Electronic Preprint for Journal of Information Processing Vol.26

Regular Paper

Logical Qubit Layout Problem for ICM Representation

Nurul Ain Binti Adnan[1,a)] Shigeru Yamashita[1,b)]

Received: March 13, 2017, Accepted: October 3, 2017

Abstract: This paper formulates the qubit layout problem for *ICM representation* which is favorable for topological quantum computation. Observing the properties of *braiding operations* in a logic circuit model of ICM, we study the potential usefulness of two-dimensional qubit layouts based on this model. We compare and contrast the efficiency of one- and two-dimensional qubit layouts in reducing the logical time steps for topological quantum circuits. This leads us to an approach to find a good gate order for two-dimensional qubit layouts. Indeed, our preliminary experimental results show the effectiveness of two-dimensional qubit layouts.

Keywords: topological quantum computation, two dimensional qubit layout, braiding operation

1. Introduction

Topological quantum computation [1], [2], [3] has proven to be one of the most promising ways to realize fault-tolerant quantum computation. In this paper, we focus on the computation model in Ref. [2] which is based on the 2-D nearest neighbor coupled lattice of qubits with the so-called *surface code*. There is a completely different computation model, *anyonic quantum computation*, which is based on topological states of matter. The two computation models are both routinely referred to as "topological quantum computation" within the literature but are vastly different. In the following, we refer to the first model as "topological quantum computation."

For topological quantum computation, there is a recently proposed quantum circuit representation called *ICM representation*, which consists of qubit (I)nitialization, (C)ontrolled-NOT gates and (M)easurements with respect to different bases [4]. Any quantum circuits consisting of any quantum gates (e.g., controlled-V, Toffoli gates) can be transformed into ICM representation after decomposing all the non-deterministic gates with an exact gate list that are robust against errors. After the decomposition, all the qubit initializations are moved to the beginning of the circuits, all of the single-qubit measurements are moved to the end of the circuits, and the CNOT gates remain in the middle of the circuits. The purpose of ICM is to create a **deterministic** circuit from **probabilistic** components (e.g., measurements).

The middle part of the ICM representation consists of many CNOT gates for a practical computation. Thus we need to consider how to design a circuit that consists of only CNOT gates for the realization of topological quantum computation. As we will explain below, a CNOT gate is implemented by a special process called a *braiding operation* that can be done in one logical time step. A critical observation here is that multiple braiding operations can be done at the same time (i.e., in one logical time step) if they do not *overlap* physically. This condition will be further explained, but we can consider intuitively that two (or more) CNOT gates can be done in one logical time step if their corresponding drawings do not *overlap* in a quantum circuit diagram.

For example, the three gates, g_1, g_2 and g_3, in **Fig. 2** can be done in one logical time step in our model because their diagrams do not overlap. Thus, the number of logical time steps for the circuit in Fig. 2 is three. On the other hand, g_1, g_2 and g_3 in **Fig. 1** need three logical time steps because their diagrams do overlap. Therefore, we need eight logical time steps for the cir-

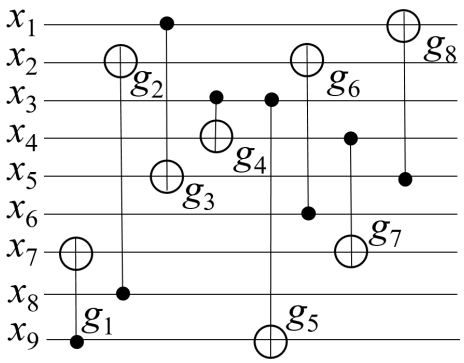


Fig. 1 An initial circuit: 8 steps.

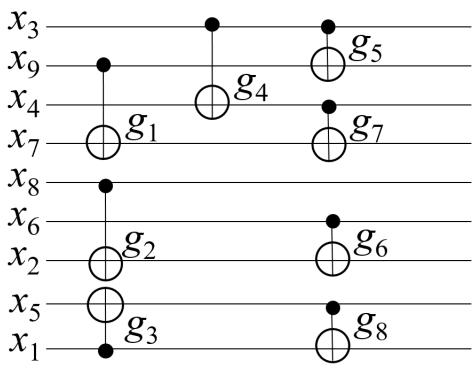


Fig. 2 Optimal one-dimensional qubit layout: 3 steps.

[1] Graduate School of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525–8577, Japan
[a)] nu_ain@ngc.is.ritsumei.ac.jp
[b)] ger@cs.ritsumei.ac.jp

cuit in Fig. 1. Note that the two circuits are logically equivalent, but the order of the logical qubits (i.e., $x_1, x_2, \cdots, x_8$) is different. Observing the above example, we must consider the order of the logical qubits when we design a circuit that only consists of CNOT gates for the ICM representation, unlike conventional quantum circuits.

One single qubit order does not allow us to perform the circuit in Fig. 1 with two time steps if we use the one-dimensional qubit layout. For example, at the one-dimensional qubit layout of the qubit order in Fig. 2, the three gates, g_4, g_1 and g_5 (and g_7), overlap so we need at least three time steps. In contrast, if we layout the qubits two-dimensionally (**Fig. 3**), we can perform the circuit with only two logical time steps because the two-dimensional qubit layout allows us to simultaneously perform g_1, g_2, g_3 and g_4, as shown on the left-hand side of Fig. 3.

Considering the above motivational example, we propose to use two-dimensional qubit layouts. In this paper, we show how we can reduce the logical time steps by using two-dimensional qubit layouts. Note that a two-dimensional logical qubit layout should be realized as easy as one-dimensional one because physical qubits are placed in two-dimensionally [1], [2], [3] anyway in both cases (as we will see in Fig. 4). For our purposes, we formulate a design problem so as to find a good qubit layout and a good gate order to reduce the logical time steps for topological quantum circuits. It may be obvious that we can do better by using two-dimensional logical qubit layouts, however we cannot know how good the two-dimensional qubit layouts will actually be. Thus, to compare the essential differences between one- and two-dimensional qubit layouts, we try to find the best possible layout and gate order for both layouts.

To find the best possible layout and gate order efficiently, we formulate the design problem so as to find the best set of one-dimensional qubit layouts by solving a *minimum clique partition problem* and then to find the best two-dimensional layouts that can embed as many such one-dimensional layouts as possible. (This will be discussed in more detail in Section 3). As far as we know, this is the first systematic approach for finding a good two-dimensional qubit layout.

