Co-Design of Network Topology and Qubit Allocation for Distributed Quantum Computing

Jiyao Liu*, Lei Fan†‡, Yuanxiong Guo§, Zhu Han‡, and Yu Wang*
*Department of Computer and Information Sciences, Temple University, Philadelphia, USA
†Dept. of Engineering Technology, ‡Dept. of Electrical and Computer Engineering, University of Houston, Houston, USA
§Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, USA

***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. In this paper, we propose a joint optimization framework for network topology design and qubit allocation in DQC to minimize the 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 further improve computational efficiency, we present a partially linearized version of the problem, making it solvable using any classical optimization solver. Extensive simulations on both random and real-world quantum circuits 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 Computing, Quantum Networks**

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.

Current research in DQC or quantum networks focuses on several key challenges such as optimizing qubit allocation [3], [4] or circuit distribution [5]–[14] across nodes, designing efficient network topologies [15]–[17], optimizing quantum compiler framework [18]–[20], and overcoming the inherent noise and decoherence [21]–[23] in DQC.

The work is partially supported by the US National Science Foundation (Grant No. CNS-2006604, CNS-2106761, CNS-2128378, CNS-2318663, ECCS-2045978, ECCS-2302469, CMMI-2222810, and OAC-2417716), Toyota, Amazon, and Japan Science & Technology Agency (JST) Adopting Sustainable Partnerships for Innovative Research Ecosystem under JPMJAP2326.

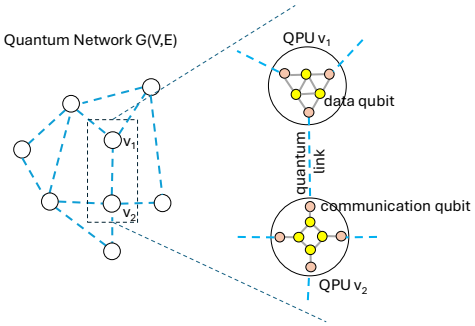


Fig. 1. Distributed quantum computing (DQC): a set of QPUs are interconnected via a quantum network, and each QPU has certain amount of physical qubits which can be used as data qubits or communication qubits.

DQC 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 [24]–[26], 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 ensures that distributed quantum computing tasks are executed

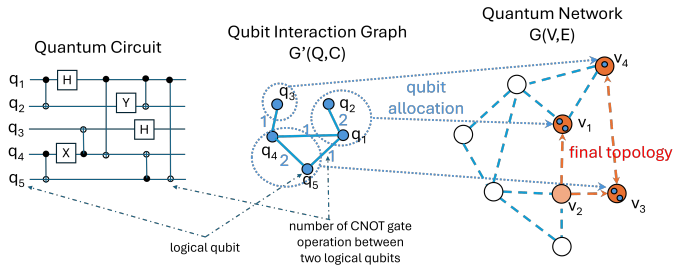


Fig. 2. Qubit allocation and topology in DQC: mapping a qubit interaction graph graph (QIG, modeled by graph G') onto the quantum network (QN, modeled by graph G) to minimize the communication cost. Here, the weight on a link in G' is the number of two-qubit gate operations between two logical quibts.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

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.

In this paper, 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.* [17] 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 [27] and CPLEX [28]. We conduct extensive simulations using both random and real-world quantum circuits. Results confirm the efficiency of our proposed formulation and solution.

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. Evaluations of the proposed method are provided in Section V. Finally, Section VI concludes the paper with possible future directions.

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, which can be categorized into *circuit partitioning and scheduling*, *network-aware qubit allocation*, and *topology co-design*.

