Labubu: Layer-buffered Bundled Optimization for Efficient Remote Gate Scheduling in Distributed Quantum Computing

Xu Xu, Yu Liu, Yingling Mao, and Yuanyuan Yang, *Life Fellow, IEEE*

***Abstract*—Distributed Quantum Computing (DQC) expands qubit capacity by interconnecting multiple Quantum Processing Units (QPUs), but remote gate execution introduces significant entanglement overhead. In this paper, we investigate the Remote Gate Scheduling problem in DQC (RGS-DQC) under a hybrid Telegate and Teledata model, provide a formal formulation, and establish its NP-hardness. To address this challenge, we propose *LABUBU*, a layer-buffered bundled optimization framework that integrates coordinate-wise pruned greedy refinement with bounded perturbation under QPU capacity constraints while maintaining linear complexity per iteration. Extensive simulations on both structured Quantum Fourier Transform circuits and unstructured random circuits show that Labubu consistently reduces entanglement cost compared with Telegate-SA, Telegate-RD, Teledata-ZS, and the competitive GateCover baseline. Experiments on QEC encoded circuits further confirm its potential for large scale fault tolerant distributed quantum computing.**

I. Introduction

Quantum computing [1] is an emerging computational paradigm. Using principles of quantum mechanics, such as superposition, entanglement, and quantum interference, it can outperform classical computing for certain problems. For instance, Shor's algorithm [2], a groundbreaking quantum algorithm, achieves polynomial-time integer factorization, providing an exponential speedup over classical algorithms. This capability presents a significant challenge to cryptographic systems such as RSA [3], a classical encryption algorithm that relies on the difficulty of integer factorization for its security. With the rapid development of quantum hardware, the capacity of Quantum Processing Units (QPUs) has now reached the scale of hundreds, and in some cases, even thousands of qubits. For example, IBM's Heron QPU is equipped with 156 physical qubits, and its newest Condor QPU has 1, 121 physical qubits [4].

However, even with such advances, executing practical quantum circuits on a single QPU remains impractical. So far, the largest number that has been experimentally factored by Shor's algorithm with a physical system is 21 [5], which remains far from practical applications. For example, factoring a large number approximately equal to 2^{1024} using Shor's algorithm requires over $5,000$ perfect logical qubits [6]. This challenge is further aggravated by the inherent noise in current QPUs, which renders quantum computing results deviate significantly from the ideal results. To achieve reliable quantum computing, Quantum Error Correction (QEC) [7] which encodes a single logical qubit using multiple physical qubits, is usually used to suppress noise. Although quantum circuits gain more robustness against noise with QEC, this encoding significantly increases the overall circuit size, making quantum circuit execution even harder. Therefore, existing QPUs with limited qubits are insufficient for real-world applications.

Distributed Quantum Computing (DQC) [8], [9] provides a scalable solution for implementing practical quantum circuits by interconnecting multiple QPUs. By doing so, it provides a promising path forward for overcoming the capacity bottleneck and laying the groundwork for large-scale quantum computing. DQC involves mapping logical qubits of a quantum circuit to physical qubits distributed across different QPUs and executing quantum gate operations on these physical qubits. Quantum gates whose qubits are mapped to the same QPU are known to be local quantum gates. In terms of local quantum gates, there have been sufficient research on the implementation of local gates inside a single QPU, and can be implemented with high fidelity. In contrast, due to the physical isolation of QPUs, the remote quantum gate (i.e., the quantum gates in which qubits involved are located in different QPUs), cannot be easily executed by directly applying the same techniques for local gates. Therefore, although DQC has the potential to scale up to larger circuits, implementing remote gates in DQC presents a significant challenge.

To address the challenge of implementing remote quantum gates, quantum entanglement plays a crucial role. As a unique phenomenon in quantum mechanics, quantum entanglement enables spatially separated qubits to interact and influence each other, forming the foundation of quantum communication. Three main entanglement-based protocols are commonly considered: Telegate, Teledata, and Cat-entanglement. Telegate directly executes remote gates between QPUs, while Teledata teleports qubits from one QPU to another to make remote gates local. However, Cat-entanglement, which allows temporary qubit copying across multiple QPUs, is impractical due to its sensitivity to local gates [10] and the frequent local operations required by gate decomposition. Therefore, this work focuses on the Telegate and Teledata protocols.

While Telegate and Teledata protocols expand distributed quantum capabilities, they introduce significant challenges. Both protocols consume pre-generated entanglement and re-

X. Xu, Y. Mao, and Y. Yang are with the Department of ECE, Stony Brook University {xu.xu, yingling.mao, yuanyuan.yang}@stonybrook.edu. Y. Liu is with the Department of Computing at Hong Kong Polytechnic University (yu-y.liu@polyu.edu.hk).

This work was supported in part by the National Science Foundation under grant numbers CNS-2231040 and CNS-2403202.

quire additional quantum gates, creating substantial resource overhead that scales with circuit complexity. In particular, entanglement generation suffers from low success rates, requiring thousands of attempts per successful entangled pair. As estimated in [11], entanglement generation is approximately 300 times more costly than local gate operations. Therefore, efficiently scheduling remote quantum gates to minimize entanglement consumption in DQC is a critical challenge in the field.

In this paper, we address the Remote Gate Scheduling problem for Distributed Quantum Computing (RGS-DQC) to minimize entanglement usage. Unlike existing approaches in the literature [12], which use Telegate solely while maintaining the existing qubit mapping, Teledata allows qubits to be dynamically remapped to other QPUs during the execution of a quantum circuit. This capability enables dynamic adjustments to the qubit mapping, thereby reducing remote two-qubit operations. However, changing the qubit mapping through Teledata incurs additional costs. To address the challenges of RGS-DQC, we propose *LABUBU*, a two-phase algorithm that incorporates both Telegate and Teledata. *LABUBU* integrates greedy search for local optima with a tailored perturbation mechanism to escape them and explore higher-quality solutions, and is further enhanced by a layer-buffered mapping update method.

This paper contributes to RGS-DQC in several significant ways. Firstly, we model the remote gate scheduling problem under a hybrid strategy that jointly considers both Telegate and Teledata approaches. Secondly, we propose an efficient algorithm for this problem, which achieves linear time complexity. We then evaluate the algorithm on distributed quantum systems of varying scales and different quantum circuits, comparing it against multiple benchmark algorithms. The results demonstrate that the proposed *LABUBU* algorithm consistently outperforms existing benchmark algorithms. Finally, to further assess its effectiveness and practicality, we conduct simulations on QEC-encoded quantum circuits with QPU capacities comparable to real-world devices. These experiments show clear advantages over circuits executed without remote gate scheduling, and specifically reveal that while *LABUBU* performs better for certain circuit widths, beyond a threshold width its variant achieves superior performance.

The remainder of this paper is organized as follows. Section II provides an overview of related works. Section III discusses preliminary concepts in quantum computing. Section IV introduces the system model and formulates the gate scheduling problem. Section V presents our proposed algorithm. Section VI presents the simulation results. Finally, Section VII concludes the paper.

II. Related Works

Distributed quantum computing has emerged as a promising approach for enabling large-scale quantum circuits beyond the capacity of a single quantum processor [8]. A fundamental challenge in DQC lies in establishing and efficiently utilizing entanglement across multiple quantum processing units (QPUs) [13], [14]. Remote quantum gates between qubits residing on different QPUs require the consumption of shared Bell states, and extensive prior work has investigated link-layer entanglement generation using a variety of quantum technologies [15]–[17]. For example, Stephenson *et al.* demonstrated deterministic Bell-state generation for trapped-ion qubits at an average rate of 182 Hz [15]. In our prior work, we proposed a non-blocking Quantum Multistage Switching Network (QMSN) architecture that enables remote entanglement establishment between arbitrary pairs of QPUs at uniform cost without path contention; building upon this network model, the present work focuses on mapping and remote gate scheduling in distributed quantum computing.

Given the high cost of remote entanglement, a central problem in DQC is to determine an effective mapping from logical qubits to QPUs so as to minimize remote operations. Mao *et al.* studied the qubit-mapping problem under different network topologies when only the TeleGate protocol is available, proposing a multistage hybrid algorithm combining local search and simulated annealing to obtain near-optimal mappings [12]. Davarzani *et al.* converted quantum circuits into bipartite graphs and applied dynamic programming to partition them into K subcircuits, thereby reducing inter-partition communication cost [18]. By exploiting cat-entanglement for CZ gates, the authors of [19] proposed a two-step heuristic algorithm that allows qubits to be logically shared across QPUs and demonstrated good performance on random circuits; however, the qubit mapping remains static throughout circuit execution.

In contrast, several works have investigated dynamic qubit remapping enabled by TeleData operations. The authors of [10] proposed two algorithms that adapt the qubit mapping during execution to reduce entanglement consumption. The "Local-Best" algorithm greedily selects target QPUs based on near-term benefits, while the "Zero-Stitching" algorithm identifies zero-cost subcircuits and stitches them together via dynamic programming. Their results show that "Local-Best" performs well on random circuits, whereas "Zero-Stitching" is particularly effective for structured circuits such as the quantum Fourier transform (QFT).

Ferrari *et al.* proposed a modular quantum compilation framework that incorporates a remote gate scheduler capable of interleaving TeleGate and TeleData operations [20]. The scheduler processes the circuit in a gate-by-gate manner and evaluates feasible remote operations using a cost function that accounts for EPR-pair consumption, gate coverage, and execution delay normalized by qubit decoherence time. This strategy demonstrates that TeleData can significantly reduce EPR consumption for circuits with repeated remote interactions, such as QFT, while offering limited benefit for circuits with sparse non-local dependencies, such as variational quantum eigensolver (VQE). However, scheduling decisions are made locally on a per-gate basis and do not explicitly consider layer-level coordination or long-term mapping evolution under capacity constraints.

Quantum error correction (QEC) has also attracted considerable attention for suppressing noise in quantum systems. Zhu *et al.* investigated mapping and scheduling techniques for QEC-encoded circuits within a single QPU [21], while Qiao *et al.* proposed a distributed QEC scheme in which physical qubits across multiple QPUs jointly encode a single logical qubit [22]. To the best of our knowledge, however, no prior work has

systematically studied the challenges of mapping and scheduling *QEC-encoded circuits* in a distributed quantum computing setting.

III. Quantum Computing Preliminaries

A. Logical and Physical Qubit

In quantum computing, just as the bit is the fundamental unit in classical computing, the qubit (quantum bit) serves as the foundational unit. Logical and physical qubits are essential concepts in theoretical computation and practical implementation, respectively. Logical qubits are idealized qubits used for theoretical analysis in quantum computing. They are assumed to be free from noise and physical constraints, such as decoherence, providing a perfect high-level abstraction of qubits for computation. In contrast, physical qubits are the actual hardware implementations of logical qubits, realized through various approaches such as trapped ions [23], superconducting circuits [24], NV centers [25], and neutral atoms [26]. Unlike their logical counterparts, physical qubits are subject to environmental noise, decoherence, and other real-world imperfections [27]. Since physical qubits are imperfect and prone to errors [28], error correction codes are used to encode a logical qubit using multiple physical qubits [29], [30], ensuring reliable quantum computation.