In general, our approach can find the optimal solutions for up to 4x4 qubit layouts, and we show our experimental result to compare the best possible logical time steps by using one- and two-dimensional qubit layouts. To compare larger cases, we also report the experimental results by using a simple SA-based search method. Moreover, we report that our proposed method can indeed decrease the number of computational time steps for a circuit from the ICM representation of a controlled-V gate compared to the case when we use one-dimensional qubit layout [4].

This paper is organized as follows. The next section explains the logic circuit model for topological quantum circuits and its design flow. Section 3 proposes our method for finding a good two-dimensional qubit layout after defining some terminology used in our method. Section 4 shows some experimental results, and Section 5 concludes our paper with future work.

2. Design Framework for Topological Quantum Computer based on ICM Representation

In this section, we first introduce a topologically error-corrected logic circuit model for topological computation. Operations to logical qubits can be visualized in a geometric description, and logical qubits can be defined as *defects*. A *braiding* operation is an interaction between two types of defects in the geometric structure.

Next we overview our whole design framework to design topological quantum circuits based on ICM representation; We explain ICM representation with an example, and then we mention what is our target in this paper.

2.1 Logic Circuit Model for Topological Quantum Computation

Surface code is a way to encode logical qubits for topological quantum computation. A logical qubit is encoded as two defects on a lattice of physical qubits. **Figure 4** shows a lattice of physical qubits in which each white circle represents a physical qubit. In general, we need a pair of defects to support each logical qubit. There are two types of defects: *primal* and *dual*. We introduce the concept of primal and dual defects to explain the structure of the logical CNOT gate in the following.

Figure 4 shows how logical qubits are placed on a lattice. The dark blue squares correspond to a defect, and a pair of defects connected with red lines represents logical qubits. Green and purple lines show *braiding* operations conceptually. A braiding operation is a movement of the status of a defect by measuring the physical qubits, and we can assume that multiple braiding operations can be done at the same time if their corresponding lines do not overlap in the physical space [2], [5]. In the figure, the two braiding operations corresponding to two purple lines can be done in one logical time step. In Fig. 4, all the defects are of the same type (say, primal) and the other type defects (not shown) are offset by a $1/2$ defect size in both the vertical and horizontal directions. A logical CNOT gate can be achieved if we *braid* a primal defect around a dual defect as we will see in the following.

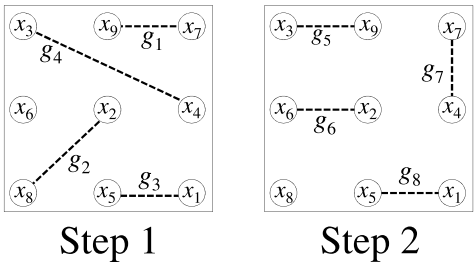


Fig. 3 Optimal two-dimensional qubit layout: 2 steps.

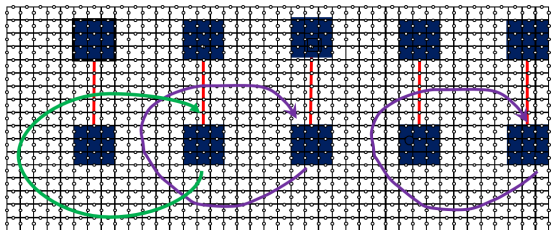


Fig. 4 Logical qubits with braiding operations.

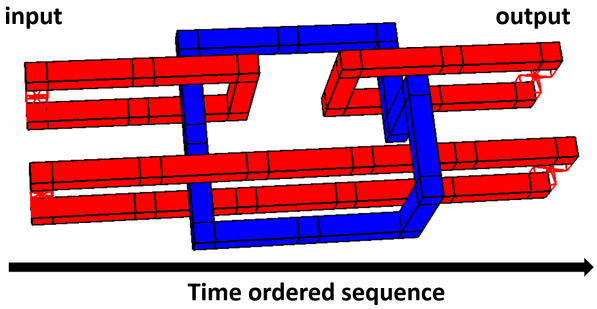


Fig. 5 Braiding for a CNOT gate.

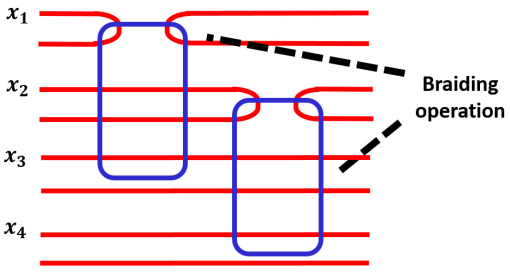


Fig. 6 Overlapped braiding operations (2 time steps).

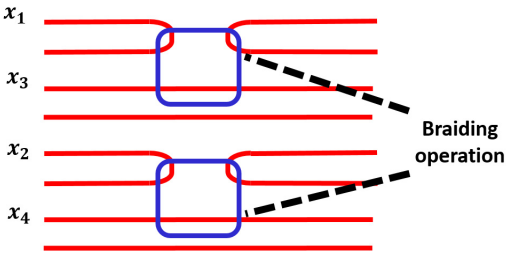


Fig. 7 Non-overlapped braiding operations (1 time step).

In a higher abstract model, the change of a defect status can be described as a strand, and a braiding can be expressed as a cross of two strands. **Figure 5** shows a CNOT gate between two primal qubits. To perform a CNOT gate between them, we need to introduce one dual qubit and perform braiding, as the figure shows, a braiding of a dual strand (blue cuboid) is applied to the two primal strands (red cuboids) to perform a logic CNOT between the two primal qubits. The upper region of the geometry corresponds to the control qubit and the lower region corresponds to the target qubit.

In respect to our ICM representation, the inputs are mapped on the left-hand side of the cuboids, and the outputs are mapped on the right-hand side. The time runs horizontally (from left to right) in the direction of the two primal strands.

In **Fig. 6**, a braiding operation between the logical qubits is denoted by a blue loop. For example, the left loop indicates a CNOT gate between x_1 and x_3. In Fig. 6, we can easily confirm that the two braiding operations for the two logical CNOT gates overlap. Thus, the two gates must be performed in two logical time steps.

