Handling Qubit Allocation and Topology Design at Scale for Distributed Quantum Computing

Jiyao Liu, *Student Member, IEEE*, Lei Fan, *Senior Member, IEEE*, Yuanxiong Guo, *Senior Member, IEEE*, Zhu Han, *Fellow, IEEE*, and Yu Wang, *Fellow, IEEE*

***Abstract*—Distributed Quantum Computing (DQC) enables the execution of quantum circuits across multiple interconnected quantum processing units (QPUs) but requiring efficient qubit allocation and network topology design to optimize computational performance. Proper qubit allocation minimizes entanglement costs across QPUs, balances computational workload, and ensures efficient execution of quantum computing tasks. Meanwhile, network topology plays a crucial role in reducing entanglement routing complexity and communication overhead for remote quantum gate operations. Therefore, in this paper, we propose a joint optimization framework for network topology design and qubit allocation in DQC to minimize communication overhead. We formulate the problem as a tractable integer nonlinear programming model that explicitly incorporates entanglement routing, thereby ensuring a more tractable optimization process. To improve computational efficiency, we present a partially linearized version of the problem, making it solvable using any classical optimization solver. To further address the scheduling of large-scale quantum circuits, we develop heuristics from edge contraction in the qubit interaction graph to pre-assign groups of qubits to QPUs, ensuring that our method remains effective at scale. Extensive simulations on real-world quantum circuits over various quantum clusters or distributed quantum networks validate the effectiveness of our proposed approach, demonstrating its capability to handle complex quantum circuits while reducing communication costs in DQC.**

***Index Terms*—Topology Design, Qubit Allocation, Quantum Networks, Distributed Quantum Computing**

I. Introduction

Distributed quantum computing (DQC) [1] represents a promising paradigm for scaling quantum computing beyond the physical limitations of individual quantum processing units (QPUs). In DQC, multiple quantum nodes, each with its own QPU processor, are interconnected via quantum communication channels (as illustrated in Fig. 1) to collaboratively execute tasks that exceed the capacity of any single node. Such an approach leverages entanglement and quantum swapping over the underlying quantum network to facilitate inter-node communication, thereby enabling the distribution of quantum states and operations. DQC has great potential for solving large-scale problems in areas like optimization, artificial intelligence [2], cryptography, and material simulation, thus positioning itself as a cornerstone for future quantum technologies.

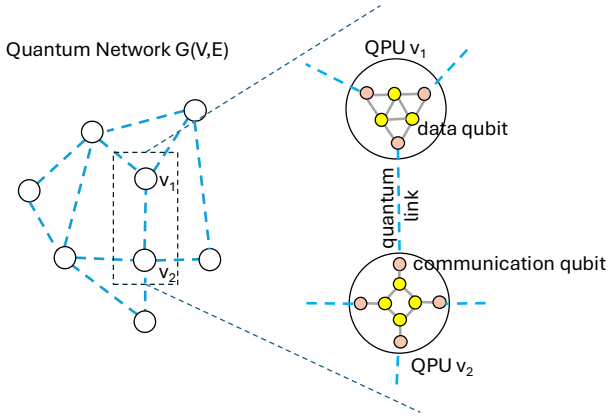


Fig. 1. Distributed quantum computing (DQC): a set of QPUs are interconnected via a quantum network, and each QPU has certain amount of physical qubits which can be used as data qubits or communication qubits. The quantum link between QPUs cloud be a multi-hop path.

Distributed quantum computing differs from classical distributed computing in several key ways. DQC leverages qubits, which can exist in superposition and entanglement, enabling parallelism beyond classical capabilities. However, unlike classical distributed systems, where data can be copied and transmitted via inter-connected networks easily, DQC relies on quantum entanglement and teleportation for quantum communication [3]–[7], facing challenges due to qubit fragility and decoherence. Therefore, while classical distributed systems scale efficiently with established resource management techniques, DQC requires more careful qubit allocation and network topology design.

In DQC, a quantum algorithm (or quantum circuit) over multiple data qubits (called logical quibts) needs to be distributed to a group of QPUs so that it can be collaboratively performed. Each QPU has certain physical qubits which can be used for either computation or communication purposes, as shown in Fig. 1. *Qubit allocation* is the process of efficiently assigning and managing qubits across multiple interconnected QPUs to optimize computational performance, as shown in Fig. 2. Proper qubit allocation is crucial for DQC because: 1) Allocating qubits must minimize the need for long-distance entanglement generation between QPUs, which is costly due to decoherence and fidelity loss; 2) Qubits are a scarce resource, and their allocation must balance computational workload and entanglement needs; and 3) Allocating qubits efficiently

J. Liu and Y. Wang are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19112. Email: {jiyao.liu, wangyu}@temple.edu. L. Fan and Z. Han are with the Department of Engineering Technology and the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004, respectively. Email: lfan8@central.uh.edu, hanzhu22@gmail.com. Y. Guo is with the Department of Information Systems and Cybersecurity, University of Texas at San Antonio, San Antonio, TX 78249. Email: yuanxiong.guo@utsa.edu. Y. Wang is the corresponding author. The work is partially supported by the US National Science Foundation (Grant No. CNS-2106761, CNS-2318683, CNS-2318663, ECCS-2302469, ECCS-2541996, and OAC-2417716) and an internal grant provided by the Office of the Vice President for Research (OVPR) at Temple University.

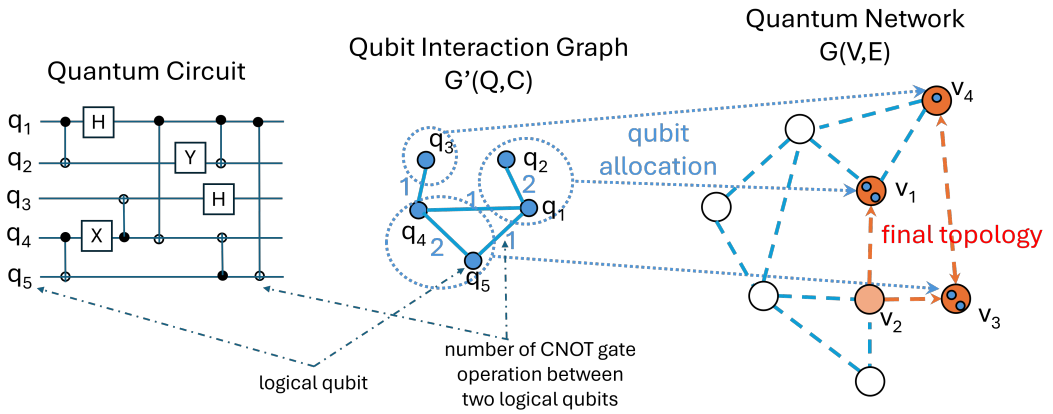


Fig. 2. Qubit allocation and topology in DQC: mapping a qubit interaction graph (QIG, modeled by graph G') onto the quantum network (QN, modeled by graph G) to minimize the communication cost. Here, the weight on a link in G' is the number of two-qubit gate operations between two logical quibts. Note that the quantum circuit is for illustration purposes only, and it is by no means meant as a realistic circuit.

ensures that distributed quantum computing tasks are executed with minimal latency and maximal parallelism. Effective qubit allocation strategies also need to consider *network topology*, available quantum links and entanglement resources, and error rates to enhance the efficiency of DQC systems.

Current research in distributed quantum computing or quantum networks focuses on several key challenges separately such as optimizing qubit allocation [8]–[10] or circuit distribution [11]–[20] across nodes to reduce communication cost, designing efficient network topologies [21], [22] to better serve given demands, optimizing quantum compiler framework [23]–[25], and overcoming the inherent noise and decoherence [26]–[28] to improve connection quality in DQC. [29] considers both qubit allocation and topology but still optimize them in separated stages.

Motivated by the above facts, we consider the co-design of network topology and qubit allocation for DQC to minimize the total communication overhead while fulfilling faster and reliable entanglement distribution for supporting the execution of remote gates in DQC tasks. Note that DQC tasks require frequent qubit iterations and entanglement swappings between QPUs in the underlying quantum network. A well-designed topology can minimize entanglement routing complexity, reducing communication cost and decoherence. While Mao *et al.* [29] have studied a similar topology design problem in DQC, they do not explicitly model the swapping routing into the optimization problem, which makes their optimization problem challenging to solve. In addition, solely relying on the shortest paths between QPUs as the entanglement route limits the performance of their proposed solution. In this paper, we first formulate the co-design problem as an integer nonlinear programming problem, where the entanglement routing has been embedded directly in the formulated problem. Then, we further convert this problem into a simpler format via partial linearization, which enable more efficient solving by the solver. Both problems can be solved by any classical solver, such as Gurobi [30] and CPLEX [31]. However, if the quantum circuit is large, directly solving even the partially linearized one is still challenging. Therefore, we further propose several heuristics to i) contract edges in the qubit interaction graph so that some logical qubits are grouped and distributed to QPUs together, and/or ii) pre-assign qubit groups to QPUs to simplify the optimization. By doing so the proposed methods are able to handle larger scale quantum circuits. Finally, we conduct extensive simulations using real-world quantum circuits over various quantum clusters and distributed quantum networks. Results confirm the efficiency of our proposed formulation and solution.

In summary, the highlights of this paper are as follows.

- We consider the joint topology design and qubit allocation problem in DQC with the goal of minimize the total communication cost while serving all demanding remote gate operation requests in DQC. We formulate this problem as an integer non-linear programming by explicitly defining the entanglement path between QPUs. We then further provide a simplified version of the problem which enable more efficient solving by the classical solver.
- To further improve the chance of efficient solving of the formulated problem over larger quantum circuits, we also propose edge contraction and pre-assignment based solutions to reduce the complexity of qubit allocation problem by grouping certain logic qubits into groups (and distributed them to the same QPU), and/or pre-assign qubit groups to QPUs.
- We conduct extensive simulations to evaluate our proposed methods and their applications in various settings and applications. Simulations have demonstrated that our proposed further simplification, edge contraction, and pre-assignment methods make the complex problem easier to solve at scale.

The remainder of this paper is organized as follows. Section II briefly reviews the existing works in DQC and Section III introduces our system model. The joint optimization problem is formulated in Section IV. Section V shows how to handle large scale quantum circuits. Evaluations of the proposed method are provided in Section VI. Finally, Section VII concludes the paper with possible future directions. A preliminary version of this paper appears as [32].

