IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 May 2026, accepted 16 June 2026, date of publication 22 June 2026, date of current version 1 July 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3706203

RESEARCH ARTICLE

Optimized Compilation for Distributed Quantum Computing

MICHELE BANDINI, DAVIDE FERRARI, (Member, IEEE), STEFANO CARRETTA, AND MICHELE AMORETTI, (Senior Member, IEEE)
University of Parma, 43124 Parma, Italy

Corresponding author: Michele Amoretti (michele.amoretti@unipr.it)

This work was funded by the European Union's Horizon Europe research and innovation programme under grant agreement No. 101102140–QIA Phase 1.

ABSTRACT In many practical applications, quantum algorithms require several qubits, significantly more than those available with current noisy intermediate-scale quantum processors. Distributed quantum computing (DQC) is considered a scalable approach to increasing the number of available qubits for computational tasks. In the DQC setting, a quantum compiler must find the best partitioning for the quantum algorithm and then perform smart non-local operations scheduling to optimize the consumption of Einstein-Podolsky-Rosen (EPR) pairs. In this work, the focus is on minimizing the use of EPR pairs when the circuit structure allows for multiple non-local gates to utilize a single TeleGate operation. This is achieved by using a greedy algorithm that explores the circuit and groups together the gates that could share an EPR pair while also changing the order of commutative gates when necessary. With this preliminary pass, the compiled circuits show reduced depth and EPR usage. Since the quality of each EPR pair quickly deteriorates, the number of non-local gates using the same EPR pair should also be bounded. This means that, depending on the features of the target quantum network, the user can achieve different levels of optimization. Here, it is shown that this approach brings benefits even while assuming a low EPR pair lifetime.

INDEX TERMS Distributed quantum computing, quantum compilation, quantum internet.

I. INTRODUCTION

Most practical applications of quantum algorithms require many more qubits than those provided by current Noisy Intermediate-Scale Quantum (NISQ) platforms. Merely augmenting the number of physical qubits on a single device is not beneficial for the quality of computation because of the increasing noise [1]. Future devices will adopt quantum error correction to extract a few high-quality logical qubits from many noisy physical qubits. Therefore, to supply users with many logical qubits, it could prove advantageous to exploit the Distributed Quantum Computing (DQC) paradigm [2], [3], [4], which involves splitting the circuits into subprograms that are distributed for execution over networked quantum processing units (QPUs).

DQC efficiency and effectiveness will also depend on a well-designed quantum compiler, which is responsible for finding a suitable partitioning of the quantum algorithm and then appropriately scheduling non-local operations (i.e., two-qubit gates across different QPUs) to keep network resources to a minimum. Moreover, the quantum compiler must compute proper local transformations for each partition.

Compiling a quantum circuit is a very challenging task, even for a single-QPU architecture [5]. Indeed, it is an NP-complete problem [6] that relatively few papers have tackled in the DQC context. An upper bound on the overhead induced by quantum circuit compilation for DQC was found by Ferrari et al. [7]. A common approach is to represent quantum circuits as edge-weighted graphs, with qubits as vertices. Edge weights correspond to the number of two-qubit gates [8], [9], [10], [11], reducing the problem to a minimum k-cut problem. Another idea is to represent circuits as bipartite graphs, with one set of vertices for the qubits

The associate editor coordinating the review of this manuscript and approving it for publication was Tomas F. Pena.

and another for the gates, and to assign qubits to QPUs by means of a dynamic programming algorithm that seeks to minimize the number of non-local operations [12]. In [13], the compilation problem is modeled as a generalization of the quickest multi-commodity flow problem and is solved by means of techniques from the Operation Research literature. The theoretical analysis of the proposed solution is very sound, and the matlab-based evaluation provides interesting insights, considering two different network topologies (meaning that entanglement is not assumed to be a network resource available to any pair of nodes). In [14], the authors introduce several techniques for solving the compilation problem, using Steiner trees to detect and reuse common connections, further reducing the cost of entanglement sharing within the network. In [15], the authors propose a gate reordering approach.

In a previous work on this topic, Ferrari et al. [16] introduced a modular compilation framework for DQC and presented the experimental evaluation of a compiler prototype. In that case, the authors considered network topologies with sparse connectivity, with EPR pairs (i.e., maximally entangled bipartite states) established between non-adjacent QPUs by making intermediate QPUs play the role of quantum repeaters – an assumption that fits network-on-chip and local area network scenarios.

In this paper, an improved framework is presented for a compiler that copes with all networking scenarios. The compiler can recognize when multiple non-local gates can be implemented with just one EPR pair, thus optimizing the consumption of network resources. The novelty of the proposed approach lies in a clever integration of non-local gate grouping, reordering, and scheduling, which results in the possibility of setting the optimization level. This is a common feature in compilers for classical computing, but it is rather new in quantum compilers.

The paper is organized as follows. In Section II, the considered DQC system model is presented. In Section III, the compilation process is illustrated in detail. In Section IV, the experimental evaluation of the compiler is presented and discussed. Finally, Section V concludes the paper with an outline of future work.

II. DQC SYSTEM MODEL

In this section, the considered DQC system model is described, elucidating the main assumptions and the way non-local quantum gates can be implemented.

A. ASSUMPTIONS

The considered quantum computers are equipped with quantum communication capabilities, as well as Local Operations and Classical Communication (LOCC). Quantum connections between QPUs may be direct or mediated by quantum repeaters [17], [18], [19]. In both cases, the quantum communication channel can provide EPR pairs, which are assumed to be in the Bell state $|\phi^+\rangle$.

Since EPR pairs enable non-local gates, they are the primary resource for DQC. Therefore, in the proposed compiler, a DQC system is modeled as a graph $G = (V, E)$ such that each vertex $v \in V$ corresponds to a QPU, and each edge $(v_1, v_2) \in E$ indicates that EPR pairs may be shared between v_1 and v_2. Fully connected graphs match the vision of the Quantum Internet [20]. Sparse graphs, on the other hand, correspond to local area networks, either on-chip or room-scale.

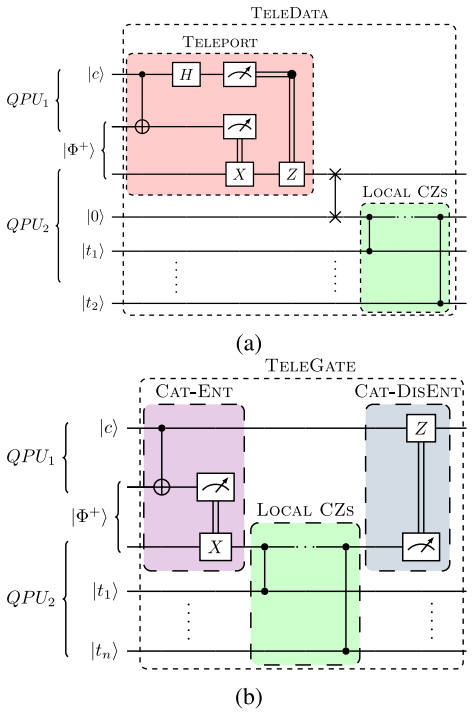


FIGURE 1. (a) Circuit representation of `TeleData` by means of the *Teleport* primitive. `TeleData` moves the quantum state of a data qubit $|c\rangle$ to one qubit of an EPR pair and, then, swap it to a free data qubit. The state of the EPR pair and $|c\rangle$ are lost in the process. Multiple `CZ` acting on the teleported qubit can then be executed. (b) Circuit representation of `TeleGate` by means of *Cat-Ent* and *Cat-DisEnt* primitives. After the *Cat-Ent* operation, the second qubit of the EPR pair participates in an entangled state with the control qubit. Multiple `CZ` with same control qubit and different target can be executed between *Cat-Ent* and *Cat-DisEnt*. It is worth noting that, between *Cat-Ent* and *Cat-DisEnt*, the control qubit is entangled with the EPR pair's one and cannot be targeted by other gates.

