Reduction of Computational Steps for Topological Quantum Computation by Inserting SWAP Gates

Koutarou Hoshi[1] Yousef Mohammed Alhamdan[1] Shigeru Yamashita[1]
Simon J. Devitt[2] Kae Nemoto [3]

[1] *Ritsumeikan University*
[2] *Ochanomizu University*
[3] *National Institute of Informatics*

1 Introduction

Topological quantum computation [2] has been drawing much attention for one of the promising ways to realize *fault-tolerant* quantum computation recently. In the topological quantum computation model, unlike the conventional quantum circuit model, we can place logical qubits in a two-dimensional layout as shown in Fig. 2 [1]. Then, the most important point is that any two operations (in a specific layout) can be performed parallelly when the *braiding operations* corresponding to the two operations are not physically *overlapped* in the layout. (More discussion about the circuit model will be given in Appendix.) For example, we cannot perform g_4 and g_7 in Fig. 2 at the same time because the two lines corresponding to the two gates are overlapped physically.

Therefore, the qubit order (i.e., qubit layout) is really important for the computation time for topological quantum computation unlike the conventional quantum circuit model. Accordingly, the existing method [3] tries to find almost the best one-dimensional qubit layout such that as many gates as possible can be performed at the same time. An SA (Simulated Annealing)-based method to find a good initial two-dimensional qubit layout has also been proposed recently [1]. In this abstract, we propose to improve the method [1] by inserting some SWAP gates in the middle. Our preliminary experiments show the effectiveness of our approach.

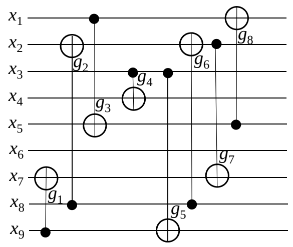


Figure 1: A quantum circuit in one-dimension.

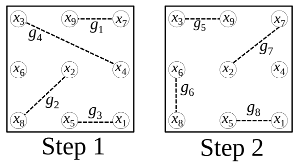


Figure 2: A two-dimensional qubit layout.

2 Our idea to use SWAP gates

In the following, we assume logical qubits, which are denoted by $x_1, x_2, \cdots, x_n$, are placed in a two-dimensional layouts as Fig. 2. Recently, a promising implementation scheme for topological quantum computation has been proposed [4]; the implementation is divided into three parts, initialization part, a large array of only CNOT gates, and the measurement part. Thus it is very important to optimize a circuit consisting of only CNOT gates; we consider to optimize a circuit consisting of only CNOT gates. In the following, the target and the control qubits of gate g_i are denoted by $T(g_i)$ and $C(g_i)$, respectively.

First we introduce a terminology "overlapped."

Definition 1 *A pair of gates g_i and g_j are 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 each other in the given two-dimensional qubit layout. If g_i and g_j are not overlapped, they are said to be* **non-overlapped** *with each other.*

For example, in the qubit layout as shown in Fig. 2, g_4 whose target and control bits are x_3 and x_4, respectively, and g_2 whose target and control bits are x_2 and x_8, respectively, are non-overlapped, whereas g_4 and g_7 whose target and control bits are x_2 and x_7, respectively, are overlapped. This is because two lines between x_3 and x_4, and between x_2 and x_8, are not crossed, but two lines between x_3 and x_4, and between x_2 and x_7, cross each other in the layout as shown in Fig. 2.

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 in our model. Thus, our task is to increase the number of CNOT gates that are non-overlapped with each other.

We can swap two CNOT gates, g_i and g_j, if $C(g_i) \neq T(g_j)$ and $T(g_i) \neq C(g_j)$. We refer this as **the swapping rule** in this abstract. 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_6 and g_7 in Fig. 1 cannot be swapped because the target qubit of g_6 and the control qubit of g_7 are the same qubit (i.e., x_2).

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

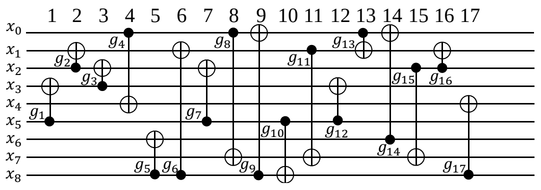


Figure 3: A quantum circuit.

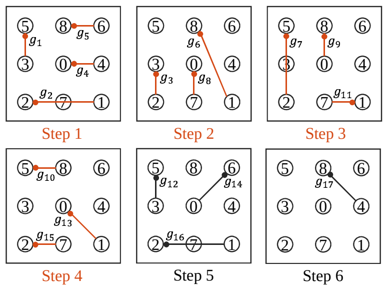


Figure 4: A two-dimensional qubit layout by the previous work.

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

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.

If two gates are adjacentable and non-overlapped in a specific qubit layout, their corresponding braiding operations can be performed parallelly in the layout, and thus the computational steps for the circuit is decreased. Thus, the qubit layout should be important, and our essential task is to find a good qubit layout to decrease the circuit depth; obviously, the problem is NP-hard, and thus we need a good heuristic. There is a previous work [1] to find a "good" qubit layout based on an SA-based heuristic such that as many adjacentable gates as possible become non-overlapped. In the following, we will propose a method to improve the previous method.