Quantum Circuit Partitioning and Scheduling. Many works have investigated how to partition quantum circuits for distributed execution in DQC. Sundaram *et al.* [5] propose efficient methods for distributing quantum circuits over multiple QPUs. Their method first partitions the given circuit across QPUs and then for each partition determines a small set of migrations (via cat-entanglement operations) that are sufficient to execute/cover all the non-local gates in the circuit. Sundaram *et al.* [6], [7] further explore teleportation-based techniques for distributing circuits across quantum networks with the objective of minimizing the number of teleportations needed to implement gates spanning multiple QPUs. Dadkhah *et al.* [8] also consider optimize the distributed quantum circuits. Their method reorders the qubit placement in a initial quantum circuit to improve the execution time, then partitions the new quantum circuit using genetic algorithm to obtain a distributed quantum circuit so that the number of teleportation costs is reduced. Baker *et al.* [9] explore time-sliced partitioning of quantum circuits to modular physical machines in a cluster to improve execution on modular architectures. For each time slice at a time, their method optimizes the mappings to reduce the cost to move data from the previous time slices. Other works [10]–[14] apply methods like hypergraph partitioning [13], dynamic programming [11] and evolutionary algorithm [14] to optimize quantum circuit distribution. These works primarily aim at minimizing inter-processor communication or the number of teleportations but do not explicitly consider the impact of network conditions and entanglement availability. Ferrari *et al.* [18], [19] also study the general quantum compiler design for DQC, analytically derive a theoretical bound of the overhead induced by quantum compilation, and present a modular quantum compilation framework for DQC that takes into account both network and device constraints.

Network-Aware Qubit Allocation. Several recent studies have incorporated more network considerations into qubit allocation in DQC. Mao *et al.* [3] 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 [4] 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. [20] 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.

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 qubit gates required 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	the maximal data/communication qubits on QPU v_i
W	the maximal number of edge in 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

Network Topology Design. The network topology has a significant impact on the efficiency of quantum network. Yu *et al.* [15] 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.* [16] 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.* [17] 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.

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 quantum servers. Major notations used in this paper are summarized in Table I.

A. Distributed Quantum Computing Model

Single qubit gates plus Controlled-NOT (CNOT) gates are universal in quantum computing [29], i.e., one can implement any quantum algorithm by only using single qubit gates and CNOT gates. Thus, in this paper, similar to previous works [3], [4], we focus on distributing quantum circuits with only single qubit gates and CNOT gates (as the example in the left part of Fig. 2). Besides, to optimize the communication cost in DQC, we can ignore single qubit gates in our design since they will be conducted on a single QPU without requirement of any communication.

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. 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.

After the qubit allocation (distributing all logical qubits to the QPUs in the underlying quantum network), when two logical qubits reside on the same processor, the CNOT gates between them can be easily done by the QPU. For example, in Fig. 2, because q_1 and q_2 are mapped to the same server v_1, the two CNOT gates can be done solely by v_1, thus require no communication even $c_{1,2} \neq 0$. However, when the two qubits are on two different processors, remote CNOT (RCNOT) should be used to replace CNOT and inter-QPU communications. For example, both q_4 and q_5 need to communicate with q_1 and they are mapped to different servers (q_4 and q_5 on v_3 while q_1 on v_4), then we need communication resources (entanglements) for two RCNOT gates collaboratively conducted by v_1 and v_3. RCNOT basically works in the same way as CNOT, but it allows the flexibility of the residence of the two qubits by additionally consuming one entanglement between the two corresponding processors. For more details about CNOT/RCNOT, we refer readers to [17].

B. Quantum Network and Resources

Our quantum network (QN) is a network of quantum processors or 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

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

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. Each of the QPU v_i can at most maintain n_i edge due to the limit of communication qubits. Note that maintaining quantum link in QN is costly, thus, we prefer a sparse network topology while maintaining the connectivity for QDC. Our formulated optimization will have a constraint on the number edge in the final topology.

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. Note that swappings may not always succeed, and the success probability upperbounds vary (e.g., 50% [30], 62.5% [31], 75% [32], or 100% [33] (no failure but inefficient)), determined by their implementations. In this paper, for simplicity, we assume that the swapping success probability is a constant ρ.

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). Notably, graph partition problems are proven to be NP-hard and, unless $P = NP$, no reasonable approximation algorithm exists. This inherently makes our joint optimization also NP-hard as it is a more general problem compared graph partition.