Depending on their intrinsic characteristics, physical qubits may have different functional roles and can be further categorized into data qubits and communication qubits. Data qubits typically offer longer coherence times, making them suitable for storing logical qubits. In contrast, communication qubits are less suited for storage, but they are well-suited for establishing entanglement across QPUs [31]. In this work, QPU capacity refers to the number of data qubits available for storing logical qubits and computation, excluding communication qubits used for remote entanglement generation. We assume that a sufficient number of communication qubits are available. For example, in SiV-based platforms [31], data qubits and communication qubits naturally appear in pairs, which is adequate for entanglement generation. We denote logical qubits by Q_l, data qubits by Q_d and communication qubits by Q_c.

B. Quantum Gate and Quantum Circuit

Quantum gates are quantum operations applied on qubits and are fundamental building blocks of quantum circuits. Based on the number of qubits involved, quantum gates are categorized into single-qubit gates, two-qubit gates, and multi-qubit gates.

A quantum circuit is a concrete representation of a quantum algorithm consisting of a set of qubits and a sequence of quantum gates. A quantum computation is completed by executing a quantum circuit and measuring the resulting quantum states, which provide the final output. Quantum gates that are executed simultaneously form a layer of a quantum circuit, which represents a time-slot in the overall circuit execution where operations are applied on different qubits in parallel. Fig. 1(a) shows an example of a quantum circuit with 16 layers and the quantum gates in the same layer are implemented in parallel.

A quantum circuit is characterized by three key parameters:

- **Width**: The width represents the total number of qubits used in the quantum circuit.
- **Depth**: The depth indicates the number of time slots in the quantum circuit, i.e., the total number of circuit layers.
- **Size**: The size refers to the total number of quantum gates used in the circuit. Since this work focuses on two-qubit gates, the circuit size refers to the number of two-qubit gates unless otherwise specified.

For the circuit in Fig. 1(a), it has a width of 3, a depth of 16, and a size of 9. Quantum circuits provide a structured framework for implementing quantum algorithms and are essential for realizing quantum computation in practice.

According to the Quantum Gate Universality Theorem [32], any quantum gate can be decomposed into single-qubit gates and two-qubit Controlled-NOT (CNOT) gates. This decomposition is of significant importance for the practical realization of quantum computation, as single-qubit gates and CNOT gates are relatively easy to implement in multi-qubit quantum systems. Therefore, in this work, we focus on quantum circuits that have already been fully decomposed into single-qubit and CNOT gates, facilitating practical implementation and aligning with the constraints of current quantum computing systems.

C. Quantum Entanglement

Quantum entanglement is a unique and fundamental phenomenon in quantum mechanics, where the states of entangled qubits become inseparably linked. In an entangled state, the state of each qubit cannot be described independently of the other's, regardless of their spatial separation. For two qubits, their maximally entangled states are referred to as **Bell states** or **EPR pairs** [1]. Bell states, expressed in the Dirac notation, are as follows:

$$|\phi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle), \quad |\psi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle).$$

These entangled states connecting two separate quantum systems are central to the Telegate and Teledata protocols for remote gate operations in DQC.

D. Quantum Processing Unit

QPU is the core computational component of a quantum computer, responsible for executing quantum circuits. Within a single QPU, the implementation and optimization of quantum gates have been extensively studied, leading to significant advancements in gate fidelity and system stability. Since intra-QPU quantum operations are well understood, this paper focuses on the more challenging domain of quantum gates across distinct QPUs, often referred to as **remote quantum gates**. Remote gates are essential for enabling DQC, where quantum circuits span across physically isolated QPUs. These gates leverage quantum communication techniques, such as quantum entanglement, to enable interactions between qubits located on different QPUs, overcoming the limitations of single-QPU systems and paving the way for scalable quantum computation.

E. Remote CNOTs Implementation

In DQC, logical qubits within a quantum circuit are distributed across different QPUs. While this significantly enhances overall qubit capacity, it introduces a critical challenge: qubits participating in the same CNOT gate may reside in different QPUs, which results in remote CNOT gates. In contrast to a local CNOT gate that acts between qubits located within the same QPU, remote CNOT gates cannot be directly implemented through internal QPU operations, posing significant challenges to DQC.

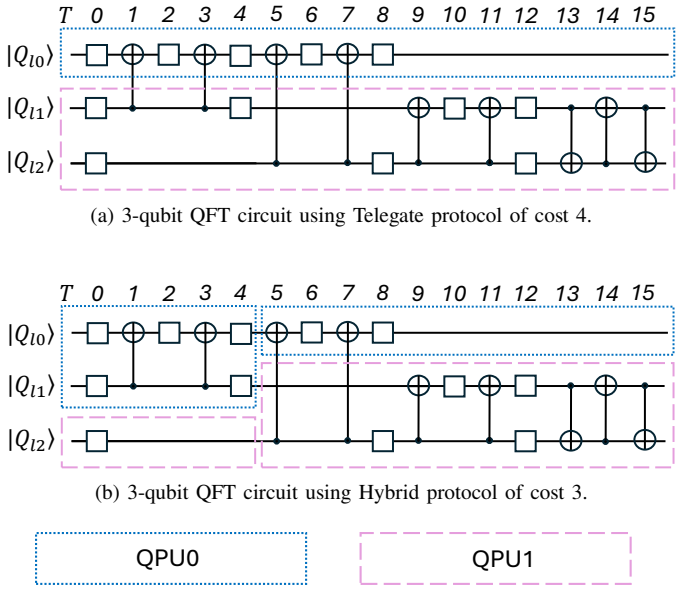
Although remote gates are more challenging to implement than local gates due to the physical isolation between QPUs, quantum entanglement provides an indirect mechanism for realizing remote CNOT operations. Leveraging entangled quantum states, two distinct protocols have been developed to enable remote quantum gates. Fig. 2 is the diagram for the two remote gate protocols. In this diagram, two logical qubits Q_{l0} and Q_{l1} are mapped onto data qubits Q_{d0} and Q_{d1}, which lie on different QPUs, respectively. A Bell state is shared between communication qubits Q_{c0} and Q_{c1}, with which a remote gate can be implemented through the following routine:

- **Telegate:** Fig. 2(a) illustrates the realization of a CNOT gate between logical qubits Q_{l0} and Q_{l1} using the Telegate protocol. First, each QPU performs a local CNOT gate, and a Hadamard gate is applied to Q_{c1}. Measurements are then performed on Q_{c0} and Q_{c1}. The outcome of Q_{c0} is transmitted to Q_{d1} via a classical channel, represented by a solid double line between Q_{c0} and Q_{d1} in the figure. This classical signal determines whether an X gate should be applied to Q_{d1}. Similarly, the measurement result of Q_{c1} dictates whether a Z gate should be applied to Q_{d0}. Finally, the CNOT operation between logical qubits Q_{l0} and Q_{l1} is completed, and their logical mappings remain unchanged.
- **Teledata:** Unlike Telegate, Teledata enables logical qubits to be transferred between QPUs, thereby converting remote gates into local gates by bringing the logical qubits together. Fig. 2(b) shows how Teledata works and thus how a remote CNOT gate is accomplished, a free data qubit Q_{d2} is initialized to $|0\rangle$. First, a CNOT gate followed by a Hadamard gate is applied to entangle Q_{d0} and Q_{c0}. Subsequent measurements dictate the application of X and Z gates on Q_{d1}. At this point, the logical qubit Q_{l0} has been successfully remapped to Q_{c1}. A SWAP gate (which can be decomposed into three alternating CNOT gates), denoted by a solid line with crosses at each end, then swaps the states of Q_{c1} and Q_{d2} , Q_{l0} is finally remapped to Q_{d2}. Finally, the remote CNOT gate between Q_{d0} and Q_{d1} is transformed into a local CNOT gate between Q_{d2} and Q_{d1}.

F. Non-blocking Switching Network

In this work, we use a non-blocking switching network proposed in our prior work to realize the identical entanglement generation cost between any two QPUs [33].

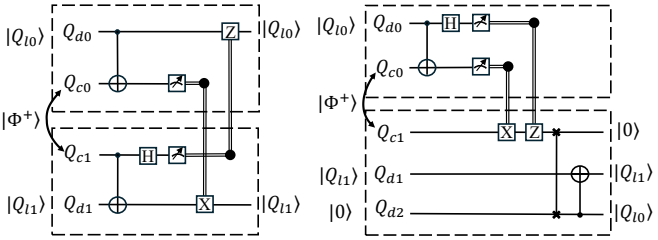
In the non-blocking switching network, we adopt the Heralded Entanglement Protocol, where qubits at two ends emit photons that interfere at a Bell-state analyzer (BSA), and a successful BSA outcome heralds the establishment of entanglement of two qubits.



(a) 3-qubit QFT circuit using Telegate protocol of cost 4.

(b) 3-qubit QFT circuit using Hybrid protocol of cost 3.

Fig. 1. Qubit mappings for 3-qubit QFT circuit.



(a) Telegate. (b) Teledata.

Fig. 2. Remote CNOT circuits with different protocols (dashed boxes represent QPUs).

In this architecture, the emitted photons are routed through a multistage switching network to a Bell-state analyzer (BSA). Importantly, any pair of photons traverses the same number of switching stages, and the network is non-blocking, meaning that ongoing entanglement generation attempts do not interfere with future requests. As a result, the expected entanglement generation costs are identical for all QPU pairs and each instance has the same entanglement genetation rate.

Although a single routing instance establishes one Bell pair at a time, the entanglement generation rate can be increased by continuously exciting multiple available (free) communication qubits within each QPU. Each communication qubit performs entanglement generation attempts in a serial manner, where an attempt must be resolved before the next attempt on the same qubit. By having multiple communication qubits available, the system can sustain multiple independent serial attempt streams, thereby increasing the aggregate attempt rate over time.

IV. System Model and Problem Formulation

In this section, we introduce the system model for DQC and formulate the remote gate scheduling problem.

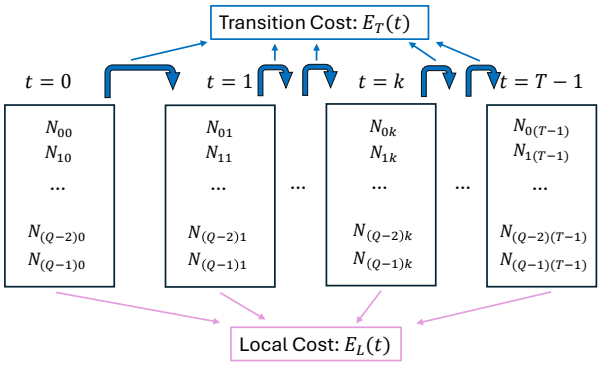


Fig. 3. Mapping matrix for DQC.

