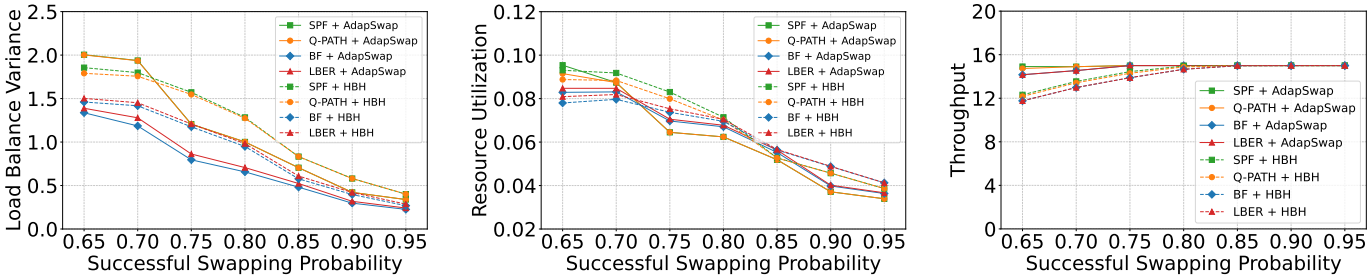
B. Computational Efficiency

To evaluate computational efficiency, we measure the runtime of each routing algorithm on random network topologies with different network scales generated using the Waxman



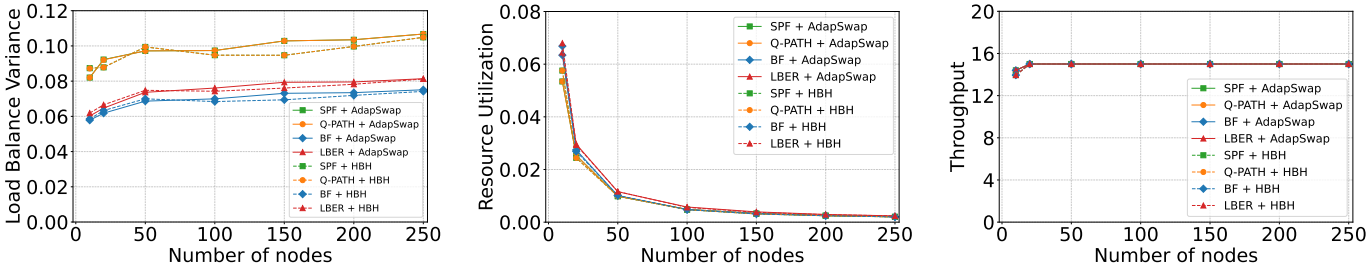
(a) Load balance variance (b) Network resource utilization (c) Throughput of request

Fig. 5: Network performance versus quantum memory size.



(a) Load balance variance (b) Network resource utilization (c) Throughput of request

Fig. 6: Network performance versus entanglement swapping success probability.



(a) Load balance variance (b) Network resource utilization (c) Throughput of request

Fig. 7: Network performance versus network scale.

Table II: Running time of algorithms

Network Scale	**LBER** (ms)	**SPF** (ms)	**Q-PATH** (ms)	**BF** (ms)
10	9.29	2.02	11.07	165.10
20	13.21	5.32	21.70	202.29
50	43.90	30.96	87.67	234.98
100	198.58	174.09	405.83	732.99
200	780.98	719.03	1187.02	1366.94
300	1881.65	1797.81	3660.37	4006.68

model [25]. The probability of a link between node u and node v is $p(u,v) = \kappa \exp \frac{-\mu(u,v)}{\mu_{max}\zeta}$, where $\mu(u,v)$ is the length between node u and v, μ_{max} is the maximum length between any two nodes, and $\kappa, \zeta \in (0,1]$ are parameters. The runtime results are shown in Table II. Consistent with our analysis, LBER has significantly lower complexity than BF. Specifically, LBER compute solutions within 800ms for networks with fewer than 200 nodes, well below the typical quantum memory qubit lifetime of 1.46s [26]. This ensures entanglement distribution decisions are made before decoherence occurs.

