Optimized Compiler for Distributed Quantum Computing

DANIELE CUOMO, Department of Physics, University of Naples Federico II
MARCELLO CALEFFI, DIETI, University of Naples Federico II
KEVIN KRSULICH, IBM Quantum, T. J. Watson Research Center
FILIPPO TRAMONTO, Kyndryl Italia Innovation Services
GABRIELE AGLIARDI, Department of Physics, Politecnico di Milano and IBM Italia
ENRICO PRATI, Department of Physics, Universitá degli Studi di Milano and IFN-CNR
ANGELA SARA CACCIAPUOTI, DIETI, University of Naples Federico II

Practical distributed quantum computing requires the development of efficient compilers, able to make quantum circuits compatible with some given hardware constraints. This problem is known to be tough, even for local computing. Here, we address it on distributed architectures. As generally assumed in this scenario, *telegates* represent the fundamental remote (inter-processor) operations. Each telegate consists of several tasks: (i) entanglement generation and distribution, (ii) local operations, and (iii) classical communications. Entanglement generations and distribution is an expensive resource, as it is time-consuming. To mitigate its impact, we model an optimization problem that combines running-time minimization with the usage of distributed entangled states. Specifically, we formulated the distributed compilation problem as a dynamic network flow. To enhance the solution space, we extend the formulation, by introducing a predicate that manipulates the circuit given in input and parallelizes telegate tasks.

To evaluate our framework, we split the problem into three sub-problems, and solve it by means of an approximation routine. Experiments demonstrate that the run-time is resistant to the problem size scaling. Moreover, we apply the proposed algorithm to compile circuits under different topologies, showing that topologies with a higher ratio between edges and nodes give rise to shallower circuits.

CCS Concepts: • **Hardware → Quantum computation**; • **Computer systems organization → Distributed architectures**; • **Mathematics of computing → Network optimization;**

Additional Key Words and Phrases: Quantum circuit compilation, Integer Linear Programming

Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti also with FLY, Future communications Laboratory.
Marcello Caleffi, Enrico Prati, and Angela Sara Cacciapuoti also with CNIT, National Inter-university Consortium for Telecommunications.
Filippo Tramonto also with IBM Client Innovation Center during his contribution to this work.
Authors' addresses: D. Cuomo, Department of Physics, University of Naples Federico II, Italy; email: daniele.cuomo@unina.it; M. Caleffi, DIETI, University of Naples Federico II, Via Claudio 62, Italy, 80126; email: marcello.caleffi@unina.it; K. Krsulich, IBM Quantum, T. J. Watson Research Center, Yorktown Heights, New York, 10598; email: kevin.krsulich@ibm.com; F. Tramonto, Kyndryl Italia Innovation Services, Via Circonvallazione Idroscalo snc, Segrate (MI), Italy, 20090; email: filippo.tramonto@gmail.com; G. Agliardi, Department of Physics, Politecnico di Milano, Piazza Leonardo da Vinci, Milano, 20133 and IBM Italia, Via Circonvallazione Idroscalo, Segrate (MI), 20090; email: gabrielefrancesco.agliardi@polimi.it; E. Prati, Department of Physics, Universitá degli Studi di Milano, Via Celoria 16, Milano, 20133 and IFN-CNR, Piazza Leonardo da Vinci 32, Milano, Italia, 20133; email: enrico.prati@unimi.it; A. S. Cacciapuoti, DIETI, University of Naples Federico II, Via Claudio 62, Italy, 80126; email: angelasara.cacciapuoti@unina.it.

2643-6817/2023/02-ART15
https://doi.org/10.1145/3579367

ACM Reference format:
Daniele Cuomo, Marcello Caleffi, Kevin Krsulich, Filippo Tramonto, Gabriele Agliardi, Enrico Prati, and Angela Sara Cacciapuoti. 2023. Optimized Compiler for Distributed Quantum Computing. *ACM Trans. Quantum Comput.* 4, 2, Article 15 (February 2023), 29 pages.
https://doi.org/10.1145/3579367

1 INTRODUCTION

Distributed architectures are envisioned as a long-term solution to provide practical applications of quantum computing [12, 22, 40, 88]. The general trend [31, 40, 41, 50, 66, 86] shows a common belief in distributed (and quasi-distributed, or multi-core) architectures as physical substrate, allowing a modular and horizontal scale-up of computing resources, rather than relying on vertical scale-up, coming from single hardware advancement. On the flip side, by linking distributed quantum processors, several new challenges arise [12, 15, 22, 30, 51, 78, 90]. Here we consider the *compilation problem*, which is generally tough to solve, even on a single processor, and for which an NP-hardness proof is available [10]. An ever growing literature arises with a variety of proposals for local computing [9, 11, 34, 45, 46, 49, 61, 67, 68, 70, 74, 83, 91, 93, 98] and for distributed computing [8, 23–25, 35, 39, 76, 80, 81, 96, 97].

Even if quantum processors are already available, distributed architectures are at an early stage and must be discussed from several perspectives. A key concept is that of *telegates* as the fundamental inter-processor operations [22, 86, 88]. Each telegate can be decomposed into several tasks, that we group as follows: (i) the generation and distribution of entangled states among different processors, (ii) local operations, and (iii) classical communications. Such tasks make the telegate an expensive resource, especially in terms of running time.[1] As a consequence, they have critical impact on the performance of the overall computation. In contrast to such a limit, telegates offer remarkable opportunities of parallelization. In fact, much circuit manipulation is possible to keep computation independent from telegate tasks. Therefore, we aim to model an optimization problem that embeds such opportunities.

1.1 Contribution

The overall objective of our work is to deeply analyse strategies to reduce the overhead caused by telegates, which are the main bottleneck in the computation on distributed architectures. Figure 1 gives a step by step overview of the paper, with particular attention to the problem modeling.

Sections 2 and 3 are devoted to detailing and justifying our **assumptions**. As a computation model we consider quantum circuits with a universal operator set. The set is based on local operations and on telegates as fundamental inter-processor operations. Here, we optimize telegates to efficiently scale with inter-processor connectivity restrictions.

We move on by defining rigorously the problem (Section 4). To come up with our **formulation** we rely on a wide literature from the Operation Research field, dealing with network scenarios. Specifically, we notice several analogies between our problem and those on dynamic networks, especially the group of *multi-commodity flow* problems [16–21, 36–38, 79, 84, 85]. The resulting formulation is particularly remarkable, as it is suitable for run-time minimization together with the minimization of resource usage, as a side objective. In an early step, the formulation is deliberately abstract, as it relies on binary relations that are not fully characterized at this stage. We believe that this enhances the modularity of the work and its readability. In fact, exploring the solution space requires to perform costly circuit manipulation, that deserves a dedicated discussion. Nevertheless, right after the abstract description of the problem structure, we proceed with

[1]Refer to [60, 94] for the state of the art on experimental implementations.

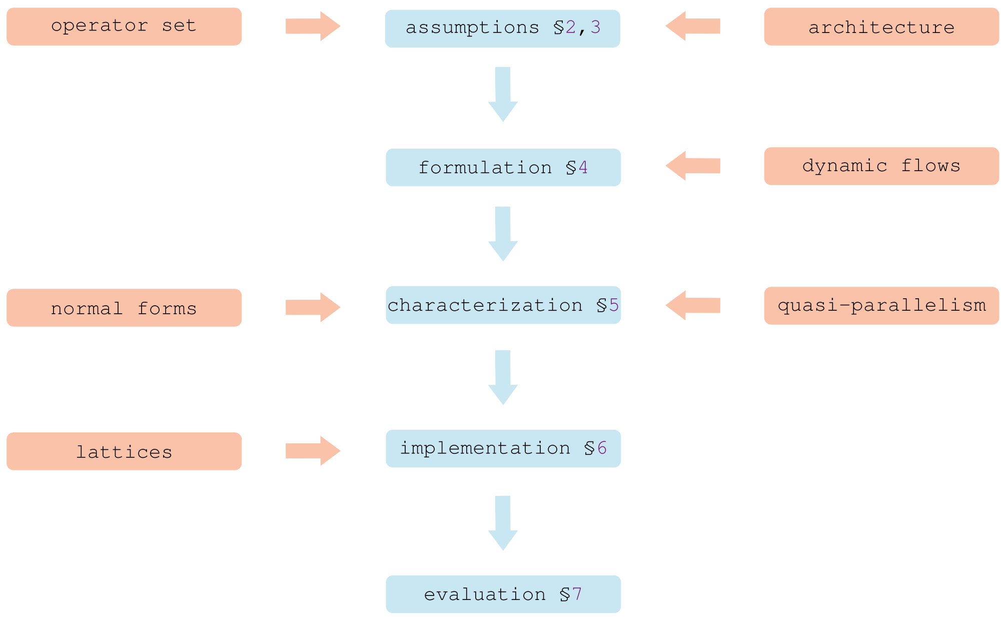


Fig. 1. Manuscript overview. Blue blocks denote the steps in the problem modeling, scanned by blue arrows. Red blocks are the main ingredients to the entry blue blocks.

the full **characterization** of the aforementioned binary relations (Sections 5.1 and 5.4). These relations define which circuit manipulations are feasible. At first, we use relations to model operations that can run in parallel, and in this context we introduce a relaxed version of parallelism, that we call *quasi-parallelism*. This relation is based on (automated) circuit manipulation which aims to gather telegates within the same time step. Section 5.1 contains a discussion on how to transform the graph, in order to adapt the model to the kind of circuit group one is tackling. After that, we relate all the operations to the partial order set induced by circuits expressed in normal forms – see Section 5.3.

We then describe our **implementation** (Section 6) and evaluate it by means of numerical **experiments** on different lattices (Section 7), showing that a square lattice gives rise to shallower circuits than a hexagon lattice, and that the compiler is able to process square lattices faster. We relate such a result to ratio between edges and nodes, which becomes an important index when choosing a topology for distributed quantum computation. Section 8 contains the summary of the findings and the conclusions.

2 DISTRIBUTED QUANTUM COMPUTING ESSENTIALS

In this section we describe the main elements, featuring a distributed quantum architecture.

One can encode a quantum processor as a set of qubits and a set of sparse tuneable couplings among qubits. If two qubits are coupled it means that they can interact. We will refer to such couplings as *local couplings*, to emphasize they belong to the same node in distributed architectures, as opposed to *entanglement links*, that are couplings between qubits in different processors. As detailed in next sub-section, two remote qubits coupled through an entanglement link cannot be used for computation: consequently, it is useful to classify qubits as either *computation qubits* or *communication qubits*, respectively.[2] While computation qubits process information during the

[2]A similar classification is available in References [12, 59].

computation, the communication qubits couple distinct processors through the entanglement. Figure 2 shows a toy architecture. The purple lines represent the couplings among distributed processors.

2.1 The Entanglement Link

To couple two processors, a communication protocol, known as *entanglement generation and distribution* [12, 13, 22], is necessary. We describe it here as three main steps:

(1) generating a two-qubits maximally entangled state[3];
(2) distributing the state between different processors[4];
(3) storing the partial states in the communication qubits.

When the protocol succeeds, the distributed qubits are correlated and can be exploited to perform non-local operations. For this reason we consider this correlation as a virtual link, which we refer to as *entanglement link*.[5] Entanglement links extend the possible interactions to any distributed computation qubits. Specifically, since the communication qubits are locally coupled with computation qubits, with entanglement links one can perform operations between remote computation qubits, referred to as *telegates*. More details on the functioning of telegates are reported in Section 3.2. However, it is important to keep in mind that, to perform a remote operation, one has to measure the states stored in the communication qubits. As a consequence, an entanglement link is a depletable resource, assigned to a single remote operation. After the measurement, a new round of entanglement generation and distribution takes place.

We now give a mathematical description of a distributed architecture, in order to formally describe the functioning of telegates.

2.2 Mathematical Description

So far, we presented the main elements occurring in a distributed quantum architecture, which we can now represent mathematically. Formally, let $\mathcal{N} = (V, P, F)$ be a network triple representing the architecture. $V = Q \cup C$ is a set of nodes describing qubits, therefore it is the disjoint union of computation qubits $Q = \{q_1, q_2, \dots, q_{|Q|}\}$ and communication qubits $C = \{c_1, c_2, \dots, c_{|C|}\}$. We can represent n processors by partitioning V into $P = \{P_1, P_2, \dots, P_n\}$. Therefore, a sub-set P_i characterizes a processor as its set of qubits/nodes.

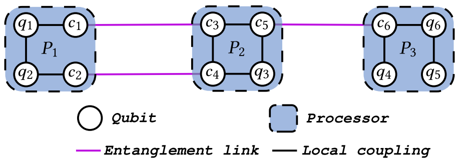


Fig. 2. Toy distributed quantum architecture with 3 processors.

$F = L \cup R$ is as a set of undirected edges. L represents the local couplings, therefore

$$L \subseteq \bigcup_i P_i \times P_i.$$

Notice that there is no particular assumption on connectivity nor cardinality within processors. This keeps the treating hardware-independent and it allows for heterogeneous architectures.

[3]The two-qubits assumption is general and can be extended to multi-qubits protocols.
[4]This step implies communication. The interested reader can find in Reference [13] three different protocols achieving the task.
[5]The interested reader can find a discussion about how to achieve practical entanglement generation and distribution, via heralded-based protocols, at Reference [59].

R represents entanglement links. Since entanglement links connect only communication qubits, we introduce, for each processor, a set of those qubits only; i.e., $C_i = C \cap P_i$. Therefore, we have

$$R \subseteq \bigcup_{i,j \,:\, i \neq j} C_i \times C_j.$$

Figure 2 shows an exemplary architecture, with three processors in P, six computation qubits in Q, six communication qubits in C, three entanglement links in R. and ten local couplings in L.

Concerning minimal assumptions, we only care about architectures actually able to perform any operation. This translated into a simple connection assumption.

3 OPERATORS

In the following, the gate model architecture of quantum computers is considered. There, a circuit describes a time-ordered quantum evolution as a sequence of quantum gates consisting of unitary operators. The set of available operators depends on the physical implementations.

