A Bridge-based Algorithm for Simultaneous Primal and Dual Defects Compression on Topologically Quantum-error-corrected Circuits*

Wei-Hsiang Tseng[1] and Yao-Wen Chang[1,2]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
whtseng@eda.ee.ntu.edu.tw; ywchang@ntu.edu.tw

ABSTRACT

Topological quantum error correction (TQEC) using the surface code is among the most promising techniques for fault-tolerant quantum circuits. The required resource of a TQEC circuit can be modeled as a space-time volume of a three-dimensional diagram by describing the defect movement along the time axis. For large-scale complex problems, it is crucial to minimize the space-time volume for a quantum algorithm with a reasonable physical qubit number and computation time. Previous work proposed an automated tool to perform bridge compression on a large-scale TQEC circuit. However, the existing automated bridging compression is only for dual defects and not for primal defects. This paper presents an algorithm to perform bridge compression on primal and dual defects simultaneously. In addition, the automatic compression algorithm performs initialization/measurement simplification and flipping to improve the compression. Compared with the state-of-the-art work, experimental results show that our proposed algorithm can averagely reduce space-time volumes by 47%.

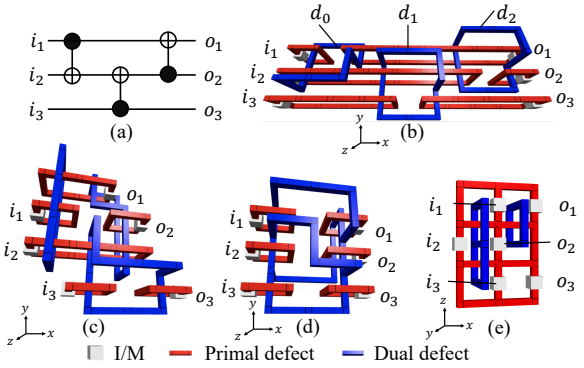

Figure 1: An example of a quantum circuit with three CNOT gates. (a) The ICM representation. (b) The canonical geometric description. (c) The compressed geometric description after topological deformation. (d) The optimized geometric description after bridge compression only on dual defects. (e) The optimized geometric description after bridge compression on primal and dual defects.

1 INTRODUCTION

Quantum computing has raised significant interest in recent years due to its potential capabilities in achieving substantial speedups on many complex problems (e.g., factorization [21] and unstructured database search [8]) that are currently intractable in classical computing. However, large-scale quantum computing on quantum devices is challenging mainly because of the decoherence caused by the noise from the environment. Decoherence causes loss of information from a system into the environment and may thus produce faulty results. Therefore, quantum devices urgently need fault-tolerant quantum circuits for better scalability and reliability.

With low failure probability, the topological quantum error correction (TQEC) scheme is among the most promising error-correcting techniques for fault-tolerant quantum computation [7]. The TQEC scheme operates on information encoded into topological cluster states in a three-dimensional (3D) lattice structure [6, 20]. In the surface code, physical qubits are placed on a two-dimensional (2D) surface. *Stabilizers* perform error correction of the quantum system. For each four neighboring physical qubits, a stabilizer measures and corrects the qubit state periodically. According to the basis for initialization and measurements, there are two kinds of stabilizers: Z stabilizer and X stabilizer. Here error correction is performed by Z and X stabilizers.

According to different encoding methods of logical qubits in the surface code, there are two primary type operations: braiding (defect-based encoding) [5] and lattice surgery (planar-based encoding) [9]. Here, we focus on braiding topological quantum circuits compression because the braiding technique is critical for applying modularization [3] to model the TQEC compression problem as a placement-and-routing problem. We detail the modularization in Section 2.3. *Defects* are specific contiguous regions where the stabilizers are turned off in the lattice. According to the type of stabilizers turned off, there are two types of defects, primal and dual defects, corresponding to stabilizers X and Z, respectively. Most encoded logical operations in the surface can be performed by defect movement. We can use a 3D diagram to model a TQEC circuit by describing the defect movement along the time axis by manipulating the defects in time. Note that a 3D diagram is composed of a 2D surface along the 1D time axis. A *Geometric description* is a 3D visual representation in the surface code, which consists of the qubit initialization/measurement (I/M), the defect configuration, and the positions of state injections and state distillation boxes [15]. We detail the geometric description in Section 2.1.

Furthermore, because the distance between defects is highly related to the error rate of a TQEC circuit, two disjoint defects cannot overlap with each other and are separated by one unit in this paper. We can use qubit initialization, CNOT gates, and measurements to decompose any fault-tolerant quantum circuit into the ICM representation [17]. Then, an ICM circuit can easily map to a canonical geometric description [18]. The space-time volume of the geometric description is the required resource of a TQEC circuit, which represents the number of physical qubits (space volume) and the required executing time steps for quantum operations (time volume).

In this paper, we use a TQEC circuit with three CNOT gates as an example. In the three CNOT gates cases, Figure 1(a) shows an ICM representation, and Figure 1(b) shows a canonical geometric description. After performing topological deformation, the result and canonical braids are topologically equivalent because the relationship between loops remains unchanged, i.e., the functionality of a TQEC circuit remains unchanged under any topological deformation [15].

The surface code has been studied for almost two decades, and still, the efficient automatic compilation of TQEC circuits is ongoing research. Paler *et al.* presented the first framework to synthesize an arbitrary quantum algorithm to a TQEC circuit [15] and then developed a general, systematic, and online method for synthesizing TQEC circuits with compact structures [16]. However, the two works [15, 16] did not consider effective volume optimization techniques. On the other hand, minimizing the number of lines or time steps in ICM models has recently attracted much attention, which can potentially reduce the space-time volume of the corresponding TQEC circuits. AlFailakawi *et al.* [2] minimized the depth of by developing a hybrid approach. Adnan and Yamashita [1] presented a two-dimensional qubit layout arrangement algorithm to reduce the time steps. However, these approaches only considered depth compression, so they may not minimize the space-time volume of a TQEC circuit.