In contrast to the above, if we change the qubit order (qubit layout) for this circuit, we can perform both gates in parallel and we can obtain a better implementation (**Fig. 7**). Here the two circuits (Figs. 6 and 7) are logically equivalent but with different qubit orders.

Note that reducing the logical time steps in the above geometric visualized model of topological quantum computation would eventually reduce the cost to realize the corresponding topological quantum computation [2], [5]. In other words, the time axis of the 3D physical resources to realize the corresponding topological quantum computation of the circuit in Fig. 7 would be half compared to the case when we realize the circuit in Fig. 6. Thus it is very important to find a qubit layout to reduce the logical time steps in the geometric visualized model of topological quantum computation.

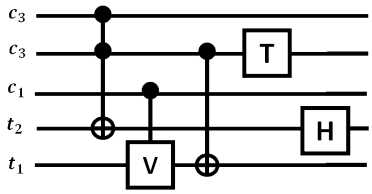


Fig. 8 Example of a quantum circuit.

Two logical CNOT gates can clearly be done at the same time (in one logical time step) if their corresponding drawings in a quantum circuit do not overlap. (We will define **overlapped** and **non-overlapped** for CNOT gates in Section 3.1.)

Another key to note is that we can perform multi-target CNOT gate braiding operations in a single braid. In other words, if we can exploit gates with multiple target lines, this helps us reduce the time step computation. These reasons motivate us to find a good qubit layout and gate order for a given circuit to reduce the computational time.

2.2 Design Flow for Topological Quantum Computation

We assume that our design flow to design topological quantum circuits starts from an arbitrary circuit (as shown in **Fig. 8**). First, we transform a given arbitrary circuit into a so-called ICM representation. An ICM representation [1] is obtained after replacing all of the non-deterministic gates from a quantum circuit with an exact gates list that only consists of qubit (I)nitialisations, (C)NOT gates, and (M)easurements. These quantum operations are relatively robust. A reader may refer to the paper Ref. [1] for how to obtain ICM representation.

Figure 9 shows the ICM representation of a controlled-V gate taken from a previous work [1]. The ICM representation for a controlled-V gate has inputs states c_{in} (control) and t_{in} and outputs c_{out} and t_{out}. The size of the circuit expands significantly due to the increased ancillas. All the ancillas initializations are moved to the beginning of the circuit and single qubit measurement are moved to the end of the circuit. The middle of the circuit is now left with CNOT gates only, proving that ICM representation replaces the non-deterministic gates with an exact gate list.

After getting the ICM representation, each single logical qubit operation can be visualized into a geometrical description, where each property of the circuit description elements has a single corresponding structure to the geometric description (see an example in Fig. 5). As previously explained, the logical qubits are defined as strands/defects here.

This paper focuses on the optimization of the middle part of ICM representation consisting of only CNOT gates. As we explain in the following, by placing the logical qubits in a good lay-

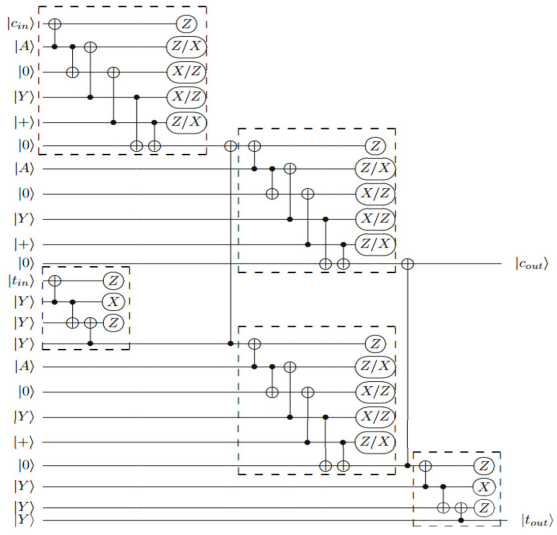


Fig. 9 ICM representation of a controlled-V gate.

out, the logical time steps for CNOT gates would be decreased. This would eventually contribute to decrease the physical resources to realize the topological quantum computation as discussed in Ref. [5].

3. Optimization of Logical Qubit Layouts for ICM

As we saw in Section 2 (e.g., the difference between two circuits (Figs. 6 and 7)), the qubit order/layout is crucial for the CNOT gates in ICM representation. Motivated by this, we present how to find a good qubit order/layout in this section. In this paper, we mainly consider two-dimensional qubit layouts. Note that a one-dimensional layout can be considered as a special case for two-dimensional layout, and it is easy to find a good one-dimensional layout if we have a method to find a two-dimensional layout. To explain our proposed method, we define some terminologies for one-dimensional and two-dimensional qubit layouts below.

3.1 Terminology Used for One-Dimensional Qubit Layout

First we review the model of a one-dimensional qubit layout that is often used in conventional logical quantum circuits. In the following, we assume that logical qubits are placed in a line, $x_1, x_2, \cdots, x_n$ for a circuit with n logical qubits. Because we focus on the part that only consists of CNOT gates in the ICM representation, our target circuits only consist of CNOT gates in the following. The target and control qubits of gate g_i are denoted by $T(g_i)$ and $C(g_i)$.

First we introduce the term **overlapped** for one-dimensional qubit layouts.

Definition 1 Pair of gates g_i and g_j is said to be **overlapped** with a given qubit order if the groups of qubits between $T(g_i)$ and $C(g_i)$ and $T(g_j)$ and $C(g_j)$ share at least one qubit with the given qubit order. If g_i and g_j do not overlap, they are said to be **non-overlapped**. If two or more gates share the same control bit, we have a special rule that also defines them as **non-overlapped** regardless of the qubit layout.

This special rule is due to the fact that we can perform a multi-target CNOT (that can be considered multiple simple CNOT gates with the same control bit) in a single braid.

For example, g_1 and g_2 in Fig. 1 are overlap with this qubit order. Since $T(g_1) = 7, C(g_1) = 9$ and $T(g_2) = 2, C(g_2) = 8$, the group of qubits placed between the control and the target qubits of g_1 are x_7, x_8, and x_9, and the group qubits placed between the control and the target qubits of g_2 are $x_2, x_3, x_4, x_5, x_6, x_7$, and x_8. Thus, the two groups of qubits share common qubits, and g_1 and g_2 are overlapped. However, if we just change the qubit order to get the circuit in Fig. 2, g_1 and g_2 become non-overlapped, as we can see from the figure.