C. Network Performance

In this experiment, we use *load balance variance* to measure the load balance performance of different approaches. It is defined as the sum of squared deviations of each node's load from the average load, calculated as $\sum_{v\in\mathcal{V}}(L_v - \bar{L})^2$, where $\bar{L}$ is the average load across all nodes. We use the US backbone network topology [27] as adopted in [12] and consider 15 E2E entanglement requests between arbitrary source-destination pairs.

Impact of quantum memory size. Figure 5 compares the load balance variance, resource utilization, and throughput of the proposed algorithms with others, where the resource utilization is evaluated by the ratio of consumed to total quantum memory. As shown in Figure 5(a), LBER achieves a lower load balance variance compared to other algorithms, with a minimal gap to BF, indicating that LBER achieves near-optimal performance. AdapSwap consistently outperforms HBH in reducing load balance variance, demonstrating its

effectiveness in improving load balance. As quantum memory size increases, the performance gap between LBER and other algorithms narrows, as load balance variance scales with quantum memory size. Figure 5(b) shows that LBER consumes slightly more network resources for load-balancing entanglement distribution. The results in Figure 5(c) confirm that all algorithms successfully establish E2E entanglements for requests. In summary, LBER-AdapSwap achieves superior load balance performance while slightly increasing resource utilization, enhancing network robustness.

Impact of entanglement swapping success probability. Figure 6 evaluates the network performance under varying entanglement swapping success probabilities. As shown in Figure 6(a), LBER-AdapSwap consistently achieves the lowest load balance variance. The gap between AdapSwap and HBH narrows as the swapping success probability increases. This is because at lower swapping success probabilities, the impact of the swapping strategy is amplified due to the need for more pre-established elementary entanglements, but this effect diminishes as the success probability increases. Figure 6(c) shows that HBH fails to meet requests when the success rate is 0.65 due to severe load imbalance near source nodes. In contrast, AdapSwap balances load by determining optimal swapping orders, even with heterogeneous success probabilities.

Impact of network scale. We generate random topologies with varying node counts using the Waxman method [25]. As shown in Figure 7(a), the load balance variances of all algorithms increase with the number of nodes due to underutilized resources in the additional nodes. However, LBER-AdapSwap consistently achieves the lowest variance due to its careful load-balancing design. Figure 7(b) indicates a decline in resource utilization as the network scale grows, reflecting unused quantum memory in larger networks. Finally, all algorithms successfully meet the request as shown in Figure 7(c).

D. Network Robustness

We investigate the robustness of quantum networks for each entanglement distribution algorithm by evaluating the *request satisfaction ratio*, defined as the fraction of successfully satisfied requests. We exclude the results of HBH and apply AdapSwap to all routing algorithms. The experiment is conducted on four topologies: a Star and a Ring each consisting of 15 nodes, a 3 by 3 grid Mesh topology, and the US Backbone topology with 39 nodes. We generate 6 requests with arbitrary source-destination pairs, where required E2E entanglements follow a uniform distribution, i.e., $n_e \in U(20, 50)$. As shown in Figure 8, LBER performs similarly to other algorithms in the Star topology, where the central node is used for all requests and becomes the bottleneck. In the other three topologies, LBER achieves a 7.73%∼31.70% higher request satisfaction ratio by leveraging multipath transmission to balance the load in the network and reduce the probability of single-point failure.

