

Exploiting Movable Logical Qubits for Lattice Surgery Compilation

Laura S. Herzog
laura.herzog@tum.de
Chair for Design Automation
Technical University Munich
Germany

Lucas Berent
Chair for Design Automation
Technical University Munich
Germany
Iceberg Quantum
Sydney, Australia

Aleksander Kubica
Yale Quantum Institute & Department of Applied Physics
Yale University
New Haven, CT, USA

Robert Wille
Chair for Design Automation
Technical University of Munich
Munich Quantum Software Company
Germany

Abstract

Lattice surgery with two-dimensional quantum error correcting codes is among the leading schemes for fault-tolerant quantum computation, motivated by superconducting hardware architectures. In conventional lattice surgery compilation schemes, logical circuits are compiled following a place-and-route paradigm, where logical qubits remain statically fixed in space throughout the computation. In this work, we introduce a paradigm shift by exploiting movable logical qubits via teleportation during the logical lattice surgery CNOT gate. Focusing on lattice surgery with the color code, we propose a proof-of-concept compilation scheme that leverages this capability. Numerical simulations show that the proposed approach can substantially reduce the routed circuit depth compared to standard place-and-route compilation techniques. Our results demonstrate that optimizations based on movable logical qubits are not limited to architectures with physically movable qubits, such as neutral atoms or trapped ions—they are also readily applicable to superconducting quantum hardware. An open-source implementation of our method is available on GitHub <https://github.com/munich-quantum-toolkit/qecc>.

1 Introduction

Quantum error correction (QEC) is essential for fault-tolerant quantum computation [1–6] and implementation of large-scale quantum algorithms [7]. A major design task of scalable fault-tolerant quantum computation is efficient logical compilation. In general, to compile a logical quantum circuit in a fault-tolerant manner, one has to pick a specific quantum hardware; then, one chooses a suitable QEC code and maps the quantum circuit into basic logical operations on the given architecture that can be further decomposed into physical operations.

This work considers planar quantum hardware with static physical qubits and local connectivity, motivated by superconducting architectures. A native choice for those architectures are topological codes in two dimensions with local qubit connectivity and logical operations implemented via *lattice surgery*. Based on this, the compilation is performed, i.e., the translation of logical circuits to operations based on lattice surgery. The key aim of this task is

to achieve the lowest possible space-time overheads—the product of the total number of required physical qubits and the schedule depth—of a computation using some QEC code.

Most existing lattice surgery compilation approaches [8–15] assume a *place-and-route* paradigm, where logical data qubits are assigned a static position at the start of the procedure (“mapping”) and logical entangling operations require finding paths (“routing”) between corresponding logical qubits. However, this assumption of static logical data qubits neglects additional degrees of freedom available through lattice surgery operations. In particular, measurement-based CNOT implementations, that are typically realized using lattice surgery, naturally allow movable logical data qubits via “free” logical qubit *teleportations*—which require the same number of lattice surgery operations as the standard CNOT scheme. Notably, we can realize movable logical qubits in quantum architectures with static physical qubits, such as superconducting quantum hardware. To the best of our knowledge, this degree of freedom has not been exploited in any previous work on lattice surgery compilation.

This work proposes a heuristic compilation routine that exploits movable logical data qubits via teleportations during CNOT operations and examines how this flexibility allows us to reduce the depth of the compiled schedules on a given architecture. Locations of logical data qubits are dynamically adjusted during the compilation process with the goal of executing more gates in parallel. The proposed approach is based on two-dimensional topological codes; for concreteness we focus on a color code architecture, because the structure of the logical operators of the color code enables us to utilize movable logical qubits via teleportations in a particularly flexible and elegant way. We build upon Ref. [16] which is the only available work in such a setup. The suggested measurement-based CNOT adaption is straightforwardly applicable to other topological codes, such as the surface code. Simulations with the proposed method identify regimes where teleportation-based movements significantly reduce routed layers, marking an initial step toward lattice surgery compilation beyond the place-and-route paradigm.

This paper is structured as follows. In Sec. 2, color codes, lattice surgery, and previous work are reviewed. This is followed by the main idea of exploiting movable logical qubits via the simultaneous