If the two logical CNOT gates are non-overlapped, the braiding operations for the two CNOT gates can be performed in one logical time step, as we discussed above. Thus, our task is to increase the number of CNOT gates that are non-overlapped.

We can swap two CNOT gates, g_i and g_j, if $C(g_i) \neq T(g_j)$ and $T(g_i) \neq C(g_j)$. This is the **swapping rule**. For example, g_3 and g_4 in Fig. 1 can be swapped. Also g_4 and g_5 in Fig. 1 can be swapped, and thus we can change the order of g_3, g_4, g_5 in any order. However, g_4 and g_7 in Fig. 1 cannot be swapped because the target qubit of g_4 and the control qubit of g_7 have the same qubit: x_4.

Based on the circuit model discussed in Section 2.1, the cost of circuit Q, denoted by $Cost(Q)$, is defined as follows. Let the maximum number of non-overlapped gates at the first part of Q be k. With the swapping rule, we can move k (the maximum possible number) gates to the beginning of the circuit so that k gates are non-overlapped. Note that two non-overlapped gates can be swapped by the swapping rule. Then $Cost(Q) = Cost(Q') + 1$, where Q' is a circuit obtained from Q by removing the first k gates. This cost is due to the fact that the first k non-overlapped gates can be done in one logical time step in our circuit model.

Our essential task is to find a good qubit order among all the permutations, and thus it seems very difficult.

To explain our method, we also need to define the following terminology.

Definition 2 If g_i can only be moved next to g_j by the swapping rule, g_i and g_j are said to be **adjacentable**

For example, g_4 and g_6 in Fig. 1 are adjacentable because g_4 and g_5 (or g_5 and g_6) can be swapped.

Note that we consider the above definition of "adjacentable" is only for the logical relationship between two gates. In other words, the definition of "adjacentable" can be applied to both one-dimensional and two-dimensional qubit layouts.

For a given qubit order, if two gates are adjacentable and non-overlapped, their corresponding braiding operations can be performed in parallell, thus reducing the computational steps for the circuit. Therefore, the existing method [6] tries to find a "good" one-dimension qubit order such that as many adjacentable gates as possible become non-overlapped.

3.2 Terminology for Two-Dimensional Qubit Layout

In this paper, we propose to use two-dimensional qubit layouts and show an efficient method for finding a good two-dimensional layout. In this paper, for simplicity we assume that each qubit is

placed on one grid point in the two-dimensional layouts. Note again the motivational example (Figs. 2 and 3), where the logical time steps are three and two when the qubits are placed in one-dimension and two-dimension, respectively. The example, suggests that a two-dimensional qubit layout is always better than a one-dimensional qubit layout. This is indeed true, as stated formally in the following; our design approach is based on this fact.

Theorem 1 If a group of gates can be performed at the same time in a one-dimensional qubit layout, a two-dimensional qubit layout must exist by which we can simultaneously perform the same group of gates.

The proof is obvious recognizing that a one-dimensional qubit order can always be embedded into a two-dimensional qubit layout. For example, the qubit layout in Fig. 3 contains one-dimensional qubit orders, such as $x_3, x_9, x_7, x_4, x_2, x_6, x_8, x_5, x_1$ and $x_7, x_9, x_3, x_4, x_1, x_5, x_8, x_2, x_6$. By choosing a qubit order of $x_3, x_9, x_7, x_4, x_2, x_6, x_8, x_5, x_1$, we can perform g_5, g_6, g_7 and g_8 in Fig. 1 at the same time. In Fig. 1, g_1, g_2, g_3 and g_4 can also be performed at the same time with this qubit order: $x_3, x_9, x_7, x_4, x_2, x_6, x_8, x_5, x_1$. In other words, the qubit layout Fig. 3 provides the above two one-dimensional qubit layouts, and two time steps are sufficient if we use the two-dimensional layout.

For two-dimensional layouts, we need to modify the term "overlapped" as follows, which should be straightforward.

Definition 3 Pair of gates g_i and g_j is said to be **overlapped** with a given two-dimensional qubit layout if the line between $T(g_i)$ and $C(g_i)$ and the line between $T(g_j)$ and $C(g_j)$ cross in the given two-dimensional qubit layout. If g_i and g_j are not overlapped, they are said to be **non-overlapped**. In addition, if two or more gates share the same control bit, we have a special rule that also defines them as **non-overlapped**, regardless of the qubit layout.

For example, in the qubit layout in Fig. 3, g_i, whose target and control bits are x_3 and x_4, and g_j, whose target and control bits are x_2 and x_8, are non-overlapped but g_i and g_k whose target and control bits are x_2 and x_7, respectively, are overlapped. This is because the line between x_3 and x_4 and the line between x_2 and x_8 do not cross, but the line between x_3 and x_4 and the line between x_2 and x_7 cross in the layout, as shown in Fig. 3.

3.3 A Method to Find a Good Qubit Layout

Our essential task is to find a "good" two-dimensional qubit layout such that as many adjacentable gates as possible become non-overlapped. The difficulty is that a two-dimensional qubit layout allows many pairs of gates to be non-overlapped and unlike one-dimensional layouts there are many possibilities for a "good" layout.

Therefore, to do the search efficiently, we divide the whole problem into the following two sub-problems, each of which can be solved optimally:

- First, we divide all the gates into the smallest number of gate groups such that all the gates in each group are *possibly non-overlapped*, which is defined below.
- Second, we enumerate the possible two-dimensional qubit layouts for each gate group so that all the gates in the gate group can be non-overlapped. Let such a set of two-dimensional qubit layouts for gate group G_i be P_i. After that, we can find a good layout that is included in as many P_i as possible.

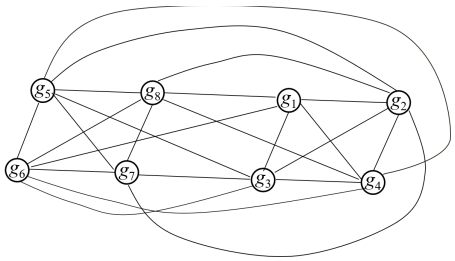


Fig. 10 Graph at Step 1 for circuit in Fig. 1.

