# A Bridge-based Compression Algorithm for Topological Quantum Circuits

Chen-Hao Hsu[1], Wan-Hsuan Lin[2], Wei-Hsiang Tseng[1], and Yao-Wen Chang[1,2]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan
chhsu@eda.ee.ntu.edu.tw; b06901054@ntu.edu.tw; whtseng@eda.ee.ntu.edu.tw; ywchang@ntu.edu.tw

***Abstract*—The topological quantum error correction (TQEC) scheme is promising for scalable and reliable quantum computing. A TQEC circuit can be modeled by a three-dimensional diagram, and the implementation resource of a TQEC circuit is abstracted to its space-time volume. Implementing a quantum algorithm with a reasonable physical qubit number and reasonable computation time is challenging for large-scale practical problems. Therefore, minimizing the space-time volume of a TQEC circuit becomes a crucial issue. Previous work shows that bridge compression can greatly compress TQEC circuits, but it was performed only manually. It is desirable to develop automated compression techniques for TQEC circuits to achieve low-overhead, large-scale quantum computations. In this paper, we present the first work that can automatically perform bridge compression on TQEC circuits. Compared with the state-of-the-art method, experimental results show that our proposed algorithm can averagely reduce space-time volumes by 83%.**

## I. Introduction

Quantum computing has attracted much attention in recent years due to its capabilities in achieving substantial speedup on several classes of problems (e.g., factorization [1]) that are considered intractable in classical computing. However, large-scale quantum computing is challenging because quantum devices could suffer from significant noise from the environment and may thus produce faulty results. Therefore, fault-tolerant quantum circuits are needed for the scalability and reliability of quantum computing.

The topological quantum error correction (TQEC) scheme is promising for scalable fault-tolerant quantum computation [2]. Based on the Raussendorf code [3], quantum information is encoded into topological cluster states in a 3D lattice structure [4] consisting of physical qubits. Quantum computation is achieved by manipulating so-called defects, specific contiguous regions where the physical qubits are removed in the lattice [2]. According to the basis used for initialization and measurements, a defect is either primal ($X$-basis) or dual ($Z$-basis). A logical qubit is formed by a pair of same-type defects, and a logical controlled-NOT (CNOT) gate is implemented by braiding primal defects around dual defects. Furthermore, a quantum algorithm can be synthesized to a visual representation called *geometric description* in the surface code [2], which describes the qubit initialization/measurement (I/M), the defect configuration, and the positions of state injections and state distillation boxes [5]. Besides, we can convert any quantum circuits into the ICM representation [6] that consists of qubit (I)nitialization, (C)NOT gates, and (M)easurements, and an ICM circuit can be mapped to a canonical geometric description [7]. For example, Figure 1(a) shows an ICM circuit with three CNOT gates, and Figure 1(b) shows the corresponding canonical geometric description with three dual loops $l_1$, $l_2$, and $l_3$ braided around primal loops. The functionality of a TQEC circuit remains unchanged under any topological deformation, which means that the resulting braids are topologically equivalent to the canonical braids [5].

The required resource of a TQEC circuit is abstracted to its space-time volume of the geometric description. The space volume represents the 2D quantum hardware resources (i.e., the number of physical qubits) for quantum computing, and the time volume represents the required executing time steps for quantum operations. To solve difficult problems in the real world, minimizing the space-time volume for large-scale TQEC circuits becomes crucial.

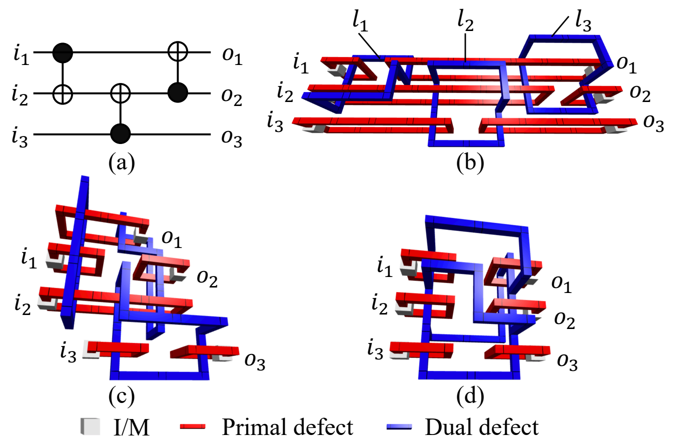

Fig. 1. An example of a quantum circuit with three CNOT gates. (a) The ICM representation. (b) The canonical geometric description. (c) The compressed geometric description after topological deformation. (d) The optimized geometric description after bridge compression.

During the past decade, many approaches to synthesize and optimize TQEC circuits have been proposed. Paler *et al.* [5] presented the first framework to synthesize a TQEC circuit from an arbitrary quantum algorithm. Paler *et al.* [8] developed a TQEC circuit synthesis method with compact structures while considering online scheduling. Both of the works [5], [8] focus on synthesizing general TQEC circuits and do not contain effective volume optimization techniques. Fowler and Devitt [9] presented a non-topological deformation called *bridge compression* to compactify a TQEC circuit with manual efforts substantially. Paetznick and Fowler [10] presented an efficient force-directed algorithm to minimize the space-time volume. Nevertheless, the forced-directed algorithm may be stuck in a local minimum, and the solution quality is highly related to the initial canonical geometric description. Lin *et al.* [11] proposed an efficient depth minimization technique for TQEC circuits with 1D and 2D qubit arrangements, which selects non-conflict dual-defect routing patterns by solving a maximum weighted independent set problem. They also proved the NP-hardness of the qubit routing problem in the layout synthesis of TQEC circuits. However, their approach only considers the depth compression (along the time axis), so the space-time volume of a TQEC circuit may not be globally minimized.