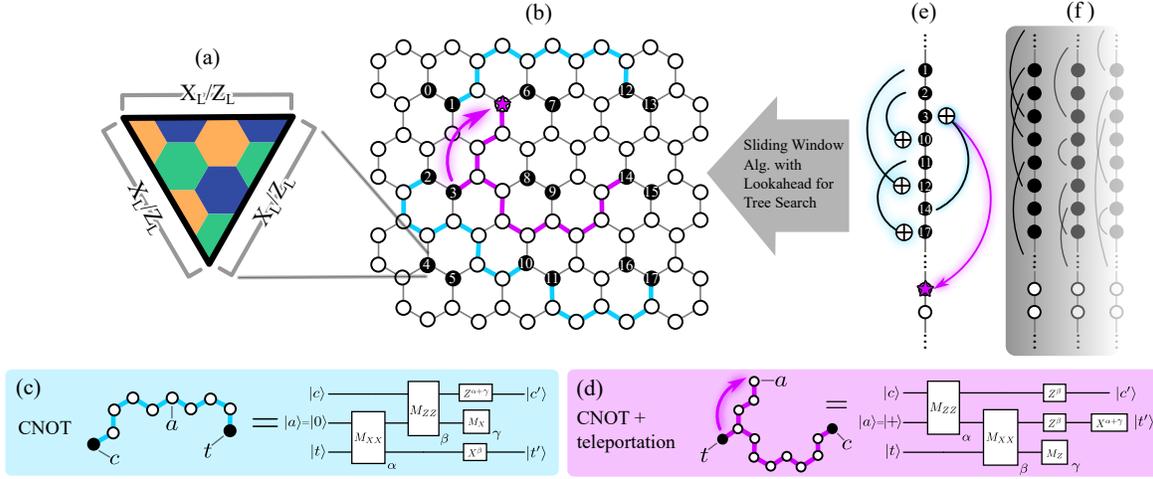


Figure 1: Overview of the movable logical qubits approach. (a) A color code patch encodes a single logical qubit which can be used as either a logical data or ancilla patch. (b) Macroscopic routing graph \mathcal{R} ; black and white nodes represent data and ancilla patches, respectively. Paths in blue indicate standard CNOT gates performed via lattice surgery with the measurement-based scheme shown in (c); the pink tree depicts a CNOT with teleportation as shown in (d). The example displays that the target qubit with label 3 is teleported to the position marked by a pink star. (e, f) display the data and ancilla patches as 1D linearized view of the routed layers. (f) shows the future routed layers that determine the teleportation-based movements in (e).

CNOT + teleportation protocol and how this can be used to reduce routed depths in Sec. 3. Based on this, the proposed heuristic compilation routine, which makes use of the movable logical qubits via teleportations, is presented in Sec. 4. The performance of the method is studied numerically in Sec. 5. We conclude in Sec. 6.

2 Background and Related Work

QEC codes use n physical qubits to encode $k < n$ logical qubits. Topological codes are among the most important class of QEC codes—they have favourable properties, such as geometric locality, which make them compatible with superconducting quantum hardware. There are various types of topological codes [17], including the *surface code* [18–20], which is very broadly used due to good available decoders, its high threshold, and ease of physical implementation; and the *color code* [21, 22], which has certain benefits compared to the surface code. For instance, transversal Clifford gates and supporting both logical X_L and Z_L operators along each boundary. Even the decoding task—which is known to be challenging for color codes—has been shown to come closer to the performance of the surface code [23–25]. The color code can be implemented on a triangular patch as detailed in the following example.

EXAMPLE 1. Consider the triangular color code patch in Fig. 1a that encodes one logical qubit. Physical data qubits are placed on vertices of the lattice and both X and Z stabilizers are associated with the faces within the triangular patch. The distance is $d = 5$, which determines how many errors $t = (d - 1)/2$ can be corrected in principle. Minimum-weight logical operator representatives X_L and Z_L can be found along each side of the triangular patch.

To perform lattice surgery compilation, which translates a logical circuit into operations on a 2D architecture, one abstracts away from

the specific form of the color code patches and represents each of them as vertices $V_{\mathcal{R}}$ on a macroscopic routing graph $\mathcal{R} = (V_{\mathcal{R}}, E_{\mathcal{R}})$ as shown in Fig. 1b. Black nodes are logical data qubits and white nodes represent logical ancilla qubits. One refers to choosing a layout of logical data and ancilla patches, and placing specific data qubit labels on the respective data patches as *mapping*. In order to execute logical two-qubit gates like a CNOT, the respective control and target qubits need to be connected on the macroscopic routing graph \mathcal{R} following the basic connectivity given by the edges $E_{\mathcal{R}}$. This constitutes a *routing* task, where one finds many non-overlapping paths such that the final schedule is as short as possible. Examples of such paths for CNOT gates are displayed in blue in Fig. 1b. Along each of these paths, a measurement-based CNOT scheme (Fig. 1c) is performed. Thus, a path must contain an additional logical ancilla patch, which can be placed on any vertex along the path if a color code architecture is assumed. In this scheme, lattice surgery is performed for the $Z_L Z_L$ and $X_L X_L$ measurements.

Given a layout, the main objective of lattice surgery compilation is to reduce the depth of the final schedule \tilde{d} . This means, if the input logical circuit allows to perform gates in parallel, one tries to simultaneously fit as many of these parallel gates with non-overlapping paths on the macroscopic routing graph \mathcal{R} . The input logical quantum circuit is split into d_L logical layers, where in each layer gates can be executed in parallel as they have disjoint logical qubit support; its depth d_L provides the lower bound on the routed depth. The logical layer structure is based on the *directed acyclic graph* (DAG) of the circuit [26, 27]. Mapping and routing on some macroscopic routing graph \mathcal{R} introduces further restrictions such that the resulting number \tilde{d} of routed layers of the final schedule is likely to be higher than d_L . Hence, a good compilation scheme attempts to lower \tilde{d} and getting as close as possible to d_L .

In general, lattice surgery [28–30] is a technique that allows to perform a CNOT gate in 2D using nearest-neighbor connectivity with the measurement-based scheme shown in Fig. 1c. The transversal CNOT implementation between two color code patches would require non-local connectivity otherwise. Originally, lattice surgery was developed for the surface code, but it can be realized for other topological codes as well [31]. Furthermore, the technique can also be applied to more general QLDPC codes [32–35]. Together with single-qubit Clifford operations—which do not require code deformation and can be applied transversally in the color code—as well as T state injection, a universal gate set can be obtained.

Related work on lattice surgery compilation mostly focuses on the surface code [8–15], leaving a largely unaddressed gap in the literature for other topological codes. Another related but unresolved issue is whether directly compiling Clifford+T gates (as in this work) to lattice surgery operations or translating the gates into multi-qubit-Pauli measurements through Clifford conjugation first [36] is to be favored, despite of early studies [37, 38]. Furthermore, one should note that with T state generation becoming cheaper [39], optimizing logical Cliffords and taking into account their compilation cost become increasingly important [38].