II. Related Works

DQC [1] has three different architypes: multi-core, multi-computer, and multi-farm architectures. Regardless of the DQC types, qubit allocation plays a crucial role in optimizing

computation efficiency, minimizing communication overhead, and ensuring the reliability of quantum circuits across multiple QPUs. In this section, we briefly review existing works on qubit allocation in DQC.

A. Quantum Circuit Partitioning and Scheduling

Many works have investigated quantum-circuit partitioning for distributed execution in DQC. Sundaram *et al.* [11] first partition a circuit across QPUs and then determine a small set of qubit migrations, based on cat-entanglement operations, to execute non-local gates. Their subsequent works [12], [13] explore teleportation-based circuit distribution over quantum networks, aiming to minimize the number of teleportations required for gates spanning multiple QPUs. Dadkhah *et al.* [14] reorder qubit placement to improve execution time and then use a genetic algorithm to partition the circuit while reducing teleportation costs. Baker *et al.* [15] propose time-sliced circuit partitioning for modular quantum architectures, optimizing the mapping of each time slice to reduce data-movement costs between consecutive slices. Other works [16]–[20] employ techniques such as hypergraph partitioning [19], dynamic programming [17], and evolutionary algorithms [20] to optimize circuit distribution.

These works primarily minimize inter-QPU communication or teleportation costs, generally without explicitly considering network conditions and entanglement availability. Ferrari *et al.* [23], [24] study DQC compiler design, derive theoretical bounds on compilation overhead, and develop a modular compilation framework that considers both network and device constraints. More recently, Zhang *et al.* [33] optimize quantum-communication scheduling in quantum data centers by overlapping cross-rack communication through collective and parallel entanglement generation while avoiding deadlock and buffer congestion.

Overall, existing circuit-distribution methods primarily optimize remote operations or teleportations over a fixed or implicit interconnect. A direct objective-value comparison with our method is therefore not like-for-like, as our decision space additionally includes network-link selection and entanglement-path routing. In Section VI, we instead compare against simulated annealing used by [29] and Kernighan–Lin graph partitioning [34], using a common downstream solver and time budget.

B. Network-Aware Qubit Allocation

Several recent studies have incorporated more network considerations into qubit allocation in DQC. Mao *et al.* [10] formulate a qubit allocation problem where the network cost of a remote CNOT is assumed linear to the number of hops of the shortest path between two QPUs in a quantum network. They propose a heuristic local search algorithm and a multistage hybrid simulated annealing algorithm to solve the formulated qubit allocation problem. The same authors [8] further extend the problem to a probability-aware qubit-to-processor mapping model, where the network overhead between two QPUs is determined through probabilistic analyses based on the link entanglement generation rates. A multistage hybrid simulated annealing algorithm with multi-flow routing is then proposed to minimize the total network overhead. Cuomo et al. [25] also explore compiler optimizations tailored for QDC, aiming to minimize communication costs with inter-processor connectivity restrictions. They formulate the distributed compilation problem as a dynamic network flow problem, where a flow is a set of entanglement paths used by the telegates.

C. Network Topology Design

The network topology has a significant impact on the efficiency of quantum network. Yu *et al.* [21] discuss quantum internet topology design, emphasizing on the impact of topology on the rate of entanglement distribution and the fidelity of entangled pairs, when serving multi-commodity quantum communication demands. Liu *et al.* [22] also investigate how network topology can be optimized alongside resource allocation and entanglement distribution strategies to improve the performance of quantum networks.

Only recently has the joint optimization of network topology and qubit allocation gained attention in DQC. Mao *et al.* [29] consider such a joint topology design problem in DQC for a set of test quantum circuits, with the goal of minimizing the overall communication overhead after distributing the quantum circuits. The communication overhead depends on an "implicit function" of distance between each pair of GPUs based on the shortest paths in the topology. Such implicit function make the mathematical deduction and simplification of the formulated problem much harder. The authors design a meta-heuristic algorithm based on simulated annealing to find the near-optimal topology for random quantum circuits. They also provide an extended algorithm to generate dedicated network topologies tailored for specific quantum circuits. Since their proposed methods limit the entanglement path to the shortest path in the topology, the solution space is restricted to a much smaller space than the real feasible space which hurts their performances. In this paper, we consider the similar topology co-design with qubit allocation for DQC, but provide a formulation of the problem with an explicit expression of entanglement paths so that the formulated problem can be solved by existing classical solvers directly.

There are also efforts to build quantum data center (QDC) where multiple QPUs are integrated into a cohesive infrastructure for next-generation DQC. Usually, a network topology similar to classical date center network is used for interconnection. For example, CISCO [35], [36] uses topologies of FatTree [37] and BCube [38] for their QDC design, while SwitchQNet [33] uses FatTree for their QDC. However, these topologies are static and designed for general traffic demands. In this paper, we consider the co-design of topology with qubit allocation with known DQC demands.

III. System Model

In this section, we first present our DQC model and the model for underlying quantum network (QN), then introduce the qubit allocation and topology formation problem where the communications among logical qubits are transferred to the communications among QPUs. Major notations used in this paper are summarized in Table I. We distinguish two levels of qubits throughout the paper. A *logical qubit* q_a is a state carrier

TABLE I
LIST OF SYMBOLS.

Symbol	Description
Q, q_a	the set of logical qubits and a-th logical qubit
C	the set of edge (CNOT gates between qubits) in QIG
$c_{a,b}$	the number of CNOT instances between q_a and q_b
V, v_i	the set of QPUs and i-th QPU
E	the set of edges between QPUs in QN
$b_{u,z}$	the weight/cost of quantum link between v_u and v_z
m_i, n_i	computation/communication qubits capacities on QPU v_i
W	the maximum number of edges in the final topology
$x_{a,i}$	whether logical qubit q_a is assigned to QPU v_i
$p_{i,j,u,z}$	whether the path between v_i and v_j uses edge (v_u, v_z)
$w_{u,z}$	whether edge (v_u, v_z) used by selected paths
$y_{i,j}$	whether at least one RCNOT demand between v_i and v_j

specified by the input quantum circuit and is represented by a QIG vertex. A *physical data qubit* (or computation qubit) is a hardware resource inside a QPU. Qubit allocation assigns every logical qubit to one QPU, where it occupies one physical data-qubit slot.

A. Distributed Quantum Computing Model

Arbitrary quantum circuits can be decomposed into single-qubit and Controlled-NOT (CNOT) gates, which form a universal gate set [39]. Single-qubit gates are always local, while a CNOT is local if both qubits reside on the same QPU and remote otherwise. Thus, following prior DQC allocation studies [8], [10], our communication model only records CNOT interactions after basis-gate decomposition.

Since we concentrate on the communication cost of DQC, any quantum circuit can be abstracted into a *Qubit Interaction Graph* (QIG) to reflect the needs of number of CNOT operations without loss of generality. As shown in the middle of Fig. 2, QIG depicts between which two qubits there are CNOT gates and how many CNOT gates are needed. In the QIG, each node represents a logical qubit q_a, and each edge (q_a, q_b) represents the existence of CNOT gates between two qubits q_a and q_b. If there are more than one CNOT gates required between these two qubits, the edge weight $c_{a,b}$ is the number of CNOT gates required between qubits pair (a, b). Given a quantum circuit, we can form its QIG $G'(Q, C)$, where Q is the set of all its qubits and C is the edge set of CNOT gates between the qubits. For example, in the QIG in Fig. 2, we have $c_{1,5} = 1$ and $c_{1,2} = 2$ since there are one CNOT gate between q_1 and q_5 and two CNOT gates between q_1 and q_2 in the circuit, respectively. Obviously, the QIG captures all possible communications in the corresponding circuit, thus informative enough in the context of communication cost minimization. Note that the weight $c_{a,b}$ aggregates the *count* of CNOT instances between q_a and q_b for communication-cost calculation; it does not combine several physical gates into a single CNOT or remote CNOT gate.

After qubit allocation, a CNOT whose control and target logical qubits reside on the same QPU is executed locally and requires no network communication. For example, in Fig. 2, since q_1 and q_2 are mapped to the same server v_1, the two CNOT gates between them can be executed entirely by v_1 without consuming pre-shared network Bell pairs, even though $c_{1,2} \neq 0$. When the two logical qubits reside on different QPUs, the CNOT is implemented as a remote CNOT

Fig. 3. Quantum swapping in QN: A swap operation at v_2 to establish the entanglement between v_1 and v_3.

(RCNOT), which consumes an end-to-end Bell pair shared by the corresponding QPUs. For example, q_4 and q_5 each have an RCNOT interaction with q_1. Since q_4 and q_5 are mapped to v_3 while q_1 is mapped to v_1, these RCNOTs require pre-shared network Bell pairs between v_1 and v_3 for their collaborative execution. We refer readers to [29] for the circuit-level implementation and operation sequence of RCNOTs. Although RCNOTs can be implemented in different ways and circuit optimizations (such as circuit transformation or remote gate scheduling) may reduce the number of required RCNOTs, these topics are beyond the scope of this study.

B. Quantum Network and Resources

The studied quantum network (QN) is a network of QPUs which hold physical qubits. We should map the logical qubits in quantum circuits to the physical qubits on quantum processors in QN to execute the circuit. Each of the QPU v_i has a memory capacity to hold up to m_i physical data qubits (or called computation qubits). Except for offering physical qubits on its nodes, quantum networks are also responsible for providing entanglements for RCNOT gates. Those entanglements are first generated by the edges in the quantum network, and then connected to form E2E entanglements between processors where there are RCNOT gates. We can form a graph $G(V, E)$ for the underlying quantum network, where V is the set of processors/QPUs v_i in the network, and E is the set of edges (possible quantum links) among the QPUs. For the quantum link (v_i, v_j) between QPUs v_i and v_j, we can define an edge weight $b_{i,j}$ as the cost factor of that quantum link. For different DQC types (such as multi-core or multi-farm), this edge weight can be defined differently and accordingly.