Fowler and Devitt [5] presented a non-topological deformation: *bridge compression* to compactify a TQEC circuit. They provided the optimized results to show the effectiveness of bridge compression with manual efforts substantially. However, performing manual optimization is not realistic on large-scale TQEC circuits. Paetznick and Fowler [14] presented two compression algorithms (force-directed and simulated annealing (SA) based algorithms) for TQEC circuits. The force-directed algorithm smoothly pushes or pulls defect segments by the greedy method without destroying the braiding relationship of the primal and dual defects. Besides, they proposed an SA-based algorithm to avoid the local minima.

*This work was partially supported by AnaGlobe, ASE, Synopsys, TSMC, Maxeda, MOST of Taiwan under Grant MOST 109-2224-E-002-009, MOST 110-2221-E-002-177-MY3, MOST 110-2224-E-002-012, and MOST 111-2622-8-002-023-SB.

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530483

Lin *et al.* [11] proposed an automatic layout synthesis of TQEC circuits with 1-D and 2-D architectures. They first arranged the logical qubits in either 1D or 2D structures for primal defects and then optimized the number of time steps under topological deformation by selecting non-conflict dual-defect routing patterns. They solved a maximum weighted independent set problem to select non-conflict dual-defect routing patterns and proved the NP-hardness of the qubit routing problem. However, their approach only considered compression along the time axis, so it may not globally minimize the space-time volume of a TQEC circuit.

Hsu *et al.* [10] proposed the first work that automatically optimizes the space-time volume of TQEC circuits by bridge compression. They applied a bridge compression technique on dual defects to compact TQEC circuits with modularization and satisfy time-ordered measurement constraints. However, their solution quality could be limited because they only perform dual bridging compression, lacking primal bridging compression.

To implement large-scale quantum computations, it is desirable to develop an automated tool to optimize the space-time volume of TQEC circuits efficiently. Fowler and Devitt [5] showed the effectiveness of bridge compression with manual efforts, but not much work applied the bridge compression technique for automatic space-time volume minimization. To the best of our knowledge, current automated tools with the bridging compression technique [10] can only bridge dual defects. However, bridging dual defects alone already can averagely reduce space-time volumes by 83% in TQEC circuits. Additionally, there is a gap in the compression efficiency and freedom between [10] and [5] that we can improve in this paper. Although dealing with the conflicts of the braiding relationship of primal and dual defects caused by the bridging sequence is difficult, the bridging compression on primal and dual defects can compress TQEC circuits much more than bridging compression on dual defects alone.

Figure 1(b) shows a canonical geometric description with a volume of 54 (9×3×2) for the quantum circuit shown in Figure 1(a). Figure 1(c) shows a compressed circuit with a volume of 32 (4×4×2) after only topological deformation. Figure 1(d) shows an optimized circuit with a volume of 18 (3×3×2) after bridge compression only on dual defects. Figure 1(e) shows an optimized circuit with a volume of 6 (2×1×3) after bridge compression on primal and dual defects. In this example, we can observe that bridge compression on primal and dual defects has excellent potential to achieve much lower space-time volume than performing bridge compression on dual defects alone. Note that two disjoint defects are separated by one unit, and the volume calculation method is (#x×#y×#z) in this paper. Here #x, #y, and #z are the number of units on the x, y, and z axis, respectively.

We summarize our main contributions as follows:

- We present an automatical tool to optimize the space-time volume of TQEC circuits by bridge compression on primal and dual defects while considering time-ordered measurement constraints and state distillation boxes.
- We design a 2D primal-dual graph (PD graph) for recording the braiding relationship and simplifying the complex geometric description for TQEC circuits.
- We propose I-shaped simplification to simplify the specific I/M that is very effective and efficient for small-scale problems.
- We develop a flipping operation to construct primal bridge structures before bridging dual defects, unlike the previous work that performs bridge compression only on dual defects. The flipping operation avoids conflict issues between the primal and dual bridging.
- We propose a primal bridging super-module to be a new type of super-module in the module placement stage, converting the primal bridging structures of a PD graph to geometric description, reducing the problem size of the SA algorithm in the 2.5D placement stage, and planning the directionality of dual segments to improve the routing results.
- Experimental results show that our proposed algorithm can averagely achieve 47% space-time volume reduction compared with the state-of-the-art work [10].

The remainder of this paper is organized as follows. Section 2 introduces geometric descriptions, time-ordered measurement constraints, modularization, a primal-dual graph, and the bridging rule, and then formulates the TQEC circuit compression problem. Section 3 details the core techniques of our proposed algorithm. Section 4 shows and analyzes the experimental results, and Section 5 concludes this paper.

2 PRELIMINARIES

This section introduces geometric descriptions, time-ordered measurement constraints, the concept of modularization, the primal-dual graph, and the bridge compression technique. Finally, we formulate the TQEC circuit compression problem.

2.1 Geometric Descriptions

For visually representing a TQEC circuit, geometric description is a 3D visual representation consisting of the qubit initialization/measurement (I/M), the defect configuration, and the positions of state injections and state distillation boxes [15]. There are two kinds of bases (basis vectors of base): X-basis ($|+\rangle$ and $|-\rangle$) and Z-basis ($|0\rangle$ and $|1\rangle$). We use red cuboids and blue cuboids to represent primal defects and dual defects, respectively, and geometric components to represent qubit I/M [15], as shown in Figure 2.