To lower the overhead of large-scale quantum computing, it is desirable to develop an automated tool to efficiently and effectively optimize the space-time volume of TQEC circuits. To the best of our knowledge, many studies focus on the optimization using topological deformation, but few works apply non-topological deformation such as bridge compression for automated space-time volume minimization,

This work was partially supported by AnaGlobe, ASE, Maxeda, Synopsys, TSMC, MOST of Taiwan under Grant MOST 108-2221-E-002-097-MY3, MOST 108-2911-I-002-544, and MOST 109-2224-E-002-009.

which can potentially compress TQEC circuits much more than the topological one. Figure 1(b) shows a canonical geometric description with a volume of 54 (9×3×2). Figure 1(c) shows a compressed circuit with a volume of 32 (4×4×2) after only topological deformation, and Figure 1(d) shows an optimized circuit with a volume of 18 (3×3×2) after bridge compression. In this example, we can observe that bridge compression has great potential to achieve much lower space-time volume than only performing topological deformation.

In this paper, we present a space-time volume optimization algorithm for TQEC circuits to realize low-overhead quantum computation. Our proposed algorithm applies bridge compression technique [9] to compactify TQEC circuits with the aid of the modularization proposed by Asai and Yamashita [12]. The main contributions of this paper are summarized as follows:

- We present the first work that automatically optimizes the space-time volume of TQEC circuits by bridge compression while considering state distillation boxes and time-ordered measurement constraints.
- We develop an iterative bridging algorithm to construct bridge structures, unlike the previous work that performs bridge compression with manual efforts.
- We propose a time-ordering-aware 2.5D placement for compaction of TQEC circuits and the satisfaction of time-ordered measurement constraints.
- We propose friend net-aware routing to reduce the required routing resource under topological deformation effectively.
- Experimental results show that our proposed algorithm can averagely achieve 83% space-time volume reduction compared with the state-of-the-art method [11].

The remainder of this paper is organized as follows. Section II introduces state distillation boxes, time-ordered measurement constraints, modularization, and the bridging rule, and then formulates the TQEC circuit compression problem. Section III details the core techniques of our algorithm. Section IV shows the experimental results, and Section V concludes this paper.

## II. Preliminaries

This section briefly introduces state distillation boxes in TQEC circuits, time-ordered measurement constraints, the concept of modularization, and the bridge compression technique. Then, we formulate the TQEC circuit compression problem.

### A. State Distillation Box

In the TQEC scheme, a single-qubit rotation gate can be implemented through teleportation with CNOT gates and logical ancillary states $|Y\rangle$ and $|A\rangle$ [5]. These ancillary states must be prepared before injected into the circuit. Additionally, distillation circuits are used to generate a single high-fidelity ancillary state from multiple low-fidelity ones. In this paper, we use a box to hold the place for a distillation circuit in geometric descriptions [5].

### B. Time-ordered Measurement Constraint

For TQEC circuits, most gates are invariant under any topological deformation. However, the measurements of certain gates (e.g., T gate) should obey a relative time ordering in the ICM representation. Figure 2(a) shows a T gate in the ICM representation, where the topmost $Z$-basis measurement must be performed before the other four selective teleportation measurements [6]. Figure 2(b) shows a valid geometric description of the circuit in Figure 2(a), which meets the time-ordered measurement constraint. Note that in this paper, time goes from left to right.

Furthermore, the selective teleportation measurements of distinct T gates operating on the same qubit in the ICM representation should also obey a time ordering [6]. The selective teleportation measurements of a T gate must be performed after those of the previous T gate are performed. Figure 2(c) shows a circuit with two T gates operating on a qubit, and the corresponding ICM representation is shown in Figure 2(d). To transform the circuit into a geometric description, we need to ensure that the selective teleportation measurements of the first T gate (the left dotted box) must be performed before the selective teleportation measurements of the second T gate (the right dotted box) are performed.

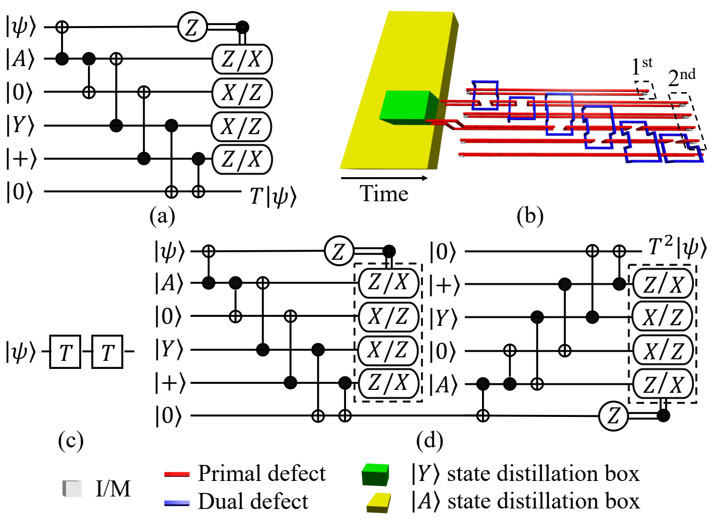

Fig. 2. (a) A T gate in the ICM representation with $|Y\rangle$ and $|A\rangle$ state injection . (b) The TQEC canonical form of a T gate with a $|Y\rangle$ box and an $|A\rangle$ box. (c) An example circuit with two T gates operating on the same qubit. (d) The ICM representation of (c).