3.1 Computation Operators

In order to achieve universal quantum computing, one may rely on a universal set of quantum logic gates capable to approximate any possible unitary operator. In the following, we consider a representative universal set of quantum gates, without loss of generality. A sufficient set for local universal quantum computing consists of the three operators $\{\mathtt{CX}, \mathtt{H}, \mathtt{T}\}$, where $\mathtt{CX}$ is the conditioned bit-flip operator, $\mathtt{H}$ is the Hadamard operator, and $\mathtt{T}$ is the $\frac{\pi}{4}$-phase shift. Indeed, with a polynomial number of repetitions of $\mathtt{H}$ and $\mathtt{T}$ one can approximate any unitary operator with arbitrary precision [54, 75]. Another sufficient set is also $\{\mathtt{CZ}, \mathtt{H}, \mathtt{T}\}$, where $\mathtt{CZ}$ is a conditioned phase-flip, thanks to the equivalence $\mathtt{CZ}_{u,v} \equiv \mathtt{H}_v \mathtt{CX}_{u,v} \mathtt{H}_v$.[6]

Nevertheless, for practical reasons that will be clear in Section 5.2, we find it convenient in the current paper to rely on the extended gate set $\{\mathtt{CX}, \mathtt{CZ}, \mathtt{H}, \mathtt{T}\}$.

Other choices of universal sets are possible, such as those based on trapped ions in a cavity [3], suitable for quantum interfaces where the photonic state is transferred to the cavity mode, and then to the electronic state of the ion via laser pulses [30, 86].

3.2 Universal Set

To extend the universality also to distributed architectures, we need at least one remote operator. Since in our gate set – $\{\mathtt{CX}, \mathtt{CZ}, \mathtt{H}, \mathtt{T}\}$ – one gate acting on two qubits (namely, $\mathtt{CX}$ or $\mathtt{CZ}$) is sufficient, then it is also enough to have one remote operator. In other words, w.l.o.g. we can show a protocol performing only a $\mathtt{CX}$ (or $\mathtt{CZ}$) between remote computation qubits. To represent such a protocol we use the notation $\mathtt{RCX}$ (or $\mathtt{RCZ}$). With the different nomenclature we highlight their physical difference. Specifically, while $\mathtt{CX}$ represents a local gate, $\mathtt{RCX}$ represents a sequence of operations that involves distant qubits. Therefore, in general, implementations of $\mathtt{CX}$ and $\mathtt{RCX}$ come with different fidelity, latency, and required resources.

Specifically to the $\mathtt{RCX}$ functioning, this is based on several fundamental steps, which we describe, in turn, by using operators.

The first operator models the entanglement link creation; we refer to that as $\mathtt{E}$ or, more explicitly, as $\mathtt{E}_{w,r}$. It sets qubits c_w and c_r to the maximally entangled state

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

[6]Here and throughout the paper, when an operator is subscripted, we are denoting the qubits it is operating on, e.g., $\mathtt{CX}_{u,v}$ is a $\mathtt{CX}$ operator with control qubit q_u and target qubit q_v.

The second operator models a measurement for a communication qubit c_w, over the computational basis. Namely, the measurements output a classical binary variable $b_w \in \{0, 1\}$. We refer to that as M_w and with a circuit component represented in Figure 3.

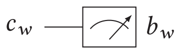


Fig. 3. Circuit component representing a measurement M_w.

Figure 4 shows a possible realization of a generic $\mathsf{RCX}_{u,v}$. Here, there are two qubits $q_u, c_w \in P_i$ and two qubits $q_v, c_r \in P_j$. Let us separate the protocol in three different steps. The first one is the creation of the entanglement link between c_w and c_r, i.e., applying $\mathsf{E}_{w,r}$. After that, the second step is the **pre-processing**: a few local operations occur and then qubits c_w, c_r are measured, getting b_w and b_r, respectively. The final step is the **post-processing**. The binary variables are used to assert whether further operations are required. Specifically, if $b_r = 1$, a Pauli Z operator applies to q_u and, if $b_w = 1$, a Pauli X operator applies to q_v. This phase can be compactly referred with the $\mathsf{Z}_u^{b_r}, \mathsf{X}_v^{b_w}$ operators. Notice that b_w is local to processor P_i and b_r is local to P_j. But P_i uses b_r and P_j uses b_w. In other words, a cross classical communication occurs between P_i and P_j.

Let us now give a look to some exemplary applications of $\mathsf{RCX}_{u,v}$ over the toy architecture of Figure 2.

Example 1. Assume one wants to run an RCX with control qubit q_2 and target q_3 – i.e., $\mathsf{RCX}_{2,3}$. Just run circuit in Figure 4, with $u = 2, v = 3, w = 2, r = 4$.

Example 2. Now assume one wants to run $\mathsf{RCX}_{1,3}$. In this case we can still use the entanglement link between c_2 and c_4. However, qubit q_1 is not coupled with c_2. To use that link we need to swap the states stored in q_1 and q_2 before and after running CX.

What happens if one wants to run, say, $\mathsf{RCX}_{1,4}$? In such a case, the qubits belong to processors having no entanglement link coupling them. There is a really efficient protocol to overcome this problem: it is called *entanglement swap* and we describe it within the next section.

3.3 The Entanglement Swap

As pointed out before, it might be the case where one wants to run an RCX operator between a couple of qubits belonging processors with no entanglement link. Formally, let P_i and P_j such processors and $R \cap (C_i \times C_j) = \emptyset$. In the basic scenario, there exists an intermediate processor P_k which has an entanglement link with both P_i and P_j, say via four communication qubits such that $c_u \in C_i$, $c_v, c_w \in C_k$ and $c_r \in C_j$. As Figure 5 shows, we exploit P_k to entangle c_u and c_r.

The entanglement swap protocol can be generalized to an arbitrary sequence of intermediate processors. To this aim we introduce the concept of *entanglement path*.

3.3.1 The Entanglement Path. Coherently with the standard definition of path of a graph, an entanglement path is a sequence of entanglement links connecting two processors. Formally, an entanglement path is a sequence $\{P_{i_1}, P_{i_2}, \ldots, P_{i_m}\}$ of m processors such that, for any j in $1 \leq j < m$, there is an entanglement link between P_{i_j} and $P_{i_{j+1}}$.

We can therefore entangle two communication qubits $c_u \in P_{i_1}$ and $c_r \in P_{i_m}$ by applying a generalization of the entanglement swap – showed in Appendix A – to $\{P_{i_1}, P_{i_2}, \ldots, P_{i_m}\}$.

Since at the end of the protocol c_u and c_r are in the entangled state $|\Phi^+\rangle$, an entanglement path is a generalization of an entanglement link.

3.3.2 RCX *with Entanglement Path.* In our scenario, the purpose of applying entanglement swap is to perform RCX. For this reason it is interesting to note that we can combine the entanglement swap protocol together with the protocol for RCX. The result is showed in Figure 6. This result

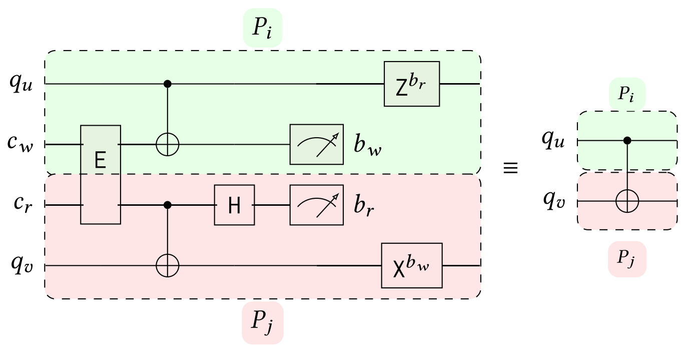


Fig. 4. Protocol performing an $\mathsf{RCX}_{u,v}$. From an operator point of view, this is equivalent to perform $\mathsf{CX}_{u,v}$. However, u and v belong to different processors and that is why we use notation RCX.

generalizes to every path, no matter the length – see Appendix A. We further discuss within next section the latency implications coming from this result.

4 DISTRIBUTED QUANTUM CIRCUIT COMPILATION PROBLEM

Usually, in the literature dealing with compiler design [35, 46, 91, 98], a circuit is encoded as a set of *layers*. Formally, a layer is a set ℓ of independent operators, meaning that each operator in ℓ acts on a different collection of qubits. A circuit is an enumeration of layers $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_{|\mathcal{L}|}\}$, where the cardinality is also commonly referred as circuit *depth*.

Notation	Description
$[n]$	An enumeration set $\{1, 2, \dots, n\}$
O	Font mainly used to denote operators
Δ_{O}	Time to run operator O
$\mathcal{Q}$	Quotient graph
$\mathcal{L}$	Circuit encoding
$\mathcal{L}_{\mathsf{O}}$	Circuit where only O operators occur
τ	Discrete time step
$\prec, \shortparallel$	Binary relations
$\mathcal{A}$	Predicate used to characterize $\shortparallel$
b	Boolean variable
f	Flow function
p_i	i-th quantum processor
$\mathbf{s}, \mathbf{t}$	sources and targets vector
d	Circuit depth

A quantum programmer writes a logical circuit, abstracting from the real architecture and assuming that qubits are fully connected, i.e., any couple of qubits can perform a CX operation directly. Such an abstraction holds also when stepping to distributed architectures.[7]

However, NISQ architectures do not provide full coupling. As a consequence, there must be a software interface – namely, a compiler – able to map an abstract circuit to an equivalent one, but meeting the real coupling. In general, such a mapping implies overhead in terms of circuit depth. Therefore, finding a mapping with minimum depth overhead is an optimization problem. We refer to it as the ***quantum circuit compilation* problem (QCC)**, which is proved to be NP-hard [10]. Its version on distributed architectures, which we refer to as the ***distributed quantum circuit compilation* problem (DQCC)**, is likely to be at least as hard as QCC. In fact, while in QCC we deal with local connectivity restrictions, in DQCC local connectivity stands alongside with remote

[7]Recall that, from a user perspective, CX ≡ RCX.

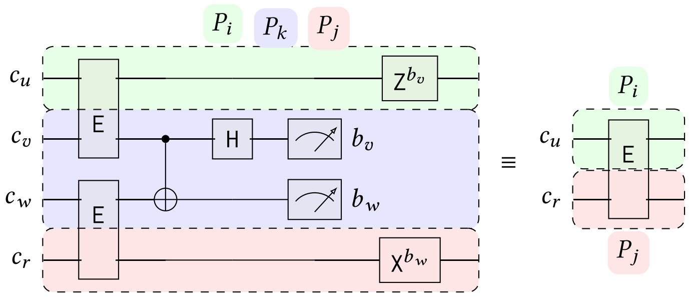


Fig. 5. Entanglement swap protocol. This scenario has three processors P_i, P_k, P_j. P_k has an entanglement link both with P_i and with P_j, created respectively by $\mathsf{E}_{u,v}$ and $\mathsf{E}_{w,r}$. At the end of the protocol c_u and c_r are in the maximal entangled state $|\Phi^+\rangle$. From an operator point of view, this is equivalent to performing $\mathsf{E}_{u,r}$.

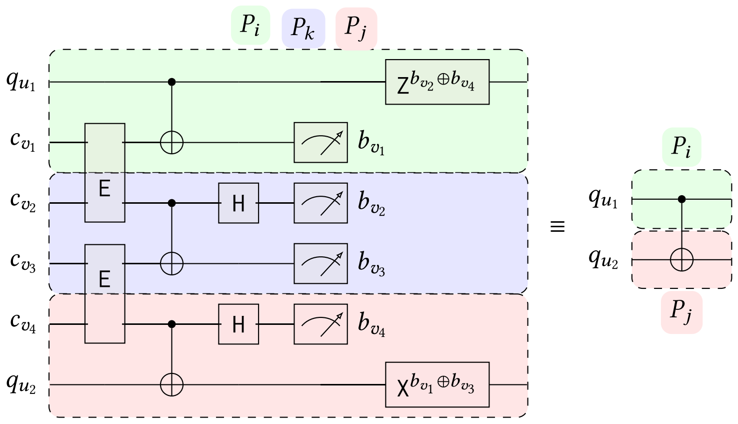


Fig. 6. RCX_{u_1,u_2} with entanglement swap.

connectivity – i.e., the entanglement links –, which is less dense than the local one.[8] Furthermore, performing a remote operation is much more time consuming than a local operation. Just consider that a remote operation relies on communication of both quantum and classical information.

The above reasons make telegates the bottleneck in distributed computing. Therefore, they are worthy of dedicated analysis to minimize their impact.

4.1 Objective Function

To optimize a circuit, the first thing we need to do is choose an objective function to rate the expected performance of a circuit. A common approach is to evaluate only those operators which are somehow a bottleneck to computation. Considering the gate set {CX, CZ, H, T}, in the context of fault-tolerant quantum computing [42], the bottleneck is the T operator [4, 82] since error correction protocols are designed for {H, CX}. Conversely, on current NISQ technologies, the bottleneck lies in the CX and CZ operators, that are more noisy as they operate on two qubits. The relevant metric can either be the number of occurrences of the subject operator O, namely the O-count, or the number of layers containing O at least once, namely the O-depth. To rate a compiled circuit on distributed architectures, we do something along the lines of this latter approach. Specifically,

[8]Because the more communication qubits there are, the less computing resources are available.

the bottleneck is the RCX and the RCZ operators, and each RCX or RCZ implies one occurrence of E. Therefore, we will rate a circuit by means of its E-depth.

As a simple example of E-depth, consider an instance of the problem: a logical circuit where some RCX operators occur. Figure 7 shows an exemplary one. Let us put in the worst-case scenario, i.e., all the four qubits belong[9] to different processors. Consequently, all the two-qubits operators are RCX. Without considering the tasks which RCX relies on, there is not much optimization to do and the E-depth is 5.