Even though not yet introduced in a compilation setup as considered in this work, there are different methods that allow to move logical information on 2D structures. Examples are standard teleportation schemes with lattice surgery [36] and more time efficient schemes as for instance sliding and gliding of patches for the surface code [40]. Note that the CNOT + teleportation utilized in this work does not impose a higher time overhead in terms of lattice surgery operations than the standard CNOT scheme. This stands in contrast to recent independent work [41] where additional time overhead is deliberately incurred to achieve denser layouts with movable logical qubits. Finally, note that for other architectures than considered here, i.e., load/store architectures with dense memories, movement of qubits is unavoidable [42].

3 Main Idea

The goal of this work is to go beyond the traditional place-and-route paradigm commonly assumed in hardware architectures with static physical qubits, such as superconducting hardware. Although physical qubits are static, logical data can be dynamically moved, enabling reductions in the routed depth during lattice surgery compilation. In particular, we utilize movable logical qubits via teleportation steps that are implemented using the measurement-based CNOT scheme. For this, we first show in Sec. 3.1 how the measurement-based CNOT circuit from Fig. 1c can be adapted to execute the required teleportation allowing for movable logical qubits. Then, Sec. 3.2 illustrates why such CNOT + teleportation steps can be useful.

3.1 Movable Logical Qubits at the Microscopic Level

Quantum computing with lattice surgery allows us to perform “free” movements of logical information via teleportation. To see this, one can adapt the measurement-based CNOT scheme from Fig. 1c by applying a measurement on the target patch, such that the target patch is teleported to the position of the logical ancilla as shown in Fig. 1d. Alternatively, the control patch can be measured such

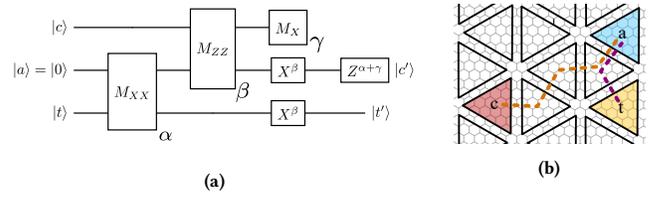


Figure 2: (a) CNOT + teleportation of control to ancilla. $\alpha, \beta, \gamma \in \{0, 1\}$ denote the measurement outcomes of the corresponding Pauli measurements. (b) Possible teleportation on the geometry with the color code on the microscopic level. The $X_L X_L$ and $Z_L Z_L$ measurements are performed along the violet and orange dashed lines, respectively.

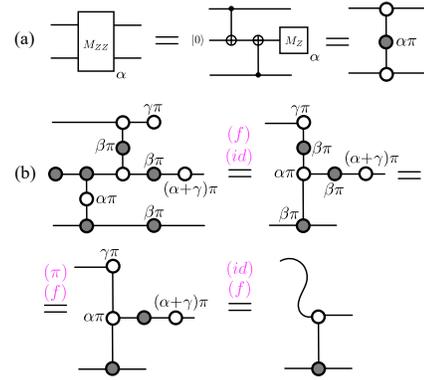


Figure 3: Derivation of CNOT + teleportation from control to ancilla with ZX calculus. Z-spiders are depicted in white and X-spiders in grey. (a) Representation of ZZ measurement as quantum circuit and corresponding ZX diagram. (b) Translation of circuit in Fig. 2a into ZX diagrams. The applied ZX rules per step are displayed in pink following [43]. The final expression is topologically equivalent to the CNOT gate in ZX calculus.

that one teleports the control to the ancilla as shown in Fig. 2a. To exploit a high level of flexibility of those teleportations, consider *tree* structures as displayed in pink in Fig. 1b, with three terminal nodes, one for each of the control, target and ancilla patches. Thus, the path between the control and target patch is extended by a branch to the position of the ancilla patch. On the microscopic level, these tree structures are well-defined as elaborated in the following example.

EXAMPLE 2. Consider Fig. 2b as example for a CNOT + teleportation step laid out on an architecture with distance $d = 7$ color codes. Assume that one wants to move the control qubit to the position of the blue patch, thus this patch is chosen as logical ancilla for the CNOT + teleportation scheme. By performing an $X_L X_L$ measurement along the route indicated by the violet dashed line, followed by a $Z_L Z_L$ measurement along the orange dashed line, as well as the corrections as displayed in Fig. 2a, one can achieve the desired teleportation-based movement of the control qubit, while executing the logical CNOT gate.

Note that two dotted lines are “entering” the blue ancilla patch. This is only possible because the color code hosts both an X_L and Z_L operator along each boundary. Due to this freedom one could place the logical ancilla on any free spot that allows for a proper

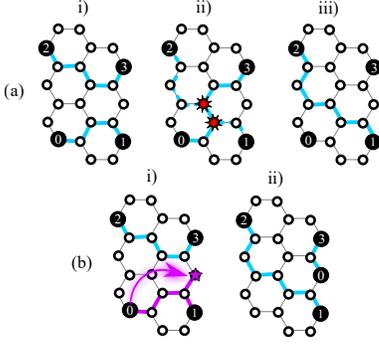


Figure 4: CNOT + teleportation at the macroscopic level. (a) Routing of the given circuit with four CNOT gates and two logical layers leads to three routed layers if the choice of qubit mapping is static. (b) With a suitable choice of a CNOT + teleportation one can route the logical circuit in two routed layers.

connection between the data patches. In case of the surface code, the choice of logical ancilla positions would be more restricted due to each boundary supporting only one logical Pauli, either X_L or Z_L , but not both.

The adaption of the measurement-based scheme for the CNOT gate can be proven straightforwardly with the ZX-calculus [43, 44] as shown in Fig. 3 for the CNOT + teleportation of control to ancilla position. The derivation works analogously for teleporting the target to the ancilla position.

3.2 Exploiting Movable Logical Qubits at the Macroscopic Level

Utilizing teleportation-based movements as introduced above enables us to go beyond the place-and-route paradigm on the macroscopic level of compilation. One can abstract away from the specific structure of the color code patches and simplify the picture to a macroscopic (hexagonal) routing graph \mathcal{R} whose vertices represent color code patches. How the overall routed depth can be reduced by exploiting teleportation-based movements is explained by the following example.