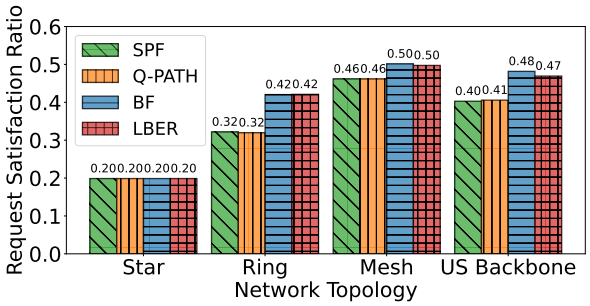


Fig. 8: Request satisfaction ratio under different network topologies.

VI. Related Work

Quantum networks are central to quantum information technology, supporting a wide range of applications [28]. A key challenge is entanglement distribution, which involves routing decisions and entanglement swapping strategies.

Routing decision. To establish E2E entanglement, routing algorithms must identify optimal paths and determine flow rates for each path. Depending on whether entanglement purification is involved, entanglement routing is categorized into *simple routing* and *routing with purification*. Pant et al. [14] proposed a greedy algorithm for path selection with the fewest hops. Q-CAST [7] used expected throughput as a routing metric, and Zhao [8] maximizes E2E entanglements through redundant routing. Nguyen [29] used linear programming to maximize the entangled routing rate while ensuring fidelity. Gu et al. [30] explored the trade-off between entanglement distribution rate (EDR) and fidelity, proposing an approximation scheme to maximize worst-case fidelity under EDR constraints. A grey-box routing algorithm in [31] used end-to-end fidelity estimations to find higher fidelity paths without detailed per-node/link information. Other works [10]–[12] also incorporated entanglement purification into routing decisions. [10] enables purification before routing but ignores fidelity degradation from entanglement swapping. To address this, Li et al. [12] proposed a greedy algorithm that optimizes both routing and purification, considering fidelity loss during swapping. However, these approaches often overuse resources along selected paths, increasing the risk of network failure, and highlighting the need for load-balancing entanglement routing algorithms to enhance network robustness.

Entanglement swapping strategy. Swapping strategies are classified as *memory-less swapping* and *memory-based swapping* depending on whether quantum memory is used. Memory-less swapping is generally impractical due to stringent synchronization requirements, while memory-based swapping offers better performance by relaxing the synchronization constraints. Various memory-based swapping schemes have been proposed [19], [32]–[34] to optimize entanglement distribution. Hop-by-hop swapping strategy performs swapping sequentially but introduces significant delays on long paths. The nested swapping strategy [32] optimally merges swapping in parallel but is restricted to paths of lengths $2^i + 1$ ($i \geq 1$). To overcome this, concurrent swapping [19]

introduces an asymmetric merging order for the arbitrary-length path. Additionally, Wang et al. [35] analyzed flow loss in sequential swapping, while Islam et al. [36] applied reinforcement learning for proactive swapping. However, existing swapping strategies struggle in heterogeneous scenarios where swapping success probabilities vary across nodes and overlook load-balancing across nodes.

Unlike previous studies, our approach integrates routing decisions and swapping strategies to achieve load-balanced entanglement distribution, significantly enhancing the robustness of quantum networks.

VII. Conclusion

In this paper, we address the load-balanced entanglement distribution problem to enhance the robustness of the quantum network. Given its NP-hardness, we decompose it and separately optimize routing decisions and the swapping strategy for each path. We propose a load-balanced entanglement routing (LBER) algorithm to optimize routing decisions efficiently and design an adaptive swapping strategy, AdapSwap, to compute the optimal swapping order along each path. Extensive experiment results demonstrate that our approach achieves lower load balance variance and a 7.73%∼31.70% higher request satisfaction ratio.

References

[1] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical computer science*, vol. 560, pp. 7–11, 2014.

[2] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai *et al.*, "Satellite-based entanglement distribution over 1200 kilometers," *Science*, vol. 356, no. 6343, pp. 1140–1144, 2017.

[3] J. I. Cirac, A. Ekert, S. F. Huelga, and C. Macchiavello, "Distributed quantum computation over noisy channels," *Physical Review A*, vol. 59, no. 6, p. 4249, 1999.