B. IMPLEMENTATION OF NON-LOCAL QUANTUM GATES

Non-local gates can be realized using three communication primitives referred to as *Teleport* [21] (quantum state teleportation), *Cat-Ent* (cat-entanglement), and *Cat-DisEnt* (cat-disentanglement) [22], [23]. These primitives enable the execution of two distinct non-local operations, specifically `TeleData` and `TeleGate` [1], [24], [25], as shown in Figure 1.

III. COMPILATION PROCESS

The compilation process is illustrated by the flowchart in Figure 2. The proposed compiler performs a number of *passes*, with some degree of parallelism between them. The first one is qubit assignment, which deals with the partitioning

of the input circuit. Then, the passes that deal with non-local gates (grouping, reordering, and scheduling) can be executed at the same time as the local mapping pass. The final pass is the local routing one.

In situations of practical concern, the availability of entanglement is limited, and its usage is costly. This means that minimizing the number of required EPR pairs can produce important performance and cost benefits and can even impact the effective feasibility of some quantum circuits. To address this challenge, non-local gate grouping and non-local gate reordering have been included as optional passes in the compiler. Their activation within the compilation process enables optimized output circuits but requires a specific non-local gate scheduling strategy.

The three passes concerning non-local gates are the main contributions of this work. Their detailed descriptions are provided in the following subsections.

The local mapping and routing passes were thoroughly presented in previous works [16], [26], [27]. Therefore, in this paper, they are not described in detail, nor are they included in the performance evaluation (Section IV). The local mapping pass associates the qubits of each sub-circuit with the qubits of the QPU that will execute that sub-circuit. The local routing pass scans each sub-circuit and, for every gate that has to be executed across qubits of the QPU that are not directly connected, according to the coupling map of the device, computes the shortest sequence of necessary SWAP gates.

The current implementation of the compiler is available on GitHub [28]. Input circuits are described using OpenQASM [29], while output circuits are represented using Qoala [30], a unified program format for hybrid interactive classical-quantum programs. This means that the compiler's output is ready to be executed on real DQC platforms whose nodes support the Qoala language.

A. QUBIT ASSIGNMENT

The goal of the qubit assignment pass is to partition the circuit in order to minimize the communication cost, i.e., the number of consumed EPR pairs. The approach used here is similar to the one presented in [16], with one key difference: since grouping techniques may be employed later to utilize one EPR pair for more than one non-local operation, the goal cannot simply be to minimize the number of non-local gates, but also to create a situation where the grouping strategy described in Section III-B works better.

As a premise, the current qubit assignment pass tries to split the input circuit by assigning the same number of qubits to each QPU. That is, if the input circuit has n_q qubits and the target DQC system has N QPUs, then n_q/N qubits will be tentatively assigned to each QPU.

An undirected weighted graph $G_c(V_c, E_c)$ is used to represent the circuit c, where each vertex $v \in V_c$ represents a qubit in c, and the weight $W(e)$, for each $e \in E_c$, represents the number of EPR pairs needed to connect the two qubits in case they are in two different QPUs. Then the problem simplifies to one of graph partitioning, where the objective is to compute a k-way partitioning such that the sum of inter-partition edge weights is minimized. Here, the same strategy adopted in [16] is used.

The harder task is to properly count the number of EPR pairs that are needed. As can be seen in Figure 3, if the count was done on the number of two-qubit gates, the partition computed as the best one would need two EPR pairs to execute all the non-local gates, as shown in Figure 3b. With the grouping strategies described in Section III-B, the result is that only one EPR pair is used, as shown in Figure 3c. Here, the three CZ gates are considered to have q_1 as their control and q_0 as their target, and therefore the H gate does not cause problems since it acts on the target. The R_z can commute, as per the rules stated in Section III-C. Unlike the grouping strategy described in Section III-B, two-qubit gates that only share one qubit are always considered to interfere with each other because a proper check can only be done with the qubit mapping finalized.

B. NON-LOCAL GATE GROUPING

This optimization pass works in a greedy manner by grouping non-interfering gates into a single multi-qubit instruction before the compilation phase. The pseudo-code in Algorithm 1 describes the optimization process. It accepts a gate array GA as input and, as it progresses through the process, creates an output circuit OC that contains single or grouped gates. There is a limit D_{max} to the number of consecutive gates that can be included in a group. The reasons for this are better explained in Section III-B2.

1) ALGORITHM BREAKDOWN

For each qubit, a working memory is maintained, which initially points to nothing but could later point to a gate group that is created during the search for non-local gates. All the gates in GA are scanned in order. Every time the optimization algorithm finds a new gate that has no qubit present in any qubit's working memory, it checks if it is local; in that case, it simply puts it into OC; otherwise, it creates a new group that contains only the gate and makes every memory relative to the qubits of the gate point towards the group.

If the gate instead has some qubits in a working memory, either its own or another, the gate-grouping pass checks if they all point to the same group: if they do, the pass checks whether the gate interferes; if they do not, the gate is considered as interfering with both groups. Otherwise, a gate is considered as interfering if it acts on a qubit which is being used as control for a non-local gate, or if it tries to establish a TeleGate using a qubit that is currently being used as target or control.

If it does not interfere, the gate-grouping pass adds it to the group in question. This triggers a control that checks if the group has reached the maximum number of gates per EPR pair D_{max}: if it has, the group is added to OC, and the memory is released.

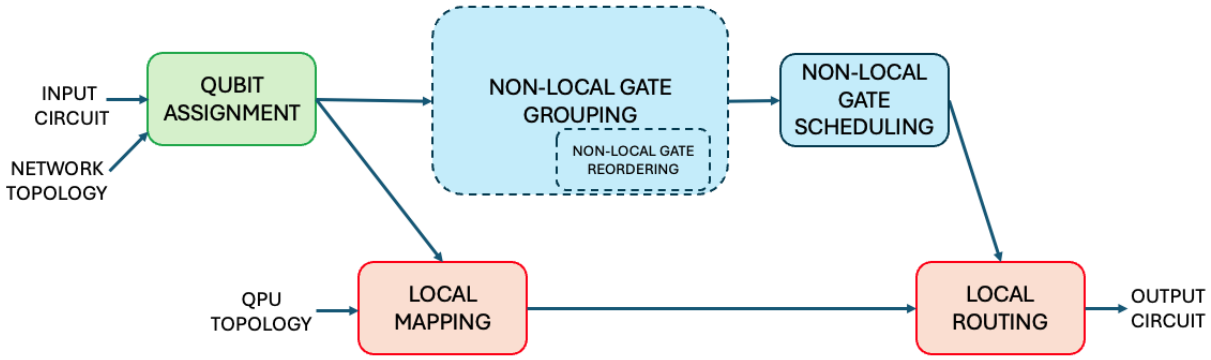


FIGURE 2. Flowchart describing the compilation process. The input includes an abstract description of a quantum circuit and a high-level description of the network configuration. The mandatory passes are qubit assignment, non-local gate scheduling, local mapping and local routing. Optional passes are non-local gate grouping and non-local gate reordering. The passes concerning non-local gates are outlined, as they are the novel contribution of this work.

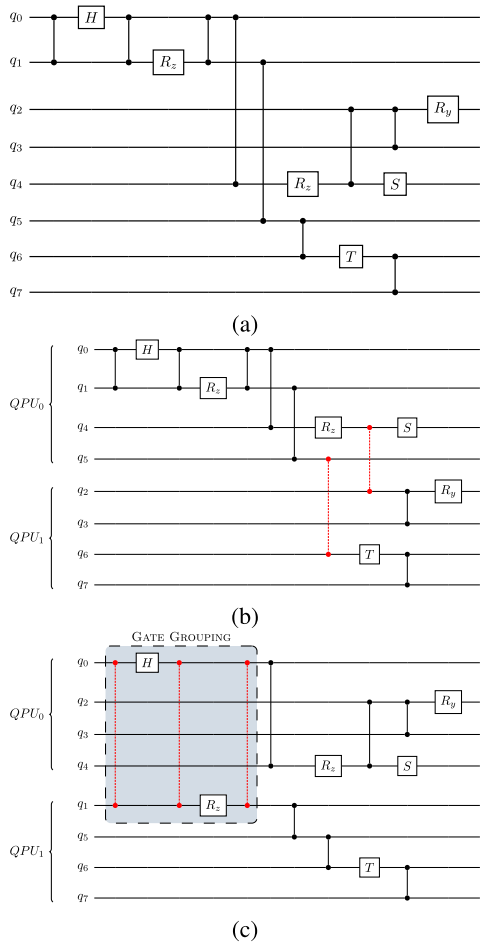


FIGURE 3. Example of qubit assignment. (a) Input circuit. (b) Baseline qubit assignment as in [16]. (c) Qubit assignment with non-local gate grouping.