We focus on the joint optimization of logical-qubit assignment and inter-QPU network topology[1] (including entanglement routes) to facilitate circuit execution. Accordingly, we abstract QPU v_i by its capacities of m_i physical data qubits and n_i communication qubits/interfaces. We assume that a local compiler can route an assigned data qubit to an available communication interface, and we do not charge the resulting intra-QPU SWAP cost. Applying the same abstraction to all methods isolates the inter-QPU co-design problem studied here. It is not physically feasible to implement an arbitrary number of direct quantum links when a DQC system contains many QPUs, especially for an on-chip or tightly integrated system. For N QPUs, a complete graph requires $N(N-1)/2$ physical links and $N-1$ communication interfaces at every QPU. These requirements grow rapidly with N, while every link also consumes communication qubits or ports, interconnect channels, layout area, stabilization resources, and control

[1]The internal design of an individual QPU, including its physical-qubit coupling graph and the placement of communication qubits, can affect the local SWAP operations, latency, and fidelity of an RCNOT. However, optimizing that internal design is outside the scope of this paper.

hardware. Thus, forming a complete or dense physical graph becomes infeasible at scale. Prior DQC topology co-design and QDC-network studies similarly consider restricted link sets, node degrees, or port counts [29], [33], [35]. We model these physical limitations at two levels: n_i limits the number of links incident to QPU v_i, and W limits the total number of physical links that may be implemented across the network. These cardinality limits represent implementable topology resources, whereas $b_{i,j}$ separately represents the communication cost of using link (v_i, v_j).

To build E2E entanglements between two QPUs which are not directly connected by quantum links in G, quantum swappings are performed on the intermediate nodes. Fig. 3 shows how two adjacent entanglements can be connected. At the beginning, we have two crude entanglements between (v_1, v_2) and (v_2, v_3). They are generated by the quantum link between those nodes. There is no direct link between (v_1, v_3) so we are unable to obtain an entanglement between them directly. However, we can perform a swapping operation (using Bell State Measurement (BSM) gates) on node v_2 to connect entanglements (v_1, v_2) and (v_2, v_3), after which we can obtain an entanglement between (v_1, v_3). Such swapping operations can be repeated to obtain a pair of E2E entanglements on desired node pairs in QN. These E2E entanglements between processor pairs can be used by RCNOT gates. The classic communications parts in RCNOT or swapping operations are ignored here as they do not affect our design. Entanglement generation, swapping, and RCNOT have distinct roles. Generation creates a Bell pair on one physical link; swapping uses a Bell-state measurement at an intermediate QPU to connect adjacent link-level pairs into an end-to-end pair; and RCNOT consumes that end-to-end pair to execute a circuit gate. Swapping may fail, and reported implementation bounds on its success probability vary (e.g., 50% [40], 62.5% [41], 75% [42], or deterministic but less efficient operation [43]). Accurately quantifying a swapping failure is complicated because it may affect entanglement-regeneration and recovery procedures, latency, throughput, and end-to-end fidelity. We therefore do not claim to model these effects accurately.

For tractability, we optimize a static weighted communication cost. The objective explicitly penalizes a long entanglement path by including its weighted path length, $\sum_{u,z} b_{u,z} p_{i,j,u,z}$, so every additional link increases the cost when link weights are positive. This term is a proxy for the additional link-level entanglements and swapping operations required by a long path; it is not a stochastic model of failures or retries. The present formulation also does not explicitly optimize gate timing, throughput, fidelity evolution, failure recovery, or intra-QPU routing. More accurate joint modeling of these effects is a valuable direction for future work.

C. Qubit Allocation and Topology Formation

With the given QIG G' and QN G, we can consider a joint topology design and qubit allocation problem to optimize the total communication cost of DQC. As shown in Fig. 2, a qubit allocation solution is essentially a graph partition: qubits assigned to the same QPU are one group, and the cross-group edges are the communications among the QPUs. In Fig. 2, the group of q_1 and q_2 are mapped to v_1 and the group of q_4 and q_5 are mapped to v_3. Then, the cross-edges between the two groups are the communications between v_1 and v_3 (2 RCNOTs in this case). Additionally, if there is no direct quantum link between v_1 and v_3, the RCNOT between q_1 and q_4 can only be served by creating E2E entanglements by the QN, i.e., creating one entanglement between (v_1, v_2) and (v_2, v_3) separately and connect them by a swapping on v_2 as done in Fig. 3. In Fig. 2, the result of this joint topology design and qubit allocation include both the qubit mapping from G' to G and the resulting network topology (a subgraph of G including all quantum links and entanglement paths for inter-QPU communications).

The joint problem is NP-hard by restriction. Fix the QN topology and all entanglement routes, use uniform inter-QPU communication costs, and retain only the logical-qubit assignment variables. The remaining problem partitions the weighted QIG among capacity-limited QPUs so as to minimize the total weight of cross-group edges; this is capacitated weighted graph partitioning, an NP-hard problem. Hence, a polynomial-time algorithm for our joint problem would solve this NP-hard special case, proving that the more general joint qubit-allocation and topology-design problem is NP-hard.

IV. Co-Design of Network Topology and Qubit Allocation: A Joint Optimization

We now formally define the optimization problem of our topology co-design with qubit allocation. We first present it in the most intuitive original formulation, then reformulate it into an equivalent but more tractable problem. Both versions can be solved by the existing integer programming solvers, such as Gurobi and CPLEX.

In our model we use two graphs G' and G to model QIG and QN respectively. Both G' and G are undirected. However, routing E2E entanglements in QN requires finding the best paths between any two nodes in G. This is challenging while the topology itself is to be determined. Note that [29] uses an iterative heuristic solution to optimize topology and allocation which is complicated. In our formulation, we use a directed QN topology so that the entanglement path can be implicitly embedded into our joint optimization as constraints, which enable efficient solutions from existing solver.

A. Problem Formulation

We first define our objective function, which is minimizing the total communication cost of the DQC, i.e., the summation of the communication cost incurred between any QPU pairs (v_i, v_j) in QN G. For each node pair (v_i, v_j), its entanglement is routed via an entanglement path between them. Therefore, the optimization objective is

$$\sum_{i<j} \left(\sum_{u,z} b_{u,z} p_{i,j,u,z} \sum_{a,b} c_{a,b} x_{a,i} x_{b,j} \right). \tag{1}$$

Here, $x_{a,i}$ is a binary indicator whether logical qubit $q_a \in Q$ is assigned to server $v_i \in V$, and $p_{i,j,u,z}$ is a binary indicator whether the entanglement path between servers v_i and v_j uses the directed edge (v_u, v_z). In other words, $x_{a,i}$ and $p_{i,j,u,z}$

are the decision variables for qubit allocation and network topology, respectively. Recall that $c_{a,b}$ is the number of CNOT gates between q_a and q_b in QIG, thus $\sum_{a,b} c_{a,b}x_{a,i}x_{b,j}$ is the total number of RCNOT required between servers v_i and v_j. $b_{u,z}$ is the weight/cost of edge (v_u, v_z), and therefore $\sum_{u,z} b_{u,z}p_{i,j,u,z}$ is the weighted length of the entanglement path between v_i and v_j. Thus, the cost of a long path is incorporated directly into the objective: every additional physical link adds its weight to the cost. With unit link weights, this term is the hop count and captures the additional link-level Bell pairs and swapping operations required along a longer path. Scenario-dependent weights can additionally represent distance or other coarse link costs. This weighted-path-length term does not, however, accurately quantify stochastic swapping failures, retry latency, throughput loss, failure recovery, or fidelity changes. Our overall objective is to minimize the communication cost of all RCNOT gates between all server pairs in the network.

We now present the original formulation of our joint optimization problem as follows.

$$\textbf{P1}: \min_{x,p} \sum_{i<j} \left(\sum_{u,z} b_{u,z} p_{i,j,u,z} \sum_{a,b} c_{a,b} x_{a,i} x_{b,j} \right) \tag{2}$$

s.t. qubit allocation

$$\sum_i x_{a,i} = 1, \qquad \forall q_a \in Q, \tag{2a}$$

$$\sum_a x_{a,i} \le m_i, \qquad \forall v_i \in V, \tag{2b}$$

topology sparsity

$$\sum_z \left[1 - \prod_{i<j} (1 - p_{i,j,u,z})(1 - p_{i,j,z,u}) \right] \le n_u, \forall v_u \in V, \tag{2c}$$

$$\sum_{u<z} \left[1 - \prod_{i<j} (1 - p_{i,j,u,z})(1 - p_{i,j,z,u}) \right] \le W, \tag{2d}$$

entanglement paths

$$\left[\sum_u p_{i,j,i,u} - \sum_z p_{i,j,z,i} - 1 \right] \sum_{a,b} x_{a,i} x_{b,j} = 0, \forall i < j, \tag{2e}$$

$$\sum_z p_{i,j,u,z} - \sum_z p_{i,j,z,u} = 0, \quad \forall i < j, \forall u \notin \{i, j\}, \tag{2f}$$

$$\left[\sum_u p_{i,j,j,u} - \sum_z p_{i,j,z,j} + 1 \right] \sum_{a,b} x_{a,i} x_{b,j} = 0, \forall i < j, \tag{2g}$$

$$\sum_{u,z} p_{i,j,u,z} \le (|V| - 1) \sum_{a,b} x_{a,i} x_{b,j}, \quad \forall i < j, \tag{2h}$$

decision variables

$$x_{a,i}, p_{i,j,u,z} \in \{0, 1\}. \tag{2i}$$

Constraints (2a) and (2b) are the qubit allocation constraints, where the former ensures that each qubit is assigned to one and only one QPU and the latter limits the number of assigned qubits on each QPU not to exceed its memory capacity.

Constraint (2c) limits the number of edges that QPU v_u can have due to the limit of its communication qubits. The term $\prod_{i,j} (1 - p_{i,j,u,z})(1 - p_{i,j,z,u})$ is 1 when $p_{i,j,u,z} = p_{i,j,z,u} = 0$ for any i and j, which means the undirected edge (v_u, v_z) is not used by any path. Thus, the left side of Constraint (2c) is the number of edges associated to QPU v_z. Constraint (2d) limits the network sparsity (i.e., the number of edges in the final topology) not to exceed a threshold W. The left side of Constraint (2d) is similar to that of Constraint (2c) but it summarize over all edges in the topology rather than those associated to a specific QPU. Constraints (2c) and (2d) are hard physical-topology budgets rather than consequences of the usage-cost objective, following the limited-degree/link abstractions used in prior DQC topology design [29]. Constraint (2c) ensures that the degree of each QPU does not exceed its available communication interfaces, while Constraint (2d) uses W to cap the total number of implementable physical links. Without these constraints, the optimizer could select an arbitrarily dense physical graph that might not be realizable for a large number of QPUs. This separation lets the model trade a physically feasible sparse topology against the shorter entanglement paths enabled by additional links; Section VI evaluates this tradeoff by varying W.