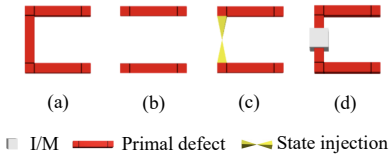

Figure 2: The components used in geometric descriptions. (a) Z-basis I/M for primal defects. (b) X-basis I/M for primal defects. (c) State injection to primal defects. (d) Generalized representation for initialization, measurement, and state injection.

In the TQEC scheme, we can implement a single-qubit rotation gate through teleportation with CNOT gates and logical ancillary states $|Y\rangle$ and $|A\rangle$ [15].
A logical ancillary states $|Y\rangle$:

$$|Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle). \tag{1}$$

A logical ancillary states $|A\rangle$:

$$|A\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle). \tag{2}$$

Note that we must prepare these ancillary states before injecting them into a circuit. We use distillation circuits to generate a single high-fidelity state from multiple low-fidelity ones to ensure state fidelity. In geometric descriptions, this paper uses green and yellow boxes to hold the place for $|Y\rangle$ and $|A\rangle$ state distillation circuits [15].

2.2 Time-ordered Measurement Constraint

For TQEC circuits, unlike most gates that are invariant under any topological deformation, the measurements of the T gate should obey a relative time ordering in the ICM representation. Therefore, two constraints come from the specific measurement ordering in a T gate or among T gates: intra-T gate constraint and inter-T gate constraint. Figure 3 illustrates the intra-T gate constraint. Figure 3(a) shows a T gate in the original quantum circuit, which can be decomposed into the ICM representation in Figure 3(b). In Figure 3(b), we need to perform the green Z-basis measurement before the other four blue selective teleportation measurements [17]. This paper defines the green Z-basis measurement as first-order, and four blue selective teleportation measurements are second-order measurements. Additionally, note that time goes from left to right. Figure 3(c) shows an accurate geometric description of the circuit in Figure 3(b), which meets the intra-T gate constraint because the first-order measurement is to the left of the second-order measurements. Figure 4 illustrates the inter-T gate constraint. Figure 4(a) shows a circuit with two T gates operating on the same qubit, which can be decomposed into the ICM representation in Figure 4(b). On the same qubit in the ICM representation, the second-order measurements of distinct T gates operation should obey a time ordering [17], the inter-T gate constraint. In the geometric description, we need to ensure that the first T gate's second-order measurements (the left dotted box) are to the left to the second T gate's second-order measurements (the right dotted box).

2.3 Modularization and Primal-dual Graph

We apply the modularization technique [3] to model the complicated TQEC compression problem as a placement-and-routing problem. The modules derived from the canonical form by breaking all dual nets into two-pin segments consist of a primal loop enclosing dual two-pin segments. Then, placing primal modules with volume optimization and routing all the dual segments to restore dual nets are a placement-and-routing problem. Note that the segments of a dual net can be reconfigured as long as the dual net can be restored. This paper performs the primal and dual bridging technique by merging the primal modules and combining the dual segments based on modularization. We apply the modularization technique to design the 2D primal-dual graph (PD graph), which records the braiding relationship between the primal modules and dual nets and simplifies the complex 3D geometric description for a TQEC circuit. Additionally, we directly perform

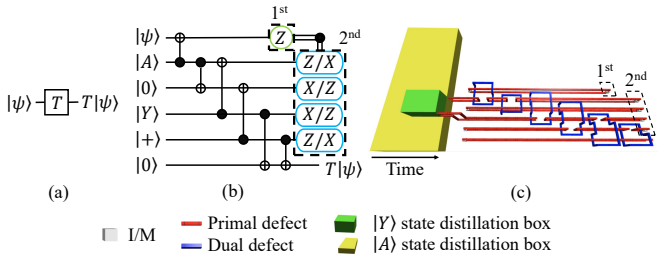

Figure 3: Illustration of the intra-T gate constraint. (a) A T gate in the original quantum circuit. (b) A T gate in the ICM representation with $|Y\rangle$ and $|A\rangle$ state injection . (c) The TQEC canonical form of a T gate with a $|Y\rangle$ box and an $|A\rangle$ box.

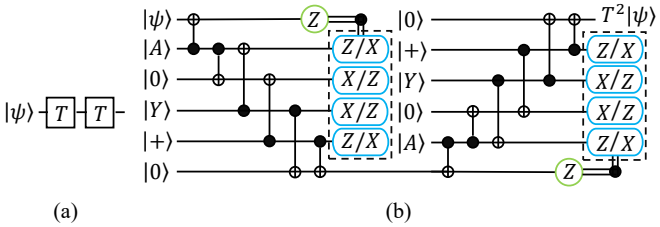

Figure 4: Illustration of the inter-T gate constraint. (a) An example circuit with two T gates operating on the same qubit. (b) The ICM representation of (a).

primal and dual bridging techniques on the PD graph. We detail converting the ICM representation into the PD graph in Section 3.1.

2.4 Bridge Compression

Fowler and Devitt [5] proposed the bridge compression technique to lower the TQEC computation overhead. We can only add a bridge between the two disjoint finite extent same-type defect structures. After adding a bridge between two primal (or dual) structures, we merge the two structures by a continuous common segment. We define the common segment from the segments of the two primal (or dual) structures, passing through the same dual (or primal) loops in identical order. Additionally, we can only add one bridge between two structures at most, using only one continuous common segment to merge the two structures. If a common segment between two structures is discontinuous, an extra loop would be produced, and the computation would be changed. Therefore, generating an extra loop is forbidden when we perform the primal and dual bridging stage. On the other hand, to achieve better compression results, we should bridge two structures with a longer continuous common segment, which means to share more volume of two structures. Unlike [10] that only adds a bridge between dual structures, we can bridge two disjoint primal/dual structures in this paper.