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

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 solver, 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 [17] 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 associated weight/cost of the edge (v_u, v_z), and therefore $\sum_{u,z} b_{u,z} p_{i,j,u,z}$ is the weighted length (or cost, determined by the definition of $b_{u,z}$[1]) of the entanglement path

[1]We can adopt different definitions of edge weight $b_{u,z}$ under different scenarios. For example, in the multi-farm DQC (e.g. remote data centers) it can be associated with the quantum link length to quantify the path cost (which is typically exponentially affected by the length); in the multi-core DQC (e.g., on-chip QPUs), we can set all $b_{u,v} = 1$ as the distances among on-chip processors are almost negligible. Besides, swappings may not always succeed, as shown by [16], given swappings succeed at probability ρ, when path length doubles, the cost of one E2E entanglement is increased to $\frac{2}{\rho}$ times. Therefore, in our model, we can simply use the path length as the cost of an E2E entanglement over the path, ignoring the co-efficient in the optimization.

between v_i and v_j. 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.

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

s.t. qubit allocation

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

$$\sum_a x_{a,i} \le m_i, \qquad \forall v_i \in V, \tag{3}$$

topology sparsity

$$\sum_z \left[1 - \prod_{i<j} (1 - p_{i,j,u,z})(1 - p_{i,j,z,u}) \right] \le n_u, \forall v_u \in V, \tag{4}$$

$$\sum_{u<z} \left[1 - \prod_{i<j} (1 - p_{i,j,u,z})(1 - p_{i,j,z,u}) \right] \le W, \tag{5}$$

entanglement paths

$$\left[\sum_u p_{i,j,i,u} - \sum_z p_{i,j,z,i} \right] \sum_{a,b} x_{a,i} x_{b,j} = \sum_{a,b} x_{a,i} x_{b,j}, \quad \forall i < j, \tag{6}$$

$$\sum_z p_{i,j,u,z} - \sum_z p_{i,j,z,u} = 0, \quad \forall i < j, \forall u \notin \{i, j\}, \tag{7}$$

$$\left[\sum_u p_{i,j,j,u} - \sum_z p_{i,j,z,j} \right] \sum_{a,b} x_{a,i} x_{b,j} = -\sum_{a,b} x_{a,i} x_{b,j}, \quad \forall i < j, \tag{8}$$

$$\sum_{u,z} p_{i,j,u,z} \le (|V| - 1) \sum_{a,b} x_{a,i} x_{b,j}, \quad \forall i < j, \tag{9}$$

decision variables

$$x_{a,i}, p_{i,j,u,z} \in \{0, 1\}. \tag{10}$$

Constraints (2) and (3) 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 (4) 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. Therefore, the left side of Constraint (4) is the number of edges associated to QPU v_z.

Constraint (5) 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 (5) is similar to that of Constraint (4) but it summarize over all edges in the topology rather than those associated to a specific QPU. Constraints (6)-(8) 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 (9) forces no entanglement path for pair (v_i, v_j) if it has no demand, avoiding disturbing topology constraints (Constraints (3)-(4)).

Finally, all decision variables $x_{a,i}$ and $p_{i,j,u,z}$ are binary. Therefore, this formulated 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 need to 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.

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

s.t. qubit allocation

$$\sum_i x_{a,i} = 1, \qquad \forall q_a \in Q, \tag{11}$$

$$\sum_a x_{a,i} \le m_i, \qquad \forall v_i \in V, \tag{12}$$

topology sparsity

$$\sum_z w_{u,z} \le n_u, \qquad \forall v_u \in V, \tag{13}$$

$$\sum_{u<z} w_{u,z} \le W, \tag{14}$$

$$w_{u,z} \le \sum_{i,j} (p_{i,j,u,z} + p_{i,j,z,u}) \le w_{u,z} \cdot (2|V|(|V| - 1)), \quad \forall u < z, \tag{15}$$

entanglement paths

$$y_{i,j} \le \sum_{a,b} x_{a,i} x_{b,j} \le y_{i,j} \cdot \frac{1}{2}|Q|, \qquad \forall i < j, \tag{16}$$

$$\sum_u p_{i,j,i,u} - \sum_z p_{i,j,z,i} = y_{i,j}, \qquad \forall i < j, \tag{17}$$

$$\sum_z p_{i,j,u,z} - \sum_z p_{i,j,z,u} = 0, \quad \forall i < j, \forall u \notin \{i, j\}, \tag{18}$$

$$\sum_u p_{i,j,j,u} - \sum_v p_{i,j,v,j} = -y_{i,j}, \qquad \forall i < j, \tag{19}$$

$$\sum_{u,z} p_{i,j,u,z} \le y_{i,j}(|V| - 1), \qquad \forall i < j, \tag{20}$$

decision variables

$$x_{a,i}, p_{i,j,u,z}, w_{u,z}, y_{i,j} \in \{0, 1\}. \tag{21}$$

In **P2**, first, the objective and the qubits assignment constraints (11) and (12) 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),

TABLE II
LIST OF USED CIRCUITS.