Constraints (2e)-(2g) are the flow constraints to form the entanglement paths among QPUs. Ignoring the term $\sum_{a,b} x_{a.i}x_{b,j}$, they are typical three flow constraints for the source, intermediate, and destination nodes on the entanglement path: out-degree minus in-degree should be 1 at the source node, 0 at the intermediate nodes, and -1 at the destination node. $\sum_{a,b} x_{a.i}x_{b,j}$ is added at both sides to cancel the constraints if no demand over this path.

Constraint (2h) forces no entanglement path for pair (v_i, v_j) if it has no demand, avoiding disturbing topology constraints (Constraints (2b)-(2c)).

Finally, all decision variables $x_{a,i}$ and $p_{i,j,u,z}$ are binary. Therefore, this formulated joint optimization problem is a binary nonlinear programming.

B. Reformulation

The original formulation (**P1**) might still be hard for existing classic solvers (e.g., Gurobi and CPLEX) to solve because it has several high degree terms, which cloud be linearized by adding intermediate variables. Therefore, in this subsection we further simplify it into a partially linearized problem **P2**, which is equivalent to the original one but can be solved more efficiently by the classic solvers.

In **P2**, first, the objective and the qubits assignment constraints (3a) and (3b) are unchanged. We then introduce a binary indicator $w_{u,z}$ for edge existence: $w_{u,z} = 1$ if any entanglement paths use the directed edges (v_u, v_z) or (v_z, v_u), otherwise $w_{u,z} = 0$, and add Constraint (3e). With the help of this indicator, we can simplify the topology constraints (Constraints (3c) and (3d)). Similarly, we introduce a demand indicator $y_{i,j}$ by adding Constraint (3f): $y_{i,j} = 1$ if there is at least one RCNOT demand between servers v_i and v_j, otherwise $y_{i,j} = 0$. This simplifies path constraints (Constraints (3g), (3i), and (3j)). By introducing $w_{u,z}$ and $y_{i,j}$,

Constraints (2c), (2d), (2e), (2g) and (2h) are all reduced to linear. These reduction, while not changing the NP-hardness of the problem, greatly reduced the number of total variables (including intermediate variables) in the problem, enabling us to solve problems of larger scales.

$$\textbf{P2}: \min_{x,p,w,y} \sum_{i<j} \left(\sum_{u,z} b_{u,z} p_{i,j,u,z} \sum_{a,b} c_{a,b} x_{a,i} x_{b,j} \right) \tag{3}$$

s.t. qubit allocation

$$\sum_{i} x_{a,i} = 1, \qquad \forall q_a \in Q, \tag{3a}$$

$$\sum_{a} x_{a,i} \leq m_i, \qquad \forall v_i \in V, \tag{3b}$$

topology sparsity

$$\sum_{z} w_{u,z} \leq n_u, \qquad \forall v_u \in V, \tag{3c}$$

$$\sum_{u<z} w_{u,z} \leq W, \tag{3d}$$

$$w_{u,z} \leq \sum_{i,j} \left(p_{i,j,u,z} + p_{i,j,z,u}\right) \leq w_{u,z} \cdot \left(|V|(|V|-1)\right), \quad \forall u < z, \tag{3e}$$

entanglement paths

$$y_{i,j} \leq \sum_{a,b} x_{a,i} x_{b,j} \leq y_{i,j} \cdot m_i m_j, \qquad \forall i < j, \tag{3f}$$

$$\sum_{u} p_{i,j,i,u} - \sum_{z} p_{i,j,z,i} = y_{i,j}, \qquad \forall i < j, \tag{3g}$$

$$\sum_{z} p_{i,j,u,z} - \sum_{z} p_{i,j,z,u} = 0, \quad \forall i < j, \forall u \notin \{i,j\}, \tag{3h}$$

$$\sum_{u} p_{i,j,j,u} - \sum_{v} p_{i,j,v,j} = -y_{i,j}, \qquad \forall i < j, \tag{3i}$$

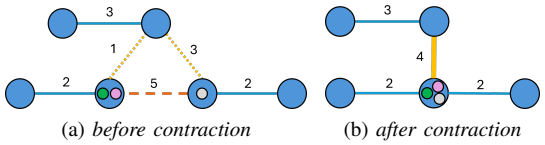
$$\sum_{u,z} p_{i,j,u,z} \leq y_{i,j}(|V|/2), \qquad \forall i < j, \tag{3j}$$

decision variables

$$x_{a,i}, p_{i,j,u,z}, w_{u,z}, y_{i,j} \in \{0,1\}. \tag{3k}$$

V. Tackling Large-Scale Circuits

Qubit assignment along is exactly a graph partition problem, which is known to be NP-hard. Therefore, classic solvers like Gurobi are not able to solve such problems at scale within a reasonable time. In practice, quantum compilers for DQC and/or QDC require an efficient solution to compile quantum programs in a reasonable time. In our evaluation section, Gurobi is not able to solve **P2** for many circuits with only 96 qubits in five minutes. However, practical quantum algorithms usually require hundreds to thousands of qubits. Leading industrial sectors also foresee to have hundreds of logical qubits in the near future (e.g., IBM predicts to have 230 logical qubits in five years). Therefore, scaling up the solving ability to hundreds of qubits is critical, given that existing solutions can only handle a limited number of qubits, from at most 15 in [29] to 32 in [32].



(a) *before contraction* (b) *after contraction*

Fig. 4. An example of edge contraction in QIG.

In the following subsections, we show how to conduct the pre-processing methods to handle large-scale quantum circuits. We first show how one single edge contraction works on a QIG, then devise a greedy algorithm to repeat the contraction process until the size of the QIG meets given criteria. Finally, we cloud further use the results from the greedy algorithm to pre-assign the grouped qubits to QPUs to facilitate the solving process. Note that all those pre-processing methods are performed before feeding the problem to the solver for solving **P2**.

A. Edge Contraction on QIG

To process a large Qubit Interaction Graph (QIG), we can shrink it by grouping some qubits into one group so that they will always be assigned onto the same QPU. In this way, we can reduce the number of vertices in QIG to make **P2** more tractable. This process is called edge contraction, which is a known operation in graph coarsening. In the context of QIG coarsening, Fig. 4 shows an example of the contraction on one single edge (the dash line in Fig. 4(a)) in the QIG. After the contraction, the edge is deleted, as the two group of qubits on the endpoints are now in one group (called super quibts hereafter) so no communications are required. At the same time, the two edges connecting the two nodes to their common neighbor is merged and the weights are summed as the weight of the merged edge. Other nodes and edges are kept unchanged. We use a function $\textsc{Contract}(H, c)$ to perform the contraction of edge c in a QIG H.

B. Greedy Edge Contraction

Repeating edge contraction multiple times allows us to shrink the QIG to a reasonably small size, determined by the stopping criteria. In the aggressive cases, the number of vertices in the QIG can be reduced to the same as the number of QPUs or even smaller.

We now propose a greedy contraction algorithm, as shown in Algorithm 1, which describes how we can contract a given QIG. It first sorts all edges by weight (communications requirement) in line 3, and try to pick the edge with largest weight without violating QPU capacity constraints in lines 6-11. The Feasibility-Check ensures that the node sizes S in the contracted QIG still has a feasible assignment to QUPs with capacity array M, so that the problem **P2** has at least one feasible solution. If no further contraction can be done, it terminates (lines 12-13). Otherwise, a possible edge is found and then contracted in line 15.

While the idea of greedy algorithm is simply selecting an edge with larger weight to contract, each contraction need to be carefully conduct because it may lead to infeasible problem for qubit assignment to QPUs. For example, if all QPUs can

Algorithm 1 GREEDY-EDGE-CONTRACTION

Input: Original QIG $G'(Q, C)$, QPU capacity array M.
Output: Contracted QIG H.

1: Copy G' to H
2: **repeat**
3: Sort all edges in H in descending order of their weights, denoted by $\mathcal{C}$ ▷ prioritize demanding edges
4: $c^* =$ Nill ▷ candidate edge for contraction
5: **for** each edge $c \in \mathcal{C}$ **do** ▷ find an edge for contraction
6: Assume q_a and q_b are the end nodes of edge c
7: Let S be the qubit size array of all nodes Q in H, i.e., $S = \{q.size | \forall q \in Q\}$
8: $S = S \backslash \{q_a.size, q_b.size\} \cup \{q_a.size + q_b.size\}$
9: **if** FEASIBILITY-CHECK(M, S) **then**
10: $c^* = c$
11: **break** ▷ a possible edge for contraction found
12: **if** $c^* =$ Nill **then**
13: **return** H ▷ no possible edge contraction
14: **else**
15: $H =$ CONTRACT(H, c^*) ▷ contract edge c^* in H
16: **until** no more edge can be contracted

hold at most 8 qubits, then no group of qubits in the contracted QIG should be greater than this size. Another infeasible case is, in heterogeneous systems, when there are 4 QPUs of size 8 and 4 QPUs of size 4, we should not have 5 groups of size 7. Simply figuring out whether a feasible assignment exists or not is NP-complete: the *Partition Problem* (a known NP-complete problem) can be reduced to this problem.

Therefore, we devise Algorithm 2 to perform the feasibility check efficiently. The trade-off here is that we only hope it can help to prevent infeasible QIGs, so it does not have to be accurate (missing some feasible cases). It tries to assign the largest super qubit to the QPU with the largest remaining capacity, until all super qubits are assigned. This algorithm not only tells the feasibility of a given contraction, but also outputs a feasible qubit allocation solution, which can be used to simplify the joint optimization (as discussed in Section V-C). Given $|S| \geq |M|$, it is easy to see that the complexity of this FEASIBILITY-CHECK implementation is $O(|S|^2 \log |S|)$, i.e., $O(|Q|^2 \log |Q|)$. Finally, the overall complexity of Algorithm 1 is $O(|C|(|C| \log |C| + |C||Q|^2 \log |Q|)) = O(|C|^2 \log |C| + |C|^2|Q|^2 \log |Q|)$. Here, $|C|$ and $|Q|$ are the number of edges and nodes in the QIG.

Note that after the edge contraction, we can use the new QIG H to formulate our joint optimization problem, such as **P2**. But it also needs a slight modification: constraint (3b) needs to be modified accordingly to reflect the memory usage of each super qubit (which can be larger than one).