[4] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, 1982.

[5] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, "Quantum repeaters: the role of imperfect local operations in quantum communication," *Physical Review Letters*, vol. 81, no. 26, p. 5932, 1998.

[6] Y. Zeng, J. Zhang, J. Liu, Z. Liu, and Y. Yang, "Multi-entanglement routing design over quantum networks," in *Proc. of IEEE INFOCOM 2022*, 2022, pp. 510–519.

[7] S. Shi, X. Zhang, and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," *IEEE/ACM Transactions on Networking*, 2024.

[8] Y. Zhao and C. Qiao, "Redundant entanglement provisioning and selection for throughput maximization in quantum networks," in *Proc. of IEEE INFOCOM 2021*, 2021, pp. 1–10.

[9] L. Yang, Y. Zhao, H. Xu, and C. Qiao, "Online entanglement routing in quantum networks," in *Proc. of IEEE IWQoS 2022*, 2022, pp. 1–10.

[10] C. Li, T. Li, Y.-X. Liu, and P. Cappellaro, "Effective routing design for remote entanglement generation on quantum networks," *npj Quantum Information*, vol. 7, no. 1, p. 10, 2021.

[11] Y. Zhao, G. Zhao, and C. Qiao, "E2e fidelity aware routing and purification for throughput maximization in quantum networks," in *Proc. of IEEE INFOCOM 2022*, 2022, pp. 480–489.

[12] J. Li, M. Wang, K. Xue, R. Li, N. Yu, Q. Sun, and J. Lu, "Fidelity-guaranteed entanglement routing in quantum networks," *IEEE Transactions on Communications*, vol. 70, no. 10, pp. 6748–6763, 2022.

[13] A. Farahbakhsh and C. Feng, "Opportunistic routing in quantum networks," in *Proc. of IEEE INFOCOM 2022*, 2022, pp. 490–499.

[14] M. Pant, H. Krovi, D. Towsley, L. Tassiulas, L. Jiang, P. Basu, D. Englund, and S. Guha, "Routing entanglement in the quantum internet," *npj Quantum Information*, vol. 5, no. 1, p. 25, 2019.

[15] S. Zhang, J. Shi, Z. Cui, Y. Wang, Y. Wu, L. Duan, and Y. Pu, "Realization of a programmable multipurpose photonic quantum memory with over-thousand qubit manipulations," *Physical Review X*, vol. 14, no. 2, p. 021018, 2024.

[16] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels," *Physical review letters*, vol. 70, no. 13, p. 1895, 1993.

[17] Anonymous, "Achieving load-balancing entanglement distribution in quantum networks (technical report)," Tech. Rep., 2025. [Online]. Available: https://anonymous.4open.science/r/LBER-Appendix-769D

[18] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[19] Z. Wang, J. Li, K. Xue, D. S. Wei, R. Li, N. Yu, Q. Sun, and J. Lu, "An efficient scheduling scheme of swapping and purification operations for end-to-end entanglement distribution in quantum networks," *IEEE Transactions on Network Science and Engineering*, 2023.

[20] L. Chen, K. Xue, J. Li, N. Yu, R. Li, J. Liu, Q. Sun, and J. Lu, "A heuristic remote entanglement distribution algorithm on memory-limited quantum paths," *IEEE Transactions on Communications*, vol. 70, no. 11, pp. 7491–7504, 2022.

[21] A. Abane, M. Cubeddu, V. S. Mai, and A. Battou, "Entanglement routing in quantum networks: A comprehensive survey," *arXiv preprint arXiv:2408.01234*, 2024.

[22] H. Karloff, *Linear programming*. Springer Science & Business Media, 2008.

[23] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. of ACM STOC 1984*, 1984, pp. 302–311.

[24] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, "Dependent rounding and its applications to approximation algorithms," *Journal of the ACM*, vol. 53, no. 3, pp. 324–360, 2006.