Circuit	Qubit Number	CNOT Number
adr4	14	1,498
clip	14	14,772
co14	15	7,840
rand24	24	18,600
rand32	32	75,600

otherwise $w_{u,z} = 0$, and add Constraint (15). With the help of this indicator, we can simplify the topology constraints (Constraints (13) and (14)). Similarly, we introduce a demand indicator $y_{i,j}$ by adding Constraint (16): $y_{i,j} = 1$ 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 (17), (19), and (20)). By introducing $w_{u,z}$ and $y_{i,j}$, Constraints (4), (5), (6), (8) and (9) 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.

V. 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 [27] using both real-world and randomly-generated 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. Then we also verify the affect of quantum resources and network topology sparsity on the overall performances.

A. Simulation Settings

Quantum Circuits. We pick both real-word and randomly generated circuits as our testing circuits, as shown in Table II. The circuits *adr4*, *clip*, *co14* are chosen from RevLib [34], and their compiled files are available at [35]. The maximum number of the circuits from [35] is no more than 15, so we also randomly generated circuits with 24 and 32 qubits for testing. For these random circuits, we pick $3/4$ of all possible qubit pairs and each qubit pair requests a random number of CNOT gates, which is uniformly picked from $(1, 100)$. We name these circuits *rand24* and *rand32*.

Quantum Clusters. We consider three different QPU clusters with 8 homogeneous GPUs and the QPU in each cluster has different quantum resources (i.e., different number of data/computation qubits m_i). We name them *small*, *medium*, *large*, respectively. We choose similar settings with [17], and the detail parameters is given by Table III. All QPUs are fully connected via an underlying QN G. For simplicity, we set the link cost $b_{u,z}$ for all quantum links in G to one unit.

Baseline Methods. We use the following baselines solutions in our evaluations, which we call *TACO* methods for *Topology-Allocation Co-Optimization*.

- *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.

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

Cluster	QPU #	Comp. Qubit # m_i	Comm. Qubit # n_i	W
small	8	2	4	8
medium	8	3	4	10
large	8	4	4	12

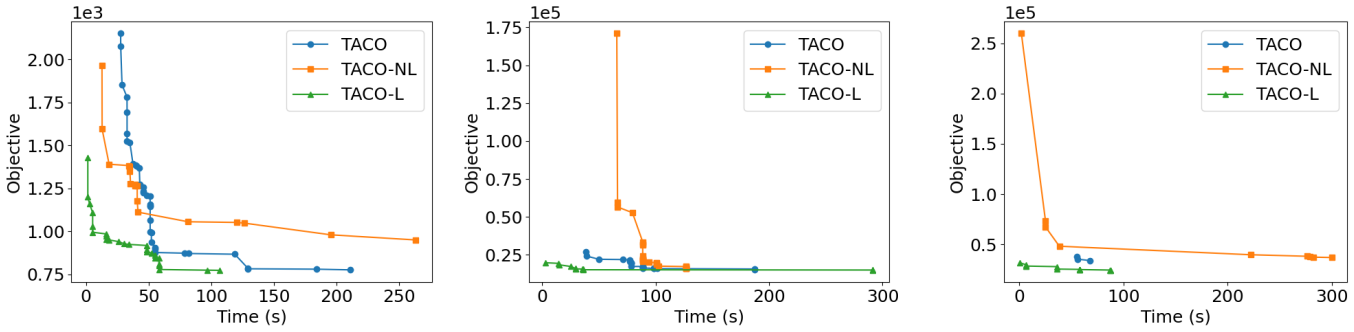
- *TACO-NL*. We use the original formulation of **P1** and directly feed it to Gurobi with the newest non-linear expression feature. Gurobi is responsible for automatically decomposing all terms to linear.
- *TACO-L*. We use Gurobi to diectly solve the reformulated problem of **P2**, which is partially linearized.