2.5 Problem Formulation

The TQEC circuit compression problem can be formally defined as follows:

Problem 1 (TQEC Circuit Compression Problem). *Given a quantum circuit synthesized from a quantum algorithm, minimize the space-time volume to generate an accurate 3D geometric description for TQEC computation, satisfy the time-ordered measurement constraints, and integrate state distillation boxes.*

3 PROPOSED ALGORITHMS

The space-time volume optimization of a TQEC circuit is a complicated problem because our algorithm needs to optimize the space-time volume, maintain the braiding relationships, inject the state distillation boxes, and consider the time-ordered measurement constraints. As a result, our proposed algorithm consists of seven major stages: (1) preprocess including gate decomposition, (2) PD graph including modularization, (3) I-shaped simplification, (4) flipping operation of primal bridging, (5) iterative dual bridging, (6) module placement, and (7) dual-defect net routing. Figure 5 shows the overall flow of our proposed algorithm. The following subsections detail each stage.

3.1 Process and PD Graph Generation

For TQEC computations, we apply the method [15] to decompose a quantum circuit into the ICM representation in the preprocess stage. In the three

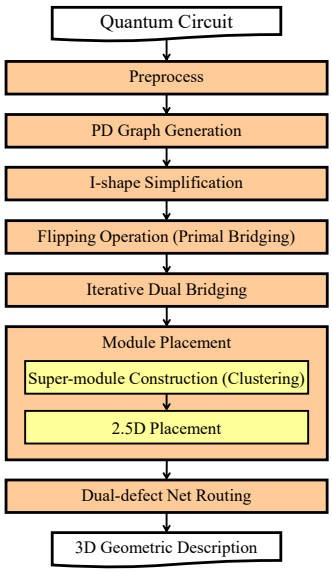

Figure 5: Overview of our algorithm.

CNOT gates case, Figure 6(a) shows its canonical geometric description, which illustrates the braiding relationship between primal and dual defects. We convert the relationship between the primal and dual defects from the ICM representation into the two-dimensional graph: primal-dual graph in Figure 6(b). The PD graph is stored in the data structure of Figure 6(d) in the order of gates. Figure 6(d) is a two-dimensional vector, and each element in this container is a primal module. Each module records which dual nets pass through it. A CNOT gate is composed of a dual net and three primal modules. The dual net passes through two primal modules on the control side, and it passes through one primal module on the target side. Therefore, we can generate rules for constructing Figure 6(d). On the control side, the first module that the dual net traversed is the current element, and the second module is an innovative element. On the target side, the module that the dual net traversed is the current element.

Next, we introduce how to record the PD graph in the data structure of the three CNOT gates case. First, we put the first CNOT gate with d_0 in the data structure. There is no current element in the first row (control side), so we add a p_0 primal module, which records the pass-through by d_0 dual net. Then, we add a p_1 primal module as an innovative element, which records the pass-through by d_0 dual net. There is no current element in the second row (target side), so we add a p_2 primal module, which records the pass-through by d_0 dual net. Then, we put the second CNOT gate with d_1 in the data structure. In the third row (control side), the process is the same as the first gate control side. There is a p_2 current element in the second row (target side), so we record the d_1 dual net in the p_2 current element. Finally, we put the third CNOT gate with d_2 in the data structure. There is a p_2 current element in the second row (control side), so we record the d_2 dual net in the p_2 current element. Then, we add a p_5 primal module as an innovative element, which records the pass-through by d_2 dual net. There is a p_1 current element in the first row (target side), so we record the d_2 dual net in the p_1 current element. After the above operation, the data structure is the same as Figure 6(d), and its geometric description is shown in Figure 6(c).

3.2 I-shaped Simplification

This subsection explains I-shaped simplification for specific I/M modules. When the I/M is on the control side's module of a CNOT gate, we can perform the simplification in Figure 7. In Figure 7(a), the two primal modules can correspond to p_0 and p_1, and the dual net can correspond to d_0 in the three CNOT gates case. p_0 and p_1 can combine into a new module in Figure 7(b).

Next, we use Figures 8 and 9 to illustrate the correctness of the simplification in Figure 7. In Figure 8(a), when the I/M is the X-basis I/M for a primal defect, p_0 is not a closed structure. We can simplify p_0 to get the result of Figure 8(b). In Figure 9(a), when the I/M is the Z-basis I/M for a primal defect, p_0 is a closed structure. Then, a bridge is added between p_0 and p_1 primal modules shown in Figure 9(b). Figure 9(c) shows the resulting bridging structure of two primal modules by merging the common segment. Figure 9(d) shows the result of maximally extending the common segment. Note that the d_0 dual net still passes through the original p_0 and p_1 primal modules here, but most p_0 and p_1 primal modules merged. When the I/M is

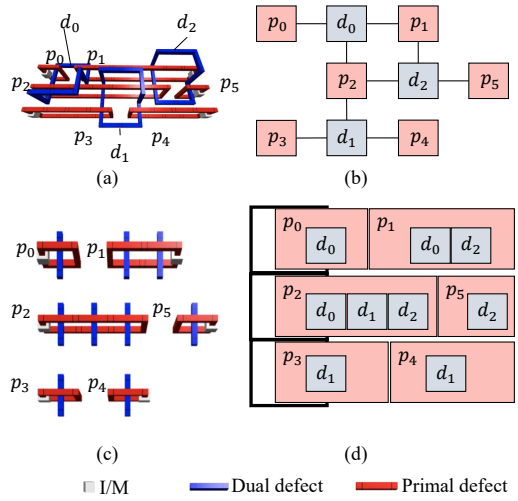

Figure 6: A three CNOT gates case of the PD graph generation. (a) The canonical geometric description. (b) The PD graph of the canonical geometric description. (c) The primal modules and dual nets derived from Figure 6(a). (d) The data structure recording the PD graph.

state injection to a primal defect, in this case, it is similar to that the I/M is the Z-basis I/M for the primal defect. Figures 8 and 9 illustrate the correctness of Figure 7.