### C. Modularization

Asai and Yamashita [12] proposed modularization that transforms the complicated TQEC compression problem into a placement-and-routing problem. Modules of a TQEC circuit are derived from the canonical form by breaking all dual loops into some two-pin nets. The parts of dual loops penetrating a primal loop are kept in modules to preserve the braiding information, so a module consists of a primal loop enclosing dual segments. For instance, Figure 3(a) shows six modules and nine nets derived from the canonical form in Figure 1(b) by breaking all dual loops, where $m_i$ denotes the $i^{th}$ module. Figure 3(b) shows the labels of pins in each module, where $p^i_{j,k}$ denotes the $k^{th}$ pin of the $j^{th}$ dual segment enclosed by module $m_i$. After placing modules with volume optimization, all the dual nets will be routed to restore dual loops. Note that the nets of a dual loop can be reconfigured as long as the dual loop can be restored. For example, both $\{(p^2_{3,2}, p^5_{1,2}), (p^5_{1,1}, p^4_{2,1}), (p^4_{2,2}, p^2_{3,1})\}$ and $\{(p^2_{3,2}, p^5_{1,2}), (p^5_{1,1}, p^4_{2,2}), (p^4_{2,1}, p^2_{3,1})\}$ are valid net sets for $l_3$ in Figure 3.

### D. Bridge Compression

The bridge compression technique is proposed to lower the TQEC computation overhead by Fowler and Devitt [9]. A bridge can only be added between two disjoint finite extent same-type defect structures. After adding a bridge, the two structures are merged by the continuous common segment, defined as the segments of the two structures that pass through the same loops in the identical order. To achieve better circuit compaction, we should bridge two structures with a longer continuous common segment. Figure 4(a) shows an initial TQEC circuit. To obtain the longest continuous common segments, we can topologically deform the TQEC circuit, as shown in Figure 4(b). Then, a bridge is added between the two dual loops (structures) shown in Figure 4(c), and Figure 4(d) shows the resulting bridge structure consisting of two dual loops by merging the common segment.

Note that we can add only one bridge between two structures; that is, the two structures would be merged by only one continuous segment. Otherwise, an extra loop would be induced, which will change the computation and thus is forbidden. For example, Figure 4(e) shows a wrong bridge structure from Figure 4(a) because there are two merged

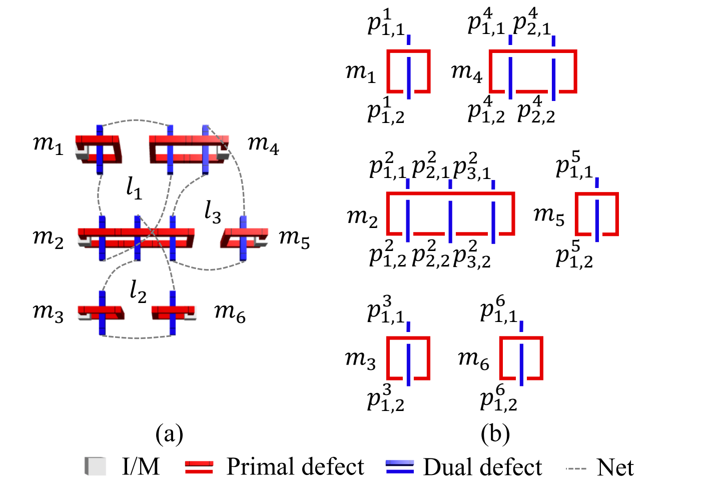

Fig. 3. (a) The modules and nets derived from Figure 1(b). (b) The labels of pins in each module.

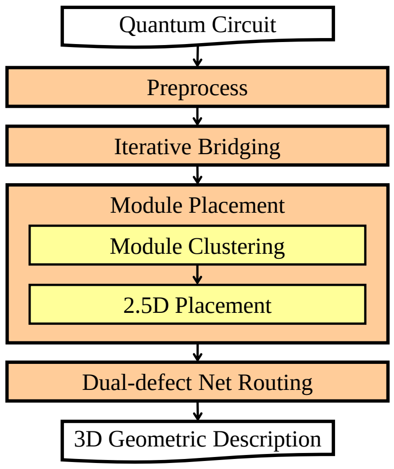

Fig. 5. Overview of our algorithm.

common segments, and an extra dual loop is thus induced in the middle. Although we can bridge two disjoint primal/dual structures, however, we only add a bridge between dual structures to simplify and tackle the TQEC circuit compression problem.

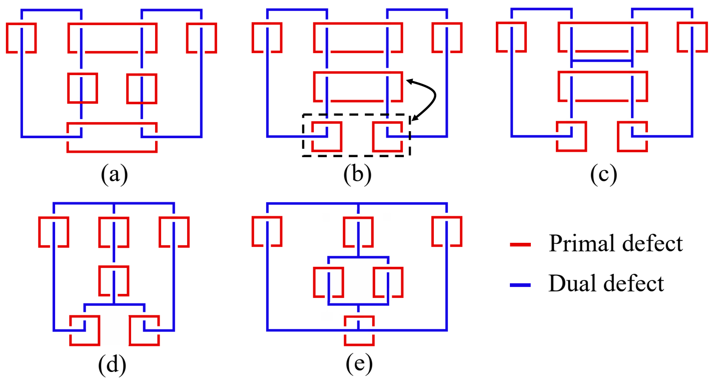

Fig. 4. (a) The initial TQEC circuit. (b) The circuit after topological deformation. (c) A bridge added between two dual loops. (d) A dual bridge structure consisting of two dual loops derived from (a). (e) A wrong bridge structure.