If the gate interferes with the group, it checks if it can commute with the next gate; in that case, it repeats the process with the next gate in *GA*. This can also happen repeatedly until either the gate can be put into a group, the next gate does not commute, or the end of *GA* is reached. When this ends, the optimizing algorithm starts again from the first (remaining) qubit.

If the gate interferes with a group and cannot be solved by reordering, the optimization module simply adds all the gates in the group to *OC*, releases the memory, and repeats the process again. In either case, every time a gate is placed in a group or directly in *OC*, it is removed from *GA*, and the optimization process starts again by scanning the first (remaining) gate.

2) OPTIMIZATIONS WHEN ENTANGLEMENT DURATION IS LIMITED

So far, in the literature on compilers for DQC, the approach has always been to maximize the number of non-local quantum gates that can be implemented with a single EPR pair. As anticipated above, the non-local gate grouping pass proposed in this work defines a settable limit D_{max} on the maximum depth of a group, which is a fair representation of the time spent keeping the same EPR pair in use.

This approach is realistic, as it is difficult to keep an entangled pair for a very long time in practical contexts, particularly over the Quantum Internet. Of course, this reduces the performance gains from the gate grouping, but as shown in Section IV, the results are still positive even with a very small D_{max}. This makes the proposed algorithm suitable for real applications. In this paper, we always set D_{max} manually to be the same for all links, but it could also be changed adaptively to reflect the quality of the network links in terms of EPR pair fidelity.

C. NON-LOCAL GATE REORDERING

As shown in Figure 3, sometimes, to better group the non-local gates and minimize EPR usage, gate commutativity can be capitalized on to achieve improved results. This is better done after the qubit assignment (described in Section III-A) has been completed. Therefore, the non-local gate reordering strategy is used inside the non-local gate grouping pass (lines 26–29 in Algorithm 1).

Algorithm 1
Input: Array of gates GA, network layout N, number of qubits m, maximum duration of EPR pair D_{max}
Output: circuit OC with grouped-together gates

```
function GATE-GROUPING(GA, N, m, D_max)
    OC ← Null
    M ← Null
    for i from 1 to m do
        M[i] ← Null
    end for
    j ← 0
    while length(GA) > 0 do
        CHECK-GATE(j, OC, M, D_max)
    end while
end function

function CHECK-GATE(j, OC, M, D_max)
    g ← GA[j]
    if ∀i ∈ g.qubits, M[i] == Null then
        if g is local then
            append g to OC
        else
            G ← [g]
            for all i ∈ g.qubits do
                M[i] points to G
            end for
        end if
    else if ∀i ∈ g.qubits, M[i] points to the same Group, where g can be added then
        G ← the Group which M[i] points, ∀i
        append g into G
    else if GA[j+1] exists and can commute with g then
        j ← j + 1
        return
    else
        G ← the first Group s.t. ∃i s.t. M[i] points to it
        append G into OC
        ∀i ∈ g.qubits, M[i] ← Null
        j ← 0
        return
    end if
    if G.length == D_max then
        append G into OC
        ∀i ∈ g.qubits, M[i] ← Null
    end if
    delete GA[j]
    j ← 0
    return
end function
```

In general, when compiling, the objective is to obtain an equivalent circuit. Sometimes, gates can be applied in different orders without changing the resulting unitary. This occurs when the matrices representing the gates share an eigenbasis. For a lengthy circuit with many 2-qubit gates, there may be a need to make many computations, which could add up to be costly.

However, when compiling, only a few basis gates are used, as they form a complete set. This means that, knowing the computational basis, some rules can be defined for the considered gates and applied to verify whether two gates can be commuted.

The main rules adopted by this pass are listed below.

- An R_x gate commutes with a CX gate if the rotation is applied to the target qubit
- An R_z gate commutes with a CX gate if the rotation is applied to the control qubit
- An R_z gate always commutes with a CZ gate
- Two consecutive CX gates commute if their control qubit is the same
- Two consecutive CX gates commute if their target qubit is the same
- Two consecutive CZ gates always commute

This means that both the type of the gates and the qubits to which they are applied, in order, are important to check commutativity.

D. COMPUTATIONAL COMPLEXITY ANALYSIS

This section is focused on the complexity analysis of the new passes introduced in this article. The analysis of the other passes can be found in [16].

Without the non-local gate reordering pass, the asymptotic complexity is $O(N)$, where N is the number of gates in the input circuit. This is straightforward to prove: each gate is scanned once; thus, even if for each gate a number of operations are performed, the complexity still scales linearly with N. On the other hand, if the non-local gate reordering is considered, the worst-case complexity rises to $O(N^2)$, since for each gate it is possible that every subsequent gate needs to be scanned.

However, experimentally the worst-case is not very representative of the real computational complexity: the execution time by deactivating non-local gate reordering is only marginally better. This can be explained by observing that non-local gate reordering happens rarely — only when a gate interferes with the previous gates, and only if it is followed by a commuting gate. The worst case scenario is unrealistic: it needs multiple consecutive commuting gates, all interfering with each other, which means that the circuit could probably be rewritten in a simpler form.

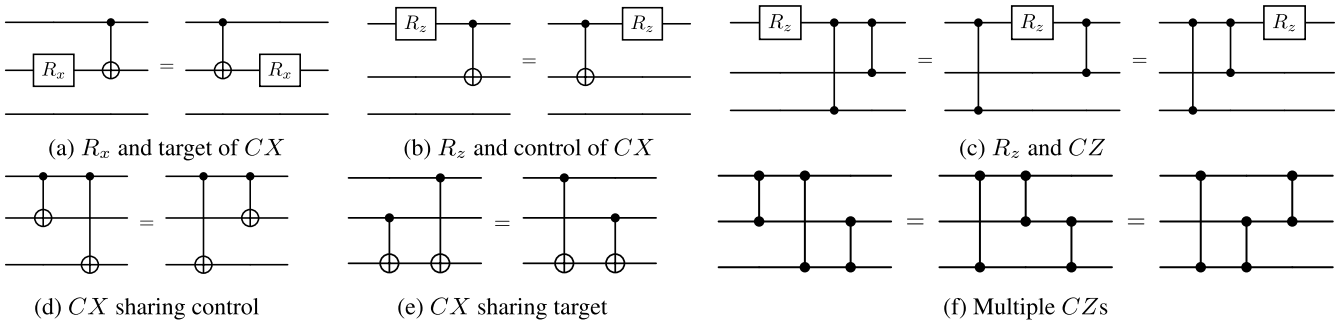
In practice, for well-written quantum circuits, the optimization complexity can be considered roughly linear even with non-local gate reordering. Therefore, the optimization passes can be efficiently employed even for large input circuits.

Importantly, grouping non-local operations is beneficial for the non-local gate scheduling and local routing passes, whose complexities depend on the number of non-local operations [16].

E. NON-LOCAL GATE SCHEDULING

Gate scheduling is performed in a simple way after gate grouping (described in Section III-B) has been completed. Every non-local gate is inside a group, even if the group includes only one non-local gate. Therefore, every other gate is local and can be scheduled directly, as explained in [16].

When a group is reached, a *Cat-Ent* is inserted. Then, every gate in the group is compiled in the same manner as described in [16]. At the end of each group, a *Cat-DisEnt* is inserted, completing the `TeleGate` primitive. If the network topology is complete, there is always a direct connection between the two QPUs. Otherwise, the shortest path between



(a) R_x and target of CX (b) R_z and control of CX (c) R_z and CZ

(d) CX sharing control (e) CX sharing target (f) Multiple CZs

FIGURE 4. Commutation rules.

them has to be computed, and intermediate nodes are used as quantum repeaters.