A. System Model for DQC

In this work, we consider a distributed quantum computing (DQC) system composed of multiple quantum processing units (QPUs) interconnected by an entanglement distribution network. We assume a *logically fully connected* setting, where remote entanglement can be established between any pair of QPUs with uniform expected cost, as proposed in [33], [34]. Such a fully connected interconnection model maximizes flexibility in qubit placement and remote gate execution, and is physically plausible in photonic and spin–photon platforms where entanglement between arbitrary nodes can be realized via optical channels [35], [36].

a) QPU resource model: We consider a homogeneous DQC system in which all QPUs have identical capacity L. Here, the capacity L refers to the number of physical *data qubits* available on each QPU for logical qubit storage and local computation. In addition to data qubits, each QPU is assumed to be equipped with a sufficient number of *communication qubits* dedicated to remote entanglement generation and buffering. Under this assumption, communication qubits do not constitute a limiting resource in our model, and the primary capacity constraint arises from data qubit availability. Accordingly, we model communication qubits as an abundant resource and focus on data-qubit capacity constraints.

b) Noise model and performance metric.: Remote two-qubit operations in DQC rely on shared entangled Bell pairs with finite fidelity F. Following [37], [38], we model each imperfect Bell pair as a Bell-diagonal (Werner) state

$$\rho = F\,|\Phi^+\rangle\langle\Phi^+| + \frac{1-F}{3}\left(|\Phi^-\rangle\langle\Phi^-| + |\Psi^+\rangle\langle\Psi^+| + |\Psi^-\rangle\langle\Psi^-|\right). \tag{1}$$

For teleportation-based remote gate implementations, only Pauli errors that anticommute with the measured operator lead to logical faults. As shown in [38], the corresponding effective Pauli error probability induced by a single imperfect entangled pair is

$$p_{\text{ent}} \triangleq \frac{2}{3}(1-F). \tag{2}$$

For a DQC task involving n remote entanglement operations, the overall task failure probability is

$$p_{\text{task}} = 1 - (1-p_{\text{ent}})^n, \tag{3}$$

which is monotonically increasing in n. Therefore, the total number of remote entanglement operations directly determines the accumulated noise introduced during circuit execution and serves as the primary performance metric in this work.

c) Remote gate execution protocols.: We focus on two representative protocols for executing remote CNOT gates: *Telegate* and *Teledata*, whose implementations and properties have been introduced in the Preliminary section (Sec. III-E). Telegate executes remote gates without modifying the logical-to-physical qubit mapping, whereas Teledata enables dynamic relocation of logical qubits across QPUs during circuit execution. These two protocols exhibit complementary trade-offs in entanglement consumption under QPU capacity constraints. Motivated by this observation, this work jointly considers Telegate and Teledata within a unified scheduling framework, with the objective of minimizing total entanglement consumption (and thus p_{task}).

Other approaches, such as *cat-entanglement* [39], are not considered in this work. As discussed in [10], cat-entanglement is fragile under frequent single-qubit operations and highly sensitive to decoherence, which limits its effectiveness in practical DQC settings.

d) On communication qubit contention.: Communication qubit contention is not explicitly modeled during optimization. We adopt a logically fully connected abstraction in which remote entanglement is available on demand, enabling us to isolate the primary objective of minimizing total entanglement consumption without introducing implementation-specific constraints. In execution, the computed schedule can be realized by pre-generating and buffering entangled pairs on communication qubits. Given the temporal sparsity of remote gates in typical DQC circuits, a modest number of communication qubits suffices to render contention negligible in practice.

B. Inter-QPU Communication Model

We assume a non-blocking photonic switching network for inter-QPU entanglement distribution, as introduced in our prior work [33]. In this architecture, photons emitted from any pair of QPUs are routed through a multistage optical switching network to a Bell-state analyzer (BSA). All routing paths traverse the same number of switching stages, and the network is non-blocking, ensuring that entanglement generation between different QPU pairs incurs an uniform expected cost.

Entanglement generation between QPU pairs is performed in an on-demand manner. With continued advances in photonic components—including photon generation, collection and coupling efficiency, optical loss reduction, and BSA detection quality—the rate of remote entanglement generation can approach that of local quantum gate operations and therefore does not dominate the critical execution path of the circuit.

In practical quantum circuits, two-qubit gates such as CNOTs are typically sparsely distributed over time rather than appearing in every circuit layer. This structural sparsity allows entanglement required for upcoming remote operations to be generated in advance and buffered on communication qubits, thereby avoiding execution stalls caused by entanglement unavailability.

Under these conditions, from the perspective of circuit execution and scheduling, the interconnection network can be

abstracted as a logically fully connected network with uniform inter-QPU communication cost. This abstraction is architectural rather than algorithmic and is not tied to a specific physical platform. While the switching network was originally analyzed in the context of trapped-ion systems, similar photonic or hybrid interconnects make this communication model applicable to other platforms, such as superconducting quantum processors.

C. QEC-Encoded Circuits in DQC Systems

Quantum devices are inherently subject to noise and decoherence, which limit the reliability of large-scale quantum computation. Quantum error correction (QEC) mitigates these effects by encoding each logical qubit into multiple physical qubits and detecting errors through syndrome measurements without directly collapsing the encoded quantum information.

While QEC significantly improves robustness, it introduces substantial resource overhead in terms of qubit count and memory footprint. Even relatively compact stabilizer codes require multiple ancillary qubits for syndrome extraction, resulting in a large expansion of physical resources. This overhead poses a major scalability challenge for monolithic quantum processors and further motivates the adoption of DQC architectures for fault-tolerant quantum computation.

To align our remote gate scheduling framework with fault-tolerant execution settings, we evaluate the proposed algorithm on both unencoded circuits and QEC-encoded fault-tolerant circuits. In this setting, gate operations must satisfy fault-tolerance constraints to prevent uncontrolled error propagation at the logical level. In many stabilizer codes, Clifford gates—including Hadamard, Phase, and CNOT—can be implemented transversally, whereas non-Clifford gates typically require additional resource-intensive procedures such as magic-state distillation.

To isolate the core scheduling challenges imposed by QEC while avoiding the added complexity of non-Clifford resource management, we restrict our study to Clifford-only circuits. Although such circuits are not universal for quantum computation, they form the backbone of many fault-tolerant protocols and provide a controlled setting to evaluate the impact of QEC constraints on scheduling policies, communication cost, and logical qubit placement in distributed systems.

D. Problem Formulation

In this subsection, we formally define the *Remote Gate Scheduling for Distributed Quantum Computing (RGS-DQC)* problem and present a rigorous mathematical formulation. Since this work focuses on remote gate scheduling, all entanglement considered here refers to *physical entanglement* established between *communication qubits*, unless otherwise specified.

1) Motivating Example: Fig. 1 illustrates how appropriate scheduling decisions combining Telegate and Teledata can reduce entanglement consumption in a distributed 3-qubit QFT circuit. When only Telegate is used (Fig. 1(a)), logical qubits remain stationary across QPUs throughout execution, and all remote CNOT gates are realized via Telegate, resulting in four entanglement consumptions. In contrast, by allowing Teledata-based qubit remapping during execution (Fig. 1(b)), one logical qubit is relocated between layers 4 and 5, converting several subsequent remote CNOT gates into local ones. As a result, the circuit completes using only three entanglements, achieving a 25% reduction. This example highlights the benefit of jointly optimizing qubit mapping and remote gate execution.

2) Circuit Abstraction: Since CNOT gates dominate inter-QPU communication cost in DQC systems, we abstract a quantum circuit by explicitly modeling only its CNOT structure. For a circuit decomposed into single-qubit gates and CNOT gates, we define a circuit matrix

$$G \in \mathbb{Z}^{Q\times T},$$

where Q denotes the number of logical qubits and T denotes the circuit depth. Each entry G_{qt} encodes the CNOT participation of logical qubit q at layer t:

- If qubit q participates in a CNOT gate with qubit $\bar{q}$ at layer t, then $G_{qt} = \bar{q}$ and $G_{\bar{q}t} = q$.
- If qubit q does not participate in any CNOT gate at layer t, we set $G_{qt} = -1$.

The circuit matrix compactly captures all CNOT dependencies layer by layer.

3) Mapping Variables and Constraints: To describe the placement of logical qubits throughout execution, we define a mapping matrix

$$N \in \mathbb{Z}^{Q\times T},$$

where N_{qt} denotes the QPU to which logical qubit q is assigned at layer t. The mapping must satisfy the following constraints:

$$N_{qt} \in \{0, 1, \ldots, M-1\}, \forall q \in \mathcal{Q}, \forall t \in \mathcal{T} \tag{4}$$

$$\sum_{m=0}^{M-1} \delta(N_{qt}, m) = 1, \forall q \in \mathcal{Q}, \forall t \in \mathcal{T} \tag{5}$$

$$\sum_{q=0}^{Q-1} \delta(N_{qt}, m) \le L_m, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6}$$

Constraints (4) and (5) ensure that each logical qubit is assigned to exactly one valid QPU at every layer, while Constraint (6) enforces the capacity limit L_m of each QPU.

4) Cost Model: Fig. 3 illustrates that the cost in RGS-DQC arises from both intra-layer local cost $E_L(t)$ and inter-layer transition cost $E_T(t)$.

We introduce a reversed Kronecker delta function to indicate whether two qubits reside on different QPUs:

$$\delta'(a,b) = \begin{cases} 1, & a \neq b, \\ 0, & a = b. \end{cases}$$

a) Intra-layer (Telegate) cost: Once the mapping $N_{\cdot t}$ for a layer t is fixed, all remote CNOT gates in that layer must be executed using Telegate. The entanglement cost incurred at layer t is defined as

$$E_L(t) = \frac{1}{2} \sum_{q: G_{qt} \neq -1} \delta'\left(N_{qt}, N_{(G_{qt})t}\right), \tag{7}$$

where the factor $1/2$ avoids double-counting each CNOT gate.

b) Inter-layer (Teledata) cost: If a logical qubit changes its assigned QPU between two consecutive layers, the transition is realized via Teledata, consuming one entanglement. The transition cost at layer t is defined as

$$E_T(t) = \sum_{q=0}^{Q-1} \delta'\big(N_{qt}, N_{q(t-1)}\big), \tag{8}$$

which counts the number of qubits whose mappings differ between layers $t-1$ and t.

5) Objective Function: The total entanglement consumption over the entire circuit execution is given by

$$\mathbb{E} = \sum_{t=0}^{T-1} E_L(t) + \sum_{t=1}^{T-1} E_T(t). \tag{9}$$

The RGS-DQC problem is thus formulated as minimizing $\mathbb{E}$ subject to constraints (4)–(6), given the circuit matrix G.

6) Computational Complexity: The RGS-DQC problem can be formulated as a nonconvex binary quadratic programming problem, which is NP-hard in general [40], [41]. Moreover, a closely related but simpler problem, QA-DQC, has been proven NP-hard in [12]. We next show that RGS-DQC is also NP-hard via a polynomial-time reduction from QA-DQC.

Theorem IV.1. *The RGS-DQC problem is NP-hard.*

Proof. We prove NP-hardness by reduction from QA-DQC.