EXAMPLE 3. Consider a logical circuit comprising the first logical layer $CNOT(2,3)$ and $CNOT(0,1)$, and the second layer $CNOT(0,3)$ and $CNOT(2,1)$. Its logical depth is $d_L = 2$. Given a specific, fixed logical data qubit mapping as displayed in Fig. 4a, the first two gates can be routed simultaneously in layer (i). In layer (ii), one can route $CNOT(0,3)$ but the attempt of routing $CNOT(2,1)$ in parallel inevitably fails due to a crossing of paths such that the latter gate must be relegated to layer (iii). Thus, the routed depth is $\tilde{d} = 3$. In contrast to this, starting with the same fixed mapping in (b) but extending one of the gates in layer (i) to a CNOT + teleportation step (pink tree), it is possible to route the remaining two gates simultaneously in layer (ii), since crossings could be avoided. By exploiting movable logical qubits, the routed depth could be reduced to the logical depth of the logical circuit, i.e., $\tilde{d} = d_L = 2$.

To illustrate how to use this degree of freedom in practice let us return to Fig. 1. Assume a given macroscopic routing graph \mathcal{R} as displayed in Fig. 1b, and four logical gates in a logical/routed layer:

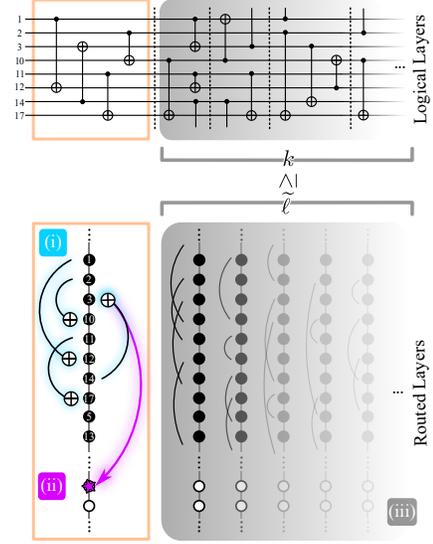


Figure 5: Heuristic optimization of routed depth with CNOT + teleportation. Top: The logical circuit is split in logical layers (separated by dashed lines), with the first enclosed by an orange box. Bottom: Logical circuit compilation into routed layers. First, the CNOTs of the first logical layer are routed in the standard way as indicated in blue (i). Then, based on the next k logical layers, tree structures (pink) are searched that minimize the number of routed layers $\tilde{\ell}$ (ii). Afterwards (iii), the $\tilde{\ell}$ resulting routed layers are fixed in the schedule. The procedure is repeated with the next logical layer (i).

$CNOT(1,12)$, $CNOT(2,10)$, $CNOT(11,17)$, and $CNOT(14,3)$. All these gates can be routed without overlaps. Beyond the standard CNOT gates in Fig. 1b, CNOT combined with teleportation (Fig. 1c) enables us to reduce the routed depth in subsequent logical layers. Given the routing of the current layer as displayed in a 1D linearized view in Fig. 1e one can search for beneficial tree structures (indicated in pink) based on the routings of a finite number k of subsequent logical layers in the circuit as indicated in Fig. 1f. For example, the teleportation based on the tree structure in Fig. 1b reduces the routed depth $\tilde{\ell}$ of k future logical layers from Fig. 1f. This heuristic procedure is detailed in the next section.

4 Proposed Heuristic Approach

The overall aim is to route the gates of a given circuit with the lowest routed depth \tilde{d} . To this end, it is useful to exploit movable logical qubits during the execution of CNOT gates, as detailed in the previous section. We now explain the heuristic “sliding-window” strategy with lookahead in Sec. 4.1, followed by an explanation of a simulated annealing based optimization subroutine in Sec. 4.2, and the underlying shortest-first routing routine in Sec. 4.3.

4.1 Sliding Window Approach

The algorithm used in this work is summarized in Fig. 5. (i) Starting with the first logical layer $j = 0$ of the circuit (top orange box), one routes as many gates as possible by a shortest-first routing (Sec. 4.3) as indicated by blue shaded gates (bottom orange box). This constitutes the first routed layer. Gates that cannot be routed anymore

because of path overlaps are pushed to the next logical layer. The logical structure of the upcoming layers is updated correspondingly. (ii) Having fixed the routes for the first routed layer, one looks for extensions of the paths to trees and corresponding teleportations of the control-to-ancilla or target-to-ancilla position, based on the next k logical lookahead layers of the circuit. If an optimized choice of tree extensions reduces the number of routed layers $\tilde{\ell} \geq k$, the teleportations are applied. (iii) Finally, one fixes the corresponding $\tilde{\ell}$ layers in the schedule. Then, the next logical layer $j' = j + k + 1$ is considered and the procedure is repeated until all gates of the logical circuit are routed. Note that if, in step (ii), no trees are found, one directly jumps to the next logical layer $j' = j + 1$ and repeats the procedure with step (i).

Teleportations optimized for the next k logical lookahead layers may temporarily create configurations in which certain logical data qubits become effectively disconnected from the routing space, as they are fully enclosed by other logical data patches. To permit such configurations—temporarily advantageous but harmful in the long run—idle data qubits are teleported during step (i) via standard lattice surgery teleportation into “gaps” generated when logical data qubits were moved in previous iterations. In this way, a large variety of CNOT + teleportation steps can be exploited while preserving a consistent layout structure in the long term, such as the pair layout shown in Fig. 1b.