## *E. Problem Formulation*

We formally define the TQEC circuit compression problem below:

*Problem 1 (TQEC Circuit Compression Problem):* Given a quantum circuit synthesized from a quantum algorithm, generate a 3D geometric description for TQEC computation such that the space-time volume is minimized while the time-ordered measurement constraints are satisfied and state distillation boxes are integrated.

## III. Proposed Algorithms

The space-time volume optimization of TQEC circuits is complicated due to the maintenance of braids, the integration with state distillation boxes, and the time-ordered measurement constraints. Therefore, we propose an algorithm to optimize the space-time volume of TQEC circuits by bridge compression and topological deformation with the aid of modularization [12]. Our proposed algorithm consists of four major stages: (1) preprocess including gate decomposition and modularization, (2) iterative bridging, (3) module placement, and (4) dual-defect net routing. Figure 5 shows the overall flow of our proposed algorithm. The following subsections detail each stage.

## *A. Preprocess*

Each quantum computing architecture supports a specific gate set for universal computations. For TQEC computations, we apply the method [5] to decompose the gates of an arbitrary quantum circuit into a list of CNOT gates and ancillas (i.e., ICM representation). Next, the ICM circuit is mapped to the canonical geometric description. According to the canonical form, a TQEC circuit is decomposed into modules and nets by modularization.

## *B. Iterative Bridging*

After the preprocessing stage, the iterative bridging is performed to merge dual loops into bridge structures by adding bridges. A bridge structure is composed of several bridged dual loops. For example, there is one bridge structure in Figure 4(d). During the bridging process, each dual loop maintains a set of *chains* for the flexibility of bridging. Each chain is a sequence of consecutive pins, and the starting pin and the ending pin are both called *endpoint pins*. Initially, for every loop, the two pins of each penetrated module form a chain. After a loop is merged to a bridge structure, the chains of loops in the bridge structure may be extended since they form a continuous common segment with the merged loop. Besides, a loop in a bridge structure is said to be *reconstructable* if its chains can be restored to a single loop by connecting all of its chains. Moreover, if two dual loops penetrate the same module, the module is called a *common module* of the two loops. For example, in Figure 3(a), module $m_4$ is a common module of $l_1$ and $l_3$ since both dual loops penetrate $m_4$. Furthermore, if loop $l_i$ has at least one common module with loop $l_j$, then $l_i$ is called a *relative loop* of $l_j$ and vice versa.

Algorithm 1 shows our proposed iterative bridging algorithm to iteratively add bridges and merge dual loops into a bridge structure. First, all dual loops are pushed into set $O$ and marked as unprocessed (line 1). Once a dual loop is merged to a bridge structure, it will be removed from $O$ and marked as processed. In each iteration, we select an unprocessed loop $l_i$ from $O$ as the initial bridge structure $b$ (line 4) and push the unprocessed relative loops of $l_i$ into the max-priority queue $Q$ (lines 5–6), which are candidate loops that could be merged to $b$. The priority of a loop in $Q$ is determined by its number of common modules with $b$ since we tend to merge a loop into a bridge structure with a potentially longer continuous common segment. Moreover, the loops in $Q$ are extracted sequentially until $Q$ is empty (lines 8–9), and we check whether the extracted loop $l_e$ can be merged to $b$ (lines 10–12). If $l_e$ fails to be merged to $b$, it would never be pushed into $Q$ in the current iteration. On the other hand, if $l_e$ can be successfully merged to $b$ (line 13), the chains of loops in $b$ will be updated according to the continuous common segment of $b$ and $l_e$ (line 14). Then, $l_e$'s unprocessed relative loops are pushed into $Q$ as merged candidates (line 15). Besides, the keys to loops in $Q$ would be updated accordingly since $l_e$ is merged to $b$ (line 16). Finally, $l_e$ would be removed from $O$ and marked as processed (line 17).

**Algorithm 1** *IterativeBridging(D)*

```
Input: $D$: the set of dual loops.
Output: $B$: the set of bridge structures.
 1: Push all dual loops in $D$ into set $O$
 2: $B \leftarrow \emptyset$
 3: while $O$ is not empty
 4:     Select loop $l_i$ from $O$ as initial bridge structure $b$
 5:     Initialize max-priority queue $Q$
 6:     Push the unprocessed relative loops of $l_i$ into $Q$
 7:     Remove $l_i$ from $O$ and mark $l_i$ processed
 8:     while $Q$ is not empty
 9:         Extract loop $l_e$ from $Q$
10:         Construct bridge graph $G_{b,l_e}$
11:         Determine the order of critical vertices
12:         Perform path finding for critical vertices
13:         if a valid path exists in $G_{b,l_e}$
14:             Merge $l_e$ into $b$ and update the chains of loops in $b$
15:             Push the unprocessed relative loops of $l_e$ into $Q$
16:             Update the keys to loops in $Q$
17:             Remove $l_e$ from $O$ and mark $l_e$ processed
18:     $B \leftarrow B \cup b$
19: return $B$
```

Loop $l_e$ could be merged to bridge structure $b$ if we can find a continuous common segment that penetrates all common modules of $b$ and $l_e$, and the continuous common segment cannot destroy the reconstructability of all loops in $b$. To find a continuous common segment of $b$ and $l_e$, we construct a bridge graph $G_{b,l_e} = (V, E)$. The bridge graph construction consists of two steps: vertex construction and edge construction.