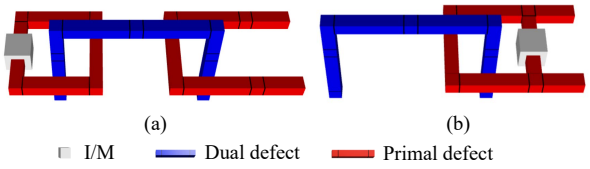

Figure 7: The I-shaped simplification of the I/M. (a) The original canonical geometric description. (b) The result of I-shaped simplification.

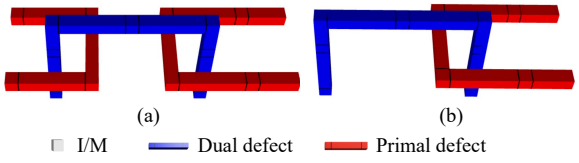

Figure 8: The I-shaped simplification of the X-basis I/M for primal defect. (a) The original canonical geometric description. (b) The result of I-shaped simplification.

Figure 10 illustrates that the algorithm performs I-shaped simplification on the three CNOT gates case. Figure 6(b) converts to Figure 10(a) after the I-shaped simplification. In Figure 10(a), d_0 is connected to p_0p_1 instead of p_0 because d_0 passes through p_0 and p_1, bridging to form a p_0p_1 module. Figure 10(b) is a geometric description of Figure 10(a). In Figure 10(b), the simplification of the dual net into a ring is like a letter I (so we call this process the I-shaped simplification). Although this simplification can only perform on specific I/M, it is very effective and fast for small-scale complex problems. We only need to check each I/M once and perform simplification when the conditions are met, so the time complexity of this algorithm is $O(n)$, where n is the number of I/M's.

3.3 Flipping Operation (Primal Bridging)

This subsection details the flipping operation of primal bridging; this operation is effective and essential for large-scale complex problems. Figure 11 shows the result of directly performing primal or dual bridging (see Figures 11(c) and 11(b)) after performing the I-shaped simplification on the three CNOT gates case. When primal bridging is executed, dual bridging is

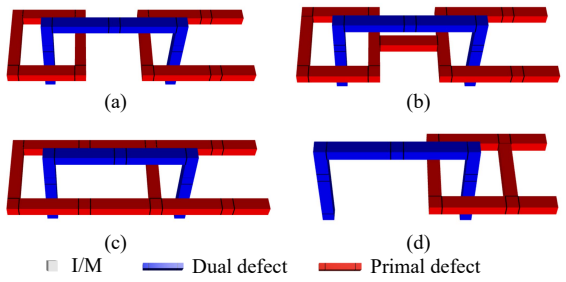

Figure 9: The I-shaped simplification of the Z-basis I/M for primal defect. (a) The original canonical geometric description. (b) The geometric description of adding a bridge to connect the two primal modules. (c) The primal bridge structure before extending the common segment. (d) The result of I-shaped simplification.

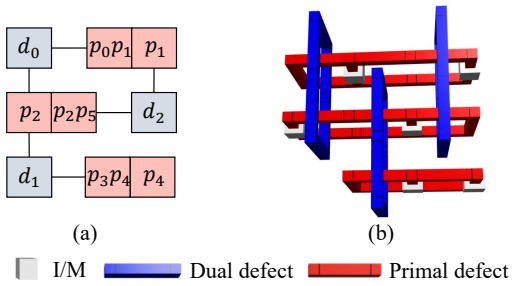

Figure 10: Illustration of the I-shaped simplification on the three CNOT gates case. (a) The PD graph after performing the I-shaped simplification. (b) The geometric description after performing the I-shaped simplification.

blocked and cannot be completed, vice versa. However, if we run the flipping operation first, bridging of primal and dual can be performed simultaneously, as shown in Figure 12(b). We flip the three primal modules to the same layer on the y axis in Figure 12(a). Note that this operation does not change the braiding relationship between dual nets and primal modules. The algorithm will perform the flipping operation first in the primal bridging stage, ensuring that primal bridging will not affect the possibility of dual bridging.

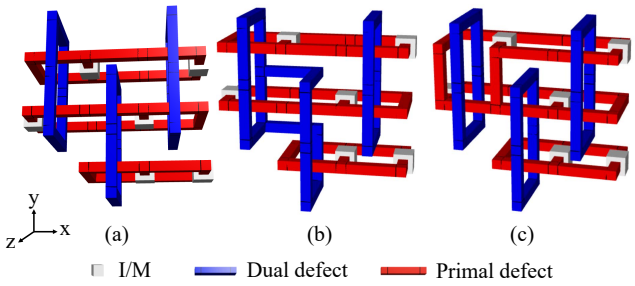

Figure 11: A three CNOT gates case of direct bridging. (a) The geometric description before bridging. (b) The geometric description after direct dual bridging. (c) The geometric description after direct primal bridging.