The following is the definition of *possibly non-overlapped*:

Definition 4 Two gates are said to be **possibly non-overlapped** if $T(g_i)$ and $C(g_i)$ are different from neither $T(g_j)$ nor $C(g_j)$, and the two gates are adjacentable. In addition, if gates g_i and g_j have the same set of control lines, both gates are also said to be **non-overlapped**.

Equivalently, if two gates are possibly non-overlapped, at least one qubit layout allows the two gates to be non-overlapped.

Unlike the one-dimensional case, a two-dimensional qubit layout allows many pairs of gates to be non-overlapped. So we expect that possibly non-overlapped gates become non-overlapped with one specific qubit layout more often than the one-dimensional case. If that happens, we can perform all the gates in one group of possibly non-overlapped gates at one time step; this means that the number of entire necessary time steps is expected to be equivalent to the number of groups of possibly non-overlapped gates. Thus, in the first sub-problem, we must find the smallest number of gate groups.

Finding a group of *possibly non-overlapped* gates can be as easily formulated as finding a clique in a graph. We can find a good solution by casting the problem to a clique cover problem as follows. There are many state-of-the-art methods for it, and we just use an exact method to solve the minimum clique partition problem [7] in our experiment.

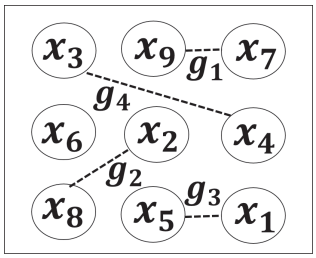
A method to solve the first sub-problem.

Step 1. Construct a graph where each node corresponds to each gate in C. We have an edge between two nodes if the corresponding two gates in the given circuit are possibly non-overlapped.

Step 2. Partition the graph obtained at Step 1 into a minimal number of cliques, $C_1, C_2, \cdots, C_m$, using a solver for clique cover problems. From each clique, we get each group, G_i, of possibly non-overlapped gates.

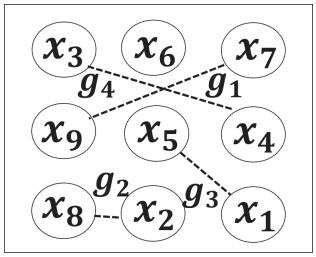
For an initial circuit shown in Fig. 1, the graph constructed at Step 1 can be shown as in **Fig. 10**. This graph can be clearly covered with two cliques: $C_1 = (g_1, g_2, g_3, g_4)$ and $C_2 = (g_5, g_6, g_7, g_8)$. Thus, the group of possibly non-overlapped gates are selected as $G_1 = \{g_1, g_2, g_3, g_4\}$ and $G_2 = \{g_5, g_6, g_7, g_8\}$ in this example. This means that in the best case we can perform the circuit in Fig. 1 in two time steps. Thus, we try to find a good two-dimensional qubit layout in the second problem to perform the circuit in two time steps.

In the following, we represent a two-dimensional qubit lay-



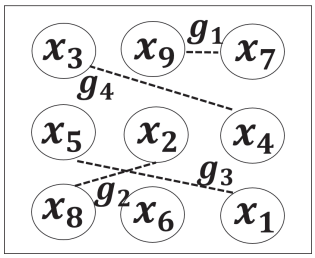
Layout 1

(a) $(x_8, x_5, x_6, x_1, x_2, x_3, x_4, x_9, x_7)$



Layout 2

(b) $(x_8, x_2, x_9, x_1, x_5, x_3, x_4, x_6, x_7)$



Layout 3

(c) $(x_8, x_6, x_5, x_1, x_2, x_3, x_4, x_9, x_7)$

Fig. 11 Example of two-dimensional layouts.

out by a qubit order, which is essentially a permutation. More specifically, we order the qubits from the lower left to the upper right to represent a two-dimensional qubit layout. For example, the qubit layout for Layout 1 (**Fig. 11**) is represented by the qubit permutation, $(x_8, x_5, x_6, x_1, x_2, x_3, x_4, x_9, x_7)$, which is essentially a permutation.

To represent and manipulate a set of permutations, an efficient graph structure has been proposed, πDD [8], which we use in our method. πDD can compactly represent a set of permutations, and provide many efficient set operations, such as intersection and union for the sets of permutations represented by πDDs. Instead of explicitly enumerating all the possible qubit layouts, we implicitly represent a set of permutations using πDD to enumerate the qubit layouts.

For the second problem, our method is as follows.

A method to solve the second sub-problem.

Step 1. We choose G_i from the set of groups obtained in the first problem one by one from the beginning of the circuit and do the following Steps 2 and 3 until G_i remains.

Step 2. We initialize πDD P_i as representing all the possible permutations and go to Step 3.

Step 3. For each pair of two gates in G_i, construct a πDD, p, that represents a set of permutations where two gates are non-overlapped. Then P_i is updated as $P_i \cap p$. This update is repeated for all pairs of two gates. The final P_i represents a set of permutations that correspond to the qubit layouts by which all the gates in G_i can be done at one time step.

Step 4. Our final task is to determine the two-dimensional qubit layouts that are included in as many P_i as possible. We find such a layout by intersecting P_i one by one. If the intersection of all P_i is not empty, we can find the best qubit layout that provides the smallest computational steps for the given circuit. If the intersection becomes empty at some point, we may choose a layout in the intermediate intersection before it becomes empty.

Note that P_i might become empty during the repetition in Step 3. In such a case, there is no qubit layout which allows all the gates in G_i to be non-overlapped; we should spend more than one time step to perform the gates in G_i and divide G_i into multiple groups so that the final πDD obtained at Step 3 for each group is not empty.

If the intermediate intersection becomes empty during Step 4, we may not get the best layout. However, we do not expect such a case to happen very often. Indeed in our experiments described in the next section, the intersection does not become empty, which means that our method can find the best layout. Thus, we can use our method to find the optimal solution efficiently in many cases.