*1) Vertex Construction:* First, the pins of common modules between $b$ and $l_e$ are vertices in $G_{b,l_e}$. Second, if two chains belonging to different loops in $b$ share a common endpoint pin, then the endpoint pin is also a vertex in $G_{b,l_e}$. Note that for a bridge structure, we only need to consider one dual segment in a module. That is, each module contains only two pins for a bridge structure, so we use $m'_i$ to denote a module instead of $m_i$ in $b$. The vertex derived from pin $p'^i_j$ is denoted by $v^i_j$, where $p'^i_j$ denotes the $j^{th}$ pin of module $m'_i$ in $b$.

*2) Edge Construction:* First, if $p^{i_1}_{j_1}$ and $p^{i_2}_{j_2}$ are endpoints of different chains within a loop, then edge $e(v^{i_1}_{j_1}, v^{i_2}_{j_2})$ is constructed in $G_{b,l_e}$. Second, for each pair of consecutive pins $p^{i_3}_{j_3}$ and $p^{i_4}_{j_4}$ in a chain, then edge $e(v^{i_3}_{j_3}, v^{i_4}_{j_4})$ would be constructed if both $v^{i_3}_{j_3}$ and $v^{i_4}_{j_4}$ exist in $G_{b,l_e}$.

To merge $l_e$ to $b$, the continuous common segment should penetrate all common modules of $l_e$ and $b$. It implies that the vertices derived from the pins of common modules should be connected in series, and such vertices are called *critical vertices*. Therefore, our target is to find a path passing through all critical vertices in a specific order while the path does not destroy the reconstructability of each loop in $b$. Actually, the path indicates the continuous common segment for bridging $b$ and $l_e$. Before path searching, we need to determine the connecting order of critical vertices. A valid connecting order can be obtained by following the two pin vertices of each common module sequentially. For example, for two common modules $m'_1$ and $m'_2$, the order $\langle v^1_1, v^1_2, v^2_1, v^2_2 \rangle$ is valid, but the order $\langle v^1_1, v^2_1, v^2_2, v^1_2 \rangle$ is invalid due to the discontinuity of the pin vertices of $m'_1$ (i.e., $v^1_1$ and $v^1_2$).

After the connecting order is determined, we perform path searching on $G_{b,l_e}$ in order to find a path that follows the specific vertex order. Once a path is found in $G_{b,l_e}$, we will check if it is a valid path defined as a path that does not destroy the reconstructability of the loops in $b$. If a valid path is found, then $l_e$ would be merged to $b$, and all the chains associated with the path would be updated. For edges in the valid path, if it connects two disjoint chains within a loop in $b$, the chains would be connected as one chain. The continuous common segment becomes a chain of $l_e$.

The whole iterative bridging process terminates when all dual loops are processed (i.e., $O$ is empty). After all the bridge structures are constructed, we generate dual-defect nets by reconstructing all the loops in bridge structures, where all the nets should be connected in the following routing stage. Note that no duplicate nets will be generated.

Figure 6 shows an example of performing iterative bridging on the circuit in Figure 1(b). First, $l_1$ is selected as initial bridge structure $b$, and the chain set of $l_1$ is $\{p'^1_1–p'^1_2, p'^2_1–p'^2_2, p'^4_1–p'^4_2\}$. Because $l_2$ and $l_3$ respectively share one common module ($m'_2$) and two common modules ($m'_2$ and $m'_4$) with $l_1$, they are pushed into the max-priority queue $Q$ with keys 1 and 2 respectively. Then, extracting $l_3$ from $Q$, we construct the bridge graph $G_{b,l_3}$ for $b$ and $l_3$ as shown in Figure 6(b) to check if $l_3$ can be merged to $b$. For vertex construction, because $m'_2$ and $m'_4$ are common modules between $b$ and $l_3$, thus $v^2_1$, $v^2_2$, $v^4_1$, and $v^4_2$ are constructed. For edge construction, because $v^2_1$ and $v^4_1$ are vertices deriving from endpoint pins of different chains within $l_1$, $e(v^2_1, v^4_1)$ is constructed. Similarly, $e(v^2_1, v^4_2)$, $e(v^2_2, v^4_1)$, and $e(v^2_2, v^4_2)$ are also constructed. Besides, $e(v^2_1, v^2_2)$ is constructed since $v^2_1$ and $v^2_2$ are two consecutive pins in a chain, and similarly, $e(v^4_1, v^4_2)$ is constructed as well. Now, suppose that the connecting order is $\langle v^2_2, v^2_1, v^4_2, v^4_1 \rangle$, a valid path for continuous common segment is shown in Figure 4(b). We can see that after bridging $b$ and $l_3$ with the continuous common segment, the reconstructability of the loops in $b$ is not destroyed because both $l_1$ and $l_3$ can be restored to a single loop by connecting all chains in the chain set, as shown in Figure 6(c). Furthermore, after $l_3$ is merged to $b$, the chain set of $l_1$ is updated to $\{p'^4_1–p'^4_2–p'^2_1–p'^2_2, p'^1_1–p'^1_2\}$, and that of $l_3$ is $\{p'^4_1–p'^4_2–p'^2_1–p'^2_2, p'^5_1–p'^5_2\}$. Repeating the above steps for merging $l_2$ to $b$, the bridge graph $G_{b,l_2}$ is constructed, as shown in Figure 6(d). Note that $v^4_2$ is constructed because $p'^4_2$ is the common endpoint of the two chains belonging to $l_1$ and $l_3$. Finally, eight dual-defect nets are generated by reconstructing all the loops in $b$, as shown in Figure 6(e).