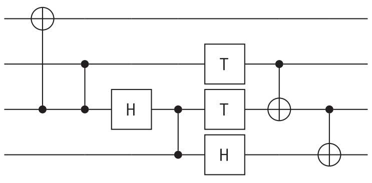


Fig. 7. Exemplary logical circuit, expressed in the universal gate set {CX, CZ, H, T}.

4.2 Modeling the Time Domain

It should be clear that E has central interest in our treating. In fact, we are also going to model the time by scanning it as E occurs. Specifically, notice that link generations among different couples of qubits are independent. For this reason we assume that all the possible links generate simultaneously and, as soon as all the states are measured, a new round of simultaneous generations begins.

Clearly, after that a measurement M generates a boolean b, there is at least one post-processing operator that needs to wait for that boolean to arrive. Generally speaking, the longer the path the more time b takes to reach its destination. We need to account for that by a proper model. To this aim, we do some observations.

Remark. Consider a generic single-qubit unitary operator U. The time required to perform U^b is largely dominated by the travel time of b, whilst the actual time taken by U can be neglected. Furthermore, the travel of b is independent from computation. Hence, we can compactly refer to the post-processing waiting-time as Δ_{U^b}. A second observation is that the travel of b is also independent by entanglement link creations, which we assume to take time Δ_{E}. It is also logical to assume $\Delta_{\mathsf{U}^b} \lesssim \Delta_{\mathsf{E}}$ for the following reasoning: even if b needs to cover a longer distance than the one covered by E, b relies on classical technologies, which are way more efficient[10] than entanglement generation and distribution protocols. For this reason, in our treating we neglect Δ_{U^b}, since it happens in parallel with Δ_{E}.

Stemming from this, we can model the time domain as a discrete set of steps $\tau \in \{1, 2, \ldots, d\}$, where d is an unknown time horizon, which is also the E-depth. At the beginning of each time step τ, the whole set of entanglement links is available for telegates. Notice that most of the local operators are expected to run during the creation of the links. Because we relate them to the following inequality

$$\Delta_{\mathsf{E}} \gg \Delta_{\mathsf{CX}}, \Delta_{\mathsf{CZ}}, \Delta_{\mathsf{H}}, \Delta_{\mathsf{T}}, \tag{1}$$

where, for a generic operator O, Δ_{O} is the time to run O. Therefore, since E is independent from local operators, we can always attempt to run these while E is running – and also while classical bits b are traveling, as explained in Section 3.3.2.

4.3 Modeling the Distributed Architecture

In light of the above observations, it is reasonable and convenient to consider the whole processor as a network node, and define a function c that provides the number of available links between two processors. Specifically, we first formalized a distributed architecture as the network graph

[9]Assigning logical qubits to physical ones – i.e., qubit mapping – is another critical step for compilation and it deserves dedicated analysis [5, 26], and is out of the scope of this work.
[10]The design of a distributed quantum architecture can easily adapt to satisfy requirements coming from assumptions on classical technologies, since these are very advanced.

$\mathcal{N} = (V, P, F)$ introduced in Subsection 2.2; this step was important to understand the interior behavior of remote operations from a qubit perspective. However, now it is useful to re-state it to a more compact encoding, which highlights the main bottleneck of a distributed quantum architecture, the entanglement links. Formally speaking, we will consider a *quotient graph* of $\mathcal{N}$.

To not further weigh down the formalism, we remodel the instance, by considering as main nodes, the processors, corresponding to an enumeration for the partition P, i.e., $P = \{p_1, p_2, \ldots, p_n\}$. All the entanglement links connecting the same couple of processors, now collapse to a single edge with integer capacity c, describing how many parallel entanglement links the two processors supply. We refer to this set of edges as

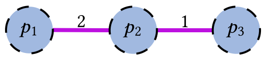


Fig. 8. Quotient graph derived from toy network of Figure 2. The processors become the nodes, the entanglement links between a couple of processors gather into one edge, with capacity equal to the number of original links.

$$E \subseteq \bigcup_{i,j \,:\, i \neq j} p_i \times p_j.$$

Hence, the new undirected graph is $\mathcal{Q} = (P, E, c)$. With this reformulation a remote operation will refer to a *control processor* and a *target processor* – i.e., $\mathsf{RCX}_{u,v}$ with $p_u, p_v \in P$.

In Figure 8 we show the quotient graph related to the toy architecture of Figure 2.

4.4 Single Layer Formulation

Consider a basic circuit expressed as the singleton $\mathcal{L} = \{\ell\}$. Assume that in ℓ there occur k RCX operators. From a logical perspective, all the k operators can run in parallel – by definition of layer. In other words, if the architecture connectivity had infinite capacity – i.e., $c(e) = \infty, \ \forall e \in E$ – we could run $\mathcal{L}$ with E-depth 1, that is optimal. As the capacity values decrease, the optimal E-depth value grows, up to E-depth k in the worst-case.

Let us formulate an optimization problem for the single-layer case – we will introduce a generalization to any circuit in Subsection 4.5. Specifically, the *quickest multi-commodity flow* [36] wraps this basic scenario.

In brief, the goal is to find a flow over time which satisfies the constraints imposed by a set of so-called commodities, which are going to represent the RCX of a quantum circuit. The less time the flow takes, the better. To formalize this problem one can directly model an objective function that evaluates a flow by the time it takes. This is an approach employed in Reference [63], but for single commodity. Alternatively, authors in Reference [36] propose to start from a formulation of the *multi-commodity flow* problem over time MCF_d, where d is a given *time horizon*,[11] namely a maximal number of time steps in which the flow is constrained. We prefer this latter way because dynamic flows like MCF_d has been deeply studied since long time ago [37, 38]. Furthermore, even if this approach has an important drawback, explained at the end of this sub-section, it does not apply to our scenario.

4.4.1 Commodities. To formulate MCF_d, first, we enumerate the occurrences of RCX in $\mathcal{L}$ as a set of commodities $[k] = \{1, 2, \ldots, k\}$. A set of couples source-sink nodes associates to the commodities. To do that, let $\mathbf{s} = (s_1, s_2, \ldots s_k)$ and $\mathbf{t} = (t_1, t_2, \ldots t_k)$ be two vectors induced by the operators RCX in $\mathcal{L}$[12] such that,

$$\mathsf{RCX}_{s_i, t_i} \in \ell \iff \exists i \in [k] \ : \ p_{s_i}, p_{t_i} \in P.$$

Namely, p_{s_i} (p_{t_i}) is the processor where the control (target) qubit of operation i occurs.

[11] The choice of using letter d should highlight that the time horizon is going to be the E-depth.
[12] We need to use vector notation to admit repetitions.

4.4.2 Decision Variables. The decision variables of the optimization problem are the time-dependent functions $f_{e,i}(\tau) \in \{0,1\}$, indicating the flow on edge $e \in E$ dedicated to operation $i \in [k]$ at time τ. The function has a binary co-domain because an operation i uses at most one entanglement link.

4.4.3 Constraints. As usual, the first constraint we introduce is the *flow conservation* constraint. Formally, $\forall i \in [k]$, $\forall \tau \in [d]$ and $\forall p_j \in P \setminus \{p_{s_i}, p_{t_i}\}$ the following holds:

$$\sum_{e \in \delta^-(p_j)} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_j)} f_{e,i}(\tau) = 0 \tag{2}$$

where $\delta^-, \delta^+ : P \longrightarrow F$ are the standard functions outputting the set of entering and exiting edges of the input node, respectively.

Since a flow $f_{e,i}(\tau) = 1$ identifies the usage of an entanglement link in e to perform i, we need to guarantee that the flow going through intermediate links of a path does not stop there. Conversely, whenever an end point of the path occurs in the control or target processor – i.e., p_{s_i} or p_{t_i} – , the *operation demand* – or *commodity demand* – constraint holds instead of the conservation constraint. Namely, $\forall i \in [k]$, this can be written as:

$$\sum_{e \in \delta^-(p_{s_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_{s_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) = -1 \tag{3}$$

$$\sum_{e \in \delta^-(p_{t_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_{t_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) = +1 \tag{4}$$

The above constraint explicitly requests that a flow dedicated to i reaches its target p_{t_i}, without exiting. Symmetrically, it leaves its control processor p_{s_i} without returning.

Notice that constraint (2) forces the operation demand to be satisfied within a single time-step.

The last constraint ensures that, at any time step, the number of operations does not exceed the entanglement resources. Hence, $\forall e \in E$ and $\forall \tau \in [d]$, we introduce a *capacity bound*:

$$\sum_{i \in [k]} f_{e,i}(\tau) \leq c(e) \tag{5}$$

Ultimately, the objective function is the total flow $f = \sum_{e \in E} \sum_{i \in [k]} \sum_{\tau} f_{e,i}(\tau)$.

By gathering the above equations, we obtain the Integer Linear Programming formulation (6), which models $\texttt{MCF}_d$. A flow f perfectly matches a set of entanglement paths used by the telegates.

$$\begin{aligned}
\text{minimize} \quad & f = \sum_{e \in E} \sum_{i \in [k]} \sum_{\tau \in [d]} f_{e,i}(\tau) & \\
\text{subject to} \quad & \sum_{e \in \delta^-(p_j)} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_j)} f_{e,i}(\tau) = 0 & \forall i \in [k], \forall \tau \in [d], \forall p_j \in P \setminus \{p_{s_i}, p_{t_i}\}, \\
& \sum_{e \in \delta^-(p_{s_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_{s_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) = -1 & \forall i \in [k], \\
& \sum_{e \in \delta^-(p_{t_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_{t_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) = +1 & \forall i \in [k], \\
& \sum_{i \in [k]} f_{e,i}(\tau) \leq c(e) & \forall e \in E, \forall \tau \in [d]
\end{aligned} \tag{6}$$

Notice that solutions with cycles are in general feasible, but are senseless in our scenario. By expressing the problem as a minimization of f, a solver will avoid any cycle and will try to use as few entanglement links as possible.

Once defined a solver for MCF_d, we just need to use it as proposed in Reference [36], namely the solver occurs as sub-routine within a binary research on the minimum time where a feasible solution exists. Since the research space is over time, the algorithm is, in general, pseudo-logarithmic. Specifically to our case, we already know that the worst solution is where all the operations run in sequence – i.e., E-depth equal to k. Therefore, the time horizon is upper-bounded by k and the binary search has $\log k$ calls to the sub-routine. Algorithm 1 shows the steps. Notice that the algorithm makes use of an undetermined solver for MCF_d. Since we are facing an NP-hard problem, this means that a real implementation would generally look for sub-optimal solutions.

ALGORITHM 1: Quickest multi-commodity flow

```
Input: $Q, [k]$
Output: $d$
1 $L \leftarrow 1, R \leftarrow k$
2 while $L \leq R$ do
3     $\bar{d} \leftarrow \lfloor \frac{L+R}{2} \rfloor$
4     $S \leftarrow \mathsf{MCF}_{\bar{d}}(Q, [k])$
5     if $S$ is feasible then
6         $d \leftarrow \bar{d}$
7         $R \leftarrow \bar{d} - 1$
8     else
9         $L \leftarrow \bar{d} + 1$
```

Unfortunately, standard MCF_d cannot catch the whole features of DQCC when $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_{|\mathcal{L}|}\}$; we need to consider that operations in $[k]$ are somehow related to each other by a logic determined by $\mathcal{L}$. Hence, in the following sub-section we are going to model such relations by introducing extra constraints.

4.5 Any Layer Formulation

As mentioned, the formulation we just gave is not enough to model the DQCC problem to any $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_{|\mathcal{L}|}\}$, because a circuit generally follows a logic which is related on the order of occurrence given by $\mathcal{L}$. Therefore, even if it might happen that two operations could run in any order, in general this is not true. One needs to define an order relation which is consistent with the logic of the circuit. From an optimization point of view, a critical matter is to choose an order relation that either wraps most of the good solutions or is prone to optimization algorithms. For this reason and for the sake of clarity, we here refer to a generic, irreflexive, order relation $\prec$ defined over $[k]$, without giving it a unique definition. Formally, for any $i, j \in [k]$, $j \prec i$ means that to run i we need to ensure that j already ran. Starting from $\prec$, we can define a constraint to add to formulation (6). Namely, $\forall i \in [k], \forall e \in \delta^-(p_{t_i})$ the following holds:

$$f_{e,i}(\tau) \leq \min_{j \prec i} \sum_{\bar{\tau} < \tau} f_{e,j}(\bar{\tau}) \tag{7}$$

The right part of the inequality is a value in $\{0, 1\}$ and takes value 1 only if all the operations logically preceding i already ran. Notice that constraint (7) is linear, as it takes the minimum value among linear functions, and it can be easily mapped to a set of independent constraints $f_{e,i}(\bar{\tau}) \leq \sum_{\bar{\tau} < \tau} f_{e,j}(\bar{\tau}), \forall j : j \prec i$.

The formulation now models DQCC. But, within next section, we refine inequality (7) to get a better solution space.

5 ENHANCING PARALLELISM

As before, from an optimization point of view, we are interested in considering as many good solutions as possible. To this aim, we propose an interesting approach which should enlarge the space of good solutions. Specifically, we notice that even if two operations $i, j \in [k]$ are such that $i \prec j$, this does not necessarily mean that they must run at different time steps. They, indeed, may run at the same time step and still respecting the logic imposed by $\prec$.

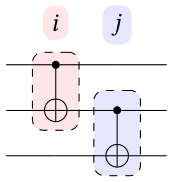


Fig. 9. RCX in logical conflict as both i and j operate on second qubit.