B. Simulation Results

Efficiency - Solving Time. We first compare the efficiency of the three baselines. We pick *adr4*, *rand24* and *rand32* as the three testing circuits, and the QPU cluster settings are picked according to the total number of qubits in these circuits. That is, we use *small*, *medium*, and *large* QPU clusters for *adr4*, *rand24*, and *rand32*, respectively. The results of performances are shown in Fig. 4.

First, our partially linearized formulation *TACO-L* is generally the most efficient for all three circuits. It not only finds the solution with much less time for all cases, but also achieves the smaller objective value (less total communication cost) in some cases. 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 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 good choice. Note that the lines from some baselines stop early (such as *TACO* for *rand32* over *large*) because they cannot find a better solution in the given time period. Comparing these three cases, obviously the larger circuit and quantum network lead to higher communication cost. But *TACO-L* can always find a good solution quickly. Therefore, we will only report its result in the following simulations.

Quantum Resources - Number of Computation Qubits. We now consider the same circuit on three different clusters which has different of number of data qubits m_i per QPU. We test *adr4*, *clip*, and *co14* over all *small*, *medium*, and *large* clusters. We pick those three circuits because they can fit into all clusters (while *rand24* and *rand32* cannot fit in small due to insufficient computation qubit). W is now fixed to 8 for all clusters to keep the graph density factor unchanged. As shown in Fig. 5, for all three circuits, more quantum resources (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.



(a) *adr4* over *small* (b) *rand24* over *medium* (c) *rand32* over *large*

Fig. 4. Objective values of the solutions from three baselines for three circuits over corresponding QPU clusters.

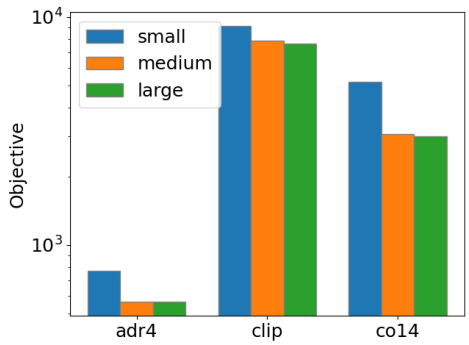


Fig. 5. Objective values of *TACO-L* for three circuits on all *small*, *medium*, *large* clusters, with the same $W = 8$.

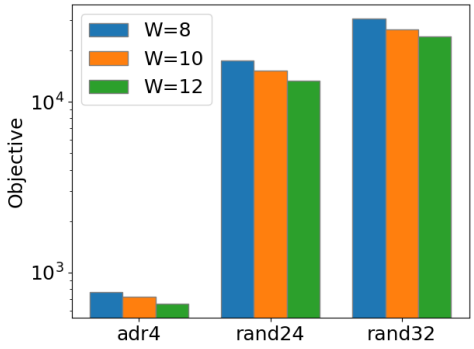


Fig. 6. Objective values of *TACO-L* for three circuits with different topology sparsity requirement W (on *small*, *medium*, *large* clusters, respectively).

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). From Fig. 6, 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 three circuits are distributed to *small*, *medium* and *large* clusters, respectively. For each circuit, only W varies but other parameters of the cluster keep unchanged.

We also show some examples of resulting topology for picked real-world circuits and the two random circuits when we use different W. Because all of the real-world circuits have no more than 16 qubits, so they are all tested on the *small* cluster. The random circuits are tested on *medium* and *large* clusters, respectively. We also change the value of W to see how it affects the final network topology. The resulting topologies are shown in Fig. 7. First, both *adr4* and *clip* have only 14 qubits so they only use 7 out of 8 available QPUs for all W, while *co14* have to use all QPUs to hold its 15 qubits. Second, it is obvious that when the network is allowed to be denser, we can have more edges to further reduce communication cost. Third, all node degree in these topologies is bounded by 4 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.

VI. 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. Simulations over both real-world and random quantum circuits confirm that the partially linearized formulation is more efficient to solve with reduced total communication costs.