IV. EXPERIMENTAL EVALUATION

The compiler has been tested using circuits from the QASM-Bench [31] benchmark suite, an OpenQASM benchmark suite for NISQ evaluation. Each circuit is characterized by the number of qubits n_q and depth. The network topologies considered are complete, which means that each quantum node in the DQC system can share EPR pairs with any other node. Quantum nodes are characterized by the number of data qubits n_d and the number of communication qubits n_c. Since the experimental evaluation regarded only non-local gate related passes, the characterization of the internal connectivity (coupling map) of the quantum nodes is not relevant.

In Table 1, compilation results are reported for a large set of practical circuits, considering two quantum computers equipped with enough data qubits (i.e., $n_q/2 \leq n_d < n_q$, n_d being a power of 2) and four communication qubits (i.e., $n_c = 4$). The performance metric considered is the required number of EPR pairs. Three compiler configurations are compared: without optimization (i.e., no gate grouping, no gate reordering), with limited optimization (i.e., grouping non-local gates whose execution fits a limited time slot with gate reordering), and with unlimited optimization (i.e., grouping as many non-local gates as possible with gate reordering). It is evident that the compiler with optimization significantly outperforms the compiler without optimization [16].

For comparison, pytket-dqc [14] was considered because it is the only other compiler for DQC that is available as open-source. Therefore, it was possible to perform a comparison using the same input circuits (from the QASM-Bench collection), the same DQC network configurations, and the same execution environment (a Linux server with two Intel Xeon Silver 4316 CPUs at 2.3 GHz, 20 cores, and 512 GB of RAM, located in the HPC facility of the University of Parma). pytket-dqc provides three different compilation strategies. The most powerful one was chosen, i.e., `CoverEmbedding`, and we compiled QFT, QV, and Multiplier circuits of increasing width n, targeting two QPUs equipped with $\lfloor n/2 \rfloor + 1$ data qubits and 4 communication qubits each. It was observed that `CoverEmbedding` produces a distribution of the input circuit whose cost (in terms of the number of EPR pairs) is calculated as if there were no limit to the number of communication qubits. To calculate the cost while taking into account the actual number of communication qubits, it is necessary to further process the distribution, which takes a very long time for large circuits. Conversely, the proposed compiler is very fast at producing the circuit distribution, as shown in Table 2. From that table, it may be observed that, for small circuits, pytket-dqc wins in terms of the number of EPR pairs (usually lower) but loses in terms of depth (usually higher); instead, for large circuits, pytket-dqc seems to produce distributed circuits

TABLE 1. The name and the number of qubits in the input circuit are in the first two columns, followed by the number of non-local gates (corresponding to the baseline number of EPR pairs when no optimization is applied), then the number of EPR pairs when limiting the EPR lifetime to 3 time slots, and finally when optimizing without limitations.

		# EPR pairs		
Circuit	n_q	no opt	limit=3 opt	unlimited opt
Adder	28	11	7	5
Adder	64	11	7	5
Adder	118	11	7	5
Cat	35	1	1	1
DNN	33	36	20	6
DNN	51	52	28	6
GHZ	40	1	1	1
Ising	34	2	2	2
KNN	41	36	18	6
Multiplier	45	290	209	159
Multiplier	75	862	604	459
Multiplier	350	13228	11108	8761
QFT	29	410	230	118
QFT	63	702	364	51
QFT	320	702	364	51
QuGAN	39	40	22	6
QuGAN	71	72	45	13
QuGAN	111	112	66	14
QuGAN	395	408	319	29
QV	32	589	589	585
QV	100	6526	6522	6442
Swap Test	25	24	12	4
Swap Test	41	36	18	6
Swap Test	83	80	40	4
W-State	76	2	2	2
W-State	118	2	2	2
Square root	45	5415	4411	3983

TABLE 2. Comparison between the proposed compiler and pytket-dqc.

		# EPR pairs		Depth		Time [s]	
Circuit	n_q	proposed	pytket-dqc	proposed	pytket-dqc	proposed	pytket-dqc
Adder	28	5	3	188	374	0.48	19
Adder	64	5	3	384	710	2.09	58
Adder	118	5	3	588	1214	5.18	978.33
Cat	35	1	1	40	72	0.2	1.06
Ising	34	2	1	16	21	0.2	3.54
KNN	41	4	1	137	167	0.23	14.7
Multiplier	45	159	37	2287	4516	4.7	3247.39
Multiplier	75	459	70	6559	12708	15.4	38543.88
Multiplier	350	8761	n.a.	134751	n.a.	1521.77	> 48 h
QFT	29	118	222	612	584	1.89	19212.58
QFT	63	51	n.a.	512	n.a.	6.06	> 48 h
QFT	320	51	n.a.	2311	n.a.	194.79	> 48 h
QV	32	585	600	1748	1062	6.56	1770
QV	100	6442	n.a.	18667	n.a.	196.15	> 48 h
Swap Test	25	4	1	90	167	0.1	5.46
Swap Test	41	4	1	136	166	0.21	14.66
Swap Test	83	7	1	266	347	0.72	92.35
W-State	76	2	2	156	383	0.75	24.14
W-State	118	2	2	240	593	1.88	83.1