Next we explain how to construct P_1 for $G_1 = \{g_1, g_2, g_3, g_4\}$, which is the first group of *possibly non-overlapped* gates for the example from Fig. 1. See the various two-dimensional qubit layout examples as shown in Fig. 11. For example, Layout 1 is represented as a qubit permutation, $(x_8, x_5, x_6, x_1, x_2, x_3, x_4, x_9, x_7)$, and allows g_2 and g_3 to be non-overlapped. Layout 2 also allows g_2 and g_3 to be non-overlapped. Thus, for a pair of gates: g_2 and g_3, a set of permutation p, created at Step 3, includes Layouts 1 and 2, but it does not include Layout 3 where a line between x_2 and x_8 and a line between x_1 and x_5 cross. For pair of gates g_1 and g_4, we also create a set of permutations that includes Layouts 1 and 3, but not Layout 2. By using primitive operations on πDDs, we can create a set of permutations to represent the set of layouts where two lines do not cross.

If we want to find a layout that allows both pairs of gates, (g_2, g_3) and (g_1, g_4) to be non-overlapped, we just perform an intersection operation between the two πDDs that represent the two sets of permutations obtained as p at Step 3 for (g_2, g_3) and (g_1, g_4). By the intersection, Layouts 2 and 3 are automatically excluded from the intermediate candidate set. In this way, we update intermediate layout candidate set P_i by excluding "bad" layouts for the current pair of two gates. Note that the intersection operation can be done very efficiently with πDDs.

In summary, for each pair of gates, we make πDDs that represent the layouts that allow the two gates to be non-overlapped and update intermediate P_i as $P_i \cap p$. This means we exclude layouts that do not allow the current pair of gates to be non-overlapped from the intermediate layout candidate set. Thus, the final P_1 after Step 3 represents a set of layouts that allows all pairs of gates in $G_1 = \{g_1, g_2, g_3, g_4\}$ to be non-overlapped.

Table 1 Comparison of three methods for small randomly generated CNOT-based circuits.

	1D	2D Optimal		2D SA	
Circuit	Steps	Time (sec.)	Steps	Time (sec.)	Steps
9 bits 50 gates	32	34.44	23 (0.72)	0.00005 (1.4×10^{-6})	23 (0.72)
9 bits 100 gates (1)	63	133.44	46 (0.73)	0.00022 (1.6×10^{-6})	46 (0.73)
9 bits 100 gates (2)	63	133.02	43 (0.68)	0.00022 (1.7×10^{-6})	43 (0.68)
16 bits 100 gates (1)	57	2985.48	30 (0.53)	0.00030 (1.0×10^{-7})	30 (0.53)
16 bits 100 gates (2)	58	2644.35	32 (0.55)	0.00029 (1.1×10^{-7})	32 (0.55)
16 bits 200 gates	119	2546.01	65 (0.54)	0.00102 (4.0×10^{-7})	67 (0.56)
25 bits 200 gates	–	–	–	0.00125	48
25 bits 300 gates	–	–	–	0.00244	83

4. Experimental Result

4.1 An SA-based Heuristic Method

As described in the previous section, our method can find the best layout if the intermediate candidate set does not become empty. There are many efficient solvers for the first sub-problem, i.e., clique cover problems. However, for the second problem, our enumeration-based method obviously cannot deal with so many qubits, even though we utilize an efficient graph structure, πDD [8], to manipulate sets of permutations.

Therefore, we implemented a simple simulated annealing (SA)-based heuristic to compare one- and two-dimensional qubit layouts even for larger problems. (Readers who are unfamiliar with simulated annealing might refer to previous work [9].) In each iteration in our implementation, we swap the location of two qubits and evaluate the depth of the circuit with the new qubit layout. As in a conventional SA-based search, the swap is accepted even though the depth increases when the *temperature* in the SA is high.

The only specific technique used in our implementation is that we do not just select a pair of qubits to be swapped randomly, but we select qubits that are used many times for gates with a higher probability. This is because swapping such qubits tends to more greatly impact the result.

4.2 Comparison between One- and Two-Dimensional Qubit Layouts

We implemented two algorithms: our proposed method mentioned in Section 3.2 and the SA-based method described in Section 4.1 by C++. Then we compared them with the existing one-dimensional optimization method [6] to show how effectively two-dimensional qubit layouts can decrease the computational steps.

It is expected that two-dimensional qubit layout can reduce the computational time steps more efficiently than one-dimensional qubit layout; we confirm this expectation by our experimental results of small cases with optimal optimization methods in **Table 1**.

Table 1 shows the optimized computational steps of the randomly generated CNOT-based circuits by the three optimization methods. Columns 1D, 2D Optimal, and 2D SA show the results for the existing one-dimensional optimization method [6], our proposed method, and the simple SA-based method, respectively. The numbers in parentheses mean the ratio of the number of steps to the one by 1D.

Table 2 Comparison of four methods for large randomly generated CNOT-based circuits.

	1D	2D	1D SA		2D SA	
Circuit	Steps	Steps	Time (sec.)	Steps	Time (sec.)	Steps
16 bits 100 gates	54	32	5.73	42	4.04	27
16 bits 500 gates	271	175	42.54	250	24.65	155
25 bits 100 gates	57	35	7.61	44	4.37	24
25 bits 500 gates	255	148	54.52	240	30.41	128
36 bits 100 gates	52	26	10.76	39	6.45	17
36 bits 500 gates	257	121	73.12	232	34.97	106
100 bits 100 gates	51	19	29.01	39	17.68	13
100 bits 500 gates	239	95	193.31	224	68.71	67

The table also reports the computational time (CPU time) for the two methods on a Linux version 2.6.27 67v15 system on AMD Phenom^{TM} II x6 1055T CPU with 4-GB memory. The numbers in parentheses mean the ratio of the CPU time by 2D SA to 2D Optimal.

The SA parameters are set as follows. The initial temperature is 100°C, which is multiplied by 0.9 at each iteration until it becomes less than 20°C. At each iteration, we tried 500 different swaps of two qubits.

2D Optimal essentially enumerates all the possible qubit layouts, and thus its computational time should increase exponentially even if we use efficient data structures for manipulating the permutations [8]. Indeed, we cannot complete the computation within ten minutes for 25 qubits as expected, whose results are shown as "–" in the table.