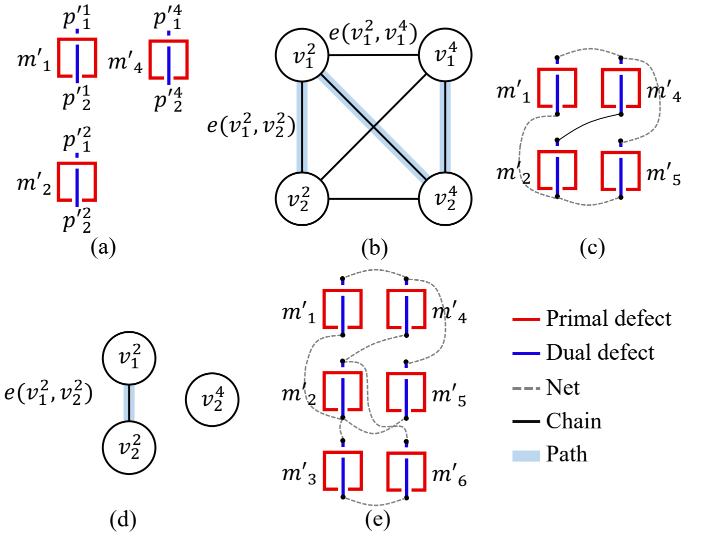

Fig. 6. An example of performing iterative bridging on the TQEC circuit in Figure 1(b). (a) The initial bridge structure $b$. (b) The bridge graph $G_{b,l_3}$. (c) The resulting bridge structure after adding $l_3$. (d) The bridge graph $G_{b,l_2}$. (e) The resulting bridge structure after adding $l_2$.

## C. Module Placement

In the placement stage, we need to place all the modules in 3D space with the optimization of total wirelength and routability while considering time-ordered measurements and state distillation boxes for universal computations. In this work, we use the optimized distillation boxes obtained in [9], 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). $|Y\rangle$ and $|A\rangle$ state distillation boxes are regarded as modules and should be placed as well. First, we cluster some modules into a *super-module* for state injections and time-ordered measurements. Next, we perform 2.5D placement while considering wirelength, routability, and time-ordered measurement constraints. The two major steps in this stage are detailed as follows.

*1) Module Clustering:* Both time-ordered measurements and distillation boxes for state injections should be placed in certain ordering along the time axis. Therefore, we propose module clustering to handle the time-ordered measurement issue and the integration of state distillation boxes. There are two types of super-modules: time-dependent super-module and distillation-injection super-module. We detail each type of super-modules below.

- **Time-dependent super-module**: The T gate measurements in the ICM representation should be performed in a specific time ordering, as mentioned in Section II-B. Therefore, we cluster the modules associated with the five measurements of a T gate into a time-dependent super-module. To meet the time-ordered constraint of T gate measurements, the module associated with the first $Z$-basis measurement must be placed on the left side of the four modules associated with the four selective teleportation measurements of the T gate. Besides, the four modules are placed vertically and aligned by their right boundaries. Figure 7(a) shows an example of the time-dependent super-module for a T gate, where the module on the left is associated with the first $Z$-basis measurement, and the four modules on the right are associated with the four selective teleportation measurements of the T gate.
- **Distillation-injection super-module**: Ancillary states $|Y\rangle$ or $|A\rangle$ should be prepared before injected into the circuit. Although a state distillation box can be placed at any position ahead of the injected module in the time axis, we cluster a state distillation box and its injected module into a distillation-injection super-module by directly connecting them to reduce the required primal-defect routing resource between them and meet the distillation-injection constraint. Figure 7(b) shows a $|Y\rangle$ state distillation-injection super-module, and Figure 7(c) shows an $|A\rangle$ state distillation-injection super-module.

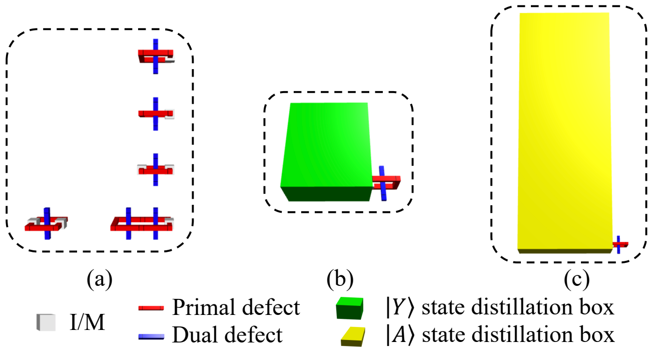
(a)
(b)
(c)
I/M
Primal defect
Dual defect
$|Y\rangle$ state distillation box
$|A\rangle$ state distillation box

Fig. 7. Examples of super-modules. (a) A time-dependent super-module. (b) A $|Y\rangle$ state distillation-injection super-module. (b) An $|A\rangle$ state distillation-injection super-module.

*2) 2.5D Placement:* Instead of general 3D architectures, we propose to place all modules in a 2.5D architecture for its higher regularity, which can benefit the routability. For a 2.5D architecture, modules are divided into different tiers, where the modules in each tier are placed in a 2D plane. All tiers are stacked up to form a 2.5D architecture in the 3D space. We use the 2.5D B*-tree representation [13] for the modules, and the placement of each tier is represented by a 2D B*-tree [14], where each node in the tree represents a module. For example, Figure 8 shows an example of a three-tier 2.5D B*-tree representation and the corresponding B*-tree of each tier. A simulated annealing (SA) engine is applied to minimize the total volume and total estimated wirelength. Similar to [13], we apply four perturbations of the 2.5D B*-tree to explore the solution space, including inter-/intra-tree node swap and inter-/intra-tree node move. Furthermore, to enhance the routability, each module is slightly expanded to preserve some routing region around the module.