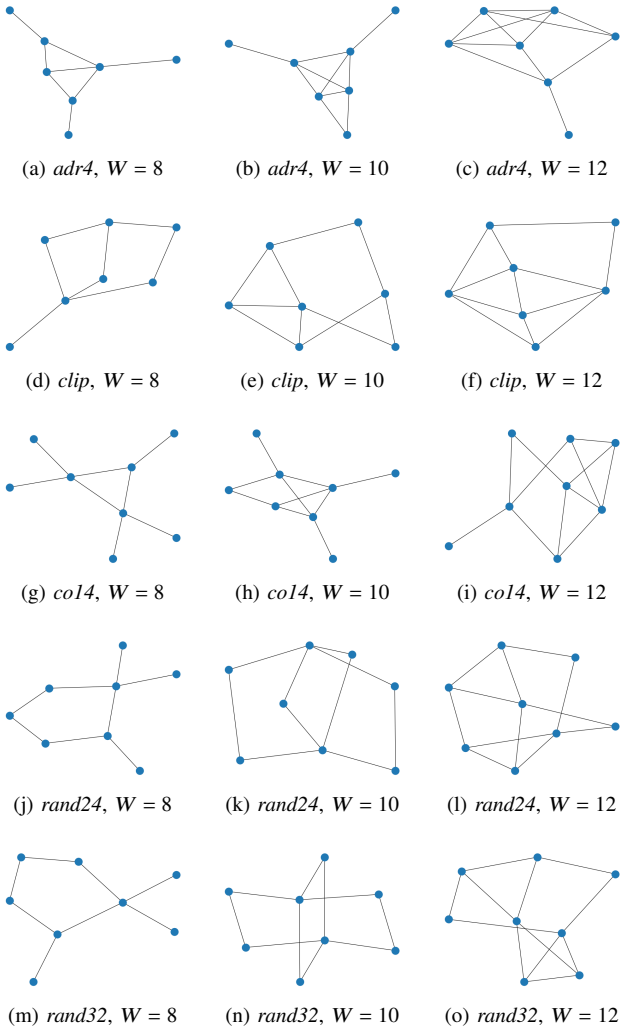
We leave further investigation on 1) testing the proposed methods on clusters with heterogeneous QPUs, 2) how to solve the optimization more efficiently than directly using the solver, and 3) extending the optimization formulation to consider dynamic quantum network, as possible future works.

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



(a) *adr4*, $W = 8$ (b) *adr4*, $W = 10$ (c) *adr4*, $W = 12$

(d) *clip*, $W = 8$ (e) *clip*, $W = 10$ (f) *clip*, $W = 12$

(g) *co14*, $W = 8$ (h) *co14*, $W = 10$ (i) *co14*, $W = 12$

(j) *rand24*, $W = 8$ (k) *rand24*, $W = 10$ (l) *rand24*, $W = 12$

(m) *rand32*, $W = 8$ (n) *rand32*, $W = 10$ (o) *rand32*, $W = 12$

Fig. 7. Optimized topologies for real-world and random circuits with different topology sparsity constraints $W = 8, 10, 12$.

[4] Y. Mao, Y. Liu, and Y. Yang, "Probability-aware qubit-to-processor mapping in distributed quantum computing," in *Pro. of ACM 1st Workshop on Quantum Net. and Distributed Quantum Compu. (QuNet)*, 2023.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] L. Sünkel, M. Dawar, and T. Gabor, "Applying an evolutionary algorithm to minimize teleportation costs in distributed quantum computing," in *Proc. of 2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2024.

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

[16] J. Liu, X. Liu, X. Wei, and Y. Wang, "Topology design with resource allocation and entanglement distribution for quantum networks," in *Proc. of IEEE Int'l Conf. on Sensing, Commu., and Net. (SECON)*, 2024.

[17] Y. Mao, Y. Liu, X. Shang, and Y. Yang, "Network topology design for distributed quantum computing," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2024, pp. 1213–1223.

[18] 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.

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

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

[21] 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.

[22] A. M. Gomez, T. L. Patti, A. Anandkumar, and S. F. Yelin, "Near-term distributed quantum computation using mean-field corrections and auxiliary qubits," *Quantum Sci. & Tech.*, vol. 9, no. 3, p. 035022, 2024.

[23] 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.

[24] 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 Int'l Conf on Quantum Communications, Networking, and Computing (QCNC 2024)*, 2024.

[25] 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.

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

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

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

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

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

[31] 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.

[32] 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.

[33] 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.

[34] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Proc. of Int'l Symp. on Multi-Valued Logic*, 2008, pp. 220–225.

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