4.2 Simulated Annealing for Tree Search

The search for trees in step (ii) is performed by applying simulated annealing [45]. The inputs for this routine are the current routed layer (routes of blue gates in (i)) and the k subsequent logical lookahead layers. A sample solution is initialized by extending the paths of the blue gates to random trees on the ancilla space. In addition, note that it is in principle also possible to place the ancilla along the path without explicitly extending the path to a tree and teleporting towards a patch *on* the path. The objective function to minimize is the number of routed layers $\tilde{\ell}$ of the next k logical lookahead layers, which is computed explicitly with the shortest-first routing (Sec. 4.3). Furthermore, a simulated annealing procedure requires the definition of a “neighborhood” to choose a new sample solution from. The neighborhood of a tree is defined by all variations of the tree in which the ancilla patch positions lie within a radius r of edges from the initial ancilla patch position on \mathcal{R} . One randomly teleports the control or target qubit to the new ancilla position. The simulated annealing procedure therefore searches tree structures based on the routed depth $\tilde{\ell}$ of the next k logical lookahead layers and attempts to find solutions which reduce $\tilde{\ell}$.

4.3 Shortest-First Routing Subroutine

The routing of k logical layers is performed in the same spirit as in [16]. Given a circuit of CNOT gates only, one can formalize the problem as *vertex disjoint path* (VDP) problem [46–48]. Based on a simple approximation scheme of the VDP problem [47], routing k layers of CNOT gates means, that given a logical layer, one begins by computing the shortest paths for all CNOT gates using Dijkstra’s algorithm. From those, one selects the shortest path and fixes it in the current routed layer and removes the used nodes from \mathcal{R} . The procedure is repeated until no further paths fit on \mathcal{R} . All routes

found in this procedure constitute a routed layer. If not all CNOT gates of the given logical layer could be routed, they are pushed in the next logical layer and all logical layers are updated correspondingly in the DAG. Then, the procedure is repeated for the next logical layer until all gates in the initially k logical layers are routed and turned into $\tilde{\ell}$ routed layers.

Note that this procedure can be greedily generalized to include T gates with a factory reset time t [16]. In the same spirit, the proposed optimization with CNOT + teleportation steps can be generalized to include T gates as well. Assume a T state injection scheme using a CNOT gate; this CNOT gate can also be used for a CNOT + teleportation step. The proposed optimization scheme with CNOT and T gates is available on GitHub <https://github.com/munich-quantum-toolkit/qecc>.

5 Performance Evaluation

This section evaluates the proposed compilation method with movable logical qubits and identifies regimes where it performs best.

For this comparison, we consider the application of the shortest-first routing described in Sec. 4.3 for the whole logical circuit as a standard routing baseline with static logical data patch locations, leading to routed depth \tilde{d}_{st} . This is compared to the proposed approach with teleportation-based movements. We denote the resulting routed depth as \tilde{d}_{opt} . The following proof-of-principle comparison is performed using random initial mappings and random CNOT circuits with varying numbers of gates per logical layer. Random initial mappings of data qubit labels are used to isolate the effect of the proposed method. Note that it is reasonable to assume that initial mapping optimization [8] is assumed to offer limited benefit for growing circuit depth.

The random circuits used here are constructed in such a way that they can vary in their ratio between number of logical CNOT gates G and their depth d_L . This means that one can choose a number g of randomly sampled CNOT gates per layer in the DAG such that one has in total $G = g \cdot d_L$ gates in the circuit.

The interplay between the density of the circuit (determined by g) and the density of the layout play a major role how the proposed method performs. The layout describes the ratio c between the data and total patches on the macroscopic routing graph and how they are placed, as for instance the pair layout in Fig. 1b. Further layouts are displayed in the left-most column of Tab. 1.

To study this interplay, let us fix the layout and the number of logical data qubits first and vary the value g with a fixed total number of gates G . For the optimization $k = 5$ logical lookahead

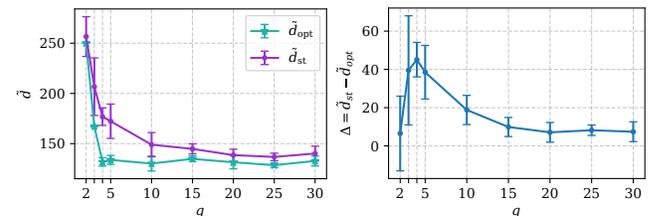


Figure 6: \tilde{d} (left) and $\Delta = \tilde{d}_{\text{st}} - \tilde{d}_{\text{opt}}$ (right) under variation of g with a fixed number of gates $G = 500$ and $q = 60$ logical data qubits in a triple layout. For each data point 10 random circuits were sampled.

Table 1: Comparison of scheduling performance across different layouts for $q = 120$ logical qubits. For each layout the smallest unit of data patches is displayed together with the asymptotic layout density c which is the ratio between the number of data patches and the total number of patches. A randomly sampled circuit type with $g = 8$ gates per logical layer was used. \tilde{d}_{st} and \tilde{d}_{opt} denote the routed depths of the standard approach and the proposed optimized approach using CNOT + teleportation steps, with both absolute values and a linear fit; d_L is the logical circuit depth. Values represent mean \pm standard deviation. For each entry 10 random circuits were sampled.