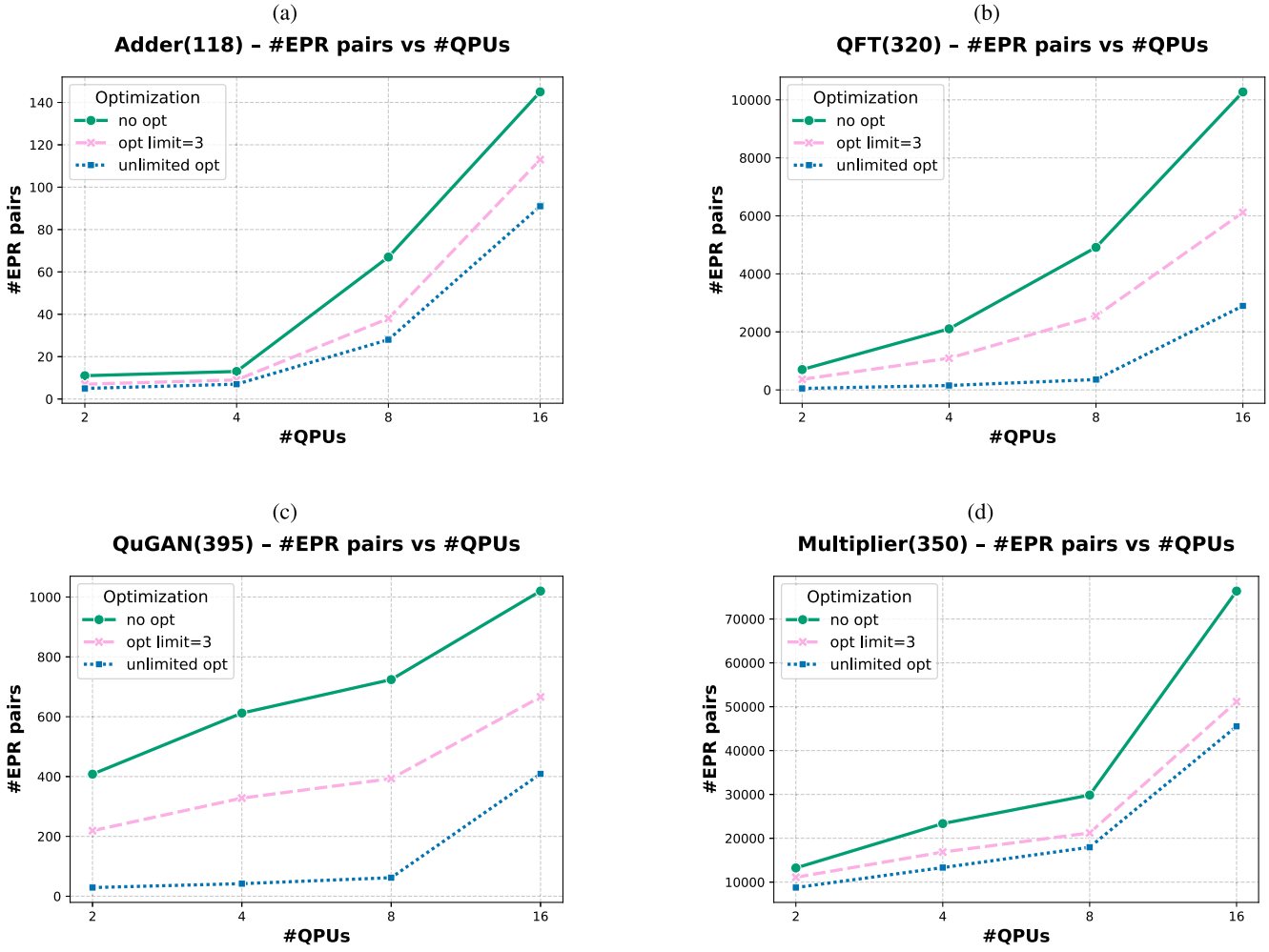


FIGURE 5. Number of required EPR pairs with respect to the number of QPUs, for selected quantum circuits with very different features. In each plot, different compiler configurations are compared. For (a) and (b), the four considered DQC architectures include devices with 64, 32, 16 and 8 data qubits, respectively. For (c) and (d), instead, the four considered DQC architectures include devices with 256, 128, 64 and 32 qubits, respectively. Each device has 4 communication qubits.

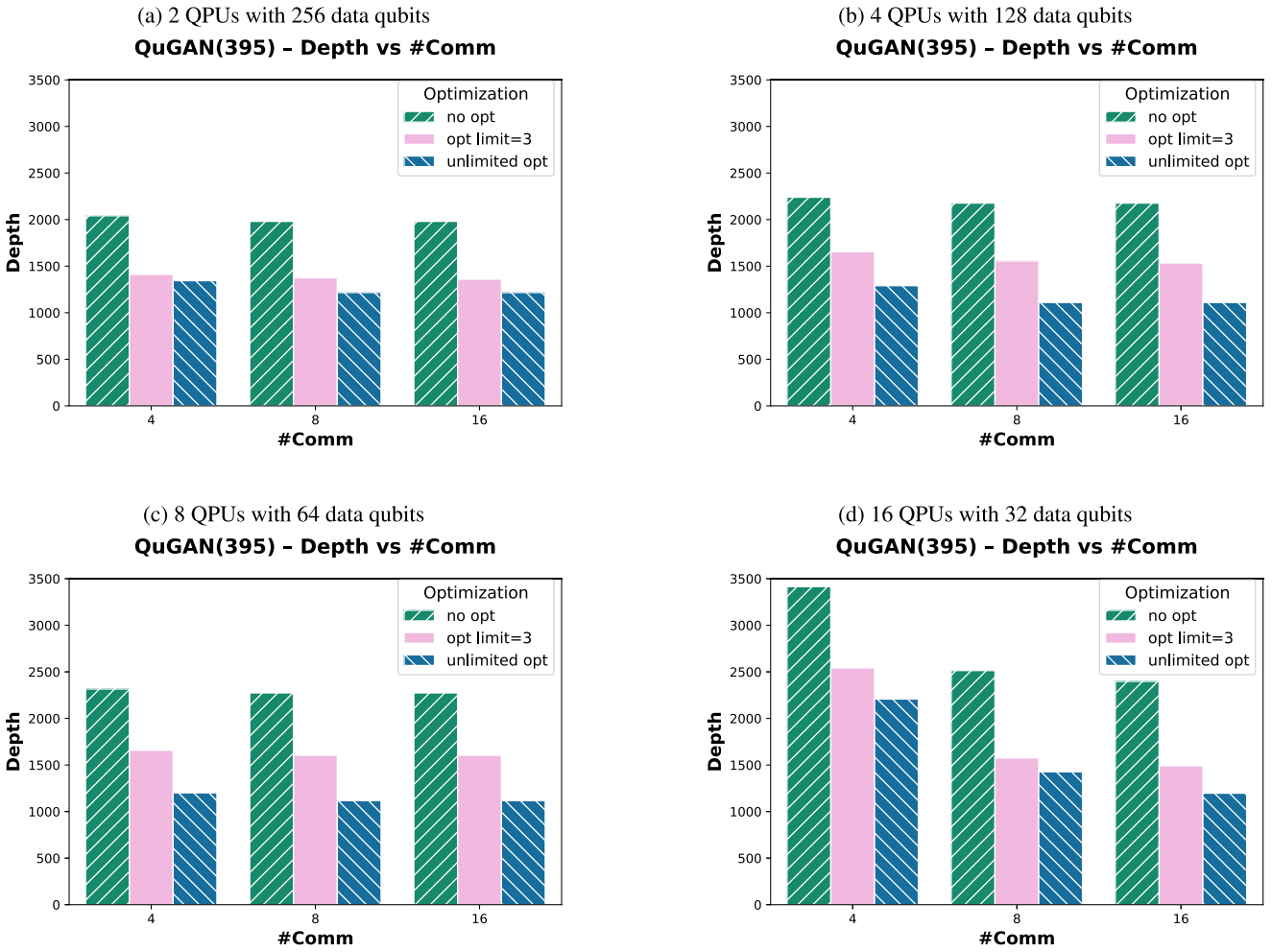


FIGURE 6. Depth of the compiled circuit (QuGAN with 395 qubits) for increasing number of communication qubits for each node. Each plot corresponds to a specific DQC architecture: (a) 2 nodes with 256 data qubits, (b) 4 nodes with 128 data qubits, (c) 8 nodes with 64 data qubits, and (d) 16 nodes with 32 data qubits.

with a higher number of EPR pairs and greater depth. In some relevant cases, pytket-dqc does not manage to produce the output in less than 48 hours, which is the limit imposed by the HPC facility of the University of Parma.

For selected circuits, Figure 5 shows how many EPR pairs are used when the circuits are compiled for different DQC architectures. Specifically, from one architecture to another, the number of QPUs grows exponentially, while the number of qubits per QPU decreases accordingly. In each plot, three compiler configurations are compared: without optimization, with unlimited optimization, and with limited optimization. It can be observed that, as expected, the number of requested EPR pairs grows with the number of QPUs for all three compiler configurations. Importantly, the growth is less steep when optimization is used.

For the QuGAN circuit with 395 qubits, Figure 6 shows the depth of the compiled circuit, considering devices with an increasing number of communication qubits (from 4 to 8 to 16) and four DQC architectures with an increasing number of QPUs and a decreasing number of data qubits. It may be observed that increasing the number of communication qubits is usually beneficial for reducing the depth of the compiled circuit. This benefit becomes more evident as the number of partitions increases. For the QFT circuit with 320 qubits and for the Multiplier circuit with 350 qubits, the same analysis is reported in Figure 7 and Figure 8. Especially in the QFT case, the benefit of the increasing number of communication qubits is glaring.