C. Pre-Assignment Scheme

Notice that the FEASIBILITY-CHECK algorithm already gives us a possible pre-assignment: we can greedily assign the largest super qubit to the QPU with the largest remaining capacity, as suggested by line 4 in Algorithm 2. We can simply use this method to pre-assign the qubit allocation (all $x_{a,i}$ in

Algorithm 2 FEASIBILITY-CHECK

Input: QPU capacity array $M = \{m_i | v_i \in V\}$, super qubit size array $S = \{s_a | q_a \in Q\}$.
Output: whether the super qubits can fit in QPUs or not.

1: **while** $|S| > 0$ **do**
2: Sort M, S in decreasing order
3: **if** $S[0] \leq M[0]$ **then**
4: $M[0] = M[0] - S[0]$ ▷ remove used capacity
5: Remove $S[0]$ from S ▷ delete assigned qubit
6: **else**
7: **return** False
8: **return** True

P2). Doing so greatly simplifies the optimization problem and helps the solver (e.g., Gurobi or CPLEX) to solve the problem more efficiently. We will see later in evaluations show that this strategy significantly increases the scale of the problems that we can solve.

VI. EVALUATIONS

In this section, we conduct several sets of simulations to evaluate the proposed formulation and solution of our joint co-design problem by Gurobi [30] using real-world quantum circuits. We first focus on comparing the difference of the two formulations, especially how our re-formulation makes the problem more efficient to solve. After that, we compare the proposed edge contraction method with Kernighan–Lin graph partitioning [34] and evaluate how both methods, with and without pre-assignment, facilitate solving large-scale problems. Finally, we also verify the affect of quantum resources and network topology sparsity on the overall performances, based on our solving methods. Codes, including quantum circuit generation, problem formulation and optimization, and data visualization, are available at https://github.com/Jiyao17/topodqc.

A. Simulation Settings

We now introduce the quantum circuits and QPU clusters we use in our simulations.

1) Quantum Circuits: We choose important real-word applications as our testing circuits: Multi-Control Multi-Target (MCMT) Gates, Quantum Fourier Transformation (QFT), and Grover's Search Algorithm (Grover). Among them, MCMT is critical for benchmarking the quantum compilers for their gate decomposition ability; QFT is the core component of Shor's algorithm, which is potential to break the RSA-based encryption schemes; Grover is a black-box searching algorithm that can reduce the search space from N (on classic computers) to $\sqrt{N}$ (on quantum computers), and it searches for a random bit string (once) in our implementation. All of them are also used as the testing tasks in [33]. We mark the number of qubits used in those algorithms with -x, e.g., QFT-32 means the QFT algorithm with 32 qubits. All those algorithms are coded using IBM qiskit library, transpiled on the basis gates of arbitrary single qubit gates and CNOT gates, and converted to .qasm [44] (quantum assembly language files). Finally, by analyzing the .qasm files, we can determine the input parameters for the

TABLE II
QUANTUM RESOURCES AND TOPOLOGY SPARSITY REQUIREMENT OF DIFFERENT QPUS CLUSTERS.

Cluster Name-*total comp. qubit*	Number of QPUs	Comp. Qubit per QPU	Comm. Qubit per QPU	W
micro-4	2	2	1	1
micro-8	2	4	1	1
tiny-16	4	4	3	5
tiny-32	4	8	3	5
small-64	8	8	3	10
small-96	8	12	3	10
small-128	8	16	3	10
small-192	16	16 & 8	4 & 3	20
medium-256	16	16	4	40
medium-384	32	16 & 8	4 & 3	80
large-512	16	32	8	40
large-768	32	16 & 32	4 & 8	80
large-1024	32	32	8	80

For heterogeneous cases, we set two different qubit numbers per QPU for each half of all QPUs, listed as a & b in the cell.

optimization problem (e.g., number of CNOT gates $c_{a,b}$ for all logic qubit pairs (a, b)).

2) Quantum Clusters: We consider both homogeneous and heterogeneous QPU clusters, and the detail parameters are given by Table II. If not mentioned otherwise, QPUs are fully connected via an underlying QN G (subject to topology optimization) and all link costs $b_{u,z}$ are set to one unit. In Section VI-F, we use various network topologies and set different link costs depending on the application scenarios.

B. Compared Methods and Baselines

1) Solving Methods: We use the following baselines solutions from Section IV in our evaluations, which we call *TACO* methods for *Topology-Allocation Co-Optimization*. Considering that joint optimization is relatively new in this field, we can only find [29] as the most relevant baseline.

- *TACO*. We manually decompose the terms in the original formulation of **P1** whose degrees are higher than two to quadratic terms and use Gurobi to solve it.
- *TACO-NL*. We use the original formulation of **P1** and directly feed it to Gurobi with its newest non-linear expression feature. Gurobi is responsible for automatically decomposing all terms to linear/quadratic.
- *TACO-L*. We use Gurobi to diectly solve the reformulated problem of **P2**, which is partially linearized.
- *SA*. Solving **P2** with the meta-heuristic method Simulated Annealing. This method is also used in [29] to solve a similar problem.

2) Pre-Processing Methods: We evaluate the effectiveness of the proposed edge contraction and pre-assignment methods from Section V. To compare edge contraction against an established graph-partitioning approach, we also implement a capacity-aware recursive version of the Kernighan–Lin method [34]. EC and KL are alternative QIG pre-processing methods, and PA can be applied after either method.

- *EC*. Applying the greedy edge contraction (Algorithm 1) to the original QIG before solving the optimization problem.
- *KL*. Applying recursive Kernighan–Lin graph partitioning to the original weighted QIG before solving the optimization problem. The CNOT counts are used as edge weights, and partition sizes are selected according to QPU computation-qubit capacities.
- *PA*. Applying pre-assignment (Algorithm 2) to the output of EC or KL to pre-solve all $x_{a,i}$ in **P2** and simplify the problem.

TABLE III
EFFICIENCY OF THE FOUR BASELINE SOLVING METHODS ON DIFFERENT QUANTUM CIRCUITS (OBJECTIVE VALUE / TIME TO BEST IN SECONDS).

DQC Task	SA	TACO	TACO-NL	TACO-L
MCMT-4	8 / 1.5	8 / 0.002	**8 / 0.001**	8 / 0.002
MCMT-8	26 / 2.1	**16 / 0.001**	**16 / 0.001**	**16 / 0.001**
MCMT-16	-	204 / 4.9	204 / 15.1	**204 / 1.9**
MCMT-32	-	570 / 22.7	**570 / 22.4**	570 / 24.3
MCMT-64	-	5372 / 247.4	5814 / 218.7	**4378 / 227.9**
MCMT-96	-	-	-	-
QFT-4	8 / 1.02	**8 / 0.002**	**8 / 0.002**	8 / 0.003
QFT-8	32 / 1.9	**32 / 0.001**	**32 / 0.001**	32 / 0.003
QFT-16	-	256 / 0.02	256 / 1.8	**256 / 0.01**
QFT-32	-	624 / 289.0	624 / 183.6	**624 / 179.5**
QFT-64	-	3356 / 0.3	3154 / 154.2	**2764 / 117.8**
QFT-96	-	-	-	**6760 / 98.5**
Grover-4	16 / 1.1	**16 / 0.001**	**16 / 0.001**	**16 / 0.001**
Grover-8	172 / 10.0	140 / 0.006	**140 / 0.005**	**140 / 0.005**
Grover-16	-	1156 / 9.6	1156 / 10.5	**1156 / 3.0**
Grover-32	-	2968 / 33.7	2928 / 256.1	**2928 / 40.1**
Grover-64	-	17344 / 156.9	-	**10348 / 260.5**
Grover-96	-	-	-	-

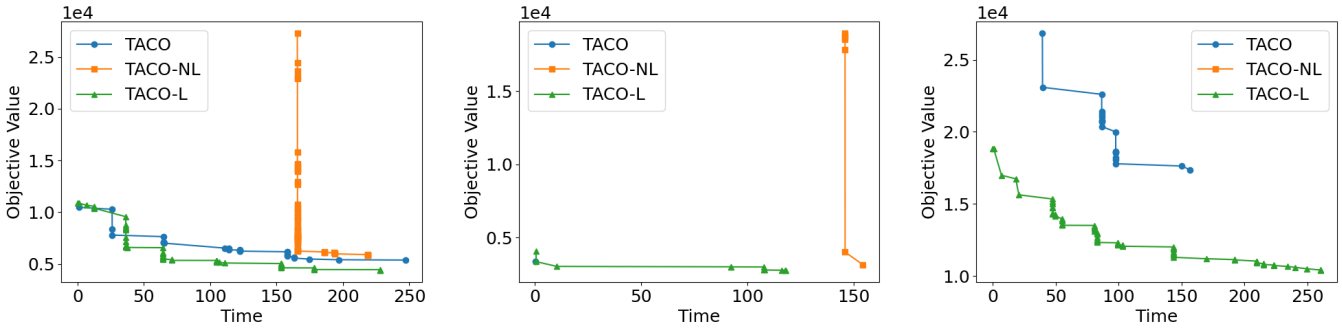
-: no feasible solution within allowed solver time; **bold**: best solution

3) Performance Metrics: To evaluate the proposed methods and baselines, we focus on the achieved objective value and solver time. The objective value is the total weighted communication cost in (1); its unit is determined by the link weights $b_{u,z}$ and is a normalized cost unit when all $b_{u,z} = 1$. The solver time is measured in seconds from the solver start and records when the reported best objective value is first found within the allowed solver time. We do not explicitly model swapping failures in our default evaluation, as their effects on entanglement regeneration, latency, throughput, fidelity, and recovery require a dynamic stochastic formulation. Instead, all methods use the same static proxy, $\sum_{u,z} b_{u,z} p_{i,j,u,z}$, which penalizes longer paths requiring more link-level entanglements and swaps. Thus, the reported objective value represents normalized communication cost rather than considering actual failure rate or retry latency. Scenario-dependent link weights in case study of Section VI-F can represent coarse link costs, but accurate stochastic modeling is left for future work.

C. Comparison of Solving Methods

We first compare the efficiency of the four baselines. The details of all testing circuits and the results are shown in Table III. The QPU cluster settings are selected according to the total number of qubits in these circuits. Each cell gives the best objective value obtained within 300 seconds and the solver time in seconds when that objective first appears.