Layout	Logical Depth d_L	Standard Method \tilde{d}_{st}	Proposed Method \tilde{d}_{opt}	$\tilde{r} = \frac{\tilde{d}_{\text{opt}} - d_L}{\tilde{d}_{\text{st}} - d_L}$	$\Delta = \tilde{d}_{\text{st}} - \tilde{d}_{\text{opt}}$	$\frac{\Delta}{\tilde{d}_{\text{st}}}$
Single $c = \frac{1}{8}$ 	40	47.9 \pm 2.7	40.2 \pm 0.4	3.8 \pm 8.0%	7.7 \pm 2.8	15.8 \pm 5.1%
	80	95.4 \pm 3.7	80.3 \pm 0.5	2.4 \pm 3.8%	15.1 \pm 4.0	15.7 \pm 3.6%
	160	192.5 \pm 6.8	160.4 \pm 0.7	1.5 \pm 2.3%	32.1 \pm 7.1	16.6 \pm 3.1%
	320	384.5 \pm 13.1	320.7 \pm 0.8	1.2 \pm 1.3%	63.8 \pm 13.3	16.5 \pm 2.9%
	Fit:		$\tilde{d}_{\text{st}} = 1.2 d_L - 0.4$	$\tilde{d}_{\text{opt}} = 1.0 d_L + 0.1$		
Pair $c = \frac{1}{4}$ 	40	60.5 \pm 6.2	48.3 \pm 1.4	43.5 \pm 12.3%	12.2 \pm 5.9	19.4 \pm 7.5%
	80	119.5 \pm 9.3	97.5 \pm 2.3	46.2 \pm 11.0%	22.0 \pm 8.8	18.0 \pm 5.8%
	160	237.6 \pm 16.0	193.9 \pm 3.9	45.0 \pm 8.5%	43.7 \pm 14.2	18.1 \pm 4.8%
	320	473.6 \pm 31.0	385.6 \pm 7.1	43.9 \pm 7.6%	88.0 \pm 27.2	18.3 \pm 4.5%
	Fit:		$\tilde{d}_{\text{st}} = 1.5 d_L + 1.5$	$\tilde{d}_{\text{opt}} = 1.2 d_L + 0.8$		
Triple $c = \frac{3}{10}$ 	40	78.3 \pm 9.6	60.1 \pm 3.4	54.9 \pm 11.6%	18.2 \pm 8.1	22.3 \pm 8.0%
	80	156.5 \pm 17.4	120.3 \pm 5.2	54.4 \pm 9.1%	36.2 \pm 14.8	22.4 \pm 6.8%
	160	312.6 \pm 30.5	238.8 \pm 8.0	53.2 \pm 8.8%	73.8 \pm 26.4	23.0 \pm 6.3%
	320	625.1 \pm 56.6	480.0 \pm 15.7	53.7 \pm 7.6%	145.1 \pm 47.7	22.7 \pm 5.6%
	Fit:		$\tilde{d}_{\text{st}} = 2.0 d_L + 0.2$	$\tilde{d}_{\text{opt}} = 1.5 d_L - 0.1$		
Hex $c = \frac{3}{2}$ 	40	90.9 \pm 4.6	75.9 \pm 3.0	70.9 \pm 6.7%	15.0 \pm 4.4	16.4 \pm 4.2%
	80	181.0 \pm 8.6	151.0 \pm 4.7	70.7 \pm 6.1%	30.0 \pm 8.4	16.4 \pm 3.9%
	160	365.3 \pm 18.1	304.4 \pm 10.4	70.6 \pm 5.3%	60.9 \pm 15.3	16.5 \pm 3.4%
	320	731.0 \pm 40.4	609.7 \pm 18.0	70.9 \pm 5.1%	121.3 \pm 31.7	16.4 \pm 3.5%
	Fit:		$\tilde{d}_{\text{st}} = 2.3 d_L - 1.2$	$\tilde{d}_{\text{opt}} = 1.9 d_L - 1.0$		

layers were chosen with radius $r = 10$. In Fig. 6 one can see that the optimization performs best with rather low choices of g . If the circuit is too dense, then there are already too many paths on the routing graph once one attempts to find tree extensions for optimizations such that the benefit saturates. On the other hand, if the density of the circuit is too low, there is not enough parallelism to exploit such that the optimizations become negligible or unreliable.

This study allows us to fix g in a suitable regime for the proposed method. Varying the logical depth d_L and the layouts provides further information about the aforementioned interplay as displayed in Tab. 1. With $q = 120$ logical data qubits in total and $g = 8$, both the layouts and the logical depth were varied. One can observe that the relative layer overhead \tilde{r} is very low for the single layout, which is the sparsest layout. In other words, the proposed method can yield routed depths very close to the logical depth, which is the lower bound on the routed depth. With increasing layout density, also \tilde{r} grows, since parallel routing and tree search become more challenging. Without taking the relation to the logical depth into consideration, the relative layer reductions $\Delta/d_{\text{st}} = (\tilde{d}_{\text{st}} - \tilde{d}_{\text{opt}})/\tilde{d}_{\text{st}}$ are the highest for layouts with intermediate density, such as the triple layout.

Irrespective of the specific layout, the absolute difference $\Delta = \tilde{d}_{\text{st}} - \tilde{d}_{\text{opt}}$ increases with growing logical depth for any layout. This illustrates that consecutive teleportation-based movements of data patches can indeed lead to consecutive improvements throughout the whole circuit. This crucial observation is also underpinned by the lower slopes in the linear fits for \tilde{d}_{opt} than for \tilde{d}_{st} .

Overall, our results demonstrate that exploiting teleportation-based movements during lattice surgery compilation can yield significant reductions in routed depths compared to a standard routing with fixed data qubit positions.

6 Conclusion

Our work constitutes the first attempt to go beyond the place-and-route paradigm for quantum hardware architectures with static physical qubits and with focus on reducing schedule depth. We provided a proof-of-principle software implementation that exploits the inherent degree of freedom of teleporting the control or target logical during the logical lattice surgery CNOT gate. Furthermore, we explored the relevant trade-offs in the density of the layout and the logical circuit, identifying regimes in which significant reductions can be achieved.