Consider the example from Figure 9. Since operations i and j operate over a common qubit, they are in logical conflict. Hence, it is reasonable to think that $i \prec j$ should hold. However, when considering i and j in their extended form – i.e., where communication qubits are explicit –, we notice that their logical conflict does not map over all the operations involved. As Figure 10 shows, the left part of the equivalence is a naive implementation of i followed by j, where the extended form completely inherits the logical conflict. Instead, the right part of the equivalence is way more efficient and it is still an implementation of circuit of Figure 9. As consequence, even if i and j are in logical conflict, they can run at the same time step. We refer to this property as *quasi-parallelism.* For this reason we introduce a new binary relation between operations in $[k]$, which we refer to with the intuitive symbol $\shortparallel$. As before, we do not give here a unique definition of $\shortparallel$. Specifically, for any $i, j \in [k]$, we write $i \shortparallel j$ to mean that operations i and j can run at the same time step, but we did not fix a criterion to establish when $\shortparallel$ holds. Clearly, operations $i, j \in [k]$ which can run in full parallelism, are a special case of quasi-parallelism and $i \shortparallel j$ holds. We can now split the constraint (7), by discriminating between operations which can run in quasi-parallelism and the ones which cannot. Formally, $\forall i \in [k], \forall e \in \delta^-(p_{t_i})$ we introduce two new constraints

$$f_{e,i}(\tau) \leq \min_{j \prec i \wedge j \nshortparallel i} \sum_{\bar{\tau} < \tau} f_{e,j}(\bar{\tau}) \tag{8}$$

$$f_{e,i}(\tau) \leq \min_{j \prec i \wedge j \shortparallel i} \sum_{\bar{\tau} \leq \tau} f_{e,j}(\bar{\tau}) \tag{9}$$

To sum up, we propose (10) as Integer Linear Programming formulation of the DQCC problem. C is the set of constraints coming from the standard MCF formulation given in (6). In what follows we propose a characterization for relation $\shortparallel$.

$$\begin{aligned}
\text{minimize} \quad & f = \sum_{e \in E} \sum_{i \in [k]} \sum_{\tau \in [d]} f_{e,i}(\tau) \\
\text{subject to} \quad & C, \\
& f_{e,i}(\tau) \leq \min_{j \prec i \wedge j \nshortparallel i} \sum_{\bar{\tau} < \tau} f_{e,j}(\bar{\tau}) \quad \forall i \in [k], \forall e \in \delta^-(p_{t_i}), \forall \tau \in [d], \\
& f_{e,i}(\tau) \leq \min_{j \prec i \wedge j \shortparallel i} \sum_{\bar{\tau} \leq \tau} f_{e,j}(\bar{\tau}) \quad \forall i \in [k], \forall e \in \delta^-(p_{t_i}), \forall \tau \in [d]
\end{aligned} \tag{10}$$

5.1 Characterization

Our goal is to model $\shortparallel$ to catch as many solutions as possible, while keeping them feasible to the hardware. With this in mind, we propose the following criterion: given any i, j, $i \shortparallel j$ holds whenever i and j can run within a certain "small enough" time lapse. Specifically, the time lapse depends on the coherence time of communication qubits, which are assumed to be much more affected by noise than computing qubits.

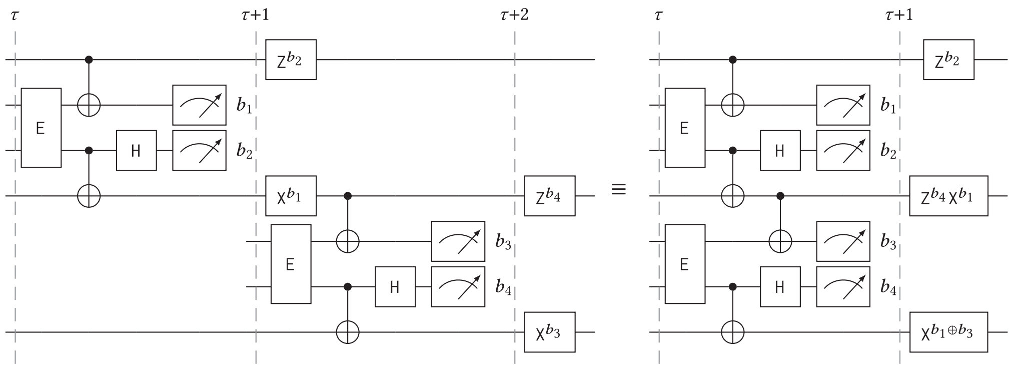


Fig. 10. Example of how to achieve quasi-parallelism for two RCX in logical conflict.

Notice that, when two operations i, j run in quasi-parallelism, the life-time of the employed communication qubits might grow. Therefore, we need to ensure that it does not exceed the coherence time of the entanglement. Formally, let us assume Δ_Φ being the coherence time of the entanglement – hence, it starts from the moment E ends, up to the beginning of the measurements M.

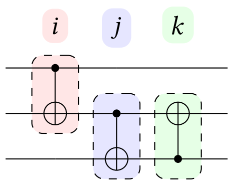


Fig. 11. Three RCX operators in logical conflict.

A complication arises from the fact that $\shortparallel$ is, in general, an *intransitive* relation. To understand why this is true, consider the circuit in Figure 11. In such a scenario we are faced with multiple choices. Namely, running

(1) all i, j, k at different time steps;
(2) all i, j, k at the same time step;
(3) i, j together and k afterwards;
(4) i only, followed by j, k together.

Case (1) is not of interest, because it is the worst solution and no optimization applies. Case (2) is the best solution, but it is not necessarily feasible. In fact, for Δ_Φ small enough, we are forced to split the operations, as in one of the cases (3) and (4). This explains the non-transitivity, since $i \shortparallel j$ and $j \shortparallel k$, but $i \nshortparallel k$.

We still need to characterize $\shortparallel$, hence, we introduce a predicate method which aims to bring RCX closer to each other, so that quasi-parallelism is achievable.

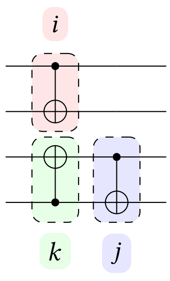


Fig. 12. Two independent RCX – i.e., i and j – belonging to different layers.

5.2 A Recursive Predicate for the Quasi-parallelism Relation

As said above, we are now going to introduce a method which verifies if any two telegates can run in quasi-parallelism. Therefore, this method, say $\mathcal{A}(i, j, \Delta_\Phi)$, is a predicate, which is true whenever the operations in input can run in quasi-parallelism. We can finally characterize $\shortparallel$:

$$i \shortparallel j \iff \mathcal{A}(i, j, \Delta_\Phi).$$

$\mathcal{A}$ works in a recursive fashion with three different scenarios as the base case.

Base case (i): given two operations i, j, if they belong to the same layer, clearly they can run in full parallelism, therefore $\mathcal{A}(i, j, \Delta_\Phi)$ is true.

Base case (ii): similarly to (i), if i, j belong to different layers and they are completely independent,[13] $\mathcal{A}(i, j, \Delta_\Phi)$ is true. Circuit of Figure 12 gives an example with i, j in contiguous layers.

Base case (iii): assume i, j contiguous – i.e., in contiguous layers – and both operating on, at least, one common qubit. We want to introduce, with this base case, the possibility that multiple operators may run simultaneously, as exemplified in Figure 10. For this reason, algorithm $\mathcal{A}$ considers all the operators involved to perform an RCX – recall protocol from Figure 4. Namely, $\mathcal{A}$ pushes forward the post-processing of i – i.e., the Pauli operations $\mathtt{Z}^b$ or $\mathtt{X}^b$ – after the pre-processing of j – i.e., the CX operations. One can do that by using the following transformation rules:

- $\mathtt{CX}(\mathtt{X}^b \otimes \mathtt{I}) \equiv (\mathtt{X}^b \otimes \mathtt{X}^b)\mathtt{CX}$
- $\mathtt{CX}(\mathtt{I} \otimes \mathtt{Z}^b) \equiv (\mathtt{Z}^b \otimes \mathtt{Z}^b)\mathtt{CX}$
- $\mathtt{CX}(\mathtt{I} \otimes \mathtt{X}^b) \equiv (\mathtt{I} \otimes \mathtt{X}^b)\mathtt{CX}$
- $\mathtt{CX}(\mathtt{Z}^b \otimes \mathtt{I}) \equiv (\mathtt{Z}^b \otimes \mathtt{I})\mathtt{CX}$

Similarly, when a CZ occurs, the following rules apply:

- $\mathtt{CZ}(\mathtt{X}^b \otimes \mathtt{I}) \equiv (\mathtt{X}^b \otimes \mathtt{Z}^b)\mathtt{CZ}$
- $\mathtt{CZ}(\mathtt{Z}^b \otimes \mathtt{I}) \equiv (\mathtt{Z}^b \otimes \mathtt{I})\mathtt{CZ}$

After the application of these rules, some post-processing operation, might have been *propagated* also to communication qubits. Specifically, it may happen that an operation $\mathtt{X}^b$ should precede a measurement. However, one can always reduce the depth of the circuit by sending b to the target(s) of the measurement. This is indeed what happens in our first example – Figure 10 –, where, instead of performing $\mathtt{X}^{b_1}$ in the communication qubit, we opt to put it in combination with $\mathtt{X}^{b_3}$, achieving a single operation $\mathtt{X}^{b_1 \oplus b_3}$ – see also Figure 13 for a circuit representation. At the end of the circuit manipulation, the life-time of the communication qubits may have risen. If it does not exceed Δ_Φ, then $\mathcal{A}(i, j, \Delta_\Phi)$ is true; otherwise, $\mathcal{A}(i, j, \Delta_\Phi)$ is false.

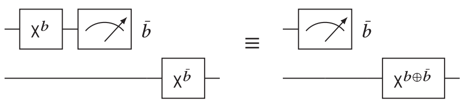


Fig. 13. Propagation of $\mathtt{X}^b$. First wire no longer needs information of b. Second wire needs information given by $b \oplus \bar{b}$. Notice that measured $\bar{b}$ is not the same value in the two cases.

Recursion: consider now the case where i and j are separated by a sequence of local operations $\mathtt{O}_1, \ldots, \mathtt{O}_n$,[14] assumed to be confined to the universal set $\{\mathtt{CX}, \mathtt{CZ}, \mathtt{H}, \mathtt{T}\}$. In this case, $\mathcal{A}$ applies, recursively, transformations for both i and j. Specifically, as long as possible, it *pushes forward* the post-processing of i by using former rules together with:

- $\mathtt{T}\mathtt{Z}^b \equiv \mathtt{Z}^b\mathtt{T}$
- $\mathtt{H}\mathtt{X}^b \equiv \mathtt{Z}^b\mathtt{H}$

Ultimately, as long as possible, $\mathcal{A}$ *pushes backward* the pre-processing of j by using the following standard rules:

- $\mathtt{CX}(\mathtt{T} \otimes \mathtt{I}) \equiv (\mathtt{T} \otimes \mathtt{I})\mathtt{CX}$
- $\mathtt{CZ}(\mathtt{T} \otimes \mathtt{I}) \equiv (\mathtt{T} \otimes \mathtt{I})\mathtt{CZ}$
- $\mathtt{CZ}(\mathtt{I} \otimes \mathtt{T}) \equiv (\mathtt{I} \otimes \mathtt{T})\mathtt{CZ}$
- $(\mathtt{CX}_{u,v} \otimes \mathtt{I})(\mathtt{I} \otimes \mathtt{CX}_{w,v}) \equiv (\mathtt{I} \otimes \mathtt{CX}_{w,v})(\mathtt{CX}_{u,v} \otimes \mathtt{I})$
- $(\mathtt{CX}_{u,v} \otimes \mathtt{I})(\mathtt{I} \otimes \mathtt{CX}_{u,w}) \equiv (\mathtt{I} \otimes \mathtt{CX}_{u,w})(\mathtt{CX}_{u,v} \otimes \mathtt{I})$

[13]Namely, what i does to its qubits does not affect the qubits j operates on.
[14]Namely, operations $\mathtt{O}_1, \ldots, \mathtt{O}_n$ belong to layers between the ones of i and j.

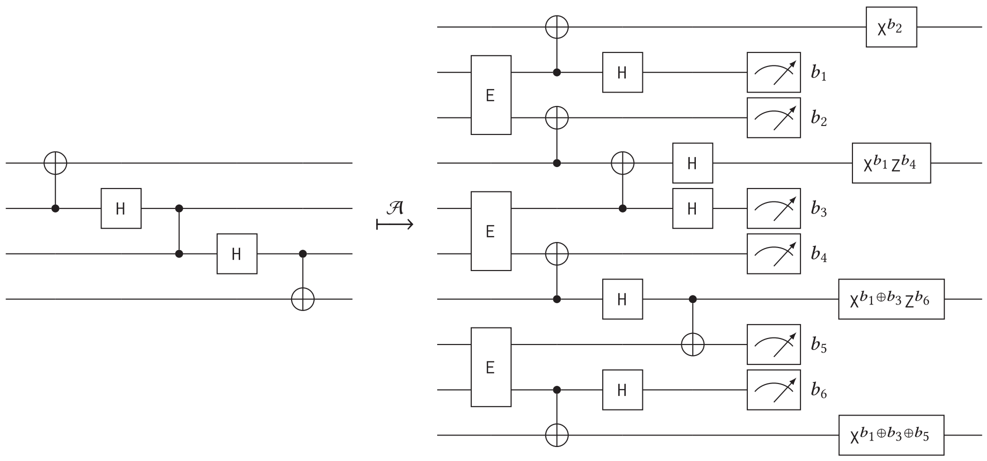


Fig. 14. An expansion, obtained by applying rules from $\mathcal{A}$. In this example scenario, RCX and RCZ are interspersed with single-qubit local operators. Notice that boolean variables travel simultaneously. Hence, the assumption we made in Section 4.2 – i.e., $\Delta_{U^b} \lesssim \Delta_E$ – holds also for complex evaluations as $Z^{b_1 \oplus b_4}$ and $X^{b_6}Z^{b_3}$.

- $CX_{u,v}(H \otimes H) \equiv (H \otimes H)CX_{v,u}$
- $CZ(I \otimes H) \equiv (I \otimes H)CX$
- $CX(I \otimes H) \equiv (I \otimes H)CZ$

If $\mathcal{A}$ manages to make i post-processing and j pre-processing contiguous, the validity check reduces to the base case scenario. Otherwise, $\mathcal{A}(i, j, \Delta_\Phi)$ is false.