Consider an arbitrary instance of QA-DQC defined by a set of logical qubits $\mathcal{Q}$, a set of QPUs $\mathcal{M}$ with capacities $\{L_m\}$, and an interaction weight matrix $\{y_{ij}\}$. We construct a corresponding instance of RGS-DQC by building a circuit whose CNOT interactions encode the same weighted interaction graph. Specifically, let the circuit depth be $T = \sum_{i<j} y_{ij}$, and construct the circuit matrix G such that the same CNOT interaction structure induced by $\{y_{ij}\}$ is repeated across all layers. Thus, each layer induces an identical qubit interaction graph.

Now consider the subclass of solutions with time-invariant mappings,

$$N_{qt} = N_q, \quad \forall q \in \mathcal{Q}, \forall t.$$

This restriction satisfies all feasibility constraints (4)–(6). Under this restriction, the Teledata cost vanishes, i.e., $E_T(t) = 0$ for all t, and the objective reduces to

$$\mathbb{E} = \sum_{i<j} y_{ij}\, \delta'(N_i, N_j),$$

which coincides exactly with the QA-DQC objective under the same capacity constraints.

Therefore, QA-DQC reduces in polynomial time to a special case of RGS-DQC. Since QA-DQC is NP-hard, RGS-DQC is NP-hard. □

V. LABUBU: LAyer-BUffered BUndled Optimization Algorithm

In this section, we present our proposed optimization framework, *LABUBU* (*Layer-Buffered Bundled Optimization*), which consists of two tightly integrated stages, both employing layer-buffered mapping updates to address the RGS-DQC problem.

Algorithm 1: Coordinate-wise_Pruned_Greedy(CPG)

```
Input: N, G, Depth, Width, qpu_lim
Output: N
for t ← 0 to Depth − 1 do
    buffer ← an empty vector of length Width;
    for q ← 0 to Width − 1 do
        S ← ∅;
        if t < Depth − 1 then
            S ← S ∪ {N_{q(t+1)}};
        if t > 0 then
            S ← S ∪ {N_{q(t−1)}};
        if G_{qt} ≠ −1 then
            S ← S ∪ {N_{(G_{qt})t}};
        if S = ∅ then
            S ← {N_{qt}};
        buffer[q] ← MajorityVote(S);
    N_{·t} ← Mapping_Update(buffer, N_{·t}, qpu_lim);
return N;
```

Labubu operates in two phases:

- a coordinate-wise pruned greedy (CPG) mapping phase that selects locally optimal qubit-to-QPU assignments, and
- a bounded perturbation phase that introduces controlled stochasticity to escape local minima.

Both phases utilize a *layer-buffered update strategy*, where mapping decisions are temporarily stored and only committed at the end of each circuit layer. This design enables more flexible qubit remapping by decoupling individual decisions from immediate QPU capacity constraints, allowing the algorithm to explore better configurations without being prematurely restricted.

A. Coordinate-wise Pruned Greedy Mapping Decision (CPG)

As discussed in the previous section, the NP-hardness of RGS-DQC makes it intractable for deterministic optimization at realistic circuit scales. We therefore adopt a heuristic approach in this phase to obtain high-quality solutions.

In DQC, the circuit width and depth determine the size of the mapping matrix N. For large circuits, N may contain tens of thousands of variables, rendering global, simultaneous optimization computationally prohibitive. Moreover, the mapping decision exhibits strong *temporal coupling*: assigning a qubit at layer t affects the transition cost to layers $t-1$ and $t+1$, while two-qubit gates induce additional *spatial coupling* between interacting qubits within the same layer.

Under these conditions, *coordinate-wise* optimization becomes a natural design choice. A naive greedy update of N_{qt} would require evaluating all M candidate QPUs. However, by examining the objective structure, the candidate set can be pruned to at most three values. Specifically, for a given logical qubit q at layer t, the objective terms that depend on N_{qt} are limited to: (i) the transition costs to two adjacent layers, and (ii) the intra-layer Telegate cost if q participates in a CNOT at layer

t. Accordingly, we define the objective terms that depend on N_{qt} as

$$\mathbb{E}'(q,t) = \delta'(-1, G_{qt})\,\delta'\big(N_{qt}, N_{(G_{qt})t}\big) + \delta'\big(N_{qt}, N_{q(t-1)}\big) + \delta'\big(N_{qt}, N_{q(t+1)}\big). \tag{10}$$

where $\delta'(-1, G_{qt})$ indicates whether the CNOT-related term is applicable. Equation (10) implies that $\mathbb{E}'(q,t)$ depends only on the three mappings $\{N_{q(t-1)}, N_{q(t+1)}, N_{(G_{qt})t}\}$ (when available). Therefore, it suffices to restrict the greedy choice of N_{qt} to this (at most three-element) candidate set, rather than searching over all QPUs.

Intuitively, if all available candidates are identical, choosing that value makes $\mathbb{E}'(q,t) = 0$. If two candidates agree and the third differs, choosing the majority value yields $\mathbb{E}'(q,t) = 1$. In the worst case where all three candidates differ, any choice among them gives $\mathbb{E}'(q,t) = 3$. The same reasoning applies when only one or two candidates exist, in which case selecting the majority (or the only) candidate minimizes $\mathbb{E}'(q,t)$ over the pruned set. This motivates the MajorityVote rule used in CPG.

To avoid premature capacity conflicts, Labubu adopts a layer-buffered update mechanism. Within each layer t, we first compute tentative assignments for all qubits and store them in a buffer, while keeping the current mapping unchanged during the coordinate-wise scan. After all qubits in the layer have produced buffered decisions, we invoke the layer-wise Mapping_Update procedure (Sec. V-D) to enforce QPU capacity constraints and commit a feasible layer mapping.

Algorithm 1 summarizes the CPG phase. CPG performs a coordinate-wise sweep over all layers and qubits, where each N_{qt} is updated by majority voting over the pruned candidate set. Since a single sweep may be insufficient, Labubu repeats CPG for multiple iterations under a bounded iteration budget. Empirically, the objective typically stabilizes after a finite number of iterations, but CPG may still become trapped in a local optimum.

The behavior of CPG also reveals two challenges:

a) **Local-optimum trap:** repeated greedy updates can quickly settle into a locally stable configuration.
b) **Insufficient diversity:** if the mapping is temporally uniform (e.g., identical across layers at initialization), then the pruned candidate sets may provide limited opportunities for remapping, making CPG ineffective.

To address the first challenge, we introduce a structured perturbation mechanism (Sec. V-B) that injects controlled stochasticity while preserving useful structure. To address the second challenge, we design a tailored initialization strategy (Sec. V-C) that intentionally breaks temporal uniformity and creates meaningful remapping opportunities for subsequent CPG iterations.

B. Bounded Perturbation (BP) for Escaping Local Optima

Due to its greedy nature, CPG may converge to a locally stable configuration that is suboptimal globally. To mitigate this issue, Labubu incorporates a *Bounded Perturbation (BP)* mechanism that enables controlled exploration beyond the current local optimum.

Algorithm 2: Bounded_Perturbation(BP)

```
Input: N, G, Depth, Width, qpu_lim
Output: N
for t ← 0 to Depth − 1 do
    buffer ← an empty vector of length Width;
    for q ← 0 to Width − 1 do
        S ← ∅;
        if t < Depth − 1 then
            S ← S ∪ {N_{q(t+1)}};
        if t > 0 then
            S ← S ∪ {N_{q(t−1)}};
        if G_{qt} ≠ −1 then
            S ← S ∪ {N_{(G_{qt})t}};
        if S = ∅ then
            S ← {N_{qt}};
        buffer[q] ← RandomChoice(S);
    N_{·t} ← Mapping_Update(buffer, N_{·t}, qpu_lim);
return N;
```

In heuristic optimization, perturbation is a common strategy for escaping local minima. However, indiscriminate random perturbations applied to all qubits and layers would destroy the structured properties of the current solution and negate prior optimization efforts. Conversely, overly small perturbations affecting only a few variables may be insufficient to alter the search trajectory, causing the algorithm to return to the same local minimum.

BP strikes a balance between these extremes. As shown in Algorithm 2, each qubit is reassigned to a randomly selected QPU from its pruned candidate set, thereby preserving structural relevance while intentionally breaking temporal consistency in the mapping. This introduces moderate stochasticity without discarding useful structural information accumulated during previous CPG iterations.

Although the BP phase may temporarily increase entanglement consumption, the subsequent CPG phase rapidly refines the perturbed mapping and often leads to improved solutions. This iterative interplay between structured perturbation and greedy refinement is illustrated in Fig. 4.

C. Initial Mapping

Due to the strong temporal correlation between adjacent layers, a mapping decision at layer t tends to influence neighboring layers. Consequently, the CPG phase favors temporally consistent mappings, which may limit opportunities to convert remote CNOT gates into local ones.

To alleviate this issue, we construct an initial mapping in which, for each layer, qubits participating in the same CNOT gate are assigned to the same QPU. Under this initialization, the intra-layer cost satisfies

$$\sum_{t=0}^{T-1} E_L(t) = 0,$$

indicating that no remote CNOT gates are present at the initial stage. However, this comes at the expense of increased transition cost $\sum_{t=1}^{T-1} E_T(t)$, since mappings may differ significantly across consecutive layers.

Furthermore, to promote efficient utilization of limited QPU resources, logical qubits are preferentially assigned to already activated QPUs whenever feasible, thereby reducing unnecessary fragmentation of capacity across the system.

This initialization not only eliminates initial intra-layer communication cost but also introduces deliberate inter-layer diversity in the mapping matrix. Such diversity prevents the algorithm from stagnating in temporally uniform configurations where no effective updates can be generated by CPG.

In addition, we adopt a simple yet effective capacity reservation strategy during initialization. Specifically, instead of fully utilizing each QPU up to capacity L, we restrict the usable capacity to $L - k$, where k is a small positive integer. The remaining k data qubits per QPU are left idle.

This reserved capacity substantially improves the flexibility of Teledata operations. When a QPU is fully occupied, Teledata-based qubit relocation may become infeasible due to lack of available data qubits. By maintaining limited slack capacity, the algorithm can accommodate temporary qubit migrations, thereby enhancing robustness and improving overall scheduling performance. The impact of this strategy is further evaluated in Section VI-B4.

D. Layer-Buffered (Labu) Mapping Update

A key innovation of LABUBU lies in its **layer-buffered mapping update strategy**, which significantly enhances the flexibility of qubit mapping under QPU capacity constraints. In conventional coordinate-wise (Cowi) update methods, a qubit is immediately assigned to a target QPU once selected. Such a decision may be infeasible if the target QPU is temporarily full, even though subsequent updates could later release sufficient capacity. This limitation may result in overly conservative decisions and suboptimal mappings.

To address this issue, LABUBU adopts a layer-buffered update mechanism. Instead of applying mapping decisions immediately, candidate assignments produced during the CPG or BP phase are first stored in a temporary buffer. After the mapping selection for the entire layer is completed, the buffered decisions are applied in a unified manner, referred to as Labu-CPG or Labu-BP, respectively. By deferring the application of mapping changes, this mechanism mitigates premature capacity conflicts and enables more globally adaptive scheduling decisions within each layer.