First, the solving ability of SA on this problem is limited. It is not able to obtain a feasible solution within the given time for all circuits with more than 16 qubits even with partially linearized version. These results are similar to those in [29] where only circuits with at most 15 qubits are tested. Besides, for 8-qubit circuits, SA only finds the optimal solution for



(a) *MCMT-64* (b) *QFT-64* (c) *Grover-64*

Fig. 5. Convergence towards best objectives over time when solving the optimization for diffrent circuits with the baselines.

TABLE IV
EFFECTIVENESS OF EC AND KL PRE-PROCESSING ON HOMOGENEOUS QPU CLUSTERS.

DQC Task	EC	KL	TACO-L + EC	TACO-L + KL	TACO-L + EC + PA	TACO-L + KL + PA
MCMT-256	0.073	0.036	-	85228 / 231.8	48168 / 32.4	**44712 / 83.6**
MCMT-512	0.316	0.086	244724 / 300.8	200828 / 43.5	141988 / 2.3	**118452 / 0.9**
MCMT-1024	1.668	0.192	-	-	611608 / 447.7	**591460 / 599.9**
QFT-256	0.224	0.081	5508 / 147.5	5508 / 525.4	**5508 / 0.1**	5508 / 0.1
QFT-512	1.100	0.209	5700 / 142.7	5700 / 93.9	**5700 / 0.0**	5700 / 0.0
QFT-1024	4.604	0.554	-	-	**11780 / 0.9**	11780 / 0.9
Grover-256	0.065	0.035	44784 / 203.1	57296 / 557.8	44784 / 0.4	**44640 / 1.9**
Grover-512	0.273	0.077	73512 / 165.6	73592 / 493.8	**73512 / 0.1**	73592 / 0.2
Grover-1024	1.201	0.188	-	-	**178856 / 1.8**	178936 / 4.9

EC and KL entries are pre-processing time (s); TACO-L entries are objective / time to best (s).
-: no feasible solution within allowed solver time; **bold**: better result between the final two columns

TABLE V
EFFECTIVENESS OF EC AND KL PRE-PROCESSING ON HETEROGENEOUS QPU CLUSTERS.

DQC Task	EC	KL	TACO-L + EC	TACO-L + KL	TACO-L + EC + PA	TACO-L + KL + PA
MCMT-192	0.045	0.022	-	-	**28576 / 360.3**	31610 / 323.9
MCMT-384	0.187	0.064	-	-	148260 / 490.5	**123688 / 523.9**
MCMT-768	0.836	0.141	-	-	362314 / 581.7	**326564 / 539.0**
QFT-192	0.123	0.063	5028 / 346.2	5236 / 333.2	**5028 / 0.2**	5100 / 0.5
QFT-384	0.622	0.155	-	-	**10448 / 4.1**	10790 / 8.7
QFT-768	2.581	0.383	-	-	**11588 / 1.0**	11588 / 1.0
Grover-192	0.037	0.023	33728 / 292.0	41888 / 473.3	**33104 / 0.9**	38808 / 72.0
Grover-384	0.152	0.053	-	-	**85792 / 142.6**	95136 / 124.2
Grover-768	0.629	0.119	-	-	**128008 / 1.9**	139680 / 18.6

EC and KL entries are pre-processing time (s); TACO-L entries are objective / time to best (s).
-: no feasible solution within allowed solver time; **bold**: better result between the final two columns

QFT-8. Generally, SA does not perform as good as all other 3 baselines in terms of both the best objective and solving time.

Second, our partially linearized formulation *TACO-L* is generally the most efficient for all non-trivial cases with more than 2 QPUs. It not only achieves the best objective value (least total communication cost) in all cases, but also finds the best solution with least time for all almost all cases. The only exception is MCMT-32, where TACO-L uses more time than TACO-NL but still very close to the best time. This means it is suitable to solve the joint optimization in DQC with larger scales compared with *TACO* or *TACO-NL*. Then, interestingly, *TACO-NL*, while uses the newest automatic linearization feature of Gurobi for high-degree non-linear expressions, cannot beat *TACO*, which only repeatedly decomposes high degree expressions to quadratic by adding intermediate variables. This suggests that directly using the non-linear expression may not be sufficient, and manually decomposing high degree expressions may still be a better choice.

We are also interested in the progress (best objective by time) of those baselines when they solve the problem. Fig. 5 shows the best objective found by the baselines for the three circuits with 64 qubits. It is clear that our reformulated TACO-L method can generally find the best solution (compared to other baselines) at most times during the solving period.

D. Effectiveness of Pre-Processing Methods

Table III shows that directly solving TACO-L becomes difficult for some circuits with 96 qubits. We therefore evaluate EC and KL on larger circuits containing up to $1,024$ qubits. Table IV and Table V report the results on homogeneous and heterogeneous QPU clusters, respectively. The EC and KL columns give pre-processing time in seconds; each TACO-L column gives the best objective and the solver time at which that objective was first found. A dash indicates that no feasible

solution was found within the given time budget. For every configuration, we use a common 600-second budget and set the solver time limit to $600 - t_{\text{prep}}$ seconds, where t_{prep} is the measured EC or KL time. Thus, the partitioning overhead is charged against the solver budget. In the last two columns, boldface identifies the better result between TACO-L+EC+PA and TACO-L+KL+PA: the smaller objective is preferred, with time to the best solution used to break an objective tie.

KL has lower pre-processing time in all 18 cases: its observed times range from 0.022 to 0.554 seconds, compared with 0.037 to 4.604 seconds for EC. Nevertheless, the faster partitioner does not always produce the better input for the downstream joint optimization. With PA enabled, EC gives a smaller objective in eight cases and KL in six cases; the objectives tie in the other four cases, for which EC reaches the objective earlier. Accordingly, TACO-L+EC+PA is the better final result in 12 of 18 cases, whereas TACO-L+KL+PA is better in six. Both PA variants find feasible solutions in every case. Without PA, feasible solutions are obtained in only seven cases with EC and eight with KL, which confirms that pre-assignment is important at these scales. Overall, the comparison with a standard graph-partitioning baseline validates that the proposed EC method remains competitive and more often yields the better end-to-end optimization result, despite KL's smaller pre-processing overhead. More importantly, integrated with both EC or KL and PA, TACO-L can find either better solution or the same solution in shorter time for these larger scales of circuits.

E. Results and Topologies over Difference Settings

1) Quantum Resources - Computation Qubits: We now consider the same circuit on three different clusters which has different of number of data qubits m_i per QPU. As shown in Fig. 6, when running the 64 qubits version of the algorithms, more qubits on one single QPU (larger m_i) always leads to lower communication cost because more qubits can reside in the same processor so fewer inter-QPU communications are required.

2) Topology Sparsity Requirement: We now investigate how the topology sparsity requirement W can affect the total communication cost (i.e, the objective of our optimization). Recall that W caps the total number of implementable physical links and prevents an unrealizable dense topology. For this set of tests, we run the 64 qubits version of the algorithms on the *small-64* cluster, only varying W as indicated. From Fig. 7, we can see that for the three testing circuits, the achieved objective value is always smaller if the network topology is allowed to be denser. This is because a denser graph always has shorter point-to-point paths, thus smaller communication cost. The only exception is Grover: $W = 12$ has the same objective value compared to $W = 10$. This is because no more edge is necessary so the final topology of $W = 12$ is actually the same as the topology of $W = 10$. This can be verified by the resulting topologies shown in Figs. 8(h) and 8(i).

We also show some examples of resulting topology for the three circuits when we use different W. The resulting topologies (under the same settings used in the last part) are shown in Fig. 8. First, it is obvious that when the network is allowed to be denser, we can have more edges to further reduce communication cost. Second, all node degree in these topologies is bounded by 3 which is the number of communication qubits of each QPU. Last but not least, different circuits have different communication patterns, so they may have different optimized topology even running on the same set of QPU clusters.

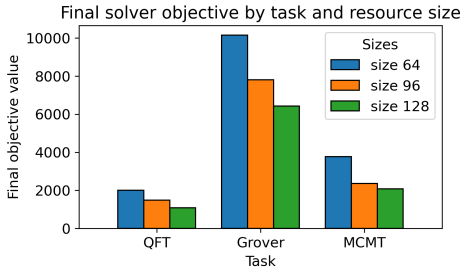


Fig. 6. Objective values of *TACO-L+EC+PA* for different size m_i.

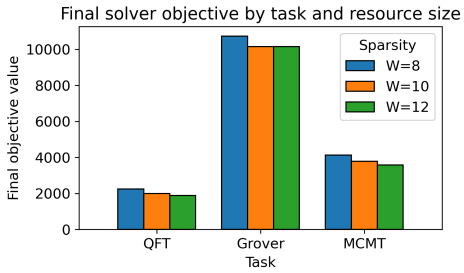
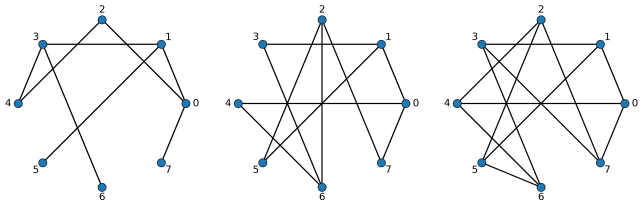
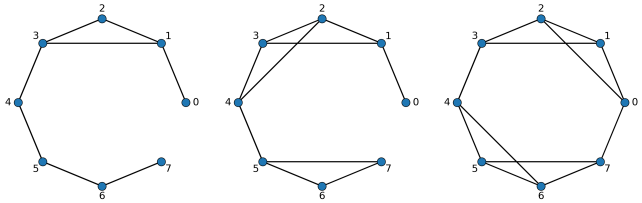


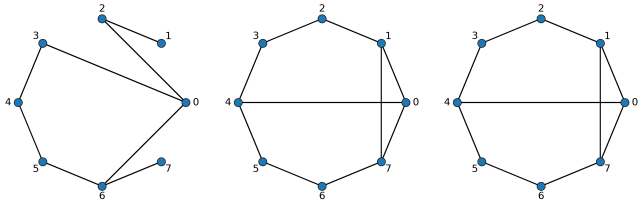
Fig. 7. Objective values of *TACO-L+EC+PA* for different W.