We present a greedy method to perform the primal bridging with the flipping operation. In Figure 12, to perform the primal bridging with the flipping operation, each primal module can be bridged with up to two primal modules on the z-axis (positive and negative directions). Therefore, this problem is converted into a connection problem between the primal modules via the dual net on the PD graph. In Figure 13(a), the algorithm will randomly select the starting point on an edge, and here the p_0p_1 module is chosen as the starting position. Since the p_0p_1 and p_1 modules are bridged together through the I-shaped simplification, they are regarded as the same point and considered together. Note that the primal module performs the I-shaped simplification on the x-axis and primal bridging with flipping operation on the z-axis, so they would not conflict with each other. The traversed modules are p_0p_1 and p_1. We have two choices: go through d_0 net to reach the p_2

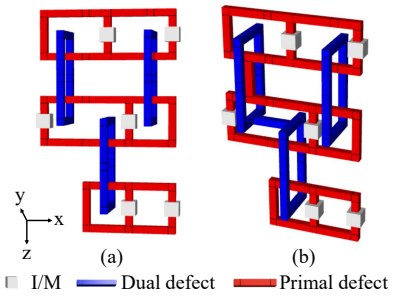

Figure 12: A three CNOT gates case of flipping operation. (a) The geometric description of the flipping operation before bridging. (b) The geometric description of the flipping operation after adding bridges.

module or go through d_2 net to get the p_2p_5 module. The greedy algorithm will choose the module which connects with the most dual nets. We can formulate the cost function Φ of the greedy method:

$$\Phi(M) = \max\left\{\sum_{n=1}^{d} D_n^M\right\}, \tag{3}$$

$$D_n^M = \begin{cases} 0, & \text{M has been traversed,} \\ 1, & \text{Otherwise,} \end{cases} \tag{4}$$

where d is the number of dual nets passing through the current module, and M represents the modules connected to the d dual nets. In this case, the greedy algorithm chooses the p_2 module, and the p_2p_5 module was added together by the I-shaped simplification. The traversed modules are p_0p_1, p_1, p_2, and p_2p_5. Having traversed the p_1 module, the algorithm goes through the d_1 net to reach the p_3p_4 module, and the p_4 module is added together by the I-shaped simplification. The greedy algorithm is then completed because it has traversed all modules. In other cases, if there are modules not been traversed, the greedy algorithm executes again until it traverses all modules. Figure 13(b) shows the PD graph after the greedy algorithm is completed.

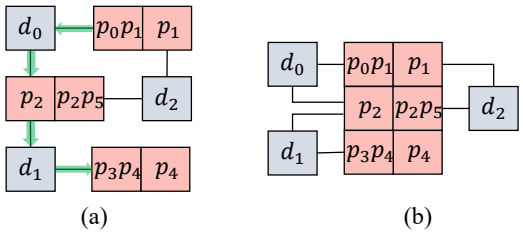

Figure 13: The flipping operation of primal bridging on the PD graph. (a) The order of primal modules bridging. (b) The result of the flipping operation on the PD graph.

3.4 Iterative Dual Bridging

After the primal bridging stage, the iterative dual bridging is performed to merge dual nets. Iterative dual bridging is similar to iterative bridging [10]; the difference is that the bridging module generated by the I-shaped simplification needs to consider. The green box in Figure 14(a) shows the original p_1 primal module range, which d_0 and d_2 dual nets cross. Therefore, d_0 and d_2 dual nets have the opportunity to bridge in the iterative bridging algorithm [10]. However, after the I-shaped simplification stage, d_0 dual net passes through p_0 and p_1 primal modules simultaneously. If d_0 and d_2 dual nets bridge in the original p_0 module, an error will occur. In addition, the original p_1 module separates d_0 and d_2 dual nets. The original p_1 module is split into two parts in the PD graph. One part forms a new p_0p_1 module with the p_0 module, representing the two modules' bridging part. The other part of the original p_1 module remains in the new p_1 module. Therefore, performing iterative bridging [10] in the PD graph forms iterative dual bridging, which can altogether avoid the results of d_0 and d_2 bridging. Only d_0 and d_1 dual nets pass through the p_2 module simultaneously in the PD graph. After the iterative dual bridging stage, the PD graph is shown in Figure 14(b).

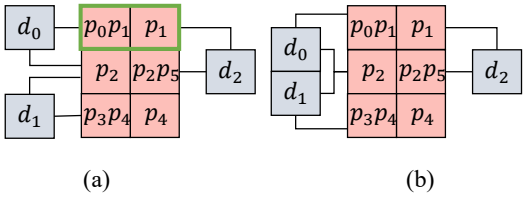

Figure 14: Illustration of iterative dual bridging. (a) The PD graph before performing iterative dual bridging. (b) The PD graph after performing iterative dual bridging.

3.5 Module Placement

In the placement stage, we need to place all the primal modules in the three-dimensional space, optimizing the total wirelength and routability. We not only need to consider time-ordered measurements and state distillation boxes but also primal bridging modules. We use the optimized distillation boxes that come from [5], where the volume of a $|Y\rangle$ box is 18 (3×3×2), and that of an $|A\rangle$ box is 192 (16×6×2). Similar to [10], both time-ordered measurements and distillation boxes for state injections should be placed in specific ordering along the time axis and are respectively categorized as a kind of super-module. In this work, the primal bridging module is also in the type of super-module. Therefore, there are three super-module types: *time-dependent super-module*, *distillation-injection super-module*, and *primal bridging super-module*. We use time-dependent and distillation-injection super-modules to handle the time-ordered measurement issue and the integration of state distillation boxes.