However, since capacity constraints in Eq. (6) are not enforced during the buffering phase, the resulting buffered mapping may violate QPU capacity limits. Direct application of such a mapping may therefore lead to infeasible assignments. To ensure feasibility, we introduce a bounded capacity-repair procedure that incrementally adjusts the buffered mapping toward a valid configuration.

Algorithm 3 presents the pseudocode of the Layer-Buffered Mapping Update. The input *buf_map* denotes the buffered mapping obtained after a CPG or BP phase, while *cur_map* represents the original mapping prior to the update. The parameter *qpu_lim* specifies the capacity limit of each QPU.

The repair procedure first identifies overloaded QPUs and computes their excess usage. For each overloaded QPU, a limited number of qubits that were newly migrated to that QPU are reverted to their original assignments in *cur_map*. The number of such rollbacks is bounded by the corresponding excess usage, thereby ensuring that repair actions are both conservative and minimal. This mechanism guarantees that no additional violations are introduced while progressively reducing overload.

Since a single repair pass may not eliminate all violations, the procedure performs a bounded number of repair iterations. If a feasible mapping is obtained within the repair budget, it is accepted and applied to the entire layer. Otherwise, the algorithm falls back to *cur_map*, leaving the mapping unchanged for the current layer and preserving feasibility.

The parameter *max_repair* specifies an upper bound on the number of repair iterations. In our implementation, *max_repair* is set to 500 as a conservative safeguard. However, in practice, the actual repair depth is typically far below this bound.

The detailed behavior of the mapping update procedure is analyzed in Section VI-B5. These empirical characterizations provide insight into the repair dynamics and support the selection of a conservative upper bound for *max_repair*.

Algorithm 3: MAPPING_UPDATE

Input: *buf_map*, *cur_map*, *num_qpu*, *qpu_lim*, *max_repair*
Output: *corrected_map*

```
repair_cnt ← 0;
// Iteratively repair capacity
   violations via bounded rollback
while repair_cnt < max_repair do
    if IsValid(buf_map, qpu_lim) then
        return buf_map;
    // Identify overloaded QPUs and
       their excess usage
    violations ← GetViolations(buf_map, qpu_lim);
    new_map ← buf_map;
    // Rollback migrated qubits on
       overloaded QPUs
    foreach logical qubit q do
        i ← buf_map[q];
        if i is overloaded and cur_map[q] ≠ i and
          rollback budget of i not exhausted then
            new_map[q] ← cur_map[q];
            decrease rollback budget of i;
    // Accept if repaired
    if IsValid(new_map, qpu_lim) then
        return new_map;
    buf_map ← new_map;
    repair_cnt ← repair_cnt + 1;
// Fallback if repair budget is
   exhausted
return cur_map;
```

Algorithm 4: *LABUBU*

Input: G, *Depth*, *Width*, *MaxIter*, $Interval$, qpu_lim
Output: $N^\star$
Initialize N;
$cost^\star \leftarrow +\infty$;
$N^\star \leftarrow N$;
for $i \leftarrow 1$ **to** *MaxIter* **do**
  if $i \bmod Interval = 0$ **then**
    // Checkpoint before disruptive BP step
    $cost \leftarrow \textsc{ComputeCost}(N, G)$;
    if $cost < cost^\star$ **then**
      $cost^\star \leftarrow cost$;
      $N^\star \leftarrow N$;
    $N \leftarrow \textsc{Labu-BP}(N, G, Depth, Width, qpu\_lim)$;
  else
    $N \leftarrow \textsc{Labu-CPG}(N, G, Depth, Width, qpu\_lim)$;
// Final checkpoint
$cost \leftarrow \textsc{ComputeCost}(N, G)$;
if $cost < cost^\star$ **then**
  $cost^\star \leftarrow cost$;
  $N^\star \leftarrow N$;
return $N^\star$;

E. Layer-Buffered Bundled Optimization (Labubu)

Having introduced the CPG and BP phases, we now present LABUBU as a unified bundled optimization framework that integrates both mechanisms within an iterative schedule. In each iteration, Labubu applies the CPG phase to greedily refine the qubit mapping, and periodically invokes the BP phase to introduce controlled perturbations that facilitate escape from local optima. This alternating design combines steady local improvement with structured global exploration.

Algorithm 4 outlines the overall procedure. The algorithm runs for a fixed number of iterations, denoted by *MaxIter*, while the BP phase is triggered every *Interval* iterations. Since BP introduces stochasticity, different executions may converge to different local optima. To enhance robustness, we perform multiple independent runs with distinct random seeds and report the solution with the minimum entanglement cost among all runs.

F. Computational Complexity Analysis

Let Q denote the circuit width and T the circuit depth. In each iteration, LABUBU processes the circuit layer by layer. For every layer, the algorithm performs a constant number of linear scans over all qubits to evaluate candidate assignments and to enforce QPU capacity constraints via the `MAPPING_UPDATE` routine. Although `MAPPING_UPDATE` may invoke recursive feasibility checks, the recursion depth is explicitly bounded, resulting in only constant-factor overhead. Therefore, the computational cost per iteration is $O(QT)$.

The algorithm runs for a fixed number of iterations I, which is set as a constant upper bound to ensure predictable runtime. Accordingly, the overall time complexity of LABUBU is $O(QTI)$. Under a fixed iteration budget, the algorithm scales linearly with respect to both the circuit width Q and the circuit depth T.

Evaluating the global entanglement cost is significantly more expensive than local mapping updates. For this reason, the cost is not recomputed after every iteration. Instead, it is evaluated only immediately before executing the BP phase and once after all iterations have completed. This strategy is sufficient because the CPG phase is greedy and does not increase the entanglement cost, whereas BP may temporarily increase it in order to escape local optima. The final solution is selected as the best one observed among all recorded cost evaluations.

VI. PERFORMANCE EVALUATION

In this section, we provide a comprehensive evaluation of the proposed algorithm's performance.

TABLE I
SUMMARY OF TESTED CIRCUITS

	QFT		RD		QEC-Encoded RD		
Width	**Depth**	**Size**	**Depth**	**Size**	**Width**	**Depth**	**Size**
30	232	915	108	696	240	1409	21072
40	312	1620	152	1231	320	1975	38857
50	392	2435	183	1913	400	2355	58391
60	472	3250	235	2847	480	2986	88329
70	552	4065	273	3826	560	3488	120022
80	632	4880	325	5056	640	4120	161152
90	712	5695	351	6349	720	4492	198883
100	792	6510	400	7902	800	5065	248514

A. Simulation Settings

We conduct simulations using two representative types of quantum circuits: Quantum Fourier Transform (QFT) circuits and randomly generated circuits (RD). QFT circuits constitute a fundamental building block in quantum algorithms and are widely adopted in prior DQC studies [10], [42], [43]. RD circuits consist of randomly generated single- and two-qubit operations and are commonly used to benchmark quantum computing performance; for example, similar random circuit constructions were employed in Google's quantum supremacy experiments [44].

All circuits are generated using Qiskit [45] and decomposed into a basis gate set consisting of single-qubit gates and CNOT gates. To distinguish circuit scales, we append the circuit width to the circuit name; for instance, QFT100 denotes a 100-qubit QFT circuit. Table I summarizes the circuit parameters used in our experiments, including QEC-encoded variants. Here, *Size* denotes the total number of CNOT gates.

We compare LABUBU with four representative baselines from the literature.

- **Telegate-RD**: logical qubits are randomly mapped to QPUs with a fixed mapping throughout execution, and all remote gates are implemented via the Telegate protocol.
- **Telegate-SA** [12]: a fixed mapping is obtained using Simulated Annealing, and remote gates are executed using Telegate.

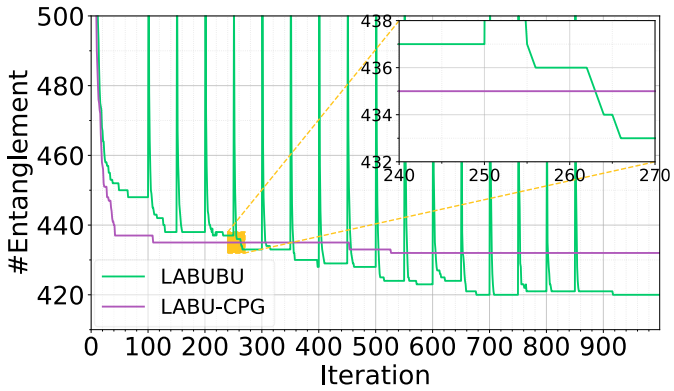


Fig. 4. Iterative progress of LABUBU and LABU-CPG on QFT100.

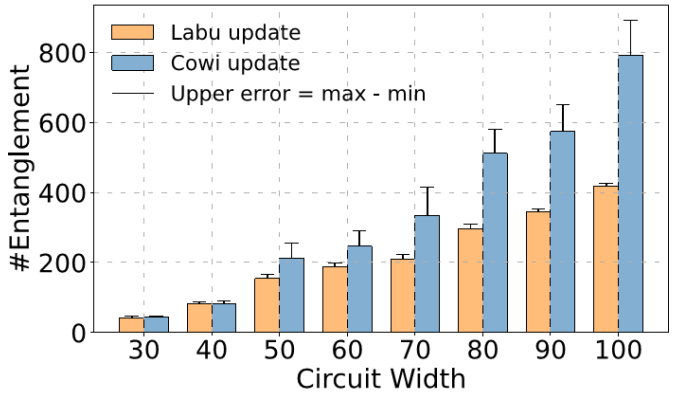


Fig. 5. Performance of different update Strategy on QFT100.

- **Teledata-ZS** [10]: adopts the Teledata protocol and applies a dynamic programming strategy for remote gate scheduling.
- **GateCover (GC)** [20]: a greedy heuristic that schedules Teledata operations to cover as many future remote CNOT gates as possible.

In addition, we evaluate a hybrid configuration, **GC+LABUBU**, in which the solution produced by GC is used as the initialization for LABUBU.

In our simulations, LABUBU is executed 18 times with distinct random seeds, and the solution with the minimum entanglement cost is reported. Unless otherwise specified, the parameters *max_repair* and *MaxIter* are set to 500 and 1000, respectively.

B. Performance Evaluation

1) Convergence: Fig. 4 illustrates the iteration progress of LABUBU and LABU-CPG on a QFT circuit with width 100, where each QPU has a capacity of 25 qubits. LABU-CPG converges after approximately 40 iterations, with occasional cost drops in later stages, indicating its ability to achieve rapid convergence. LABUBU bundles CPG and BP together; after each BP execution, the cost experiences a surge, which is then quickly reduced by the subsequent CPG phase to reach a lower-cost solution. By alternately executing LABU-CPG and LABU-BP, LABUBU continually explores better solutions. On QFT-100, after 1000 iterations, LABUBU achieves an approximately 3.2% performance improvement over LABU-CPG.