So far, we defined $\mathcal{A}$ only for i, j without any other remote operation in between. Before generalizing the method to any i and j we prove that our definition of $\mathcal{A}$ can be implemented so that it runs in polynomial time. We need this requirement to keep things tractable.

Theorem 1. *$\mathcal{A}$ has $O(n)$ complexity, with n being the number of operations $\mathcal{A}$ considers.*

Proof. Assume there occur n local operations, say $O_1, \dots, O_n$, between i and j. If $\mathcal{A}$ manages to push i forward O_1, it means that its post-processing run after O_1 and it may only propagate *vertically*, over different qubits – by construction of the rule set. As consequence, the depth of the circuit has not increased. Furthermore, the post-processing is still composed by Pauli operations of the kind Z^b or X^b. Hence, this holds for any $O_{1 \leq \bar{n} \leq n}$ and the recursion is upper-bounded by $O(n)$.

Symmetrically, if $\mathcal{A}$ manages to push j backward O_n, it means that the pre-processing can run before O_n. Also in this case, the depth has not increased and the pre-processing is still composed by two independent CX operations – again, by construction of the rule set. Hence, this holds for any $O_{1 \leq \bar{n} \leq n}$ and the recursion is upper-bounded by $O(n)$. □

We can now move on to the general case. Formally, between i and j a remote operation k may occur, which is also in logical conflict with both. For such a scenario, we just add a recursive rule. Namely, $\mathcal{A}(i, j, \Delta_\Phi)$ holds iff the following holds:

$$\exists \varepsilon \in [0, 1] \ : \ \mathcal{A}(i, k, \varepsilon \cdot \Delta_\Phi) \wedge \mathcal{A}(k, j, (1 - \varepsilon) \cdot \Delta_\Phi).$$

Take a moment to appreciate why this kind of recursion is feasible. Specifically, one might think that validity of $\mathcal{A}(i, k, \varepsilon \cdot \Delta_\Phi)$ and $\mathcal{A}(k, j, (1 - \varepsilon) \cdot \Delta_\Phi)$ are not independent, because they both operate on k. However, in the former function, $\mathcal{A}$ evaluates the pre-processing of k, while, in the latter, it evaluates its post-processing. Therefore they can be evaluated independently.

THEOREM 2. *Generalized $\mathcal{A}$ has $O(m)$ complexity, with m being the number of operations $\mathcal{A}$ considers.*

PROOF. Assume there occur $k_1, \ldots, k_m$ between i and j. For the purpose of the proof let m be a power of 2. $\mathcal{A}(i, j, \Delta_\Phi)$ can choose any of the $k_1, \ldots, k_m$ operations for the recursion. To keep symmetry, let $\mathcal{A}(i, k_{\frac{m}{2}}, \varepsilon \cdot \Delta_\Phi)$ and $\mathcal{A}(k_{\frac{m}{2}}, j, (1-\varepsilon) \cdot \Delta_\Phi)$ be the recursive call. Notice that operations considered by $\mathcal{A}(i, k_{\frac{m}{2}}, \varepsilon \cdot \Delta_\Phi)$ are $\frac{m}{2}$, as well as the ones considered by $\mathcal{A}(k_{\frac{m}{2}}, j, (1-\varepsilon) \cdot \Delta_\Phi)$. The result is a recursive binary tree of height $\log m$ and, therefore, $O(m)$ calls to $\mathcal{A}$. The leaves correspond to the base case of the recursion, which is proved to be tractable in Theorem 1. □

Figure 14 shows an example scenario where we used rules as in $\mathcal{A}$ – in addition to the first one of Figure 10. Clearly, our modular architecture is prone to modifications or extensions of $\mathcal{A}$, if future research highlighted more refined requirements.

Remark. Notice that we managed to define $\mathcal{A}$ to be independent by the connectivity of Q. This was possible thanks to the way we modeled telegates via efficient entanglement paths – see Appendix A. In other words, $\mathcal{A}(i, j, \Delta_\Phi)$ works for any solver and regardless of the path this chooses to perform i and j. As consequence, the characterization of $\mathcal{A}$ – and therefore also of $\shortparallel$ – is static and depends only by the logical circuit and global factors, i.e., Δ_Φ. Furthermore, we may relate coherence time and entanglement link creation to $\Delta_\mathsf{E} + \Delta_\Phi \approx \Delta_\mathsf{E}$. As consequence, whatever Δ_Φ is, $\mathcal{A}$ does not significantly affect the duration of each time step. This makes the E-depth a particularly good index for the running time of the overall computation.

5.3 The Role of the Clifford Group in Distributed Quantum Computing

In our algorithm, we tried to postpone the post-processing as much as possible, to allow classical information to travel across remote computers in the meantime. An ideal result would be to push it to the end of the circuit: indeed, since the post-processing is made only of Pauli gates, if it were located at the end of the circuit, it could be efficiently replaced by a classical computation, removing the need of the quantum state to remain coherent while the information travels. We show in the next subsection that pushing the post-processing to the end is possible if the circuit belongs to a particular class, namely the *Clifford* group, generated by the operators $\{\mathsf{CX}, \mathsf{CZ}, \mathsf{H}, \mathsf{T}^2\}$ (or by the minimal sets $\{\mathsf{CX}, \mathsf{H}, \mathsf{T}^2\}$ and $\{\mathsf{CZ}, \mathsf{H}, \mathsf{T}^2\}$). Let us introduce here some basic facts about such a group.

The interest in the Clifford group derives from the fact that it covers a wide spectrum of circuits, but does not include the complexities of the T gate. The Clifford group can also be efficiently simulated on a classical computer. We already discussed that the T gate represents the most error-prone gate in the fault-tolerant context. On the other hand, it is obvious that the Clifford group together with the T gate is universal [75]. For this reason, it makes sense to represent an arbitrary circuit in terms of a Clifford circuit plus as little T gates as possible. This was attempted in literature in two ways:

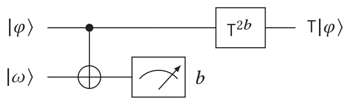


Fig. 15. Example of T gate *injection*.

- decompose circuits, with the goal of minimizing the number of T gate occurrences [4, 82];
- *inject* T gates into a Clifford circuit, by means of state preparation [62, 92, 95].

A basic example of T *injection* is shown in Figure 15, where injection is performed through one auxiliary qubit, prepared in the state

$$|\omega\rangle = \mathsf{TH}\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{i\pi}{4}}\,|1\rangle). \tag{11}$$

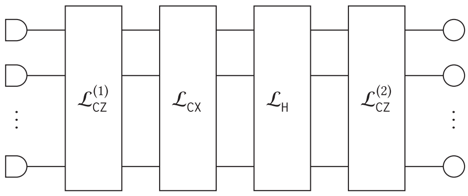

$\mathcal{L}^{(1)}_{\mathtt{CZ}}$
$\mathcal{L}_{\mathtt{CX}}$
$\mathcal{L}_{\mathtt{H}}$
$\mathcal{L}^{(2)}_{\mathtt{CZ}}$

Fig. 16. Normal form coming from the ZX-rules applied in Reference [29].

Other facts about the Clifford group are worth being reported. Specifically, distributed architectures based on trapped ions [50, 73, 86] are well fitted to work with state injection on Clifford circuits. Indeed, experimental results show that single-qubit gates can run with 99.9999% fidelity [43] and that CX (or CZ) operators, can achieve a 99.9% fidelity [7]. Furthermore the local connectivity for such a processor is complete [64]. This means that a T injection would give a fidelity of $\sim$ 99.8997%, if prepared as in Equation (11) and circuit of Figure 15, without the need of distillation nor of local routing. As a consequence, future architectures relying on entanglement generation and distribution, are likely to supply some T injection module too.

5.4 Circuit Normal Forms for the Clifford Group and Implications on the Post-processing

As said at the beginning of Section 5.3, important benefits could be achieved by postponing the post-processing to the end of the circuit, where they can be computed classically. An attempt in this direction is available in Reference [65], where authors delay Pauli operations together with non-Pauli ones. Instead, our approach is to show that the result can always be achieved on the Clifford group, by relying on the **normal forms** [1, 27, 29, 71]. Such a form results particularly useful for distributed computing and, more in general, for *measurement-based* computation.

It was shown [29] that any Clifford gate acting on a Pauli state, can be represented in the normal form depicted in Figure 16. This normal form is of practical interest as it can be obtained starting from any Clifford circuit, which is in general not in normal form. Such a result comes from the employment of a ZX-*calculus* reasoner – e.g., [53]. ZX-calculus [29, 87] is a graphical language arisen as an optimizer for quantum circuits, that translates a quantum circuit into a ZX-*diagram*. The main difference between the diagram and the original circuit is that the former works with ZX-*rules*, which serve as a reasoning tool to smartly generate a new circuit, equivalent to the original one. ZX-calculus was recently introduced in the literature, with the main objective of minimizing a circuit gate-depth, and its potentiality is still being explored, raising increasing interest for its versatility. In fact, we use it here to perform architecture-compliant optimization.

Let us describe the few tools and properties we need to benchmark our compiler, while the interested reader can refer to the bibliography for a more extended dissertation. Coming back to Figure 16, we use the circuit symbol to express a generic Pauli state preparation. Similarly, the symbol expresses a generic Pauli measurement. $\mathcal{L}_{\mathtt{O}}$ is a set of layers where only the O operator occurs. For example $\mathcal{L}_{\mathtt{CZ}}$ encodes a circuit composed by CZ operators.[15]

The following remark is a consequence of dealing with Clifford circuits in normal form.

Remark. While predicate $\mathcal{A}$ is running, only Pauli and Hadamard operations concur to its evaluation. Hence, all the post-processing operations can be pushed forward, up to end of the circuit

[15]Notice that $\mathtt{CZ}_{u,v} \equiv \mathtt{H}_v \mathtt{CX}_{u,v} \mathtt{H}_v$. Thus, we do not need to expand our assumptions on the gate set.

and can be computed efficiently by a classical computer. Furthermore, since no post-processing occurs during quantum computation, the entanglement path length has a negligible impact.

The normal form suggests that the problem can be separated into three parts, corresponding to $\mathcal{L}_{\mathtt{CZ}}^{(1)}$, $\mathcal{L}_{\mathtt{CX}}$ and $\mathcal{L}_{\mathtt{CZ}}^{(2)}$. For two of them – i.e., $\mathcal{L}_{\mathtt{CZ}}^{(1)}$ and $\mathcal{L}_{\mathtt{CZ}}^{(2)}$ – the order relation is trivial (as all CZ commute), and therefore we can use any quickest multi-commodity flow solver to get a feasible compilation. On the contrary, the optimal characterization of the order relation for $\mathcal{L}_{\mathtt{CX}}$ is a conceptually complex task. Indeed, a set of relations with minimal size may not be the best characterization from a practical point of view, if many of the relations involve remote qubits. The topic of optimal CX order relations deserves a dedicated analysis and is the subject of future work.

Let us emphasize the importance of $\mathcal{L}_{\mathtt{CZ}}$ circuits, by pointing out some facts from Reference [71]. The authors therein introduce the *Boolean degrees of freedom* as a way to count how many different algorithms can be implemented with a class of gates, and show that a generic $\mathcal{L}_{\mathtt{CZ}}$ "has roughly half the number of the degrees of freedom" compared to a generic $\mathcal{L}_{\mathtt{CX}}$, and roughly a quarter compared to the Clifford group. We validate our compiler performance by solving $\mathcal{L}_{\mathtt{CZ}}$ circuits on different architectures in Section 7. So, being able to exploit normal forms to isolate two highly expressive blocks $\mathcal{L}_{\mathtt{CZ}}^{(1)}$ and $\mathcal{L}_{\mathtt{CZ}}^{(2)}$ that can be compiled without recurring to order relations, is a very relevant result.

Before discussing the implementation details, let us make a final remark on ZX-calculus. We introduced it in the context of the Clifford group, but it is designed to work more broadly with any circuit [6, 14, 47, 52]. Therefore, we aim to expand our analysis in future works, by investigating normal forms for universal circuits. An interesting result in this sense is available in Reference [44], where authors show that a universal circuit can be split into three steps:

(1) the system is prepared in a *non-Clifford state*, this involves auxiliary qubits which will do the work of injecting non-Clifford phases – e.g., the T gate;
(2) an $\mathcal{L}_{\mathtt{CX}}$ circuit;
(3) a measurement-based sequence of Clifford operations (which can still be treated with ZX-calculus [28]).

6 IMPLEMENTATION TECHNIQUES

6.1 Time-expansion

Formulation (10) is a particular case of $\mathtt{MCF}_d$, as it slightly recedes from the standard formulation. As expected, the problem is still intractable. To understand that, consider this simple scenario: an instance $[k]$ with $k = 2$ such that $1 \shortparallel 2$. We can restate the problem as follows: assert if there exists a solution at first time step. If not, just put operation 2 at second time step. Unfortunately, asserting if such a solution exists is NP-hard. Indeed, in Reference [32], authors proved the hardness of such a decision problem, even for single capacity edges. Therefore, it is reasonable to look for approximations of DQCC. To this aim, we think a good line of research would be to follow a common technique for tackling $\mathtt{MCF}_d$: the *time-expansion* [38]. Namely, a re-definition of the instance graph, from Q to a new graph Q_d. Such a technique is useful because, instead of tackling $\mathtt{MCF}_d$ over Q, one can tackle its static version MCF over Q_d. Let us introduce it formally for our scenario.

A time-expansion of Q is a graph $Q_d = (P_d, E_d)$. Accordingly to this criterion, an edge $(p_i, p_j) \in E$ taking discrete travel time θ would translate into directed edges $(p_i(\tau), p_j(\tau + \theta)), (p_j(\tau), p_i(\tau + \theta)) \in E_d$, with a shared constraint on the capacity. Nevertheless, edges in Q are assumed to have null travel time. Hence, a time-expansion of Q is particularly efficient, since one just needs to introduce a repetition of Q for each time-step τ, which we refer to as $Q(\tau) = (P(\tau), E(\tau))$. As consequence, time-dependent sets $\mathbf{s}(\tau)$ and $\mathbf{t}(\tau)$ replace $\mathbf{s}$ and $\mathbf{t}$. We keep using $\mathbf{s}$ and $\mathbf{t}$ as the nodes encoding the commodities, non-localized in time. For each i and τ, we introduce edges $(p_{s_i}, p_{s_i}(\tau))$ and $(p_{t_i}(\tau), p_{t_i})$, both with unit capacity.