The selective teleportation measurements among the T gates operating on the same qubit should obey a relative time ordering, as mentioned in Section II-B. To maintain the ordering during the SA process, we create a time-dependent super-module list (TSL) for each qubit. The time-dependent super-modules associated with the T gates operating on the same qubit are added to a TSL. Before the SA engine is applied, the super-modules in a TSL are resized to the same size. After a perturbation operation is performed on the 2.5D B*-tree, the positions of time-dependent super-modules would be identified. According to their relative time ordering, the time-dependent modules in a TSL will be reallocated to the identified positions. By the reallocation, the order of T gates in a TSL is maintained after a perturbation operation. Besides, the reallocation does not affect the positions of other modules since the super-modules in a TSL are resized to an identical size.

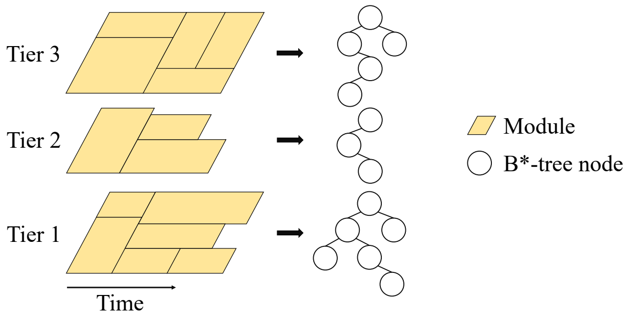

Fig. 8. An example of a 2.5D B*-tree representation with three tiers and the corresponding B*-tree for each tier.

### D. Dual-defect Net Routing

In the routing stage, we should route all the nets to restore the dual loops. First, all the nets are routed sequentially in the non-decreasing order sorted by the Manhattan distance of each net, and the A* search algorithm is applied to route each net within a restricted search region. Besides, the negotiation-based rip-up and reroute technique [15] is adopted to alleviate routing congestion, which is a common technique used in electronic design automation (EDA). Moreover, if a net fails to be routed, the net search region will be slightly expanded in the next iteration to explore more routable regions.

Next, we introduce the concept of the *friend net*. If two nets share the same pin, then they are defined as a friend net to each other with respect to the pin. Once net $n_i$ is routed, the friend nets of $n_i$ with respect to one of $n_i$'s pin $p$ can end on any point of the routed path of $n_i$ instead of $p$. This rule is a kind of topological deformation that does not change the braiding relationship and thus is valid. Figure 9(a) shows an example that $n_1$ and $n_2$ are friend nets with respect to pin $p$. Therefore, after $n_1$ is routed, as shown in Figure 9(b), the unrouted net $n_2$ can end on any point of the routed path of $n_1$ instead of $p$, as shown in Figure 9(c). The friend net-awareness routing can indeed reduce the required routing resource and also enhance the routability.

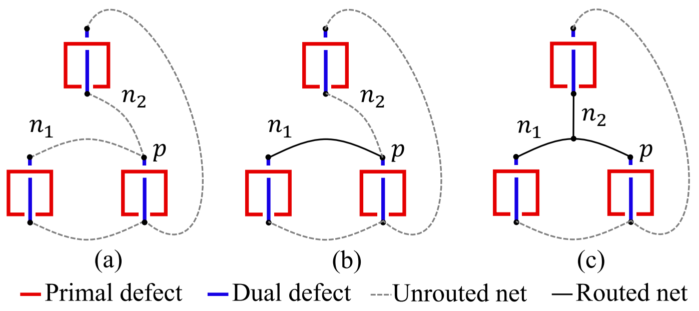

Fig. 9. The concept of the friend net. (a) All nets are unrouted. (b) $n_1$ is routed. (c) $n_2$ can end on any point of $n_1$ because they share the pin.

TABLE I
BENCHMARK STATISTICS.

| Benchmark | #Qubits | #CNOTs | #$\|Y\rangle$ | #$\|A\rangle$ | #Modules | #Nets |
|---|---|---|---|---|---|---|
| 4gt10-v1_81 | 131 | 168 | 42 | 21 | 362 | 483 |
| 4gt4-v0_73 | 257 | 341 | 84 | 42 | 724 | 978 |
| rd84_142 | 897 | 1162 | 294 | 147 | 2500 | 3339 |
| hwb5_53 | 1307 | 1729 | 434 | 217 | 3687 | 4982 |
| add16_174 | 1394 | 1792 | 448 | 224 | 3857 | 5167 |
| sym6_145 | 1519 | 1980 | 504 | 252 | 4255 | 5688 |
| cycle17_3_112 | 1911 | 2478 | 630 | 315 | 5321 | 7119 |
| ham15_107 | 3753 | 4938 | 1246 | 623 | 10560 | 14215 |

TABLE II
COMPARISON OF SPACE-TIME VOLUME FOR THE STATE-OF-THE-ART WORK [11] AND OURS.