Going forward, it will be useful to extend the evaluation of our software implementation to studies on specific quantum algorithms, or to adapt it to multi-qubit Pauli measurements [36]. Finally, as a promising direction for future work we envision that the proposed methods can be used to tackle growing schedule depths if defects are considered—thereby acting as the logical counterpart to existing proposals on the physical level [49, 50].

Acknowledgments

We thank Tom Peham for fruitful exchange regarding ZX calculus. L.H., L.B. and R.W. acknowledge funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (grant agreement No. 101001318) and Millenion (grant agreement No. 101114305). This work was part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, No. 563402549). A.K. acknowledges support from the NSF (QLCI, Award No. OMA-2120757), IARPA and the Army Research Office (ELQ Program, Cooperative Agreement No. W911NF-23-2-0219).

References

- [1] Peter W. Shor. "Fault-tolerant quantum computation". In: *Proceedings of 37th conference on foundations of computer science*. IEEE, 1996, pp. 56–65. doi: 10.1109/SFCS.1996.548464.
- [2] Andrew M. Steane. "Efficient fault-tolerant quantum computing". In: *Nature* 399.6732 (1999), pp. 124–126.
- [3] Peter W. Shor. "Scheme for reducing decoherence in quantum computer memory". In: *Physical Review A* 52.4 (1995), R2493–R2496. doi: 10.1103/physreva.52.r2493.
- [4] John Preskill. "Reliable quantum computers". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 385–410. doi: 10.1098/rspa.1998.0167.
- [5] A Yu Kitaev. "Quantum computations: algorithms and error correction". In: *Russian Mathematical Surveys* 52.6 (1997), p. 1191.
- [6] Jens Eisert and John Preskill. *Mind the gaps: The fraught road to quantum advantage*. 2025. doi: 10.48550/arXiv.2510.19928. arXiv: 2510.19928[quant-ph].
- [7] Alexander M. Dalzell et al. *Quantum algorithms: A survey of applications and end-to-end complexities*. 2023. arXiv: 2310.03011 [quant-ph].
- [8] Abtin Molavi et al. "Dependency-Aware Compilation for Surface Code Quantum Architectures". In: *Proceedings of the ACM on Programming Languages* 9.OOPSLA1 (2025), pp. 57–84. doi: 10.1145/3720416.
- [9] Michael Beverland, Vadym Kliuchnikov, and Eddie Schoute. "Surface Code Compilation via Edge-Disjoint Paths". In: *PRX Quantum* 3.2 (2022). Publisher: American Physical Society, p. 020342. doi: 10.1103/PRXQuantum.3.020342.
- [10] Daniel Herr, Franco Nori, and Simon J. Devitt. "Lattice surgery translation for quantum computation". In: *New J. Phys.* 19.1 (2017). Publisher: IOP Publishing, p. 013034. doi: 10.1088/1367-2630/aa5709.
- [11] Mingzheng Zhu et al. "Ecmas: Efficient Circuit Mapping and Scheduling for Surface Code". In: *2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. 2024, pp. 158–169. doi: 10.1109/CGO57630.2024.10444874.
- [12] L. Lao et al. "Mapping of lattice surgery-based quantum circuits on surface code architectures". In: *Quantum Sci. Technol.* 4.1 (2018). Publisher: IOP Publishing, p. 015005. doi: 10.1088/2058-9565/aadd1a.
- [13] George Watkins et al. "A High Performance Compiler for Very Large Scale Surface Code Computations". In: *Quantum* 8 (2024), p. 1354. doi: 10.22331/q-2024-05-22-1354. arXiv: 2302.02459[quant-ph].
- [14] Allyson Silva et al. "Multi-qubit Lattice Surgery Scheduling". In: *19th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2024)*. Vol. 310. 2024, 1:1–1:22. doi: 10.4230/LIPIcs.TQC.2024.1.
- [15] Alan Robertson, Haowen Gao, and Yuval R. Sanders. *A Resource Allocating Compiler for Lattice Surgery*. 2025. doi: 10.48550/arXiv.2506.04620. arXiv: 2506.04620[quant-ph].
- [16] Laura S. Herzog et al. *Lattice Surgery Compilation Beyond the Surface Code*. 2025. doi: 10.48550/arXiv.2504.10591. arXiv: 2504.10591[quant-ph].
- [17] H. Bombin. *An Introduction to Topological Quantum Codes*. 2013. doi: 10.48550/arXiv.1311.0277. arXiv: 1311.0277[quant-ph].
- [18] A.Yu. Kitaev. "Fault-tolerant quantum computation by anyons". In: *Annals of Physics* 303.1 (2003), pp. 2–30. doi: 10.1016/s0003-4916(02)00018-0.
- [19] Eric Dennis et al. "Topological quantum memory". In: *Journal of Mathematical Physics* 43.9 (2002), pp. 4452–4505. doi: 10.1063/1.1499754.
- [20] Austin G. Fowler et al. "Surface codes: Towards practical large-scale quantum computation". In: *Phys. Rev. A* 86.3 (2012), p. 032324. doi: 10.1103/PhysRevA.86.032324. arXiv: 1208.0928[quant-ph].
- [21] H. Bombin and M. A. Martin-Delgado. "Topological Quantum Distillation". In: *Phys. Rev. Lett.* 97.18 (2006), p. 180501. doi: 10.1103/PhysRevLett.97.180501. arXiv: quant-ph/0605138.
- [22] Aleksander Kubica. "The ABCs of the Color Code: A Study of Topological Quantum Codes as Toy Models for Fault-Tolerant Quantum Computation and Quantum Phases Of Matter". PhD thesis. 2018. doi: 10.7907/059V-MG69.