(a) *MCMT-64*, $W = 8$ (b) *MCMT-64*, $W = 10$ (c) *MCMT-64*, $W = 12$

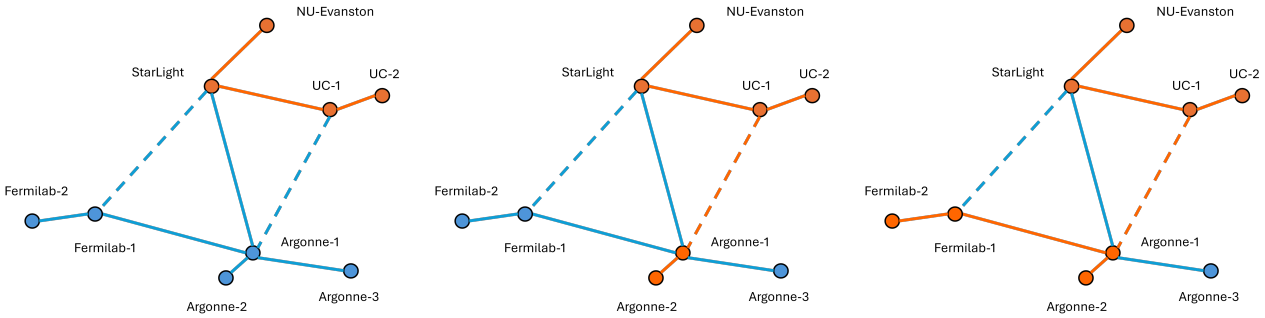


(d) *QFT-64*, $W = 8$ (e) *QFT-64*, $W = 10$ (f) *QFT-64*, $W = 12$



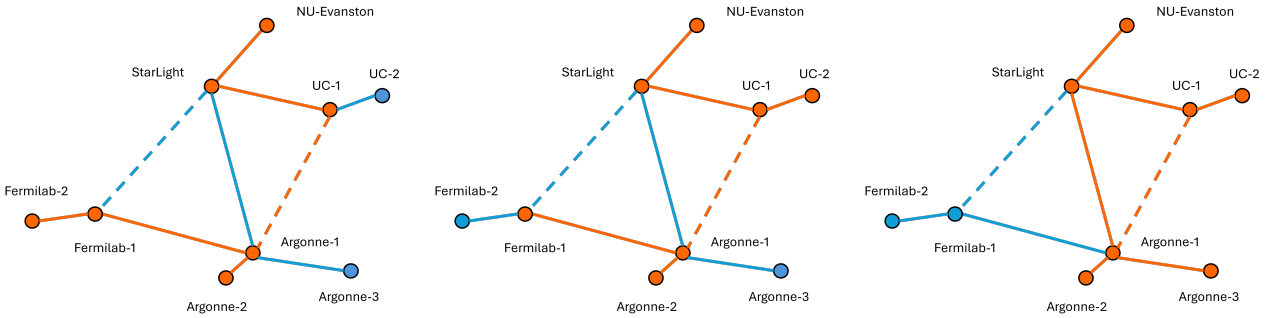
(g) *Grover-64*, $W = 8$ (h) *Grover-64*, $W = 10$ (i) *Grover-64*, $W = 12$

Fig. 8. Optimized topologies for the real-world circuits with different topology sparsity constraints $W = 8, 10, 12$.



(a) *QFT-1024* (b) *QFT-1536* (c) *QFT-2048*

Fig. 9. Resulting topologies of Chicago QN with homogeneous computing resources (each node has 256 qubits). The red nodes/links are selected nodes/links in the resulting topology from our algorithm.



(a) *QFT-1536, heterogeneous nodes* (b) *QFT-1536, heterogeneous nodes, w/ repeaters* (c) *MCMT-1536, w/ repeater*

Fig. 10. Resulting topologies of Chicago QN with heterogeneous computing resources (each main node has 256 qubits and affiliated node has 128 qubits) and repeaters. The red nodes/links are selected nodes/links in the resulting topology from our algorithm.

F. Case Studies with DQC Applications

We show how to apply our method to real-word distributed quantum computer design on two possible application scenarios: Chicago quantum network [45] for metropolitan scale networks and multi-rack networks for quantum data centers.

1) Chicago Quantum Network: A quantum network [45] built around the Chicago city, connecting multiple institutions, including the Argonne National Lab, Fermi Lab, Northwestern University, StarLight, and University of Chicago. As shown in Fig. 9, this network has nine sites, and solid/dashed lines represent existing/planned links.

First, we consider running different scales of circuits within this network when each site has 256 qubits. The edge level cost $b_{u,v}$ is set exponential to the link length to model the photon loss of direct links. Fig. 9 shows which nodes and edges (in red) are selected to form the resulting topology of DQC for three scales of QFT algorithms: QFT-1024, QFT-1536, and QFT-2048 after using our proposed algorithm. As each node has 256 qubits, we can see that 4, 6, and 8 nodes are selected for those circuits separately to fulfill the computing qubit requirement and to save communication cost. An interesting observation is that, in this setting, the right side enhancing link is preferred to connect the south part of the network as it is the shortest among the three links connecting the 4 north nodes and the 5 south nodes.

We also consider heterogeneous settings where the main nodes (i.e., NU-Evanston, StarLight, UC-1, Fermilab-1, Argonne-1) have 256 qubits, and affiliated nodes (UC-2, Fermilab-2, Arggone-2, and Argonne-3) have 128 qubits. The edge level cost $b_{u,v}$ is set linear to the link length to model links segmented by repeaters with memory [22]. The resulting topologies, i.e., selected nodes and links are shown in Fig. 10. We can see that they are different from their corresponding topology under the homogeneous setting.

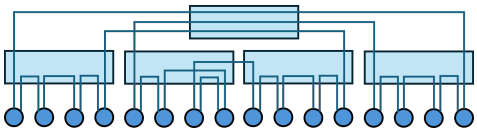


Fig. 11. Resulting topology of an optimized switch network for a QFT-1024 algorithm. Each QPU has at most two ports.

2) Quantum Data Center Networks: Quantum Data Center Networks (QDCNs) connect multiple QPUs deployed in one single quantum data center, which is a promising pathway to large scale quantum computing. We adopt the model in [33] that considers in-rack and cross-rack links to form a large scale QDCN. Similar to [33], $b_{u,v}$ of cross-rack links are set to 800 times to that of in-rack links. Here, the factor of 100 reflects the lower throughput, and 8 stands for resources used for purification to mitigate the lower fidelity. In this setting, each QPU has 64 qubits, and the whole network is formed by 4 racks with 4 QPUs in each rack. Note that this multi-rack quantum data center network is also one of the designs of Cisco's QDCN [35].

Fig. 11 shows an instance of resulting topology of the QDCN by our proposed algorithm for QFT-1024. The squares

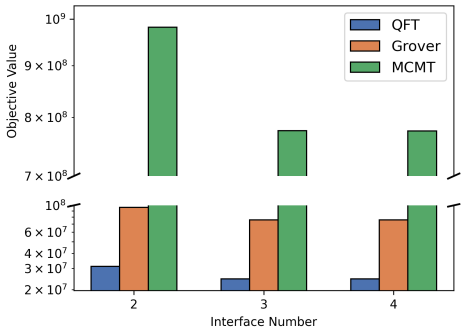


Fig. 12. Communication cost is reduced by more ports allowed at each QPU (i.e., denser resulting topology) in QDCN.

are switches and dots are QPUs. In this experiment, each QPU has at most 2 ports (i.e. $n_i = 2$), similar to the setting in [33]. We are also interested in the case where each QPUs has more ports. We test the 1024 qubits version of three quantum algorithms, given the port number on each QPUs varies from 2 to 4. Fig. 12 shows the results. As expected, with more ports allowed the resulting topology is denser which lead to communication cost reduction in the objective values. We can see that there is around 20% cost reduction by increasing the port number from 2 to 3, while there is only marginal reduction when increasing from 3 to 4. This may suggest that too many interfaces are not necessary for reducing communication cost. However, this depends on the payload or traffic pattern of quantum algorithms in QDCN. With more quantum algorithm payload, denser topology may be needed.

VII. Conclusion

In this paper, we first formulate a tractable integer nonlinear programming problem for a joint optimization of network topology and qubit allocation in distributed quantum computing. This simple formation allows the entanglement paths are explicitly incorporated into the joint optimization. We then further convert the original problem into an equivalent but partially linearized version so that it can be solved more efficiently by classical optimization solver. To tackle the scheduling of larger-scale quantum circuits, we further propose heuristics from contracting edges in qubit interaction graph to pre-assign grouped qubits to QPUs so that the proposed optimization problem can be solved effectively. Simulations over real-world quantum circuits on various DQC settings confirm that 1) the partially linearized formulation is more efficient to solve with reduced total communication costs, 2) both edge contraction and pre-assignment can help to solve larger quantum circuits, and 3) the proposed joint optimization can be applied to various quantum applications (both DQC and QDC).

We leave further investigation on (1) how to solve the optimization more efficiently than directly using the solver, (2) extending the optimization formulation to consider time dependency among quantum gates [36], [46], circuit transformation [47], or remote-gate scheduling [48], and (3) jointly modeling local physical-qubit connectivity, the placement of communication qubits, intra-QPU routing, and communication/data-qubit or quantum-channel resource adjustment. Other important extensions include accurate models of stochastic entanglement generation and swapping, fidelity evolution, latency, throughput, retry scheduling, and failure recovery.

References

[1] 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.

[2] S. DiAdamo, M. Ghibaudi, and J. Cruise, "Distributed quantum computing and network control for accelerated VQE," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–21, 2021.

[3] X. Wei, L. Fan, Y. Guo, Z. Han, and Y. Wang, "Optimizing satellite-based entanglement distribution in quantum networks via quantum-assisted approaches," in *Proc. of IEEE International Conf. on Quantum Communications, Networking, and Computing (QCNC)*, 2024.

[4] X. Wei, J. Liu, L. Fan, Y. Guo, Z. Han, and Y. Wang, "Optimal entanglement distribution problem in satellite-based quantum networks," *IEEE Network*, vol. 39, no. 1, pp. 97–103, 2025.

[5] X. Wei, L. Fan, et al., "Entanglement from sky: Optimizing satellite-based entanglement distribution for quantum networks," *IEEE/ACM Trans. on Networking*, vol. 32, no. 6, pp. 5295–5309, 2024.

[6] J. Liu and Y. Wang, "Statistical modeling and latency optimization for entanglement routing in quantum networks," in *Proc. of IEEE Int'l Conf. on Quantum Computing and Engineering (QCE)*, 2025.