Our idea to improve the previous method can be explained by the following example. Suppose we want to realize a circuit consisting of 9 qubits and 17 gates as shown in Fig. 3. By the previous method [1], we can find a good qubit layout as shown in Fig. 4. As we see from Fig. 4, all gates in each step (in Fig. 4) are adjacentable and non-overlapped. Therefore, all the gates in one step can be done at the same time, and thus the number of the total logical computational steps is 6. This is the best initial qubit layout. However, we can decrease the number of time steps to be 5 by our method.

Two gates, g_{14} and g_{17} are adjacentable but overlapped at the qubit layout as shown in Fig. 4. Thus, we need Step 6 to perform only g_{17} separately. If we can swap x_4 and x_6 to modify the layout to be Step 5 in Fig. 5 before performing g_{17} and g_{14}, we can perform g_{17} together with g_{12}, g_{14} and g_{16}.

We can use a SWAP operation to change the location of x_4 and x_6. For a SWAP operation, we need three CNOT gates, so we need three logical operations (three time steps in our model) between x_i and x_j to swap the locations of x_i and x_j. Our idea is that we can do so

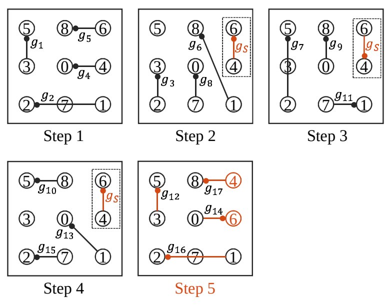


Figure 5: Reduction of steps by the proposed method.

with no overhead (i.e., no increase of the time steps) if such three operations are non-overlapped in some steps before the step we want to do the swap. In the example, we can do the three CNOT operations (denoted as g_s in Fig. 5) for the SWAP in Step 2 to Step 4. In conclusion, the number of the total time steps is decreased to be 5 as Fig. 5.

The detailed optimization procedure based on our idea can be found in Appendix. We compared our method with the previous method [1]; our method can reduce the circuit depth by up to 10%. We show experimental results in Appendix.

3 Conclusion

In this abstract, we propose to use SWAP gates in the middle to reduce the total computational steps for topological quantum computation. Our idea is to change a two dimensional layout in the middle of the circuit so that the computational steps are decreased for the remaining part. Our preliminary experiment show that our idea would be useful. However, our current implementation is naive and thus very time-consuming for large circuits. Thus, we should develop a more efficient method to treat larger circuits, and then perform more comprehensive comparisons with the previous methods.

References

[1] Nurul Ain Binti Adnan, Shigeru Yamashita, Simon J. Devitt, and Kae Nemoto. 2d qubit layout optimization for topological quantum computation. In *Reversible Computation*, pages 176–188, 2014.

[2] Austin G. Fowler, Ashley M. Stephens, and Peter Groszkowski. High threshold universal quantum computation on the surface code. *PHYS.REV.A*, 80:052312, 2009.

[3] Shigeru Yamashita. An optimization problem for topological quantum computation. In *Asian Test Symposium*, pages 61–66. IEEE Computer Society, 2012.

[4] Alexandru Paler, Ilia Polian, Simon J. Devitt, Kae Nemoto. A Fully Fault-Tolerant Representation of Quantum Circuits To appear in *Conference on Reversible Computation*, 2015.

Appendix

Logic Circuit Model for Topological Quantum Computation

Here we explain a logical circuit model for topological quantum computation based on the implementation scheme proposed by [2]. In the scheme, first we prepare a group of physical qubits called *surface code data qubits* to express the quantum state of a single logical qubit. By the way of preparation that there are two types of logical qubit codes: *smooth* and *rough* qubits.

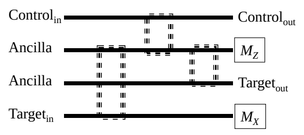
We can apply a specific elementary logical operation to a logical qubit by performing measurements to a specific group of physical qubits that are relevant to the logical qubit code. A group of measurements to a specific group of physical qubit can be illustrated as a drawing in the physical space corresponding to the qubits. Such a group of measurements is called *a braiding operation.* An important property of braiding operations is that multiple braiding operations can be done at the same time if their corresponding drawings are not overlapped with each other in the physical space.

A CNOT operation to logical two qubits that are encoded into different types of qubits (i.e., smooth and rough qubits) can be performed simply by just a single braiding operation between the two coded qubits. However, it is not always possible to encode the target and the control qubits of all the CNOT gates into different types of qubits. For example, if we want to apply three CNOT gates between x_1 and x_2, between x_2 and x_3, and between x_3 and x_1, the three pairs of qubits, (x_1, x_2), (x_2, x_3) and (x_3, x_1) should be encoded into different types; this is impossible.