Since only integral flow are allowed and the demand is exactly 1, for any operation i, only one of the edges $\{(p_{s_i}, p_{s_i}(\tau))\}_\tau$ – as well as only one in $\{(p_{t_i}(\tau), p_{t_i})\}_\tau$ – will have a non-zero flow.

Now that we gave a first intuitive way to encode the sources of the problem, let us optimize it. Notice that operation 1 can always run at time 1, and it is a waste of time and space considering other options. As consequence, for operation 1, we only introduce $(p_{s_i}, p_{s_i}(1))$ and $(p_{t_i}(1), p_{t_i})$. This extends to any operation, which can always run in a time between 1 and $\min\{i, d\}$, by assuming that a solution exists with time horizon d. Therefore, for each operation i, we introduce the sets of edges $\{(p_{s_i}, p_{s_i}(\tau)) \ : \ \forall 1 \le \tau \le \min\{i, d\}\}$ and $\{(p_{t_i}(\tau), p^T_{t_i}) \ : \ \forall 1 \le \tau \le \min\{i, d\}\}$. Figure 17 shows the final graph for instance $[k]$ with $k = 3$, time horizon $d = 2$ on an architecture with 4 processors.

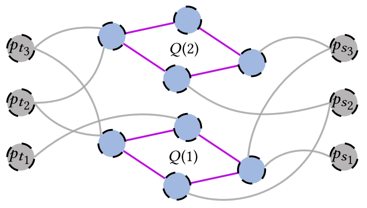


Fig. 17. Time-expanded graph of 4 processors, for an instance $[k]$ with $k = 3$ and time horizon $d = 2$.

As said, the time expansion Q_d is a common way to tackle $\texttt{MCF}_d$ as a static flow problem and it is particularly efficient in our scenario. Specifically, we could model Q_d by simply introducing d repetitions of Q and, especially, without the need of edges connecting different time-steps $Q(\tau)$, $Q(\bar{\tau})$. Because of this result, we are also able to implement a time-expansion at a logical level, without actually allocating space for d repetition of Q. This is detailed in Section 6.3.

To the best of our knowledge, even if approximation algorithms for MCF [20, 85] and variants [16, 18, 19, 79, 84] have been extensively studied, there seems to be no proposal relatable to ours, modeling DQCC. More formally, no efficient reduction seems possible from our problem to standard formulations, while approximation algorithms proposed in literature usually rely on LP-relaxation, or on greedy criteria. Theses proposals do not guarantee that constraints (8) and (9) are satisfied. Hence, further studies along this line would be useful to (i) place the problem within its most proper complexity class and to (ii) guarantee approximation ratio.

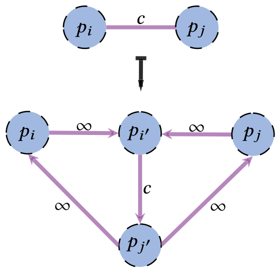


Fig. 18. Mapping from an undirected graph to a directed one working for any multi-commodity flow problem. The transformation undergoes with a constant overhead in the number of nodes and edges.

6.2 Transformation to Direct Graph

Since the literature dealing with MCF usually assume a directed graph, we here report a mapping method from an undirected graph to an equivalent one with direct edges. This would bring just a constant overhead in the space, while it would not affect any approximation factor which a solver would rely on. Figure 18 comes from [2]. It is a fast approach to map an undirected multi-commodity flow problem to a directed one. Specifically, for each couple of nodes p_i, p_j connected by an edge with capacity c, one has to introduce two extra nodes, say $p_{i'}, p_{j'}$ and connect them with the direct edge $(p_{i'}, p_{j'})$ of capacity c. The last step is creating directed cycles of infinite capacity, where the only bottleneck is c.

6.3 Compilation Through Approximation

We already discussed in Section 4.4 how to tackle DQCC as a particular case of quickest multi-commodity flow. In this way we managed to reduce the problem on the resolution of one or more static instance of the MCF. In References [55, 56] it has been shown that whenever each commodity is a source (or a target) for any other node, than solving it through LP-relaxation outputs an optimal solution to MCF. This result can be of interest when treating *fully entangling circuits*.

To keep the compiler more general, we opted to investigate algorithms with approximation boundary guaranteed [57, 58, 69]. Specifically, we implemented the pseudo-code outlined in Reference [33]. This is followed by a proof on the approximation quality for the case of capacity $c = 1$ and $c > 1$. We focus on the case $c = 1$, but it can be extended to $c > 1$.

By using our formalism, the approximation algorithm aims to run as many non-local operators – i.e., satisfying commodities demand – as possible. A computed solution is a sub-set $S \subseteq [k]$. The optimal solution is $S^* \subseteq [k]$ and $|S| \leq |S^*|$. It follows the (optimal) approximation boundary [33, 69]:

$$|S| \geq \frac{|S^*|}{O(\sqrt{m})}, \ m = |E| \tag{12}$$

Notice that the solution quality is inversely proportional to the number of entanglement links. It means that we cannot estimate an optimal solution to the DQCC, as for a given time horizon, this affects the quality of the solution space. Furthermore, the time-expansion increases the number of edges and so does the distance $|S^*| - |S|$. Ultimately, even if the allocated space by the time-expansion grows at most linearly with the number of non-local operations – see Section 4.4 –, this can seriously affect the performance when such an amount is very big.[16] On the contrary, it is possible to keep the time-expansion *abstract* and compiling iteratively as many operations as possible at each time-step. This method is detailed in Algorithm 2. Notice that each iteration guarantees the boundary of Equation (12) and, above all, since the instance decreases in size, the distance $|S^*| - |S|$ tends to decrease as well.

ALGORITHM 2: Iterative compiler

```
Input: $Q, [k]$
Output: $d$
1 $S \leftarrow [k]$
2 $d \leftarrow 0$
3 while $S \neq \emptyset$ do
4     $\bar{S} \leftarrow \mathtt{MCF}(Q, S)$
5     $S \leftarrow S \setminus \bar{S}$
6     $d \leftarrow d + 1$
```

7 EVALUATION

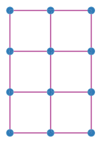
As distributed quantum architectures are still at an early stage, it is hard to predict with confidence what kind of connectivity and resources they will supply. Furthermore, it is worth mentioning that distributed computing, by its nature, presents features coming from routing models as well as compiling models. Hence, we here report and compare an interesting work available in literature dedicated on routing entangled states [77]. In such a manuscript, authors deal with unreliable optical links to create entanglement and dynamically choose a multi-path solution in order to maximise the entanglement success-rate. Even if our network topology relies on the same architecture, we model the linkage through a single path which is dedicated to the entanglement generation and distribution for a time Δ_{E}, taken big enough to guarantee a high fidelity. This is a fundamental difference, making the two models difficult to compare.

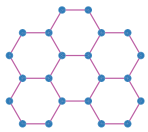
Here we evaluate the *square lattice* topology proposed in Reference [77] by comparing it with a *hexagon lattice* topology. We therefore verify the compiler performance for both the lattices in terms of:

- solution quality;
- robustness to scale-up.

We conclude the comparison with the possible implications of the results.

[16]Better upper-bounds for the worst-case solution should be investigated.


(a) Square lattice $\mathcal{S}_{\blacktriangledown}$.


(b) Hexagon lattice $\mathcal{H}$.

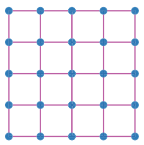

(c) Square lattice $\mathcal{S}_{\blacktriangle}$.

Fig. 19. Example of lattices used for the experimental evaluation; they all come from generator $g = 4$.

7.1 Set-up

To compare the compiler performance on different topologies, we make use of a *generator factor* g. The number of nodes and edges of each lattice will be expressed as a function of g. Because the two lattices differ by definition, it is not trivial to settle a fair comparison. To do that, we first generate a sample of hexagon lattices $\mathcal{H}$ such that

$$|P| = {}^1\!/_2 \cdot g^2 + 3g + O(1),\ |E| = {}^3\!/_4 \cdot g^2 + {}^7\!/_2 \cdot g + O(1). \tag{13}$$

We compare $\mathcal{H}$ with two square lattices, say $\mathcal{S}_{\blacktriangledown}$ and $\mathcal{S}_{\blacktriangle}$, that have sizes respectively lower and higher than $\mathcal{H}$ for each g – see Figure 19. Hence, $\mathcal{S}_{\blacktriangledown}$ is such that

$$|P| = {}^1\!/_4 \cdot g^2 + {}^3\!/_2 \cdot g + O(1),\ |E| = {}^1\!/_2 \cdot g^2 + 2g + O(1). \tag{14}$$

while $\mathcal{S}_{\blacktriangle}$ is such that

$$|P| = 2g^2 + 2g,\ |E| = g^2 + 2g + O(1). \tag{15}$$

We show in the next subsection that $\mathcal{S}_{\blacktriangle}$ and $\mathcal{S}_{\blacktriangledown}$ perform better than $\mathcal{H}$ in terms of resulting E-depth. This implies that the square lattice is a better design for distributed quantum computers, assuming that our compiler performs equally well on different topologies.

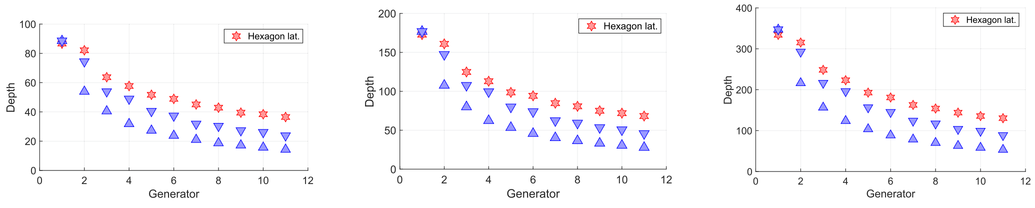
Since we use Algorithm 2, capacities are assumed to be 1. We already pointed out that such an algorithm can be extended to the case $c > 1$.

Notice that different node degrees imply different assumptions on the processor units P_i. The hexagon lattice has node degree upper-bounded by 3 and lower-bounded by 2, which means that P_i has 2 to 3 communication qubits. Similarly, the square lattice has degree upper-bounded by 4. Hence, the communication qubits per unit are 2 to 4. Since our focus here is on distributed compilation, we will assume that P_i has 1 computation qubit. This is especially reasonable when considering that real implementation of distributed architecture may use most of their local resources as *auxiliary qubits*, meant to keep the computation fault-tolerant.

Concerning the life-time of the entanglement Δ_Φ, this comes after that the operator E succeeded to store the state in the distributed system. While performing E is the hardest part – as it takes a long time Δ_{E} [48] –, once it succeeds, the storage on matter qubits is quite performing [89]. For this reason, we can just assume that the coherence time is long enough to satisfy $\Delta_\Phi > 4 \cdot \Delta_{\mathsf{CZ}}$; where the factor 4 is an upper-bound for the node degrees of lattices.

For the numerical evaluation we use a generating vector $\mathbf{g} = (1, 2, \ldots, 11)$. Hence, when the generator is fixed to 11, the size of $\mathcal{H}$ reaches $|P| = 96$ and $|E| = 131$, $\mathcal{S}_{\blacktriangledown}$ reaches $|P| = 49$ and $|E| = 84$, while $\mathcal{S}_{\blacktriangledown}$ reaches $|P| = 144$ and $|E| = 264$.

Ultimately, regarding the circuits, we have already discussed in Section 5.4 that from any Clifford circuit we can extract three separated sets of 2-qubits gates and focus on $\mathcal{L}_{\mathsf{CZ}}$ circuits. For this reason, we here consider $\mathcal{L}_{\mathsf{RCZ}}$ circuits. We generate three samples classified by their size (or number of occurring operators). Each sample is composed by 10 random circuits in order to average the results. The size of the samples are 256, 512, and 1,024.



(a) 256 RCZ (b) 512 RCZ (c) 1024 RCZ

Fig. 20. Quality scale comparison.

7.2 Results

To evaluate the results we used the `matlab` environment [72]. The employed architecture is a MacBook Air (M1, 2020, 8GB RAM).

The first result – shown in Figure 20 – is a comparison on the solution quality, a.k.a. the E-depth. As anticipated, the plots show that a square lattice gives better solutions, for any problem size. We can relate this behavior to the *ratio edges-to-nodes*. Formally, let $\mathfrak{r}_Q = \frac{|E|}{|P|}$ be such a ratio for a graph Q. Then it results that square lattices have ratio:

$$\lim_{g \to \infty} \mathfrak{r}_{\mathcal{S}} = 2. \tag{16}$$

Instead hexagon lattices have a lower ratio:

$$\lim_{g \to \infty} \mathfrak{r}_{\mathcal{H}} = {}^{3}/_{2}. \tag{17}$$

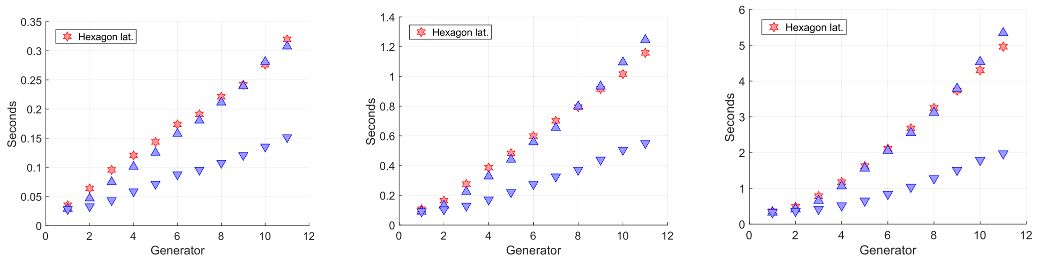
This suggests that the bigger the ratio, the better the solutions. The plots also show that the depth achieved by the different lattices may be ruled by the same polynomial function (up to some constant factor). This is in line with the intuition that a more connected topology allows for shorter depth. Furthermore, we already mentioned in Section 6.3 that, even if the approximation algorithm depends on the edges size, this is called as a subroutine that performs better and better at each iteration. All this may mean that the compiler has a convergence to an optimal depth. On the contrary, if the compiler was affected by the number of edges, the functions should swap at some point, but we never observed such phenomenon.