We detail the types of the primal bridging super-module below. The PD diagram records the bridging relationship between the primal modules (see Figure 14(b)). We need to convert the bridging relationship between the primal modules into an actual geometric description (see Figure 15(a)). Because the flipping operation bridges p_0p_1, p_2, and p_3p_4 modules in sequence, we place the three modules in line on the z-axis and simultaneously put their I-shaped simplification bridging modules on the x-axis. We can avoid conflicts by placing the bridging modules of the flipping operation and I-shaped simplification on different axes. In addition, this close-grid placement can ensure the common segment continuity and avoid generating extra loops on the primal bridging structure. On the other hand, there are five dual segments in Figure 15(a), whose direction is related to the routing result. When placing the modules on the z-axis, we need to *plan* whether the dual segment should flip. If there is no planning step, we might get poor routing results (see Figure 15(b)). Here we use a Boolean value f to record whether the dual segment has flipped in the module. The f value of the module placed at the beginning is zero, and its dual segment does not need to flip. We flip the dual segment based on the relationship between the primal module flipping operations. When the dual segment needs to flip, we proceed according to the f value of the source module:

$$f_{current} = 1 - f_{source} \tag{5}$$

Therefore, when converting a PD graph into an actual geometric description, we can decide whether to flip the dual segment based on the calculated f value. In the three CNOT gates case, we do not need to execute 2.5D placement because all primal modules are bridged into the same primal bridging super-module. In large-scale complex problems, we use the 2.5D B*-tree representation [4] for the modules and super-modules, where each node in the tree represents a module (or super-module). Besides, we use an SA engine to minimize the total volume and total estimated wirelength. Due to the primal bridging, we can drastically reduce the number of nodes in the 2.5D B*-tree. Therefore, the SA engine can generate better solutions more easily and quickly in large-scale complex problems.

3.6 Dual-defect Net Routing

In the routing stage, we should route all the dual segments to restore the dual nets. We use the A*-search algorithm to route each net within a restricted search region and the negotiation-based rip-up and reroute technique [12] to alleviate routing congestion. In the three CNOT gates case, we can get the result in Figure 1(e) after the dual-defect net routing stage.

4 EXPERIMENTAL RESULTS

We implemented our proposed algorithm in the C++ programming language with the Boost C++ libraries [13] and the Lemon graph library [19]. All experiments were performed on a Linux workstation with 4 Xeon 3.4 GHz CPUs with 64 GB memory. The experiments were conducted on the RevLib benchmarks [22], where the benchmark statistics is summarized in Table 1.

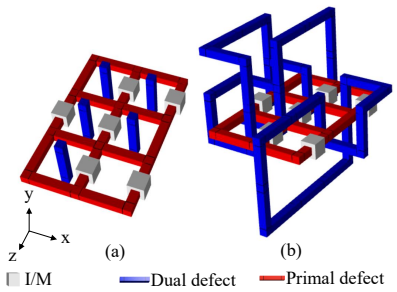

Figure 15: Examples of the geometric description results in the module placement stage. (a) The geometric description of the primal bridging structure. (b) The routing result without the planning step in the module placement stage.

Table 1: Benchmark statistics.

Benchmark	#Qubits	#CNOTs	#$\|Y\rangle$	#$\|A\rangle$	#Modules	#Nodes
4gt10-v1_81	131	168	42	21	362	18
4gt4-v0_73	257	341	84	42	724	360
rd84_142	897	1162	294	147	2500	1242
hwb5_53	1307	1729	434	217	3687	1853
add16_174	1394	1792	448	224	3857	1904
sym6_145	1519	1980	504	252	4255	2148
cycle17_3_112	1911	2478	630	315	5321	2744
ham15_107	3753	4938	1246	623	10560	5301

Table 2: Comparison of space-time volume for the previous work [11] and ours.

Benchmark	Canonical		[11] (1D)		[11] (2D)	
	Volume	Ratio	Volume	Ratio	Volume	Ratio
4gt10-v1_81	136836	6.553	98322	4.709	91116	4.364
4gt4-v0_73	535398	11.751	361152	7.927	327816	7.195
rd84_142	6287400	32.957	2805246	14.705	2744316	14.385
hwb5_53	13608294	29.215	9114828	19.568	8203548	17.612
add16_174	15028608	28.937	6449532	12.418	6173928	11.888
sym6_145	18103176	30.942	10720836	18.324	9852336	16.840
cycle17_3_112	28469700	21.444	19082448	14.373	16843884	12.687
ham15_107	111335928	30.495	69294822	18.980	63017484	17.260
Avg. Ratio		24.037		13.876		12.778

Table 3: Comparison of space-time volume for the previous work [10] and ours.

Benchmark	[10]			Ours		
	Volume	Ratio	Runtime (s)	Volume	Ratio	Runtime (s)
4gt10-v1_81	25520	1.222	15	20880	1	16
4gt4-v0_73	58696	1.288	26	45560	1	184
rd84_142	451440	2.366	262	190773	1	654
hwb5_53	1341704	2.880	447	465800	1	1295
add16_174	1069362	2.059	590	519350	1	941
sym6_145	1971840	3.370	793	585060	1	1538
cycle17_3_112	2354100	1.773	1402	1327656	1	1666
ham15_107	7331454	2.008	4901	3650985	1	4541
Avg. Ratio		2.121			1.000	

"#Qubits", "#CNOTs", "#$|Y\rangle$", and "#$|A\rangle$" denote the numbers of qubits, CNOT gates, $|Y\rangle$ ancillas, and $|A\rangle$ ancillas, respectively, after gate decomposition. "#Modules" denotes the numbers of modules before primal bridging, and "#Nodes" denotes the numbers of modules after primal bridging (i.e., the numbers of nodes in the 2.5D B*-tree). Therefore, we can calculate the problem size reduction by "#Modules" minus "#Nodes".