[25] B. M. Waxman, "Routing of multipoint connections," *IEEE journal on selected areas in communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[26] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpundefineddek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, and S. Wehner, "A link layer protocol for quantum networks," in *Proc. of ACM SIGCOMM 2019*, 2019, pp. 159–173.

[27] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[28] Z. Li, K. Xue, J. Li, L. Chen, R. Li, Z. Wang, N. Yu, D. S. Wei, Q. Sun, and J. Lu, "Entanglement-assisted quantum networks: Mechanics, enabling technologies, challenges, and research directions," *IEEE Communications Surveys & Tutorials*, 2023.

[29] T. N. Nguyen, D. H. Nguyen, D. H. Pham, B.-H. Liu, and H. N. Nguyen, "Lp relaxation-based approximation algorithms for maximizing entangled quantum routing rate," in *Proc. of IEEE ICC 2022*, 2022, pp. 3269–3274.

[30] H. Gu, Z. Li, R. Yu, X. Wang, F. Zhou, J. Liu, and G. Xue, "Fendi: Toward high-fidelity entanglement distribution in the quantum internet," *IEEE/ACM Transactions on Networking*, 2024.

[31] V. Kumar, C. Cicconetti, M. Conti, and A. Passarella, "Routing in quantum networks with end-to-end knowledge," *arXiv preprint arXiv:2407.14407*, 2024.

[32] R. Van Meter, T. D. Ladd, W. J. Munro, and K. Nemoto, "System design for a long-line quantum repeater," *IEEE/ACM Transactions On Networking*, vol. 17, no. 3, pp. 1002–1013, 2008.

[33] M. Ghaderibaneh, C. Zhan, H. Gupta, and C. Ramakrishnan, "Efficient quantum network communication using optimized entanglement swapping trees," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–20, 2022.

[34] K. Goodenough, D. Elkouss, and S. Wehner, "Optimizing repeater schemes for the quantum internet," *Physical Review A*, vol. 103, no. 3, p. 032610, 2021.

[35] Z. Wang, J. Li, Z. Li, L. Chen, N. Yu, Q. Sun, and J. Lu, "Efficient routing design based on entanglement flow loss effect in quantum networks," in *Proc. of IEEE QCNC 2024*, 2024, pp. 9–16.

[36] T. Islam, M. Arifuzzaman, and E. Arslan, "Reinforcement learning based proactive entanglement swapping for quantum networks," in *Proc. of IEEE QCNC 2024*, 2024, pp. 135–142.

APPENDIX A
PROOF OF THEOREM 1

Proof. To demonstrate that the load-balanced entanglement distribution problem is NP-hard, we provide a reduction from the well-known NP-hard Boolean satisfiability problem (SAT) to an instance of this problem. We begin by introducing the SAT problem.

Boolean satisfiability problem (SAT): Let $\mathcal{X} = \{x_1, x_2, \cdots, x_N\}$ denote a set of $\mathcal{N} = \{1, \cdots, N\}$ boolean variables, where $x_n \in \{0, 1\}$. Using logical OR ($\vee$) and NOT ($\neg$) between multiple Boolean variables, we construct the clause $A_j = t_{j1} \vee t_{j2} \vee \cdots \vee t_{jn}$, where $t_{ji} = x_i$ or $\neg x_i$. Then the set of $\mathcal{J} = \{1, \cdots, J\}$ clauses is denoted as $\mathcal{A} = \{A_1, \cdots, A_J\}$. We define the Boolean formula Φ by taking the logical AND ($\wedge$) of all clauses, $\Phi = A_1 \wedge A_2 \wedge \cdots \wedge A_J$. The goal of the SAT problem is to determine whether there exists an assignment of the variables $\mathcal{X}^\star$ that makes the formula Φ true.