| Benchmark | Canonical | | [11] (1D) | | [11] (2D) | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|
| | Volume | Ratio | Volume | Ratio | Volume | Ratio | Volume | Ratio | Runtime (s) |
| 4gt10-v1_81 | 136836 | 5.362 | 98322 | 3.853 | 91116 | 3.570 | 25520 | 1.000 | 15 |
| 4gt4-v0_73 | 535398 | 9.122 | 361152 | 6.153 | 327816 | 5.585 | 58696 | 1.000 | 26 |
| rd84_142 | 6287400 | 13.927 | 2805246 | 6.214 | 2744316 | 6.079 | 451440 | 1.000 | 262 |
| hwb5_53 | 13608294 | 10.143 | 9114828 | 6.793 | 8203548 | 6.114 | 1341704 | 1.000 | 447 |
| add16_174 | 15028608 | 14.054 | 6449532 | 6.031 | 6173928 | 5.773 | 1069362 | 1.000 | 590 |
| sym6_145 | 18103176 | 9.181 | 1072836 | 5.437 | 9852336 | 4.997 | 1971840 | 1.000 | 793 |
| cycle17_3_112 | 28469700 | 12.094 | 19082448 | 8.106 | 16843884 | 7.155 | 2354100 | 1.000 | 1402 |
| ham15_107 | 111335928 | 15.186 | 69294822 | 9.452 | 63017484 | 8.595 | 7331454 | 1.000 | 4901 |
| Avg. Ratio | | 11.133 | | 6.505 | | 5.984 | | 1.000 | |

## IV. EXPERIMENTAL RESULTS

We implemented our algorithm in the C++ programming language with the Lemon graph library [16] and the Boost C++ libraries [17]. 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 [18], where the benchmark statistics is summarized in Table I. "#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" and "#Nets" denote the numbers of modules and nets respectively after modularization and iterative bridging.

The space-time volume comparisons of our proposed algorithm and the state-of-the-art method [11] are summarized in Table II. Because both of the canonical forms of TQEC circuits and the work [11] do not consider the placement of state distillation boxes, the volumes of them shown in Table II are calculated by the volumes of the synthesized TQEC circuits directly plus the total volume of state distillation boxes for a fair comparison. Besides, the work [11] proposes 1D and 2D qubit arrangements, and the results of both arrangements are also reported in Table II, where 2D qubit arrangements result in less space-time volumes than 1D ones. Our proposed algorithm can averagely achieve 91.0% volume reduction from the canonical forms while the work [11] can only achieve 46.3% reduction, which justifies the effectiveness of our compression technique.

## V. CONCLUSION

In this paper, we have presented an effective algorithm that optimizes space-time volumes of TQEC circuits while considering time-ordered measurement constraints and the integration of state distillation boxes. We have proposed an iterative bridging technique that efficiently constructs bridge structures for dual-defect loops. Under the placement-and-routing framework, we have developed a time-ordering-aware 2.5D placement method that simultaneously optimizes the circuit volume and meets the time-ordered measurement constraints, and we have also proposed the friend net-aware dual-defect net routing that can improve the routability under topological deformation. Experimental results have shown that our proposed algorithm can compress much more space-time volume of TQEC circuits than state-of-the-art work.

## REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, no. 3, p. 032324, 2012.

[3] R. Raussendorf, J. Harrington, and K. Goyal, "Topological fault-tolerance in cluster state quantum computation," *New J. Phys.*, vol. 9, no. 6, p. 199, 2007.

[4] A. G. Fowler and K. Goyal, "Topological cluster state quantum computing," *arXiv preprint arXiv:0805.3202*, 2008.

[5] A. Paler, S. J. Devitt, and A. G. Fowler, "Synthesis of arbitrary quantum circuits to topological assembly," *Sci. Rep.*, vol. 6, no. 1, pp. 1–16, 2016.

[6] A. Paler, I. Polian, K. Nemoto, and S. J. Devitt, "A fully fault-tolerant representation of quantum circuits," in *Reversible Computation*, 2015.

[7] A. Paler, I. Polian, K. Nemoto, and S. J. Devitt, "Fault-tolerant, high-level quantum circuits: form, compilation and description," *Quantum Science and Technology*, vol. 2, no. 2, p. 025003, 2017.

[8] A. Paler, A. G. Fowler, and R. Wille, "Synthesis of arbitrary quantum circuits to topological assembly: Systematic, online and compact," *Sci. Rep.*, vol. 7, no. 1, pp. 1–16, 2017.

[9] A. G. Fowler and S. J. Devitt, "A bridge to lower overhead quantum computation," *arXiv preprint arXiv:1209.0510*, 2012.

[10] A. Paetznick and A. G. Fowler, "Quantum circuit optimization by topological compaction in the surface code," *arXiv preprint arXiv:1304.2807*, 2013.

[11] Y. Lin, B. Yu, M. Li, and D. Z. Pan, "Layout synthesis for topological quantum circuits with 1-D and 2-D architectures," *IEEE Tran. on CAD*, vol. 37, no. 8, pp. 1574–1587, 2017.

[12] K. Asai and S. Yamashita, "Compaction of topological quantum circuits by modularization," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 102, no. 4, pp. 624–632, 2019.

[13] P. Falkenstern, Y. Xie, Y.-W. Chang, and Y. Wang, "Three-dimensional integrated circuits (3D IC) floorplan and power/ground network co-synthesis," in *Proc. of ASPDAC*, 2010.

[14] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proc. of DAC*, 2000.

[15] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proc. of FPGA*, 1995.

[16] The LEMON graph library. [Online]. Available: https://lemon.cs.elte.hu/trac/lemon

[17] The Boost C++ libraries. [Online]. Available: https://www.boost.org/users/history/version_1_72_0

[18] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Proc. of ISMVL*, 2008.