2) Update Method Comparison: In Sec. V-D, we introduced the layer-buffered (LABU) update strategy, which defers mapping commitments to improve flexibility under capacity constraints. To evaluate its effectiveness, we compare the LABU and conventional coordinate-wise (Cowi) update strategies on QFT circuits of varying widths, with QPU capacity fixed at 25 (see Fig. 5). In the figure, each bar represents the best solution obtained from 18 independent runs, while the error bar indicates the range between the maximum and minimum entanglement costs across runs.

The results show that the LABU update strategy consistently outperforms Cowi. When the QPU capacity is close to the circuit width, the performance gap is relatively small. However, as the circuit width increases, the advantage of LABU becomes increasingly pronounced. For example, on the QFT100 instance, LABU achieves nearly a 50% reduction in total entanglement cost compared with Cowi.

Moreover, the error bars indicate that LABU produces more concentrated outcomes across runs, whereas Cowi exhibits significantly larger variability. These results demonstrate that the layer-buffered mechanism not only improves solution quality but also enhances robustness, particularly for large-scale circuits where capacity constraints play a more critical role.

3) Performance Comparison: Fig. 6 reports the total entanglement cost of LABUBU under different QPU capacities and circuit types, together with its relative improvement over the benchmark methods. The relative improvement is defined as

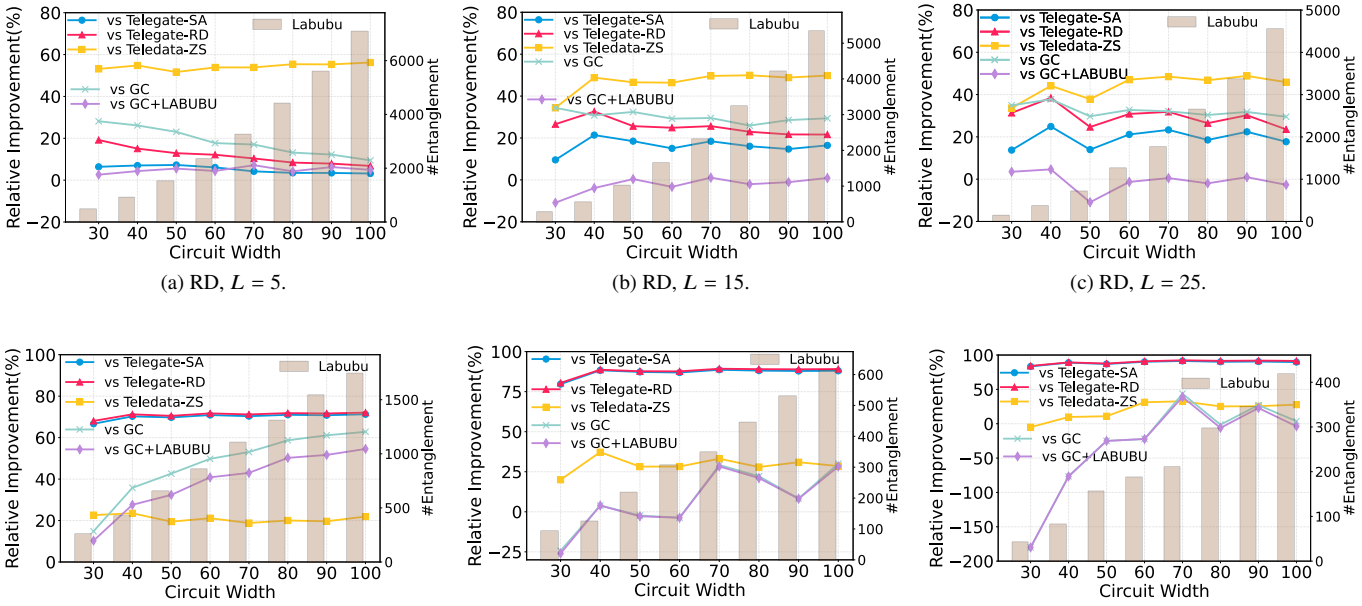
$$\frac{Cost(\text{benchmark}) - Cost(\text{LABUBU})}{Cost(\text{benchmark})}.$$

a) Performance on RD circuits.: Figs. 6(a)–(c) present the results on RD circuits. Under identical QPU capacity, LABUBU consistently outperforms Telegate-SA, with the relative improvement increasing from approximately 5% to about 20% as the QPU capacity grows. Compared with Telegate-RD, a similar trend is observed, while the overall improvement margin is roughly 10% larger.

GateCover (GC) performs slightly worse than the Telegate-based baselines. In comparison, LABUBU achieves a 15%–40% reduction in entanglement cost over GC. When GC is used as the initialization for LABUBU (denoted GC+LABUBU), a clear improvement over GC alone is observed across most configurations. This demonstrates that the proposed optimization framework is capable of effectively refining GC-generated mappings and further reducing entanglement consumption.

Compared with standalone LABUBU, the performance of GC+LABUBU is largely comparable, and in a few configurations it achieves slightly lower entanglement cost. This indicates that high-quality initialization can occasionally provide marginal gains. However, the carefully designed initialization strategy in LABUBU already offers strong performance, explaining why the additional benefit of GC initialization remains limited.

The largest performance gap is observed with Teledata-ZS, where LABUBU consistently achieves around 50% relative improvement across different capacities and widths. This result



(a) RD, $L = 5$. (b) RD, $L = 15$. (c) RD, $L = 25$.

(d) QFT, $L = 5$. (e) QFT, $L = 15$. (f) QFT, $L = 25$.

Fig. 6. Relative improvement of *LABUBU* across benchmarks and absolute entanglement cost for various quantum circuits under different QPU capacities.

suggests that Teledata-ZS struggles to exploit structural benefits in randomly generated circuits.

b) Performance on QFT circuits: Fig. 6(d)–(f) show the results on QFT circuits. In this structured setting, LABUBU achieves substantial improvements of 70%–90% over Telegate-SA and Telegate-RD. Compared with the RD case, Teledata-ZS performs significantly better on QFT circuits due to their regular interaction pattern. Nevertheless, LABUBU still maintains a consistent 20%–30% improvement across most configurations.

For small circuit widths at $L = 15$ and $L = 25$, LABUBU shows slightly negative relative improvement compared with GC and GC+LABUBU, indicating that GC is highly competitive in these specific regimes. Moreover, GC+LABUBU performs nearly identically to GC, suggesting that GC already produces near-local-optimal solutions in these structured, moderate-scale settings. However, under tighter capacity constraints (e.g., $L = 5$), LABUBU again achieves the best performance, highlighting its stronger adaptability in highly constrained scenarios.

c) Discussion: The different behaviors across RD and QFT circuits can be attributed to structural regularity. GC explicitly exploits repetitive interaction patterns and therefore performs particularly well on highly structured circuits such as QFT. However, as circuit width increases or QPU capacity decreases—leading to a larger number of QPUs—the search space grows significantly, and the advantage of purely greedy structural exploitation diminishes.

Teledata-ZS also benefits from structured gate layouts, as it relies on stitching zero-remote-gate subcircuits. In contrast, RD circuits lack such regularity, which limits the effectiveness of Teledata-only strategies.

By jointly integrating Teledata and Telegate within a layer-buffered iterative optimization framework, LABUBU consistently achieves superior performance across both unstructured and structured workloads. Moreover, under fixed QPU capacity, the absolute entanglement cost of LABUBU increases smoothly with circuit width, indicating stable scaling behavior as the problem size grows.

d) Summary.: Across both RD and QFT circuits, LABUBU demonstrates strong robustness and versatility. Its ability to adapt to both random and highly structured interaction patterns suggests that it provides a generally effective scheduling framework for distributed quantum computing systems.

4) Comparison of Initialization Methods: In Section V-C, we discussed that reserving a small number of idle physical qubits during the initialization phase may enhance the flexibility of Teledata operations and thereby improve overall performance. To quantitatively evaluate this effect, we conducted experiments on QFT circuits of various widths in a DQC system with QPU capacity $L = 25$, comparing three configurations: no idle qubits, one reserved idle qubit, and two reserved idle qubits per QPU.

As shown in Fig. 7, reserving a small number of idle qubits consistently reduces the total entanglement cost. This benefit becomes more pronounced as the circuit width increases, indicating that additional mapping flexibility is particularly valuable in larger-scale systems.

Interestingly, increasing the number of reserved idle qubits from one to two does not further improve performance; instead, the two-idle configuration performs slightly worse than the one-idle case. The reason is that excessive reservation effectively reduces the usable capacity of each QPU, which may require activating additional QPUs to host all logical qubits. This leads to a more dispersed qubit distribution and increases the likelihood of remote interactions, thereby offsetting the flexibility gains provided by additional idle space.

Based on these observations, we adopt the one-idle-qubit initialization strategy in all subsequent experiments, as it provides the best trade-off between Teledata flexibility and effective QPU utilization.

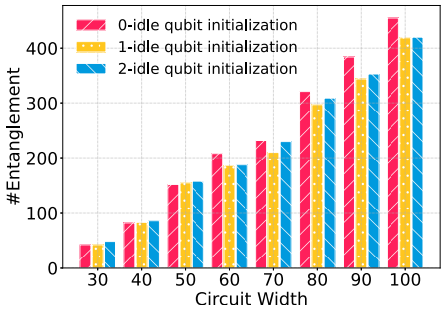


Fig. 7. Performance comparison of different initialization.

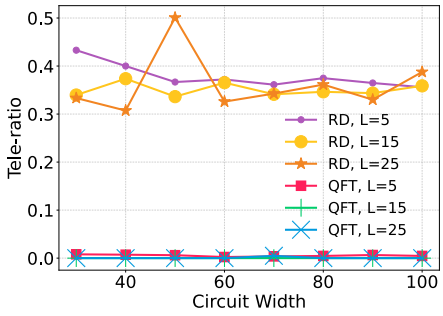


Fig. 8. Tele-ratio of LABUBU.

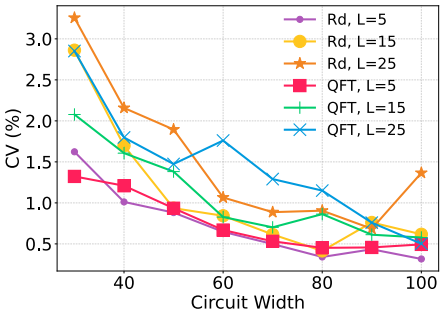


Fig. 9. Coefficient of Variation of LABUBU.

5) Behavior of the `MAPPING_UPDATE` *Procedure:* To repair invalid mapping in qubit mapping update, we introduced algorithm 3 in V-D. The behavior of algorithm 3 can be categrized into five types:

- Type I (Unchanged Mapping): The target mapping is identical to the current mapping, and no update is performed.
- Type II (Directly Feasible Mapping): The target mapping satisfies all QPU capacity constraints and is accepted immediately without invoking the repair procedure.
- Type III (Degenerated-to-Original Mapping): The target mapping violates capacity constraints and, after recursive rollback, degenerates entirely to the original mapping.
- Type IV (Repaired Feasible Mapping): The target mapping is initially infeasible but, after a finite number of recursive repair rounds, a new feasible mapping is found and accepted.
- Type V (Repair Failure): The target mapping remains infeasible after exhausting the maximum repair budget, and the original mapping is retained.