On the contrary, the SA-based heuristic is very fast and may be applicable to larger circuits. We also found that the optimization ability of the heuristic is very good and achieves almost the same reduction of steps as 2D Optimal. Thus, for a larger circuit, such a heuristic will be useful.

In order to better see the usefulness of the two-dimensional qubit layouts even for larger circuits, we also conducted an experiment for larger cases. In our experiment, we compare the time steps between one-dimensional and two-dimensional layouts for several randomly generated larger practical circuits. Note that for larger cases we cannot use 2D Optimal, so we compare the results obtained by only SA-based methods. Note also that SA-based methods would be good enough by judging from Table 1 although there is no guarantee that the results are optimal.

The results are shown in **Table 2**. Column 1D and 2D show the time steps of randomly generated initial CNOT-based circuits;

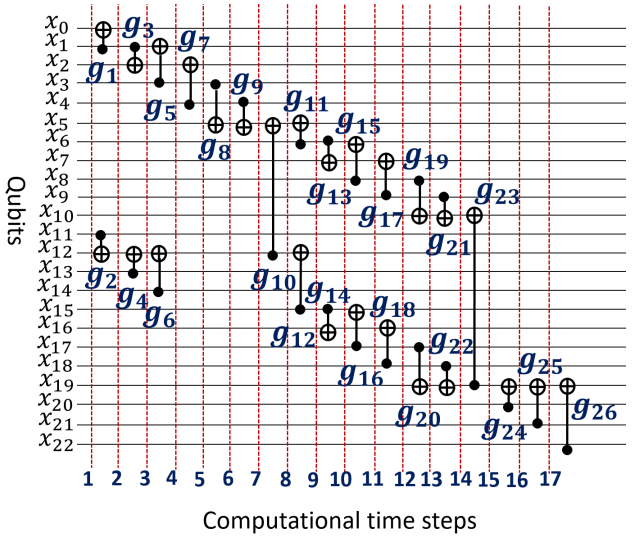


Fig. 12 The original order of CNOT gates in Fig. 9.

while Column 1D SA and 2D SA show the time steps for the optimized one-dimensional and two-dimensional layouts obtained by simple SA-based methods, respectively. The 2D results are reported to have much smaller computational time steps compared to the 1D results. Thus, these results clearly show that we can indeed decrease computational time steps by applying two-dimensional qubit layouts.

Note that the number of best two-dimensional qubit layouts is huge by the verification of our exact enumeration. This means that the problems are easy for a heuristic to find the best solution, so there was only a small difference between the two methods. However, we consider that the SA-based heuristic cannot find the best solution (like our exact method) when the number of best layouts is relatively small, which might happen in practical designs.

4.3 Case Study: Qubit Layout Problem for a Practical ICM Circuit

The previous section showed how the two-dimensional qubit layout problem is important for decreasing the computational time steps for random circuits. Here, we show a case study where we evaluate how two-dimensional qubit layouts are important for practical ICM circuits.

For our case study, we used a circuit from the ICM representation (see the original circuit in Fig. 9) taken from Ref. [4]. **Figure 12** shows the derived CNOT gates from the original ICM representation of a controlled-V gate. There were originally 17 computation steps at first (horizontallly in the graph).

Because we have more than 20 qubits, we used the SA-based method to find a good two-dimensional qubit layout. We consider a 5x5 qubit layout, as shown in **Figs. 14** and **13**. In this example, we applied the special rule introduced in Section 3.2 to generate a multi-target CNOT in a single braid to exploit their benefits. This rule allows us to find more gates to be defined as non-overlapped, regardless of the qubit layout. For example, we can perform g_{12} and g_{14} at the same time, as illustrated in Fig. 13. g_{11} and g_{13} can also be performed at the same time, as illustrated in Fig. 14.

The optimized two-dimensional qubit layouts obtained by our method are shown in Fig. 13. With this optimized qubit layout, we can perform the circuit in six time steps.

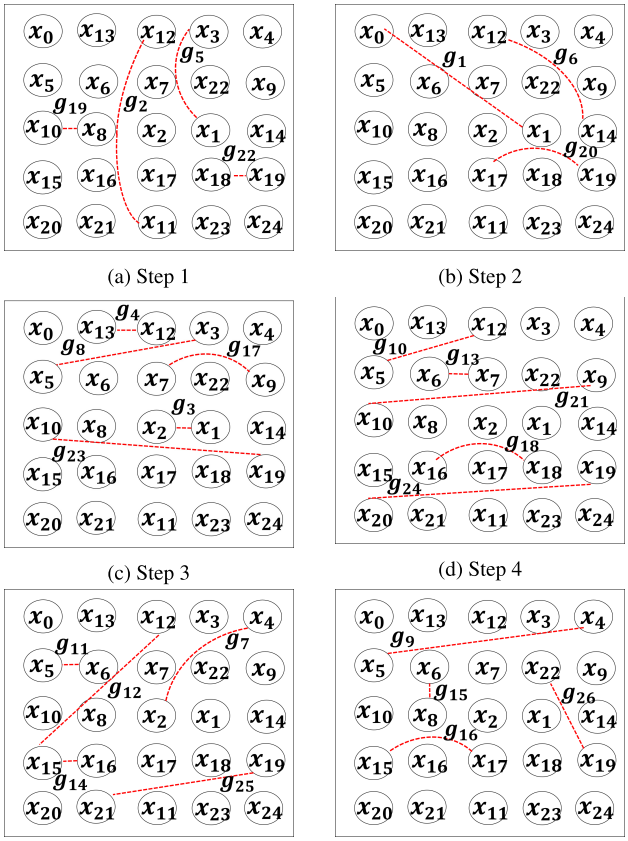


Fig. 13 With an optimized qubit layout (6 time steps).

To evaluate the importance of qubit layouts, we also found the best gate ordering with the initial qubit layout. By the best gate ordering, the number of computational time steps decreased from 17 to 7 (Fig. 14).