The space-time volume comparisons of our proposed algorithm and the previous works [11] and [10] are summarized in Tables 2 and 3 . Note that the TQEC circuits' canonical forms and the work [11] do not consider the placement of state distillation boxes, so their volumes in Table 2 are calculated by the synthesized TQEC circuits volumes directly plus the total state distillation boxes volume for a fair comparison. From the tables, our proposed algorithm can averagely achieve a 47.4% volume reduction over the work [10]. Additionally, our proposed algorithm can offer better compression efficiency on the large-scale benchmarks than the small-scale benchmarks. For the large-scale benchmarks, the work [10] in the 2.5D placement stage incurs too many nodes in the 2.5D B*-tree representation. In contrast, our proposed algorithm can bridge multiple primal modules to a primal bridging super-module, drastically reducing the number of nodes in a 2.5D B*-tree because either a primal module or a primal bridging super-module is represented as a node. Although primal bridging slightly reduces the local routability, it significantly reduces the common segment's volume and the SA problem size. Note that the runtime increases slightly due to the decreased local routability in the routing stage, i.e., taking more time to reach the estimated results of the module placement stage. Besides, our algorithm effectively bridges the primal modules by braiding the relationship of primal and dual defects (PD graph). As a result, our SA initial solution in the 2.5D placement stage is better than the work [10] because multiple primal modules traversed by the same dual net have been bridged in a super-module. The experimental results and above analysis well justify the effectiveness of our compression technique.

5 CONCLUSION

In this paper, we have presented a practical algorithm that optimizes space-time volumes of TQEC circuits by bridge compression on primal and dual defects while considering time-ordered measurement constraints and the integration of state distillation boxes. We have developed a 2D PD graph to record the braiding relationship and simplified the complex geometric description for TQEC circuits. We have proposed an effective and fast I-shaped simplification for simplifying the specific I/M. We have also developed a flipping operation that constructs primal bridge structures before bridging dual defects and avoids conflict issues between the primal and dual bridging. Under the placement framework, we have proposed a primal bridging super-module to be a new type of super-module that can convert the primal bridging structures of the PD graph to geometric description, reduce the problem size of the SA algorithm, and improve the routing results. Experimental results have shown that our proposed algorithm can compress much more space-time volume of TQEC circuits than previous works.

REFERENCES

[1] Nurul Ain Binti Adnan and Shigeru Yamashita. 2018. Logical qubit layout problem for ICM representation. *Journal of Information Processing* 26 (2018), 20–28.

[2] Mohammad AlFailakawi, Imtiaz Ahmad, Laila AlTerkawi, and Suha Hamdan. 2015. Depth optimization for topological quantum circuits. *Quantum Information Processing* 14, 2 (2015), 447–463.

[3] Kota Asai and Shigeru Yamashita. 2019. Compaction of Topological Quantum Circuits by Modularization. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences* 102, 4 (2019), 624–632.

[4] Paul Falkenstern, Yuan Xie, Yao-Wen Chang, and Yu Wang. 2010. Three-Dimensional Integrated Circuits (3D IC) Floorplan and Power/Ground Network Co-Synthesis. In *Proc. of ASPDAC*.

[5] Austin G Fowler and Simon J Devitt. 2012. A bridge to lower overhead quantum computation. *arXiv preprint arXiv:1209.0510* (2012).

[6] Austin G Fowler and Kovid Goyal. 2008. Topological cluster state quantum computing. *arXiv preprint arXiv:0805.3202* (2008).

[7] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. 2012. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A* 86, 3 (2012), 032324.

[8] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *ACM Symposium on Theory of Computing*.

[9] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. 2012. Surface code quantum computing by lattice surgery. *New Journal of Physics* 14, 12 (2012), 123011.

[10] Chen-Hao Hsu, Wan-Hsuan Lin, Wei-Hsiang Tseng, and Yao-Wen Chang. 2021. A Bridge-based Compression Algorithm for Topological Quantum Circuits. In *Proc. of DAC*.

[11] Yibo Lin, Bei Yu, Meng Li, and David Z Pan. 2017. Layout synthesis for topological quantum circuits with 1-D and 2-D architectures. *IEEE Tran. on CAD* 37, 8 (2017), 1574–1587.

[12] L. McMurchie and C. Ebeling. 1995. PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs. In *Proc. of FPGA*.

[13] The Boost organization. 2019. *The Boost C++ Libraries.* https://www.boost.org/users/history/version_1_72_0

[14] Adam Paetznick and Austin G Fowler. 2013. Quantum circuit optimization by topological compaction in the surface code. *arXiv preprint arXiv:1304.2807* (2013).

[15] Alexandru Paler, Simon J Devitt, and Austin G Fowler. 2016. Synthesis of arbitrary quantum circuits to topological assembly. *Sci. Rep.* 6, 1 (2016), 1–16.

[16] Alexandru Paler, Austin G Fowler, and Robert Wille. 2017. Synthesis of arbitrary quantum circuits to topological assembly: Systematic, online and compact. *Sci. Rep.* 7, 1 (2017), 1–16.

[17] Alexandru Paler, Ilia Polian, Kae Nemoto, and Simon J Devitt. 2015. A fully fault-tolerant representation of quantum circuits. In *Reversible Computation*.

[18] Alexandru Paler, Ilia Polian, Kae Nemoto, and Simon J Devitt. 2017. Fault-tolerant, high-level quantum circuits: form, compilation and description. *Quantum Science and Technology* 2, 2 (2017), 025003.

[19] Kovacs Peter, El-Kebir Mohammed, Juttner Alpar, and Dezso Balazs. 2017. *The LEMON Graph Library.* https://lemon.cs.elte.hu/trac/lemon

[20] Robert Raussendorf, Jim Harrington, and Kovid Goyal. 2007. Topological fault-tolerance in cluster state quantum computation. *New J. Phys.* 9, 6 (2007), 199.

[21] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.

[22] Robert Wille, Daniel Große, Lisa Teuber, Gerhard W Dueck, and Rolf Drechsler. 2008. RevLib: An online resource for reversible functions and reversible circuits. In *Proc. of ISMVL*.