To conclude our evaluation, we took the average times for each sample. The results are shown in Figure 21. Differently from what we got in the solution quality evaluation – where we noticed a similar behaviour for each architecture –, the time-scale gives new perspectives in the lattices comparison. In fact, $\mathcal{H}$ and $\mathcal{S}^{\blacktriangle}$ seems to need approximately the same time to compile any circuit, with $\mathcal{S}^{\blacktriangle}$ performing slightly worse – which is coherent with the size difference between the twos. Instead, $\mathcal{S}^{\blacktriangledown}$ outperforms the others lattices. Furthermore, it seems that it is more resistant to scale-up as the scaling seems to follow a lower degree function.

8 CONCLUSION

To conclude this manuscript, let us highlight the main benefits of our framework for treating DQCC, as well as the key findings.

(i) By expressing the problem as a quickest flow problem, we could give a formulation corresponding to a multi-commodity flow problem over fixed time. This approach is particularly well fitting with our goals, because a quickest flow expresses the need to run a circuit as fast as possible, while a flow over fixed time brings a side interest into the minimization of resource usage, which is clearly a *desideratum*, but still secondary to the overall running-time.



(a) 256 RCZ (b) 512 RCZ (c) 1024 RCZ

Fig. 21. Time scale comparison.

(ii) Quasi-parallelism, represented by constraints (8) and (9), gives the possibility to consider a wider solution space. Quasi-parallelism is grounded on the idea of gathering logically sequenced telegates within the same time step, by means of an efficient circuit manipulation – see predicate $\mathcal{A}$.

(iii) We built our model step by step, each of which is rigorously explained. The result is a highly modular work. For example, if one can consider only circuits where operations can all commute each other, formulation (6) is enough and approximation bounds are available. Instead, when considering any circuit, one can easily shape the extra constraints of formulation (10). Consider, for example, the quasi-parallelism relation $\shortparallel$, we characterized it as the predicate $\mathcal{A}$. By just extending the way $\mathcal{A}$ works, the space of good solutions gets larger.

(iv) Since we modeled the problem as a network flow problem, one can also exploit the huge related literature to get inspiration in the way of tackling the problem.

(v) We deeply investigated the literature on quantum circuits and logic in order to tackle big groups of circuits with a form which would be well fitting with the constraints coming from the architecture. This led us to focus on circuits expressed in normal forms. By tackling individual normal forms, the compiler can be modulated to a form chosen and take advantage from the properties coming from a normal form. We started by outlining a normal form for Clifford circuits up to one for universal circuits. From this step-by-step analysis of the circuit, we will be able to improve the compiler in future works, while at the same time being able to evaluate our model by means of a restricted group of circuits.

(vi) We applied our compiler on different topologies. We focused on square and hexagon lattices and showed that square lattices outperforms hexagon ones, both in terms of solution quality (E-depth) and running-time. We gave some perspective on why we obtained such results, showing that the ratio edges-to-nodes is a representative metric.

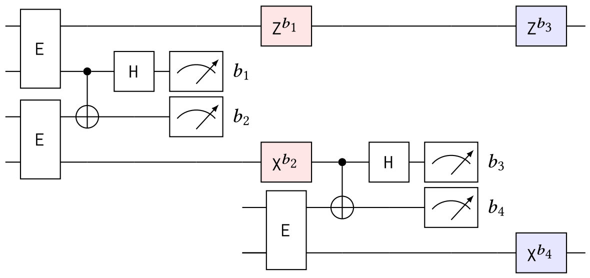
APPENDIX

A ENTANGLEMENT SWAP GENERALIZATION

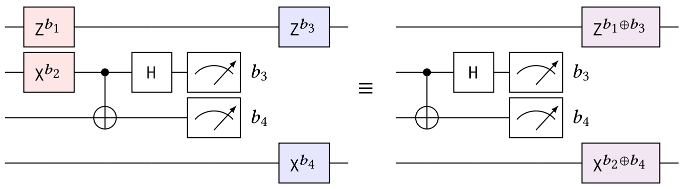
Within this section we show how to efficiently implement an entanglement path. In Section 3.3, we introduced the entanglement swap as a circuit of depth 5. We also claimed that such a depth is fixed when generalizing the entanglement swap to the entanglement path. To this aim, we give an inductive proof for such a statement, starting from the base case with entanglement path of length 2.

Theorem 3. *An entanglement path $\{P_{i_1}, P_{i_2}, \dots, P_{i_m}\}$ has an implementation with depth 5.*

Proof. Consider, as base case, that we want to create a path of length 2. Clearly, we could do that by just putting in strict sequence two entanglement swaps:

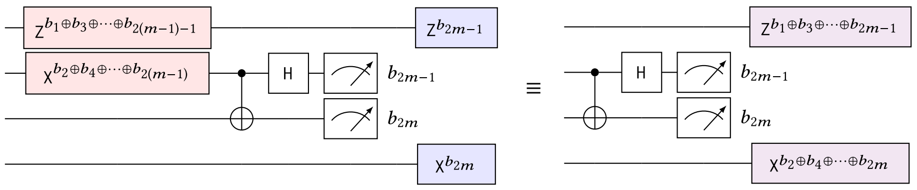


The colored operators are the only ones we are going to optimize; since the others are independent and no optimization can be applied. What follows is the base case for the induction:



Specifically, the circuit on the right of the equation has post-processing composed by $Z^{b_1 \oplus b_3}$ on first qubit and $X^{b_2 \oplus b_4}$ on last qubit. Furthermore, now the measurements are independent from other operations.

By assuming that such a shape is preserved in the inductive step, we show that this transformation can be applied to any length:

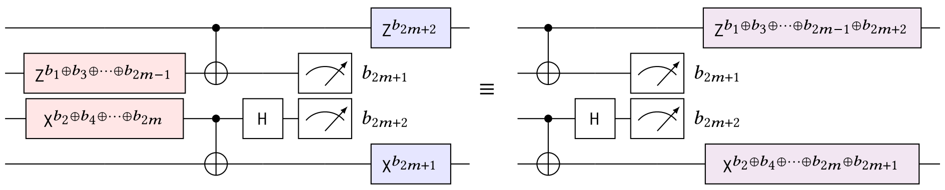

$Z^{b_1 \oplus b_3 \oplus \cdots \oplus b_{2(m-1)-1}}$
$X^{b_2 \oplus b_4 \oplus \cdots \oplus b_{2(m-1)}}$
H
b_{2m-1}
b_{2m}
$Z^{b_{2m-1}}$
$X^{b_{2m}}$
≡
H
b_{2m-1}
b_{2m}
$Z^{b_1 \oplus b_3 \oplus \cdots \oplus b_{2m-1}}$
$X^{b_2 \oplus b_4 \oplus \cdots \oplus b_{2m}}$

This proves that we can always consider an entanglement path $\{P_{i_1}, P_{i_2}, \dots, P_{i_m}\}$ to have circuit depth 5. □

We just showed an efficient implementation for the entanglement path. Now we do one last step to exploit such a result and performing a generalized remote operation efficiently.

Theorem 4. *An* RCX *of entanglement path $\{P_{i_1}, P_{i_2}, \dots, P_{i_{m+2}}\}$ has depth 5.*

Proof. Theorem 3 allows us to assume that, to perform a remote operation by using a path of length m, the computing qubits interact only with two communications qubits and depend only by Pauli operations $\mathsf{Z}^{b_1 \oplus b_3 \oplus \cdots \oplus b_{2m-1}}$ and $\mathsf{X}^{b_2 \oplus b_4 \oplus \cdots \oplus b_{2m}}$. We can further *propagate* such operations as follows:


$\mathsf{Z}^{b_{2m+2}}$
$\mathsf{Z}^{b_1 \oplus b_3 \oplus \cdots \oplus b_{2m-1}}$
b_{2m+1}
$\mathsf{X}^{b_2 \oplus b_4 \oplus \cdots \oplus b_{2m}}$
H
b_{2m+2}
$\mathsf{X}^{b_{2m+1}}$
$\equiv$
$\mathsf{Z}^{b_1 \oplus b_3 \oplus \cdots \oplus b_{2m-1} \oplus b_{2m+2}}$
b_{2m+1}
H
b_{2m+2}
$\mathsf{X}^{b_2 \oplus b_4 \oplus \cdots \oplus b_{2m} \oplus b_{2m+1}}$

In this way the measurements are independent and the depth of the circuit is not increased. □

REFERENCES

[1] Scott Aaronson and Daniel Gottesman. 2004. Improved simulation of stabilizer circuits. *Physical Review A* 70, 5 (2004), 052328.

[2] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1988. Network flows. (1988).

[3] Nitzan Akerman, Nir Navon, Shlomi Kotler, Yinnon Glickman, and Roee Ozeri. 2015. Universal gate-set for trapped-ion qubits using a narrow linewidth diode laser. *New Journal of Physics* 17, 11 (2015), 113060.

[4] Matthew Amy, Dmitri Maslov, and Michele Mosca. 2014. Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 10 (2014), 1476–1489.

[5] Pablo Andres-Martinez and Chris Heunen. 2019. Automated distribution of quantum circuits via hypergraph partitioning. *Physical Review A* 100, 3 (2019), 032308.

[6] Miriam Backens. 2014. The ZX-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics* 16, 9 (2014), 093021.

[7] C. J. Ballance, T. P. Harty, N. M. Linke, and D. M. Lucas. 2014. High-fidelity two-qubit quantum logic gates using trapped calcium-43 ions. *arXiv preprint arXiv:1406.5473* (2014).

[8] Robert Beals, Stephen Brierley, Oliver Gray, Aram W. Harrow, Samuel Kutin, Noah Linden, Dan Shepherd, and Mark Stather. 2013. Efficient distributed quantum computing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 469, 2153 (2013), 20120686.

[9] Kyle E C. Booth, Minh Do, J. Christopher Beck, Eleanor Rieffel, Davide Venturelli, and Jeremy Frank. 2018. Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. In *28th International Conference on Automated Planning and Scheduling.*

[10] Adi Botea, Akihiro Kishimoto, and Radu Marinescu. 2018. On the complexity of quantum circuit compilation. In *Eleventh Annual Symposium on Combinatorial Search.*

[11] Lukas Burgholzer, Sarah Schneider, and Robert Wille. 2022. Limiting the search space in optimal quantum circuit mapping. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 466–471.

[12] Angela Sara Cacciapuoti, Marcello Caleffi, Francesco Tafuri, Francesco Saverio Cataliotti, Stefano Gherardini, and Giuseppe Bianchi. 2019. Quantum internet: Networking challenges in distributed quantum computing. *IEEE Network* 34, 1 (2019), 137–143.

[13] Angela Sara Cacciapuoti, Marcello Caleffi, Rodney Van Meter, and Lajos Hanzo. 2020. When entanglement meets classical communications: Quantum teleportation for the quantum internet. *IEEE Transactions on Communications* 68, 6 (2020), 3808–3833.

[14] Titouan Carette, Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. 2021. Completeness of graphical languages for mixed state quantum mechanics. *ACM Transactions on Quantum Computing* 2, 4 (2021), 1–28.

[15] Davide Castelvecchi. 2018. The quantum internet has arrived (and it hasn't). *Nature* 554, 7690 (2018), 289–293.

[16] Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. 2007. Approximation algorithms for the unsplittable flow problem. *Algorithmica* 47, 1 (2007), 53–78.

[17] Kaushik Chakraborty, David Elkouss, Bruno Rijsman, and Stephanie Wehner. 2020. Entanglement distribution in a quantum network: A multicommodity flow-based approach. *IEEE Transactions on Quantum Engineering* 1 (2020), 1–21.

[18] Chandra Chekuri, Sanjeev Khanna, and Bruce Shepherd. 2004. The all-or-nothing multicommodity flow problem. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*. 156–165.

[19] Chandra Chekuri, Sanjeev Khanna, and Bruce Shepherd. 2006. An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing* 2, 1 (2006), 137–146.

[20] Dae-Sik Choi and In-Chan Choi. 2006. On the effectiveness of the linear programming relaxation of the 0-1 multi-commodity minimum cost network flow problem. In *International Computing and Combinatorics Conference*. Springer, 517–526.

[21] Claudio Cicconetti, Marco Conti, and Andrea Passarella. 2021. Request scheduling in quantum networks. *IEEE Transactions on Quantum Engineering* 2 (2021), 2–17.

[22] Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. 2020. Towards a distributed quantum computing ecosystem. *IET Quantum Communication* 1, 1 (2020), 3–8.

[23] Davood Dadkhah, Mariam Zomorodi, and Seyed Ebrahim Hosseini. 2021. A new approach for optimization of distributed quantum circuits. *International Journal of Theoretical Physics* 60, 9 (2021), 3271–3285.

[24] Omid Daei, Keivan Navi, and Mariam Zomorodi. 2021. Improving the teleportation cost in distributed quantum circuits based on commuting of gates. *International Journal of Theoretical Physics* 60, 9 (2021), 3494–3513.

[25] Omid Daei, Keivan Navi, and Mariam Zomorodi-Moghadam. 2020. Optimized quantum circuit partitioning. *International Journal of Theoretical Physics* 59, 12 (2020), 3804–3820.

[26] Zohreh Davarzani, Mariam Zomorodi-Moghadam, Mahboobeh Houshmand, and Mostafa Nouri-Baygi. 2020. A dynamic programming approach for distributing quantum circuits by bipartite graphs. *Quantum Information Processing* 19, 10 (2020), 1–18.