Another relevant insight is the observation that the aforementioned metrics (the number of EPR pairs and the depth of the output circuit) usually worsen when the number of QPUs is significantly larger than necessary. For example, having QPUs with 128 data qubits, it is sufficient to use 3 of them to execute the QFT(320) circuit, resulting in 102 EPR pairs and a depth of 2389 with the best possible optimization. Using 4 QPUs would result in 153 EPR pairs and a depth of 2467. Having QPUs with 64 qubits, it is sufficient to use 5 (resulting in 204 EPR pairs and a depth of 2545). Using 8 QPUs results in 357 EPR pairs and a depth of 2779. Similarly, using 10 QPUs with 32 data qubits is sufficient

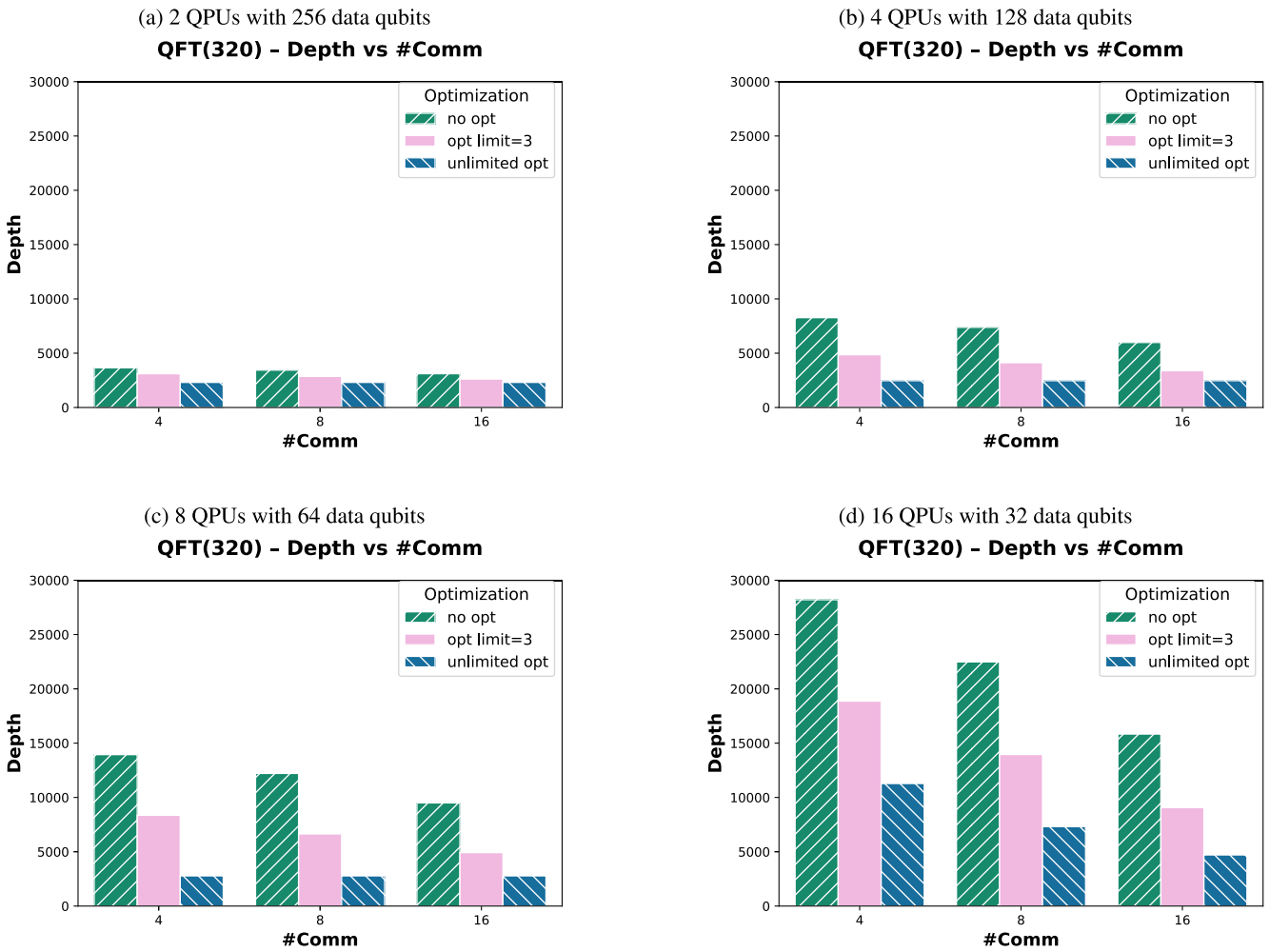


FIGURE 7. Depth of the compiled circuit (QFT with 320 qubits) for increasing number of communication qubits for each node. Each plot corresponds to a specific DQC architecture: (a) 2 nodes with 256 data qubits, (b) 4 nodes with 128 data qubits, (c) 8 nodes with 64 data qubits, and (d) 16 nodes with 32 data qubits.

TABLE 3. Compilation results for selected circuits, considering different DQC architectures. All QPUs have 4 communication qubits.

Circuit	n_q	# QPUs	QPU type	Input Depth	Output Depth (no opt)	Output Depth (limit=3 opt)	Output Depth (unlimited opt)	EPR pairs (no opt)	EPR pairs (limit=3 opt)	EPR pairs (unlimited opt)
Adder	118	2	grid_64_4	132	587	590	588	11	7	5
Adder	118	4	grid_32_4	132	589	588	586	13	9	7
Adder	118	8	grid_16_4	132	647	608	597	67	38	28
Adder	118	16	grid_8_4	132	915	661	643	145	113	91
QFT	320	2	grid_256_4	2550	3651	3108	2311	702	364	51
QFT	320	3	grid_128_4	2550	5069	3983	2389	1404	728	102
QFT	320	4	grid_128_4	2550	8273	4858	2467	2106	1092	153
QFT	320	5	grid_64_4	2550	8798	5733	2545	2808	1456	204
QFT	320	8	grid_64_4	2550	13945	8358	2779	4914	2548	357
QFT	320	10	grid_32_4	2550	15888	10108	2935	6318	3276	459
QFT	320	16	grid_32_4	2550	28206	18870	11284	10272	6119	2897
Multiplier	350	2	grid_256_4	29193	168715	157047	134751	13228	11108	8761
Multiplier	350	3	grid_128_4	29193	179932	157073	144289	21084	15103	12302
Multiplier	350	4	grid_128_4	29193	181460	158834	132837	23348	16857	13320
Multiplier	350	6	grid_64_4	29193	180657	159773	147882	25906	19220	16523
Multiplier	350	8	grid_64_4	29193	187353	156169	145806	29854	21223	17955
Multiplier	350	11	grid_32_4	29193	210719	167095	156972	48960	31174	24269
Multiplier	350	16	grid_32_4	29193	237295	176856	156865	76372	51162	45531
QuGAN	395	2	grid_256_4	396	2041	1412	1344	408	219	29
QuGAN	395	4	grid_128_4	396	2240	1656	1290	612	328	42
QuGAN	395	7	grid_64_4	396	2979	1745	1459	720	388	67
QuGAN	395	8	grid_64_4	396	2316	1658	1201	724	393	62
QuGAN	395	13	grid_32_4	396	3252	2426	2038	889	593	291
QuGAN	395	16	grid_32_4	396	3415	2542	2208	1020	666	409
Swap Test	83	2	grid_64_4	44	415	255	264	80	40	4
Swap Test	83	4	grid_32_4	44	455	215	229	120	60	15
Swap Test	83	8	grid_16_4	44	479	195	204	144	70	28
Swap Test	83	16	grid_8_4	44	487	271	216	152	79	40

and more efficient than using 16. However, there are some exceptions. For example, considering the QuGAN(395) circuit and QPUs with 64 data qubits, 7 QPUs are sufficient but less efficient than 8 QPUs in terms of depth (the number of EPR pairs is the same). All the data are reported in Table 3.