- [23] Aleksander Kubica and Nicolas Delfosse. "Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders". In: *Quantum* 7 (2023), p. 929. doi: 10.22331/q-2023-02-21-929.
- [24] Michael E. Beverland, Aleksander Kubica, and Krysta M. Svore. "Cost of Universality: A Comparative Study of the Overhead of State Distillation and Code Switching with Color Codes". In: *PRX Quantum* 2.2 (0), p. 020341. doi: 10.1103/PRXQuantum.2.020341.
- [25] Stergios Koutsoumpas et al. *Colour Codes Reach Surface Code Performance using Vibe Decoding*. 2025. doi: 10.48550/arXiv.2508.15743. arXiv: 2508.15743[quant-ph].
- [26] Ali Javadi-Abhari et al. *Quantum computing with Qiskit*. 2024. arXiv: 2405.08810 [quant-ph].
- [27] Andrew M. Childs, Eddie Schoute, and Cem M. Unsal. "Circuit Transformations for Quantum Architectures". en. In: 2019. doi: 10.4230/LIPIcs.TQC.2019.3.
- [28] Dominic Horsman et al. "Surface code quantum computing by lattice surgery". In: *New J. Phys.* 14.12 (2012). Publisher: IOP Publishing, p. 123011. doi: 10.1088/1367-2630/14/12/123011.
- [29] Austin G. Fowler and Craig Gidney. *Low overhead quantum computation using lattice surgery*. 2019. arXiv: 1808.06709 [quant-ph].
- [30] Christophe Vuillot et al. "Code deformation and lattice surgery are gauge fixing". In: *New J. Phys.* 21.3 (2019). Publisher: IOP Publishing, p. 033028. doi: 10.1088/1367-2630/ab0199.
- [31] Andrew J. Landahl and Ciaran Ryan-Anderson. *Quantum computing by color-code lattice surgery*. 2014. arXiv: 1407.5103 [quant-ph].
- [32] Lawrence Z. Cohen et al. "Low-overhead fault-tolerant quantum computing using long-range connectivity". In: *Sci. Adv.* 8.20 (2022), eabn1717. doi: 10.1126/sciadv.abn1717. arXiv: 2110.10794[quant-ph].
- [33] Alexander Cowtan. *SSIP: automated surgery with quantum LDPC codes*. 2024. doi: 10.48550/arXiv.2407.09423. arXiv: 2407.09423[quant-ph].
- [34] Andrew W. Cross et al. *Improved QLDPC Surgery: Logical Measurements and Bridging Surface Code Lattice Surgery Approaches*. 2025. doi: 10.48550/arXiv.2407.18393. arXiv: 2407.18393[quant-ph].
- [35] Nouédyne Baspin, Lucas Berent, and Lawrence Z. Cohen. *Fast surgery for quantum LDPC codes*. 2025. doi: 10.48550/arXiv.2510.04521. arXiv: 2510.04521[quant-ph].
- [36] Daniel Litinski. "A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery". In: *Quantum* 3 (2019), p. 128. doi: 10.22331/q-2019-03-05-128. arXiv: 1808.02892[cond-mat,physics:quant-ph].
- [37] Tyler LeBlond and Ryan S. Bennink. *Quantum Resource Comparison for Two Leading Surface Code Lattice Surgery Approaches*. 2025. arXiv: 2506.08182 [quant-ph].
- [38] Sam McArdle et al. *The Fast for the Curious: How to accelerate fault-tolerant quantum applications*. 2025. arXiv: 2510.26078 [quant-ph].
- [39] Craig Gidney, Noah Shutty, and Cody Jones. *Magic state cultivation: growing T states as cheap as CNOT gates*. 2024. doi: 10.48550/arXiv.2409.17595. arXiv: 2409.17595.
- [40] Matt McEwen, Dave Bacon, and Craig Gidney. "Relaxing Hardware Requirements for Surface Code Circuits using Time-dynamics". In: *Quantum* 7 (2023). Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, p. 1172. doi: 10.22331/q-2023-11-07-1172.
- [41] Sanaa Sharma and Prakash Murali. *Space-Time Optimisations for Early Fault-Tolerant Quantum Computation*. 2025. doi: 10.48550/arXiv.2511.08848. arXiv: 2511.08848[quant-ph].
- [42] Takumi Kobori et al. *LSQCA: Resource-Efficient Load/Store Architecture for Limited-Scale Fault-Tolerant Quantum Computing*. 2024. doi: 10.48550/arXiv.2412.20486. arXiv: 2412.20486[quant-ph].
- [43] John van de Wetering. *ZX-calculus for the working quantum computer scientist*. 2020. doi: 10.48550/arXiv.2012.13966. arXiv: 2012.13966[quant-ph].
- [44] Aleks Kissinger and John van de Wetering. *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation*. Preprint, 2024.
- [45] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680. doi: 10.1126/science.220.4598.671.
- [46] C De Bacco et al. "Shortest node-disjoint paths on random graphs". In: *Journal of Statistical Mechanics: Theory and Experiment* 2014.7 (2014), P07009. doi: 10.1088/1742-5468/2014/07/P07009.
- [47] Julia Chuzhoy and David H. K. Kim. "On Approximating Node-Disjoint Paths in Grids". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. 2015, pp. 187–211. doi: 10.4230/LIPIcs.APPROX-RANDOM.2015.187.
- [48] Julia Chuzhoy, David H. K. Kim, and Shi Li. "Improved approximation for node-disjoint paths in planar graphs". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 2016, pp. 556–569. doi: 10.1145/2897518.2897538.
- [49] Dripto M. Debroy et al. *LUCI in the Surface Code with Dropouts*. 2024. doi: 10.48550/arXiv.2410.14891. arXiv: 2410.14891.
- [50] Catherine Leroux et al. "Snakes and Ladders: Adapting the Surface Code to Defects". In: *PRX Quantum* 6.4 (2025), p. 040302. doi: 10.1103/815q-xjrb.