Table II presents the distribution of mapping-update behavior types within a single layer. The observed patterns differ significantly between the two circuit classes.

For QFT100, a large proportion of updates fall into Type I, meaning that the mapping remains unchanged after the update stage. This does not stem from the MAPPING_UPDATE procedure itself; rather, it indicates that the CPG phase frequently generates no new candidate mapping. In such cases, the buffered mapping is identical to the current mapping, and the update procedure is not triggered. Type I therefore reflects limited remapping opportunities identified during the search phase, rather than conservative repair behavior.

In contrast, the weight of Type I is very low for RD100, indicating that the CPG phase actively proposes remapping decisions in most layers. These candidates are either directly feasible (Type II) or become feasible after bounded repair (Type IV), demonstrating that the rollback mechanism successfully restores feasibility while retaining partial improvements.

Table III further shows that the repair process is inherently shallow. Across all configurations, the vast majority of repairs terminate within one or two iterations, and deeper repair chains are extremely rare. This indicates that capacity violations introduced by buffered updates are localized and do not trigger cascading adjustments.

TABLE II
`MAPPING_UPDATE` BEHAVIOR TYPES DISTRIBUTION

Circuit	L	Type I	Type II	Type III	Type IV	Type V
QFT100	5	60.24%	33.95%	1.77%	4.04%	0.00%
QFT100	15	69.08%	27.34%	1.97%	1.61%	0.00%
QFT100	25	73.91%	23.84%	1.50%	0.75%	0.00%
RD100	5	0.65%	17.07%	0.29%	81.99%	0.00%
RD100	15	1.21%	25.80%	3.89%	69.10%	0.00%
RD100	25	2.65%	26.02%	16.92%	54.41%	0.00%

TABLE III
DISTRIBUTION OF REPAIR DEPTH

Circuit	L	1–2 Depth	3–5 Depth	6–10 Depth	11+ Depth
QFT100	5	4.036%	0.002%	0.000%	0.000%
QFT100	15	1.610%	0.001%	0.000%	0.000%
QFT100	25	0.747%	0.000%	0.000%	0.000%
RD100	5	78.999%	2.978%	0.011%	0.000%
RD100	15	65.484%	3.573%	0.044%	0.0003%
RD100	25	51.246%	3.115%	0.049%	0.000%

Together, these empirical characterizations provide insight into the repair dynamics and support the selection of a conservative upper bound for *max_repair*. Although *max_repair* is set to 500 in our implementation, the observed repair depths remain orders of magnitude smaller than this bound, ensuring a substantial safety margin while maintaining bounded computational overhead.

6) Protocol Composition in LABUBU: To characterize how LABUBU balances Telegate and Teledata, we define the *Tele-ratio* as Tele-ratio $\triangleq \frac{n_{\mathrm{TG}}}{n_{\mathrm{TG}}+n_{\mathrm{TD}}}$, where n_{TG} and n_{TD} are the numbers of entanglement operations used by Telegate and Teledata, respectively. Fig. 8 reports the Tele-ratio across different settings: for RD circuits it stays above 0.3, indicating substantial Telegate involvement, whereas for QFT circuits it remains close to zero, showing that Teledata dominates. These results further support that Telegate is more effective on unstructured circuits, while Teledata better exploits structured circuits such as QFT.

7) Stability of LABUBU: As introduced in Sec. VI-A, LABUBU is executed 18 times in parallel using distinct random seeds. To evaluate the stability of LABUBU, we quantify the variability of its results using the coefficient of variation (CV), defined as

$$\mathrm{CV} \triangleq \frac{\sigma}{\mu} \times 100\%, \tag{11}$$

TABLE IV
COMPARISON OF LABUBU AND OPTIMAL SOLUTION UNDER DIFFERENT QPU CAPACITY LIMITS.

QPU Limit	LABUBU	Optimal-Hybrid	Optimal-Telegate
6	100	67 (80%)	288 (0%)
8	60	43 (80%)	256 (0%)
10	58	50 (50%)	200 (0%)
12	38	24 (0%)	192 (0%)
14	36	18 (0%)	168 (0%)

where μ denotes the mean entanglement cost and σ denotes the corresponding standard deviation across different runs.

Fig. 9 shows the CV of LABUBU under the different configurations presented in Fig. 6. Across all settings, the CV remains below 3.5%, indicating that LABUBU exhibits stable performance with respect to random initialization. Moreover, as the circuit width increases, the CV shows a decreasing trend, suggesting improved stability and scalability for larger circuits. Finally, similar CV values are observed across different circuit types, demonstrating that LABUBU maintains robust performance under diverse workloads.

8) Approximation Ratio: To further assess the effectiveness of LABUBU and its proximity to the optimal solutions of the Hybrid and Telegate protocols, we employed the Gurobi solver [46] to compute exact solutions. Since the underlying problem is a 0-1 quadratic program that is NP-hard and non-convex, the solver becomes prohibitively slow as the problem size increases. Therefore, we limited our evaluation to a small-scale instance, QFT20, as shown in Table IV.

Despite the relatively small size, the solver still required several days to reach optimality under certain parameter settings. To ensure practical termination, we imposed a precision threshold determined by the real-time gap between the incumbent solution and the lower bound, with the corresponding gap shown in parentheses for each entry. Once the gap fell below the threshold, the solver returned the incumbent solution. Although a gap remains, it is worth noting that much of the solver's runtime is spent tightening the lower bound, suggesting the returned solution may be closer to optimal than the gap implies.

Table IV shows that LABUBU achieves significantly lower DQC costs than the Teledata-only protocol. In cases listed, LABUBU achieves solutions within a factor of two of the optimal or near-optimal hybrid solutions produced by Gurobi under the specified optimality gaps. These results indicate that LABUBU effectively leverages the strengths of the hybrid protocol to generate solutions that are reasonably close to optimal.

C. LABUBU on QEC-Encoded Circuit

As discussed in Subsection IV-C, DQC enables QEC by effectively expanding the available system capacity beyond what a single device can support. In this subsection, we evaluate the performance of LABUBU on large-scale *QEC-encoded quantum circuits composed solely of Clifford gates*, under a DQC setting where each QPU has a capacity comparable to state-of-the-art real-world hardware. The purpose of this evaluation is to study the scalability of remote gate scheduling on encoded circuits, rather than to analyze full fault-tolerant logical execution. In addition to LABUBU, we also evaluate *LABU-CPG*, which applies only the CPG phase without the BP phase. The results show that, for extremely large QEC-encoded circuits and under a fixed runtime budget, omitting the BP phase can in some cases lead to better solutions.

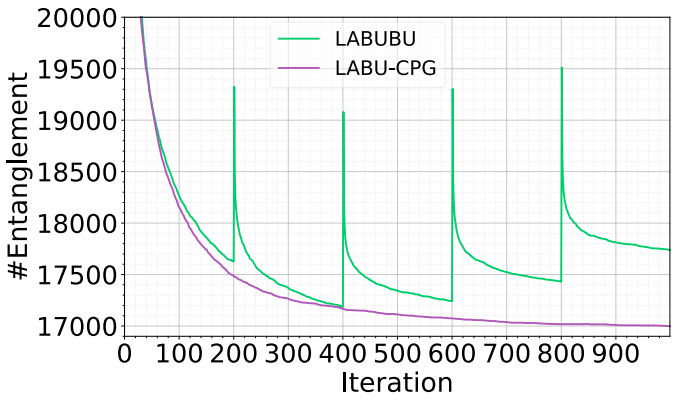


Fig. 10. Iterative progress of LABUBU and LABU-CPG on Steane code-encoded RD100.

We consider a 100-qubit random (RD) circuit encoded with the Steane code and composed entirely of Clifford gates. Each logical qubit is represented by seven physical qubits, with one additional physical qubit used for syndrome measurement. Consequently, each logical qubit expands into eight physical qubits, resulting in a QEC-encoded circuit width of 800 physical qubits. Fig. 10 illustrates the iterative optimization process of LABUBU and *LABU-CPG* for this circuit in a DQC system where each QPU has a capacity of 156 physical qubits. This capacity is chosen as a realistic reference point, reflecting the order of magnitude of currently achievable QPUs, inspired by IBM's Heron r2 processor [47], rather than as a hardware-faithful emulation target.

Compared to the results in Fig. 4, which depict a 100-qubit unencoded Clifford circuit executed on QPUs with a capacity of 25 qubits, the introduction of the BP phase in this large-scale QEC-encoded setting does not help the algorithm escape local optima. Instead, BP may drive the solution toward a higher entanglement cost, thereby negatively impacting overall performance. This behavior indicates that, at large scales, the local refinement performed by BP can interfere with the global structural improvements achieved during the CPG phase.

To further evaluate remote gate scheduling on large-scale QEC-encoded circuits, Table V reports the performance of LABUBU and *LABU-CPG* across random circuits of varying widths after Steane-code encoding. The circuit widths range from 240 to 800, with the corresponding widths of the original unencoded circuits shown in parentheses. Telegate-Random with random mapping is used as the baseline, and the lowest entanglement cost for each circuit is highlighted in bold.

The results demonstrate that active remote gate scheduling significantly reduces the total remote entanglement cost across all circuit sizes. LABUBU, which integrates the BP phase,

TABLE V
PERFORMANCE COMPARISON OVER QEC-ENCODED CIRCUITS OF DIFFERENT SIZE.

Algo \ Width	**240** (30)	**320** (40)	**400** (50)	**480** (60)	**560** (70)	**640** (80)	**720** (90)	**800** (100)
Telegate-RD	9441	20826	38611	59102	89918	122448	158840	199834
LABUBU	**2454**	**5239**	**9950**	18125	25865	46545	52950	91101
LABU-CPG	2784	5502	10867	**17043**	**25043**	**36805**	**46100**	**64813**

performs better for relatively small QEC-encoded circuits. However, once the encoded circuit width exceeds 400, *LABU-CPG* consistently outperforms LABUBU, with its advantage increasing as the circuit scale grows. This trend suggests that, for ultra-large QEC-encoded Clifford circuits, restricting the optimization to the CPG phase enables more efficient exploration of favorable suboptimal solutions, while avoiding the overhead and potential disruption introduced by fine-grained BP refinements.

VII. CONCLUSION

In this paper, we studied the Remote Gate Scheduling problem for Distributed Quantum Computing (RGS-DQC) under a hybrid execution model that jointly considers Telegate and Teledata, with the objective of minimizing total entanglement consumption. We provided a formal formulation and established the NP-hardness of RGS-DQC via a reduction from QA-DQC, highlighting the intrinsic computational complexity of the problem.

To address this challenge, we proposed LABUBU, a layer buffered bundled optimization framework that combines coordinate-wise pruned greedy refinement with a bounded perturbation mechanism. By integrating aggressive candidate pruning, iterative refinement with controlled exploration, and capacity aware layer buffered updates, LABUBU achieves scalable optimization with linear complexity per iteration.