Reduction from 3-SAT to an instance of the load-balanced entanglement distribution problem. Given an arbitrary instance of the 3-SAT problem $\Phi = \wedge_{j=1}^{J} A_j$, we construct an equivalent instance of the load-balanced entanglement distribution problem with the same optimal solution. In the instance, we assume the entanglement swapping success probabilities at all nodes are 1, i.e., $p_v^{swap} = 1$ for $v \in \mathcal{V}$, where $\mathcal{V}$ is the set of quantum node, meaning all swapping strategies have the same load balance performance. We build the quantum network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, C^{node})$ based on the Boolean formula Φ as shown in Figure 1, where $\mathcal{E}$ is the set of quantum link and C^{node} is the set of node capacity. First, we create quantum nodes $\mathcal{V}$. For each Boolean variable x_i, we create two quantum nodes: v_{x_i} and $v_{\neg x_i}$. For each clause A_j, we create one quantum node v_{A_j}, along with two additional nodes for the source s_j and destination d_j, representing a request in the quantum network. Next, we create quantum links $\mathcal{E}$. We connect all source node s_j (for $j \in \mathcal{J}$) to the nodes v_{x_n} and $v_{\neg x_n}$ (for $n \in \mathcal{N}$), and connect each node v_{A_j} to the corresponding destination node d_j with $j \in \mathcal{J}$. According to each clause $A_j = t_{j1} \vee t_{j2} \vee t_{j3}, j \in \mathcal{J}$, we create links from the corresponding nodes v_{x_n} or $v_{\neg x_n}$ (for $n \in \mathcal{N}$), to node v_{A_j} as the gray area shown in Figure 1. For example, let $t_{j1} = x_1, t_{j2} = \neg x_2$, $t_{j3} = x_3$, for $A_j = x_1 \vee \neg x_2 \vee x_3$, we connect the node v_{A_j} to node v_{x_1}, $v_{\neg x_2}$ and v_{x_3} respectively. We then set the request $r = (s_j, d_j, n_e)$ between the source node s_j and the destination node d_j, with the required end-to-end entanglements $n_e = 3$, since each clause has 3 Boolean variables. Besides, the quantum memory size for both s_j and d_j is 3, and the quantum memory size at each clause node v_{A_j} is 6, as node v_{A_j} needs to connect both the node of the Boolean variable on the left and the destination node on the right. Let T_{x_n} and $T_{\neg x_n}$ represent the total number of occurrences of the Boolean variable x_n and $\neg x_n$ in the formula Φ. The quantum memory sizes of nodes v_{x_n} and $\neg x_n$ are $2T_{x_n}$ and $2T_{\neg x_n}$, respectively, for $n \in \mathcal{N}$. If we can find a solution to the load-balanced entanglement distribution problem in the constructed quantum network, as shown in Figure 1, that satisfies all requests while minimizing the maximum load, we can derive an assignment $\mathcal{X}^\star$ that satisfies the Boolean formula Φ. Since any instance of the SAT problem can be transformed into an instance of the load-balanced entanglement distribution problem in polynomial time, and the SAT problem is NP-hard, it follows that the load-balanced entanglement distribution problem is also NP-hard.

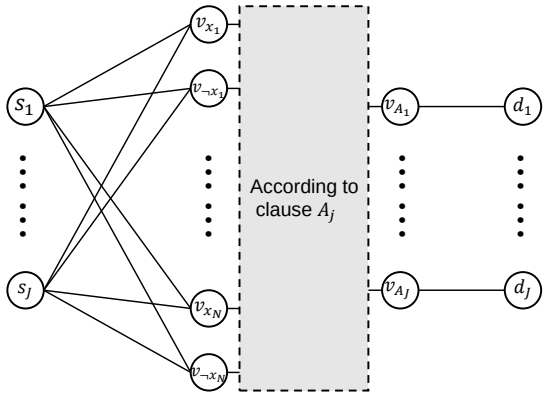


Fig. 1: The constructed quantum network

■