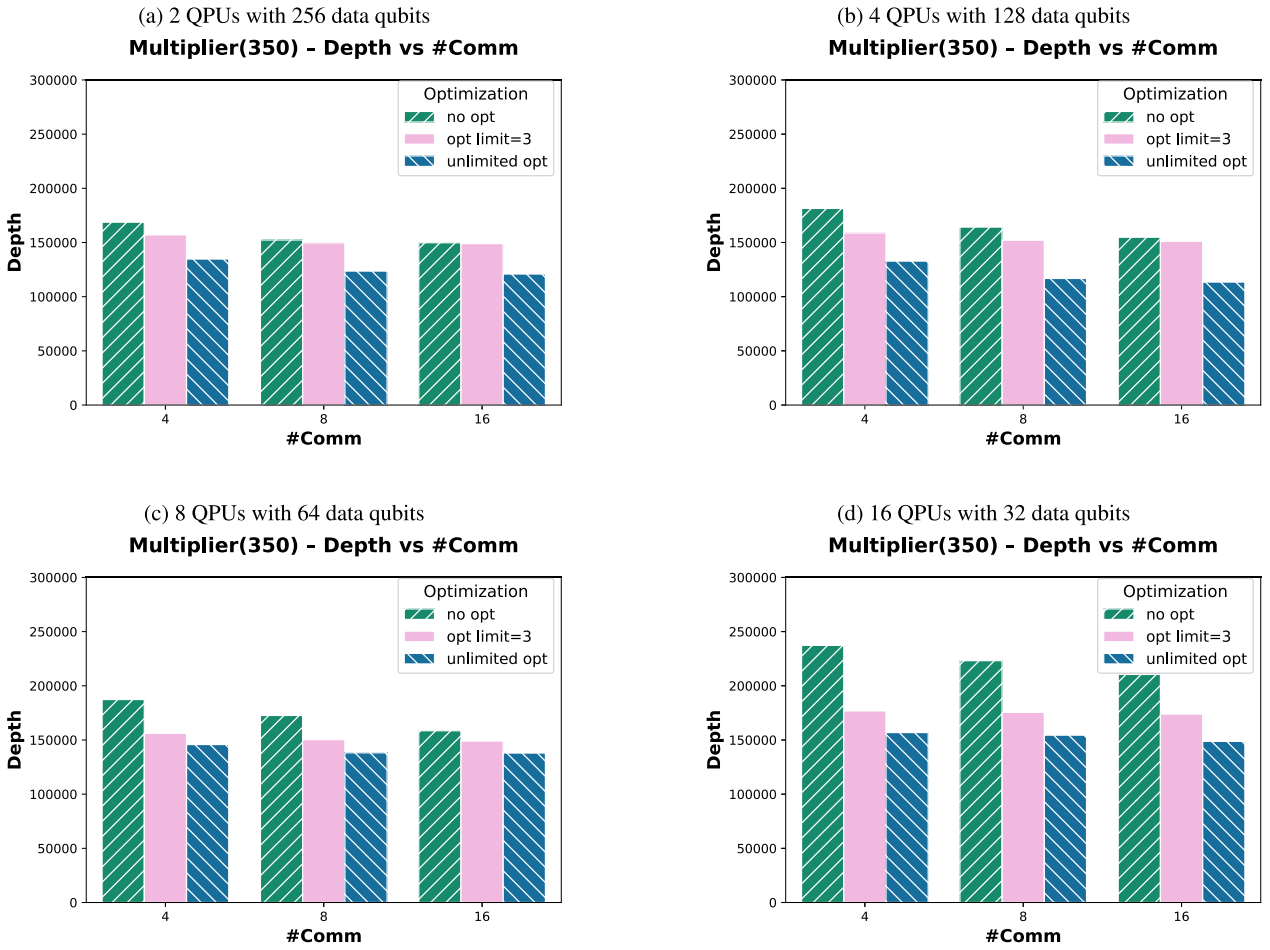


FIGURE 8. Depth of the compiled circuit (Multiplier with 350 qubits) for increasing number of communication qubits for each node. Each plot corresponds to a specific DQC architecture: (a) 2 nodes with 256 data qubits, (b) 4 nodes with 128 data qubits, (c) 8 nodes with 64 data qubits, and (d) 16 nodes with 32 data qubits.

V. CONCLUSION

In this work, a novel compiler for DQC is presented that integrates a non-local gate grouping pass, a non-local gate reordering pass, and an improved non-local scheduling pass. To this end, a greedy algorithm explores the circuit and groups together the gates that could share an EPR pair, while also changing the order of commutative gates when necessary. In this way, the compiled circuits show reduced depth and EPR usage. To take into account that the quality of EPR pairs quickly deteriorates, the number of non-local gates using the same EPR pair can be bounded. Therefore, depending on the features of the target quantum network, the user can achieve different levels of optimization. The experimental results show that the proposed approach brings benefits even when assuming a low EPR pair lifetime.

Regarding future work, there are several interesting directions. Since the scale of the circuits and networks used for the experiments is beyond what current DQC simulators can handle, we decided to leave fidelity estimation out of this work evaluation. A study of how fidelity could be estimated, following the work in [32], would be an interesting future line of research. The compiler can be further tested with sparse network topologies and specific QPU designs, thus leveraging the local mapping pass and the local routing pass. More sparsely connected network topologies would require some nodes to act as repeaters, i.e., to participate in entanglement swapping, as discussed in [16]. In such a case, we would expect an increase in the EPR consumption, although it would be linear in the diameter of the network, as well as the depth. Alternative qubit assignment passes could be introduced to cope with heterogeneous DQC systems. An adaptive D_{max} could be introduced, which changes to reflect the quality of each link in a network. A major foreseen development is the introduction of support for high-dimensional quantum systems (qudits) [33], [34], [35], paving the way for fault-tolerant DQC.

DATA AVAILABILITY

All data and code required to reproduce all plots presented in this work have been made available at https://doi.org/10.5281/zenodo.18789205

REFERENCES

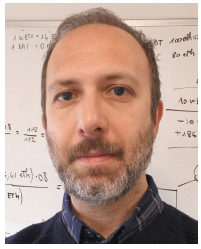
[1] R. Van Meter and S. J. Devitt, "The path to scalable distributed quantum computing," *Computer*, vol. 49, no. 9, pp. 31–42, Sep. 2016.