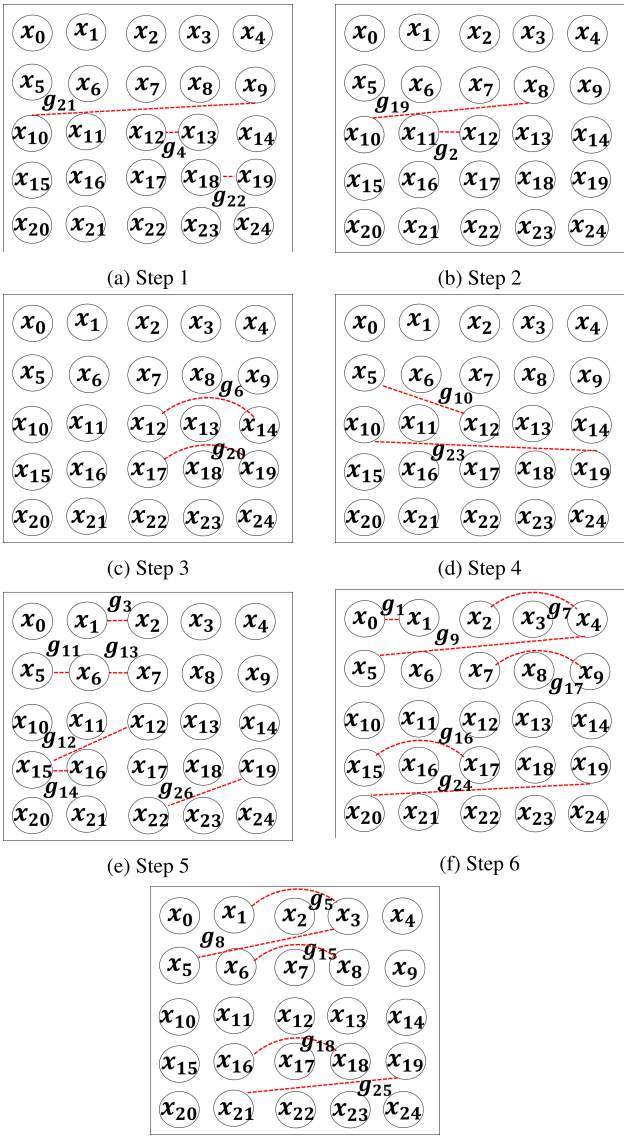
Although only gate reordering is very useful for reducing the computational steps, our case study shows that the qubit layout is also important. This circuit is a small example, but we believe the qubit layout would be much more important for a larger case.

5. Conclusions and Future Work

In this paper, we show a clear difference between one-dimensional and two-dimensional qubit layouts to reduce logical time steps for topological quantum computation. We observe the properties of braiding operations in a logic circuit model for ICM; we formulate our problem to find a good qubit layout and a good gate order. In addition, we then propose two sub-problems to find an optimal solution for the problem. Indeed we can show a clear advantage of two-dimensional qubit layouts in our experiment by our optimal solutions.

Although our proposed method can find a good two-dimensional qubit layout systematically, the method to solve the second sub-problem may not treat a large problem. Thus, we should develop an efficient heuristic to solve the second sub-problem in our future work.

Acknowledgments The authors would like to thank Simon Devitt of Macquarie University and Kae Nemoto of National In-



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

(e) Step 5 (f) Step 6

(g) Step 7

Fig. 14 With an initial qubit layout (7 time steps).

stitute of Informatics for their valuable comments and suggestions for the part of ICM representation as well as the TQC model in this paper. This work was supported by JSPS KAKENHI Grant Number 24106009 and 15H01677, and by the Asahi Glass Foundation.

References

[1] Paler, A., Polian, I., Nemoto, K. and Devitt, S.J.: A fully fault-tolerant representation of quantum circuits, Krivine, J. and Stefani, J.-B. (Eds.): *RC 2015, LNCS 9138*, pp.139–154 (2015).

[2] Fowler, A.G., Stephens, A.M. and Groszkowski, P.: High-threshold universal quantum computation on the surface code, *Phys. Rev. A 80:052312* (Nov. 2009).

[3] Raussendorf, R., Harrington, J. and Goyal, K.: Topological fault-tolerant in cluster state quantum computation, *Quantum Physics*, arXiv:0703143v1 (2007).

[4] Paler, A., Polian, I., Nemoto, K. and Devitt, S.J.: A compiler for fault-tolerant high level quantum circuits, *Quantum Physics*, arXiv:1509.02004 (Sep. 2015).

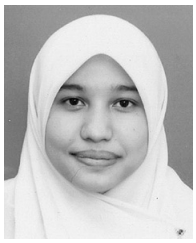
[5] Fowler, A.G. and Devitt, S.J.: A bridge to lower overhead quantum computation, arXiv:1209.0510 (2012).

[6] Yamashita, S.: An optimization problem for topological quantum computation, *Test Symposium (ATS), IEEE 21st Asian*, pp.61–66 (Nov. 2012).

[7] Brélaz, D.: New methods to color the vertices of a graph, *Comm. ACM*, Vol.22, No.4, pp.251–256 (1979).

[8] Minato, S.: πDD, A new decision diagram for efficient problem solving in permutation space, *Proc. 14th International Conference on Theory and Applications of Satisfiability Testing (SAT-2011) (LNCS 6695, Springer)*, pp.90–104 (June 2011).

[9] Aarts, E. and Korst, J.: *Simulated annealing and Boltzmann machines*, Wiley, NY (1988).



Nurul Ain Binti Adnan received her BE (B.E.) degree in Electrical and Electronic Engineering in 2012 and M.E. degree in Electronic Engineering in 2014 from the Graduate School of Engineering, Okayama University of Science, Japan. She is currently pursuing her Ph.D. degree at Graduate School of Information Science and Engineering, Ritsumeikan University, Japan. Her research interests include quantum circuit design, quantum cost reduction and synthesis of topological quantum computation.

Shigeru Yamashita is a professor at the Department of Computer Science, College of Information Science and Engineering, Ritsumeikan University. He received his B.E., M.E. and Ph.D. degrees in Information Science from Kyoto University, Kyoto, Japan, in 1993, 1995 and 2001, respectively. His research interests include new types of computation and logic synthesis for them. He received the 2000 IEEE Circuits and Systems Society Transactions on Computer-Aided Design of Integrated Circuits and Systems Best Paper Award, SASIMI 2010 Best Paper Award, 2010 IPSJ Yamashita SIG Research Award, and 2010 Marubun Academic Achievement Award of the Marubun Research Promotion Foundation. He is a senior member of IEEE, and a member of IPSJ.