[27] Jeroen Dehaene and Bart De Moor. 2003. Clifford group, stabilizer states, and linear and quadratic operations over GF (2). *Physical Review A* 68, 4 (2003), 042318.

[28] Ross Duncan. 2012. A graphical approach to measurement-based quantum computing. *arXiv preprint arXiv:1203.6242* (2012).

[29] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John Van De Wetering. 2020. Graph-theoretic simplification of quantum circuits with the ZX-calculus. *Quantum* 4 (2020), 279.

[30] Wolfgang Dür, Raphael Lamprecht, and Stefan Heusler. 2017. Towards a quantum internet. *European Journal of Physics* 38, 4 (2017), 043001.

[31] Andrew Eddins, Mario Motta, Tanvi P. Gujarati, Sergey Bravyi, Antonio Mezzacapo, Charles Hadfield, and Sarah Sheldon. 2022. Doubling the size of quantum simulators by entanglement forging. *PRX Quantum* 3, 1 (2022), 010309.

[32] Shimon Even, Alon Itai, and Adi Shamir. 1975. On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science*. IEEE, 184–193.

[33] Li Fei. 2017. Multicommodity Flows and Disjoint Paths Problem. https://cs.gmu.edu/~lifei/teaching/cs684spring17/lec8.pdf. (2017).

[34] Davide Ferrari and Michele Amoretti. 2021. Noise-adaptive quantum compilation strategies evaluated with application-motivated benchmarks. *arXiv preprint arXiv:2108.11874* (2021).

[35] Davide Ferrari, Angela Sara Cacciapuoti, Michele Amoretti, and Marcello Caleffi. 2021. Compiler design for distributed quantum computing. *IEEE Transactions on Quantum Engineering* 2 (2021), 1–20.

[36] Lisa Fleischer and Martin Skutella. 2002. The quickest multicommodity flow problem. In *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 36–53.

[37] Lester R. Ford Jr. and D. R. Fulkerson. 1958. A suggested computation for maximal multi-commodity network flows. *Management Science* 5, 1 (1958), 97.

[38] Lester R. Ford Jr. and Delbert Ray Fulkerson. 1958. Constructing maximal dynamic flows from static flows. *Operations Research* 6, 3 (1958), 419–433.

[39] Ranjani G. Sundaram, Himanshu Gupta, and C R. Ramakrishnan. 2021. Efficient distribution of quantum circuits. In *35th International Symposium on Distributed Computing*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[40] Jay Gambetta. 2022. Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing. (2022).

[41] Alysson Gold, J. P. Paquette, Anna Stockklauser, Matthew J. Reagor, M. Sohaib Alam, Andrew Bestwick, Nicolas Didier, Ani Nersisyan, Feyza Oruc, Armin Razavi, et al. 2021. Entanglement across separate silicon dies in a modular superconducting qubit device. *npj Quantum Information* 7, 1 (2021), 1–10.

[42] Daniel Gottesman. 1998. Theory of fault-tolerant quantum computation. *Physical Review A* 57, 1 (1998), 127.

[43] T. P. Harty, D. T. C. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas. 2014. High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit. *Physical Review Letters* 113, 22 (2014), 220501.

[44] Luke E. Heyfron and Earl T. Campbell. 2018. An efficient quantum compiler that reduces T count. *Quantum Science and Technology* 4, 1 (2018), 015004.

[45] Stefan Hillmich, Alwin Zulehner, and Robert Wille. 2021. Exploiting quantum teleportation in quantum circuit mapping. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 792–797.

[46] Toshinari Itoko, Rudy Raymond, Takashi Imamichi, Atsushi Matsuo, and Andrew W. Cross. 2019. Quantum circuit compilers using gate commutation rules. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 191–196.

[47] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. 2018. A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 559–568.

[48] Norbert Kalb, Andreas A. Reiserer, Peter C. Humphreys, Jacob J. W. Bakermans, Sten J. Kamerling, Naomi H. Nickerson, Simon C. Benjamin, Daniel J. Twitchen, Matthew Markham, and Ronald Hanson. 2017. Entanglement distillation between solid-state quantum network nodes. *Science* 356, 6341 (2017), 928–932.

[49] Peter J. Karalekas, Nikolas A. Tezak, Eric C. Peterson, Colm A. Ryan, Marcus P. da Silva, and Robert S. Smith. 2020. A quantum-classical cloud platform optimized for variational hybrid algorithms. *Quantum Science and Technology* 5, 2 (2020), 024003.

[50] David Kielpinski, Chris Monroe, and David J. Wineland. 2002. Architecture for a large-scale ion-trap quantum computer. *Nature* 417, 6890 (2002), 709–711.

[51] H. Jeff Kimble. 2008. The quantum internet. *Nature* 453, 7198 (2008), 1023–1030.

[52] Aleks Kissinger and John van de Wetering. 2019. Reducing T-count with the ZX-calculus. *arXiv preprint arXiv: 1903.10477* (2019).

[53] Aleks Kissinger and John van de Wetering. 2020. PyZX: Large scale automated diagrammatic reasoning. In *Proceedings 16th International Conference on Quantum Physics and Logic*, Vol. 318. Open Publishing Association, 229–241.

[54] Aleksei Yur'evich Kitaev. 1997. Quantum computations: Algorithms and error correction. *Uspekhi Matematicheskikh Nauk* 52, 6 (1997), 53–112.

[55] D. Kleitman, A. Martin-Löf, B. Rothschild, and A. Whinston. 1970. A matching theorem for graphs. *Journal of Combinatorial Theory* 8, 1 (1970), 104–114.

[56] Daniel J. Kleitman. 1971. An algorithm for certain multi-commodity flow problems. *Networks* 1, 1 (1971), 75–90.

[57] Petr Kolman and Christian Scheideler. 2002. Improved bounds for the unsplittable flow problem. In *SODA*, Vol. 2. 184–193.

[58] Bernhard Korte and Jens Vygen. 2006. Multicommodity flows and edge-disjoint paths. In *Combinatorial Optimization: Theory and Algorithms*. Springer.

[59] Wojciech Kozlowski, Stephanie Wehner, Rodney Van Meter, Bruno Rijsman, Angela Sara Cacciapuoti, and Marcello Caleffi. 2021. *Architectural Principles for a Quantum Internet*. Internet-Draft draft-irtf-qirg-principles-03. Internet Engineering Task Force. Work in Progress.

[60] Stefan Krastanov, Hamza Raniwala, Jeffrey Holzgrafe, Kurt Jacobs, Marko Lončar, Matthew J. Reagor, and Dirk R. Englund. 2021. Optically heralded entanglement of superconducting systems in quantum networks. *Physical Review Letters* 127, 4 (2021), 040503.

[61] Gushu Li, Yufei Ding, and Yuan Xie. 2019. Tackling the qubit mapping problem for NISQ-era quantum devices. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. 1001–1014.

[62] Ying Li. 2015. A magic state's fidelity can be superior to the operations that created it. *New Journal of Physics* 17, 2 (2015), 023037.

[63] Maokai Lin and Patrick Jaillet. 2014. On the quickest flow problem in dynamic networks – A parametric min-cost flow approach. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1343–1356.

[64] Norbert M. Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A. Landsman, Kenneth Wright, and Christopher Monroe. 2017. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3305–3310.

[65] Daniel Litinski. 2019. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum* 3 (2019), 128.

[66] Yehan Liu, Zlatko Minev, Thomas G. McConkey, and Jay Gambetta. 2022. Design of interacting superconducting quantum circuits with quasi-lumped models. In *American Physical Society (March Meeting)*.

[67] Liam Madden and Andrea Simonetto. 2022. Best approximate quantum compiling problems. *ACM Transactions on Quantum Computing* 3, 2 (2022), 1–29.

[68] Marco Maronese, Lorenzo Moro, Lorenzo Rocutto, and Enrico Prati. 2022. Quantum compiling. In *Quantum Computing Environments*. Springer, 39–74.

[69] Maren Martens. 2009. A simple greedy algorithm for the k-disjoint flow problem. In *International Conference on Theory and Applications of Models of Computation*. Springer, 291–300.

[70] Dmitri Maslov, Sean M. Falconer, and Michele Mosca. 2008. Quantum circuit placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 4 (2008), 752–763.

[71] Dmitri Maslov and Martin Roetteler. 2018. Shorter stabilizer circuits via Bruhat decomposition and quantum circuit transformations. *IEEE Transactions on Information Theory* 64, 7 (2018), 4729–4738.

[72] MATLAB. 2021. *R2021b*. The MathWorks Inc., Natick, Massachusetts.

[73] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim. 2014. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Physical Review A* 89, 2 (2014), 022317.

[74] Lorenzo Moro, Matteo G. A. Paris, Marcello Restelli, and Enrico Prati. 2021. Quantum compiling by deep reinforcement learning. *Nature Communications Physics* 4, 178 (2021).

[75] Michael A. Nielsen and Isaac Chuang. 2002. Quantum computation and quantum information. (2002).

[76] Eesa Nikahd, Naser Mohammadzadeh, Mehdi Sedighi, and Morteza Saheb Zamani. 2021. Automated window-based partitioning of quantum circuits. *Physica Scripta* 96, 3 (2021), 035102.

[77] Mihir Pant, Hari Krovi, Don Towsley, Leandros Tassiulas, Liang Jiang, Prithwish Basu, Dirk Englund, and Saikat Guha. 2019. Routing entanglement in the quantum internet. *npj Quantum Information* 5, 1 (2019), 1–9.

[78] Stefano Pirandola and Samuel L. Braunstein. 2016. Physics: Unite to build a quantum Internet. *Nature News* 532, 7598 (2016), 169.

[79] Julian Rabbie, Kaushik Chakraborty, Guus Avis, and Stephanie Wehner. 2022. Designing quantum networks using preexisting infrastructure. *npj Quantum Information* 8, 1 (2022), 1–12.

[80] Mohammad Beheshti Roui, Mariam Zomorodi, Masoomeh Sarvelayati, Moloud Abdar, Hamid Noori, Paweł Pławiak, Ryszard Tadeusiewicz, Xujuan Zhou, Abbas Khosravi, Saeid Nahavandi, et al. 2021. A novel approach based on genetic algorithm to speed up the discovery of classification rules on GPUs. *Knowledge-Based Systems* 231 (2021), 107419.

[81] Moein Sarvaghad-Moghaddam and Mariam Zomorodi. 2021. A general protocol for distributed quantum gates. *Quantum Information Processing* 20, 8 (2021), 1–14.

[82] Peter Selinger. 2013. Quantum circuits of T-depth one. *Physical Review A* 87, 4 (2013), 042302.

[83] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. 2018. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization.* 113–125.

[84] Aravind Srinivasan. 1997. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. In *Proceedings 38th Annual Symposium on Foundations of Computer Science.* IEEE, 416–425.

[85] Anand Srivastav and Peter Stangier. 2000. On complexity, representation and approximation of integral multicommodity flows. *Discrete Applied Mathematics* 99, 1-3 (2000), 183–208.

[86] L. J. Stephenson, D. P. Nadlinger, B. C. Nichol, S. An, P. Drmota, T. G. Ballance, K. Thirumalai, J. F. Goodwin, D. M. Lucas, and C. J. Ballance. 2020. High-rate, high-fidelity entanglement of qubits across an elementary quantum network. *Physical Review Letters* 124, 11 (2020), 110501.

[87] John van de Wetering. 2020. ZX-calculus for the working quantum computer scientist. *arXiv preprint arXiv:2012.13966* (2020).

[88] Rodney Van Meter and Simon J. Devitt. 2016. The path to scalable distributed quantum computing. *Computer* 49, 9 (2016), 31–42.

[89] Pengfei Wang, Chun-Yang Luan, Mu Qiao, Mark Um, Junhua Zhang, Ye Wang, Xiao Yuan, Mile Gu, Jingning Zhang, and Kihwan Kim. 2021. Single ion qubit with estimated coherence time exceeding one hour. *Nature Communications* 12, 1 (2021), 1–8.

[90] Stephanie Wehner, David Elkouss, and Ronald Hanson. 2018. Quantum internet: A vision for the road ahead. *Science* 362, 6412 (2018).

[91] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. 2019. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In *2019 56th ACM/IEEE Design Automation Conference.* IEEE, 1–6.

[92] Mithuna Yoganathan, Richard Jozsa, and Sergii Strelchuk. 2019. Quantum advantage of unitary Clifford circuits with magic state inputs. *Proceedings of the Royal Society A* 475, 2225 (2019), 20180427.

[93] Yuan-Hang Zhang, Pei-Lin Zheng, Yi Zhang, and Dong-Ling Deng. 2020. Topological quantum compiling with reinforcement learning. *Physical Review Letters* 125, 17 (2020), 170501.

[94] Changchun Zhong, Zhixin Wang, Changling Zou, Mengzhen Zhang, Xu Han, Wei Fu, Mingrui Xu, S. Shankar, Michel H. Devoret, Hong X. Tang, et al. 2020. Proposal for heralded generation and detection of entangled microwave–optical-photon pairs. *Physical Review Letters* 124, 1 (2020), 010511.

[95] Xinlan Zhou, Debbie W. Leung, and Isaac L. Chuang. 2000. Methodology for quantum logic gate construction. *Physical Review A* 62, 5 (2000), 052316.

[96] Mariam Zomorodi-Moghadam, Zohreh Davarzani, Ismail Ghodsollahee, et al. 2021. Connectivity matrix model of quantum circuits and its application to distributed quantum circuit optimization. *Quantum Information Processing* 20 (2021).

[97] Mariam Zomorodi-Moghadam, Mahboobeh Houshmand, and Monireh Houshmand. 2018. Optimizing teleportation cost in distributed quantum circuits. *International Journal of Theoretical Physics* 57, 3 (2018), 848–861.

[98] Alwin Zulehner and Robert Wille. 2019. Compiling $SU(4)$ quantum circuits to IBM QX architectures. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference.* 185–190.

Received 28 December 2021; revised 16 November 2022; accepted 21 December 2022