Extensive simulations on both structured QFT circuits and unstructured random circuits demonstrate that LABUBU consistently reduces entanglement cost across different circuit widths and QPU capacities. Compared with representative Telegate based, Teledata based, and structure aware baselines, including the competitive GateCover method, LABUBU maintains competitive or superior performance over a wide range of configurations. Moreover, its hybrid protocol composition adapts to circuit structure, leveraging Telegate more frequently for unstructured circuits and Teledata for structured circuits.

We further evaluated the framework on QEC encoded circuits. Both LABUBU and its CPG only variant significantly outperform Telegate based random mapping, while the deterministic variant becomes particularly effective in ultra large scale regimes.

Overall, LABUBU provides a scalable and adaptable solution for entanglement aware scheduling in distributed and fault tolerant quantum computing systems.

REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[4] J. Chow, O. Dial, and J. Gambetta, "Ibm quantum breaks the 100-qubit processor barrier," *IBM Research Blog*, 2021.

[5] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'brien, "Experimental realization of shor's quantum factoring algorithm using qubit recycling," *Nature photonics*, vol. 6, no. 11, pp. 773–776, 2012.

[6] D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill, "Efficient networks for quantum factoring," *Physical Review A*, vol. 54, no. 2, p. 1034, 1996.

[7] B. M. Terhal, "Quantum error correction for quantum memories," *Reviews of Modern Physics*, vol. 87, no. 2, pp. 307–346, 2015.

[8] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, "Distributed quantum computing: a survey," *Computer Networks*, vol. 254, p. 110672, 2024.

[9] Y. Mao, Y. Liu, and Y. Yang, "Probability-aware qubit-to-processor mapping in distributed quantum computing," in *Proceedings of the 1st Workshop on Quantum Networks and Distributed Quantum Computing*, 2023, pp. 51–56.

[10] R. G. Sundaram and H. Gupta, "Distributing quantum circuits using teleportations," in *2023 IEEE International Conference on Quantum Software (QSW)*. IEEE, 2023, pp. 186–192.

[11] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, "Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects," *Physical Review A*, vol. 89, no. 2, p. 022317, 2014.

[12] Y. Mao, Y. Liu, and Y. Yang, "Qubit allocation for distributed quantum computing," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[13] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, and G. Bianchi, "Quantum internet: Networking challenges in distributed quantum computing," *IEEE Network*, vol. 34, no. 1, pp. 137–143, 2020.

[14] D. Cuomo, M. Caleffi, K. Krsulich, F. Tramonto, G. Agliardi, E. Prati, and A. S. Cacciapuoti, "Optimized compiler for distributed quantum computing," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, pp. 1–29, 2023.

[15] L. Stephenson, D. Nadlinger, B. Nichol, S. An, P. Drmota, T. Ballance, K. Thirumalai, J. Goodwin, D. Lucas, and C. Ballance, "High-rate, high-fidelity entanglement of qubits across an elementary quantum network," *Physical review letters*, vol. 124, no. 11, p. 110501, 2020.

[16] P. C. Humphreys, N. Kalb, J. P. Morits, R. N. Schouten, R. F. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, "Deterministic delivery of remote entanglement on a quantum network," *Nature*, vol. 558, no. 7709, pp. 268–273, 2018.

[17] S. Krastanov, H. Raniwala, J. Holzgrafe, K. Jacobs, M. Lončar, M. J. Reagor, and D. R. Englund, "Optically heralded entanglement of superconducting systems in quantum networks," *Physical Review Letters*, vol. 127, no. 4, p. 040503, 2021.

[18] Z. Davarzani, M. Zomorodi-Moghadam, M. Houshmand, and M. Nouri-Baygi, "A dynamic programming approach for distributing quantum circuits by bipartite graphs," *Quantum Information Processing*, vol. 19, pp. 1–18, 2020.

[19] R. G Sundaram, H. Gupta, and C. Ramakrishnan, "Efficient distribution of quantum circuits," in *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[20] D. Ferrari, S. Carretta, and M. Amoretti, "A modular quantum compilation framework for distributed quantum computing," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–13, 2023.

[21] M. Zhu, H. Fu, J. Wu, C. Zhang, W. Xie, and X.-Y. Li, "Ecmas: Efficient circuit mapping and scheduling for surface code," in *2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2024, pp. 158–169.

[22] S. Babaie and C. Qiao, "Towards distributed quantum error correction for distributed quantum computing," in *2025 International Conference on Quantum Communications, Networking, and Computing (QCNC)*. IEEE, 2025, pp. 66–73.

[23] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, 2019.

[24] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, "Superconducting qubits: Current state of play," *Annual Review of Condensed Matter Physics*, vol. 11, no. 1, pp. 369–395, 2020.

[25] L. Childress and R. Hanson, "Diamond nv centers for quantum computing and quantum networks," *MRS bulletin*, vol. 38, no. 2, pp. 134–138, 2013.

[26] T. Graham, Y. Song, J. Scott, C. Poole, L. Phuttitarn, K. Jooya, P. Eichler, X. Jiang, A. Marra, B. Grinkemeyer *et al.*, "Multi-qubit entanglement and algorithms on a neutral-atom quantum computer," *Nature*, vol. 604, no. 7906, pp. 457–462, 2022.

[27] A. Shnirman, Y. Makhlin, and G. Schön, "Noise and decoherence in quantum two-level systems," *Physica Scripta*, vol. 2002, no. T102, p. 147, 2002.

[28] J. M. Martinis, "Qubit metrology for building a fault-tolerant quantum computer," *npj Quantum Information*, vol. 1, no. 1, pp. 1–3, 2015.

[29] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, no. 2, p. 1098, 1996.

[30] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 86, no. 3, p. 032324, 2012.

[31] C. Knaut, A. Suleymanzade, Y.-C. Wei, D. Assumpcao, P.-J. Stas, Y. Huan, B. Machielse, E. Knall, M. Sutula, G. Baranes *et al.*, "Entanglement of nanophotonic quantum memory nodes in a telecom network," *Nature*, vol. 629, no. 8012, pp. 573–578, 2024.

[32] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical review A*, vol. 52, no. 5, p. 3457, 1995.

[33] Y. Liu, Y. Mao, X. Xu, X. Shang, F. Ye, and Y. Yang, "A nonblocking multistage switching network for distributed quantum computing," *IEEE Transactions on Networking*, 2025.

[34] Y. Mao, Y. Liu, X. Xu, X. Shang, and Y. Yang, "Performance analysis of interconnection networks for distributed quantum computing," in *2024 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2024, pp. 2785–2790.

[35] H. J. Kimble, "The quantum internet," *Nature*, vol. 453, no. 7198, pp. 1023–1030, 2008.

[36] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, p. eaam9288, 2018.

[37] R. Horodecki, M. Horodecki, and P. Horodecki, "Teleportation, bell's inequalities and inseparability," *Physics Letters A*, vol. 222, no. 1-2, pp. 21–25, 1996.

[38] M. Horodecki, P. Horodecki, and R. Horodecki, "General teleportation channel, singlet fraction, and quasidistillation," *Physical Review A*, vol. 60, no. 3, p. 1888, 1999.

[39] A. Yimsiriwattana and S. J. Lomonaco Jr, "Generalized ghz states and distributed quantum computing," *arXiv preprint quant-ph/0402148*, 2004.

[40] S. Sahni, "Computationally related problems," *SIAM Journal on computing*, vol. 3, no. 4, pp. 262–279, 1974.

[41] P. M. Pardalos and S. A. Vavasis, "Quadratic programming with one negative eigenvalue is np-hard," *Journal of Global optimization*, vol. 1, no. 1, pp. 15–22, 1991.

[42] O. Daei, K. Navi, and M. Zomorodi-Moghadam, "Optimized quantum circuit partitioning," *International Journal of Theoretical Physics*, vol. 59, no. 12, pp. 3804–3820, 2020.

[43] P. Andres-Martinez and C. Heunen, "Automated distribution of quantum circuits via hypergraph partitioning," *Physical Review A*, vol. 100, no. 3, p. 032308, 2019.

[44] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

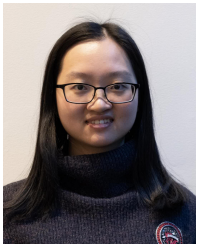
[45] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with qiskit," 2024. [Online]. Available: https://arxiv.org/abs/2405.08810

[46] B. Bixby, "The gurobi optimizer," *Transp. Re-search Part B*, vol. 41, no. 2, pp. 159–178, 2007.

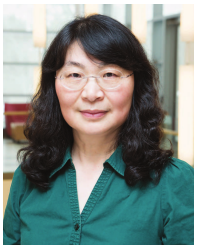
[47] Jay Gambetta and Ryan Mandelbaum. (2024) IBM Quantum delivers on performance challenge made two years ago. [Online; accessed 11-Aug-2025]. [Online]. Available: https://www.ibm.com/quantum/blog/qdc-2024

Xu Xu received the B.S. degree in Physics from Nanjing University, Nanjing, China, in 2023. He is currently pursuing a Ph.D. in the Department of Electrical and Computer Engineering at Stony Brook University. His research interests include quantum networks, distributed quantum computing, and quantum communication.

Yu Liu received the B. Eng. degree in Telecommunications Engineering from Xidian University in Xi'an, China. He received the Ph.D. degree in Computer Engineering from Stony Brook University, Stony Brook, New York. He is currently an Assistant Professor in the Department of Computing at the Hong Kong Polytechnic University, Hong Kong. His research interests encompass edge computing and networks, low-earth orbit satellite networks, online algorithm design, network function virtualization, and distributed quantum computing.



Yingling Mao received the B.S. degree in Mathematics and Applied Mathematics in Zhiyuan College from Shanghai Jiao Tong University, Shanghai, China, in 2018. She is currently working toward the Ph.D degree in the Department of Electrical and Computer Engineering, Stony Brook University. Her research interests include network function virtualization, software-defined networks, cloud computing, and distributed quantum computing.



Yuanyuan Yang received the B.Eng. and M.S. degrees in computer science and engineering from Tsinghua University, Beijing, China, and the M.S.E. and Ph.D. degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is currently a SUNY Distinguished Professor in the Department of Electrical & Computer Engineering and Department of Computer Science at Stony Brook University, New York, which she joined in 1999. From 2018-2022, she served as a program director in the National Science Foundation's Directorate of Computer and Information Science and Engineering. She directed the core computer architecture program and was on the management team of several cross-cutting programs. She has authored or coauthored more than 550 papers in major journals and refereed conference proceedings and holds seven US patents in her areas of research which include quantum computing, edge/cloud computing, and mobile computing. She has served as the Editor-in-Chief for IEEE Transactions on Cloud Computing, and the Associate Editor-in-Chief and Associated Editor for IEEE Transactions on Computers. She is currently an Associate Editor for IEEE Transactions on Parallel and Distributed Systems and IEEE Transactions on Sustainable Computing. She was also the General Chair, Program Chair, or Vice Chair for several major conferences and a Program Committee Member for numerous conferences. She is a Fellow of National Academy of Inventors (NAI) and a Fellow of IEEE.