[2] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, "Distributed quantum computing: A survey," *Comput. Netw.*, vol. 254, Dec. 2024, Art. no. 110672.
[3] D. Barral, F. J. Cardama, G. Díaz-Camacho, D. Faílde, I. F. Llovo, M. Mussa-Juane, J. Vázquez-Pérez, J. Villasuso, C. Piñeiro, N. Costas, J. C. Pichel, T. F. Pena, and A. Gómez, "Review of distributed quantum computing: From single QPU to high performance quantum computing," *Comput. Sci. Rev.*, vol. 57, Aug. 2025, Art. no. 100747.
[4] J. C. Boschero, N. M. P. Neumann, W. V. D. Schoot, T. Sijpesteijn, and R. Wezeman, "Distributed quantum computing: Applications and challenges," in *Proc. Intell. Comput.*, 2024, pp. 100–116.
[5] D. Ferrari and M. Amoretti, "Noise-adaptive quantum compilation strategies evaluated with application-motivated benchmarks," in *Proc. 19th ACM Int. Conf. Comput. Frontiers*, May 2022, pp. 237–243.
[6] A. Botea, A. Kishimoto, and R. Marinescu, "On the complexity of quantum circuit compilation," in *Proc. 11th Annu. Symp. Combinat. Search (SOCS)*, 2021, vol. 9, no. 1, pp. 138–142.
[7] D. Ferrari, A. S. Cacciapuoti, M. Amoretti, and M. Caleffi, "Compiler design for distributed quantum computing," *IEEE Trans. Quantum Eng.*, vol. 2, pp. 1–20, 2021.
[8] O. Daei, K. Navi, and M. Zomorodi-Moghadam, "Optimized quantum circuit partitioning," *Int. J. Theor. Phys.*, vol. 59, no. 12, pp. 3804–3820, Dec. 2020.
[9] R. G. Sundaram, H. Gupta, and C. R. Ramakrishnan, "Efficient distribution of quantum circuits," in *Proc. 35th Int. Symp. Distrib. Comput. (DISC)*, 2021, pp. 41:1–41:20.
[10] E. Nikahd, N. Mohammadzadeh, M. Sedighi, and M. S. Zamani, "Automated window-based partitioning of quantum circuits," *Phys. Scripta*, vol. 96, no. 3, Jan. 2021, Art. no. 035102.
[11] R. G. Sundaram, H. Gupta, and C. R. Ramakrishnan, "Distribution of quantum circuits over general quantum networks," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Sep. 2022, pp. 415–425.
[12] Z. Davarzani, M. Zomorodi-Moghadam, M. Houshmand, and M. Nouri-Baygi, "A dynamic programming approach for distributing quantum circuits by bipartite graphs," *Quantum Inf. Process.*, vol. 19, no. 10, p. 360, Oct. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s11128-020-02871-7
[13] D. Cuomo, M. Caleffi, K. Krsulich, F. Tramonto, G. Agliardi, E. Prati, and A. S. Cacciapuoti, "Optimized compiler for distributed quantum computing," *ACM Trans. Quantum Comput.*, vol. 4, no. 2, pp. 1–29, Feb. 2023.
[14] P. Andres-Martinez, T. Forrer, D. Mills, J.-Y. Wu, L. Henaut, K. Yamamoto, M. Murao, and R. Duncan, "Distributing circuits over heterogeneous, modular quantum computing network architectures," *Quantum Sci. Technol.*, vol. 9, no. 4, Aug. 2024, Art. no. 045021.
[15] R. Mengoni, W. Nadalin, M. Rennela, J. Rotureau, T. Darras, J. Laurat, E. Diamanti, and I. Lavdas, "Efficient gate reordering for distributed quantum compiling in data centers," 2025, *arXiv:2507.01090*.
[16] D. Ferrari, S. Carretta, and M. Amoretti, "A modular quantum compilation framework for distributed quantum computing," *IEEE Trans. Quantum Eng.*, vol. 4, pp. 1–13, 2023.
[17] L. Talsma, A. Inesta, and S. Wehner, "Continuously distributing entanglement in quantum networks with regular topologies," 2024, *arXiv:2402.01527*.
[18] A. Manzalini and M. Amoretti, "End-to-end entanglement generation strategies: Capacity bounds and impact on quantum key distribution," *Quantum Rep.*, vol. 4, no. 3, pp. 251–263, Jul. 2022.
[19] M. Amoretti and S. Carretta, "Entanglement verification in quantum networks with tampered nodes," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 3, pp. 598–604, Mar. 2020.
[20] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, pp. 1–9, Oct. 2018.
[21] A. S. Cacciapuoti, M. Caleffi, R. Van Meter, and L. Hanzo, "When entanglement meets classical communications: Quantum teleportation for the quantum internet," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3808–3833, Jun. 2020.
[22] J. Eisert, K. Jacobs, P. Papadopoulos, and M. B. Plenio, "Optimal local implementation of nonlocal quantum gates," *Phys. Rev. A, Gen. Phys.*, vol. 62, no. 5, Oct. 2000, Art. no. 052317.
[23] A. Yimsiriwattana and S. J. Lomonaco, "Generalized GHZ states and distributed quantum computing," *Contemp. Math.*, pp. 131–147, Feb. 2005.
[24] R. Van Meter, K. Nemoto, W. J. Munro, and K. M. Itoh, "Distributed arithmetic on a quantum multicomputer," *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 354–365, May 2006.
[25] R. Van Meter, W. J. Munro, and K. Nemoto, "Architecture of a quantum multicomputer implementing shor's algorithm," in *Proc. Theory Quantum Comput., Commun., Cryptography*, 2008, pp. 105–114.
[26] D. Ferrari and M. Amoretti, "Efficient and effective quantum compiling for entanglement-based machine learning on IBM q devices," *Int. J. Quantum Inf.*, vol. 16, no. 8, Dec. 2018, Art. no. 1840006.
[27] D. Ferrari, I. Tavernelli, and M. Amoretti, "Deterministic algorithms for compiling quantum circuits with recurrent patterns," *Quantum Inf. Process.*, vol. 20, no. 6, p. 213, Jun. 2021.
[28] M. Amoretti, M. Bandini, and D. Ferrari. (2026). *DQC-NAC Repository*. [Online]. Available: https://github.com/qslab-unipr/dqcnac
[29] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open quantum assembly language," 2017, *arXiv:1707.03429*.
[30] B. van der Vecht, A. T. Yücel, H. Jirovská, and S. Wehner, "Qoala: An application execution environment for quantum internet nodes," 2025, *arXiv:2502.17296*.
[31] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "QASMBench: A low-level quantum benchmark suite for NISQ evaluation and simulation," *ACM Trans. Quantum Comput.*, vol. 4, no. 2, pp. 1–26, Jun. 2023.
[32] P. Escofet, S. Rodrigo, A. Garcia-Sáez, E. Alarcón, S. Abadal, and C. G. Almudéver, "An accurate and efficient analytic model of fidelity under depolarizing noise oriented to large scale quantum system design," *Quantum Sci. Technol.*, vol. 10, no. 3, Jul. 2025, Art. no. 035061.
[33] D. Cozzolino, B. Da Lio, D. Bacco, and L. K. Oxenløwe, "High-dimensional quantum communication: Benefits, progress, and future challenges," *Adv. Quantum Technol.*, vol. 2, no. 12, Dec. 2019, Art. no. 1900038. [Online]. Available: https://advanced.onlinelibrary.wiley.com/doi/10.1002/qute.201900038
[34] L. E. Fischer, A. Chiesa, F. Tacchino, D. J. Egger, S. Carretta, and I. Tavernelli, "Universal qudit gate synthesis for transmons," *PRX Quantum*, vol. 4, no. 3, Aug. 2023, Art. no. 030327. [Online]. Available: https://journals.aps.org/prxquantum/abstract/10.1103/PRXQuantum.4.030327
[35] A. Chiesa, P. Santini, E. Garlatti, F. Luis, and S. Carretta, "Molecular nanomagnets: A viable path toward quantum information processing?" *Rep. Prog. Phys.*, vol. 87, no. 3, Mar. 2024, Art. no. 034501.

MICHELE BANDINI received the B.S. and M.S. degrees in mathematics from the University of Bologna, Bologna, Italy. He is currently pursuing the Ph.D. degree in information technologies with the University of Parma, Parma, Italy. His work is carried out in the context of European Quantum Flagship project Quantum Internet Alliance. His research interest includes the optimization of resources in quantum compiling for DQC.

DAVIDE FERRARI (Member, IEEE) received the Ph.D. degree in information technologies from the Department of Engineering and Architecture, University of Parma, Italy, in 2023. During the Ph.D., he worked on quantum compiling, quantum optimization, and distributed quantum computing. He has been a Research Scholar with the Future Technology Laboratory, University of Parma, working on the design of efficient algorithms for quantum compiling. He is currently a Research Fellow with the Department of Engineering and Architecture, University of Parma. He is involved in the Quantum Information Science (QIS) research initiative with the University of Parma, where he is a member of the Quantum Software Laboratory. His research interests include quantum optimization applications and efficient quantum compiling for local and distributed quantum computing. In 2020, he won the IBM Quantum Awards Circuit Optimization Developer Challenge.

STEFANO CARRETTA received the Ph.D. degree in physics from the University of Parma, Italy, in 2005. He is currently a Full Professor in theoretical physics of matter with the University of Parma. He contributed to put forward some of the first proposals for the use of magnetic molecules as qubits and the first proposal and experimental demonstration of the use of molecular nanomagnets as quantum simulators. He was a member of the Commission of Experts on quantum technologies for the 2021–2027 Italian National Research Program (PNR). He was involved into several national and European projects involving quantum technologies. He is one of the PI of European ERC Synergy Project. He is the co-author of about 180 research articles published in international journals. His research activity is mainly focused on the theoretical modeling of the quantum behavior of magnetic molecules and in quantum information processing. In 2006, he received the ''Le Scienze'' (Italian version of Scientific American) Medal and the President of Italian Republic Medal for his research on molecular nanomagnets. In 2011, he received the prestigious Olivier Kahn International Award for his contribution to the theory of molecular magnetism.



MICHELE AMORETTI (Senior Member, IEEE) received the Ph.D. degree in information technologies from the University of Parma, Parma, Italy, in 2006. In 2013, he was a Visiting Researcher with the LIG Laboratory, Grenoble, France. He is currently an Associate Professor of computer engineering with the University of Parma, where he leads the Quantum Software Laboratory. He authored or co-authored over 150 research papers in refereed international journals, conference proceedings, and books. His current research interests include quantum computing and high performance computing. He is a member of the IEC/ISO JTC 3 ''Quantum Technologies'' standardization committee. He serves as an Associate Editor for IEEE TRANSACTIONS ON QUANTUM ENGINEERING. He is the Use Case Team Lead of European HE project Quantum Internet Alliance. He is the CINI Consortium delegate to the UNI/CT 535 ''Quantum Technologies'' UNINFO Commission, which is the national mirror of the CEN/CENELEC JTC 22 ''Quantum Technologies'' standardization committee.