[7] J. Liu, X. Zhang, X. Wei, X. Liu, Y. Chen, H. Gao, and Y. Wang, "Joint swapping and purification with failures for entanglement distribution in quantum networks," in *Proc. of IEEE/ACM International Symposium on Quality of Service (IWQoS 2025)*, 2025.

[8] Y. Mao, Y. Liu, and Y. Yang, "Probability-aware qubit-to-processor mapping in distributed quantum computing," in *Proc. of the 1st ACM Workshop on Quantum Networks and Distributed Quantum Computing (QuNet)*, 2023, p. 51–56.

[9] F. Zhan, Y. Zhao, and C. Qiao, "Towards high-performance distributed quantum computing with qubit placement and provisioning," in *Proc. of 2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, 2024, pp. 1–10.

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

[11] R. G Sundaram, H. Gupta, and C. Ramakrishnan, "Efficient distribution of quantum circuits," in *Proc. of 35th International Symposium on Distributed Computing (DISC 2021)*, 2021.

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

[13] R. G. Sundaram, H. Gupta, and C. Ramakrishnan, "Distribution of quantum circuits over general quantum networks," in *Proc. of IEEE Int'l Conf. on Quantum Computing and Engineering (QCE)*, 2022.

[14] D. Dadkhah, M. Zomorodi, S. E. Hosseini, P. Plawiak, and X. Zhou, "Reordering and partitioning of distributed quantum circuits," *IEEE Access*, vol. 10, pp. 70 329–70 341, 2022.

[15] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong, "Time-sliced quantum circuit partitioning for modular architectures," in *Proc. of the 17th ACM Int'l Conf. on Computing Frontiers*, 2020, pp. 98–107.

[16] 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.

[17] 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.

[18] D. Dadkhah, M. Zomorodi, and S. E. Hosseini, "A new approach for optimization of distributed quantum circuits," *International Journal of Theoretical Physics*, vol. 60, pp. 3271–3285, 2021.

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

[20] L. Sünkel, M. Dawar, and T. Gabor, "Applying an evolutionary algorithm to minimize teleportation costs in distributed quantum computing," in *Proc. of IEEE Int'l Conf. on Quantum Comp. and Eng. (QCE)*, 2024.

[21] R. Yu, R. Dutta, and J. Liu, "On topology design for the quantum internet," *IEEE Network*, vol. 36, no. 5, pp. 64–70, 2022.

[22] J. Liu, X. Liu, X. Wei, and Y. Wang, "Topology design with resource allocation and entanglement distribution for quantum networks," in *Proc. of 21st Annual IEEE International Conf. on Sensing, Communication, and Networking (SECON)*, 2024.

[23] D. Ferrari, A. S. Cacciapuoti, M. Amoretti, and M. Caleffi, "Compiler design for distributed quantum computing," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–20, 2021.

[24] 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.

[25] D. Cuomo, M. Caleffi, K. Krsulich, F. Tramonto, G. Agliardi, et al., "Optimized compiler for distributed quantum computing," *ACM Trans. on Quantum Computing*, vol. 4, no. 2, pp. 1–29, 2023.

[26] J. Avron, O. Casper, and I. Rozen, "Quantum advantage and noise reduction in distributed quantum computing," *Physical Review A*, vol. 104, no. 5, p. 052404, 2021.

[27] A. M. Gomez, T. L. Patti, et al., "Near-term distributed quantum computation using mean-field corrections and auxiliary qubits," *Quantum Science and Technology*, vol. 9, no. 3, p. 035022, 2024.

[28] M. Prest and K.-C. Chen, "Quantum-error-mitigated detectable byzantine agreement with dynamical decoupling for distributed quantum computing," *arXiv preprint arXiv:2311.03097*, 2023.

[29] Y. Mao, Y. Liu, X. Shang, and Y. Yang, "Network topology design for distributed quantum computing," in *Proc. of IEEE 44th Int'l Conf. on Distributed Computing Systems (ICDCS)*, 2024, pp. 1213–1223.

[30] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2025. [Online]. Available: https://www.gurobi.com

[31] IBM, "IBM ILOG CPLEX Optimization Studio," 2025. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio

[32] J. Liu, L. Fan, Y. Guo, Z. Han, and Y. Wang, "Co-design of network topology and qubit allocation for distributed quantum computing," in *Proc. of IEEE International Conf. on Quantum Communications, Networking, and Computing (QCNC)*, 2025.

[33] H. Zhang, Y. Xu, H. Hu, K. Yin, H. Shapourian, J. Zhao, R. R. Kompella, R. Nejabati, and Y. Ding, "SwitchQNet: Optimizing distributed quantum computing for quantum data centers with switch networks," in *Proc. of the 52nd Annual Int'l Symposium on Computer Architecture*, 2025.

[34] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

[35] H. Shapourian, E. Kaur, T. Sewell, J. Zhao, M. Kilzer, R. Kompella, and R. Nejabati, "Quantum data center infrastructures: A scalable architectural design perspective," 2025. [Online]. Available: https://arxiv.org/abs/2501.05598

[36] S. Pouryousef, E. Kaur, H. Shapourian, D. Towsley, R. Kompella, and R. Nejabati, "Benchmarking quantum data center architectures: A performance and scalability perspective," 2026. [Online]. Available: https://arxiv.org/abs/2601.01353

[37] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE transactions on Computers*, vol. 100, no. 10, pp. 892–901, 2012.

[38] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," in *Proc. of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 63–74.

[39] A. Barenco, C. H. Bennett, R. Cleve, et al., "Elementary gates for quantum computation," *Physical review A*, vol. 52, no. 5, p. 3457, 1995.

[40] J. Calsamiglia and N. Lütkenhaus, "Maximum efficiency of a linear-optical bell-state analyzer," *Applied Physics B*, vol. 72, pp. 67–71, 2001.

[41] M. J. Bayerbach, S. E. D'Aurelio, P. van Loock, and S. Barz, "Bell-state measurement exceeding 50% success probability with linear optics," *Science Advances*, vol. 9, no. 32, p. eadf4080, 2023.

[42] F. Ewert and P. van Loock, "3/4-efficient bell measurement with passive linear optics and unentangled ancillae," *Physical review letters*, vol. 113, no. 14, p. 140403, 2014.

[43] A. Kamimaki, K. Wakamatsu, K. Mikata, Y. Sekiguchi, and H. Kosaka, "Deterministic bell state measurement with a single quantum memory," *npj Quantum Information*, vol. 9, no. 1, p. 101, 2023.

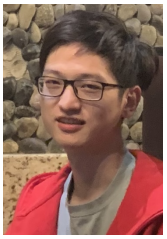
[44] "QASM," https://github.com/cda-tum/mqt-qmap/tree/main/examples.

[45] X. Wu, A. Kolar, J. Chung, et al., "Sequence: a customizable discrete-event simulator of quantum networks," *Quantum Science and Technology*, vol. 6, no. 4, p. 045027, Sep. 2021.

[46] J. Liu, L. Fan, Y. Guo, Z. Han, and Y. Wang, "Topology optimization in all-optical switched quantum data center networks," in *Proc. of IEEE Int'l Conf. on Quantum Computing and Engineering (QCE)*, 2026.

[47] X. Zhang, X. Xu, Y. Liu, Y. Mao, B. Xiao, and Y. Yang, "Joint optimization of circuit transformation and qubit mapping for distributed quantum computing," in *Proc. of IEEE INFOCOM*, 2026.

[48] X. Xu, Y. Liu, Y. Mao, and Y. Yang, "Remote gate scheduling in distributed quantum computing," in *Proc. of 2025 IEEE 45th International Conf. on Distributed Computing Systems (ICDCS)*, 2025, pp. 846–856.



Jiyao Liu (S'24) is currently pursuing the Ph.D. degree in computer and information sciences at Temple University, Philadelphia, PA, USA. His research interests span networking, quantum computing, and machine learning, with an emphasis on quantum networking, distributed quantum computing, networks for AI, and AI for networks.

Lei Fan (M'15-SM'20) is an Associate Professor with the Engineering Technology Department at University of Houston. Before this position, he worked in the electricity energy industry for several years. He received the Ph.D. degree in operations research from the Industrial and System Engineering Department at University of Florida. His research includes quantum computing, optimization methods, complex system operations, power system operations and planning.

Yuanxiong Guo (M'14-SM'19) received the B.Eng. degree in electronics and information engineering from the Huazhong University of Science and Technology, China, in 2009, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Florida, in 2012 and 2014, respectively. He is currently an Associate Professor in the Department of Information Systems and Cyber Security at the University of Texas at San Antonio. His current research interests include distributed machine learning, applied data science, and trustworthy AI with application to digital health, energy sustainability, and human-robot collaboration. He is on the Editorial Board of IEEE Transactions on Vehicular Technology and servers as the track co-chair for IEEE VTC 2021-Fall. He is a recipient of the Best Paper Award in the IEEE GLBOECOM 2011.

Zhu Han (S'01-M'04-SM'09-F'14) received the B.S. degree in electronic engineering from Tsinghua University, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. Currently, he is a John and Rebecca Moores Professor in the Electrical and Computer Engineering Department as well as in the Computer Science Department at the University of Houston. His research targets on the novel game-theory related concepts critical to enabling efficient and distributive use of wireless networks with limited resources. He received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, and the 2021 IEEE Kiyo Tomiyasu Award. He was an IEEE Communications Society Distinguished Lecturer from 2015-2018, IEEE fellow from 2014, AAAS fellow since 2019, and ACM fellow since 2024. He is a 1% highly cited researcher since 2017 according to Web of Science.

Yu Wang (S'02-M'04-SM'10-F'18) is a Professor and Chair of the Department of Computer and Information Sciences at Temple University. He holds a Ph.D. from Illinois Institute of Technology, an MEng and a BEng from Tsinghua University, all in Computer Science. His research interest includes wireless networks, smart sensing, and distributed computing. He has published over 300 papers in peer-reviewed journals and conferences. He is a recipient of *Ralph E. Powe Junior Faculty Enhancement Awards* from Oak Ridge Associated Universities (2006), *Outstanding Faculty Research Award* from College of Computing and Informatics at the University of North Carolina at Charlotte (2008), *Fellow of IEEE* (2018), *ACM Distinguished Member* (2020), and *IEEE Benjamin Franklin Key Award* (2024). He has served as Associate Editor for *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, among others.