Therefore, we have to encode logical qubits into the same type (say, smooth) code. Then, a CNOT gate between two smooth qubits can be implemented by three braiding operations by adding two ancilla qubits. This is shown in Fig. 6 where each horizontal line represents a logical qubit (smooth or rough) code, and times goes from the left to the right in the figure. In the figure, only the second line corresponds to a rough qubit code (the other lines corresponds to smooth qubit codes). The dotted rectangles in the figure represent braiding operations. M_X and M_Z in the figure mean measurements in the X and the Z bases, respectively. In the implementation of a CNOT gate, the logical qubit for the target bit is moved to the ancilla (smooth) qubit as shown in the figure. Thus, for one logical qubit, we need to use a pair of two smooth logical qubit codes.

In the following, we consider such a pair of two logical qubit (smooth) codes as one logical qubit. Also we need an additional rough logical qubit code for each CNOT gate as explained, but such additional qubits are irrelevant to our discussion, and so we will simply ignore them in the following. In conclusion, in the following, we consider only one qubit for each logical qubit, and a single CNOT can be implemented in one logical time step although there are many physical qubits and physical braiding operations for one CNOT operation.

Note that braiding operations can be done at the same time if their corresponding drawings are not overlapped



$\text{Control}_{\text{in}}$
$\text{Control}_{\text{out}}$
Ancilla
M_Z
Ancilla
$\text{Target}_{\text{out}}$
$\text{Target}_{\text{in}}$
M_X

Figure 6: A CNOT between two logical qubits of the same type.

with each other in the physical space, and thus we can assume two logical CNOT gates can be done at the same time when the CNOT gates are not overlapped physically. The above is a very intuitive perspective of the model discussed in [2], and thus an exact and detail discussions should be found in that paper.

Our proposed method

Based on our idea, we can propose the following method. Before explaining the method, we introduce the following terminologies: 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. Suppose a gate group such that each gate in the group is **possibly non-overlapped** with all the other gates in the group. Then, there is at least one qubit layout in which we can perform all the gates in the group in one time step. A qubit layout is said to be **good for a gate group** if we can perform all the gates in the group in one time step with the qubit layout.

Now we are ready to describe our method.

Step 1 Make k (k should be as small as possible) gate groups, $G_1, \cdots, G_k$, such that

- if we perform all the gates in $G_1, \cdots, G_k$ with this order, we can perform a circuit that is logically equivalent to the given circuit, and
- all the gates in the same group are all possibly non-overlapped.

Step 2 Find a good initial qubit order by an SA-based method similar to the one [1] such that the qubit order is good for the first i group (i.e., $G_1, \cdots, G_i$) where i should be as many as possible.

Step 3 Iterate the following until all the gates in one time step can be done at the same time. Suppose we have determined the layout such that $G_1, \cdots, G_m$ can be done in one time step.

- Try to insert SWAP gates during $G_1, \cdots, G_m$ such that inserted operations are not overlapped with the existing operations, and we can get a good layout for G_{m+1}.
- If the above is not succeeded, G_{m+1} is divided to several gate groups such that the layout at G_m is good for all the divided groups. In other words, G_{m+1} should be performed with more than one time step.

For Step 1, we move as many gates as possible to the input side of the circuit, and make a possibly non-overlapped gate group as large as possible in the order

Table 1: Comparison with the previous method.

	Groups	Method [1]		Our Method		Reduction
(qubits, gates)		Depth	Parallel	Depth	Parallel	(%)
(9, 100)	40	44	2.27	43	2.33	-2.27
(9, 200)	79	89	2.25	81	2.47	-8.99
(9, 300)	114	139	2.16	133	2.26	-4.32
(9, 400)	140	177	2.26	167	2.40	-5.65
(9, 500)	190	225	2.22	206	2.43	-8.44
(16, 100)	20	26	3.85	25	4.00	-3.85
(16, 200)	53	62	3.23	59	3.39	-4.84
(16, 300)	66	92	3.26	83	3.61	-9.78
(16, 400)	88	125	3.20	121	3.31	-3.20
(16, 500)	115	162	3.09	151	3.31	-6.79
(25, 100)	16	22	4.55	20	5.00	-9.09
(25, 200)	35	49	4.08	45	4.44	-8.16
(25, 300)	47	73	4.11	70	4.29	-4.11
(25, 400)	63	101	3.96	94	4.44	-10.89
(25, 500)	81	129	3.88	122	4.17	-6.98

of $G_1, \cdots, G_k$. This is a greedy method, but we expect this is good because at Step 2, we first try to find a good qubit layout from the input side of the circuit as [1].

Obviously Step 3 is not optimal and thus we need a more sophisticated method; we just show the effectiveness of our idea in the following.

Experimental Results

To compare our method with the previous method [1], we generated random CNOT-based circuits, and applied the two methods to decrease the circuit depth. The result is shown in Table 1. The first and the second columns show the circuit size (the numbers of qubits and gates), and the number of possibly non-overlapped gate groups obtained at Step 1, respectively. Note that the circuit depth cannot be less than the number of the possibly non-overlapped gate groups, and thus the value may be used to see the optimality of finding a good qubit layout in some sense. The third and the fifth columns report the depth of the circuits by the two methods. The column "Parallel" means how many gates can be done parallelly on average. The last column shows how much our method is better than the previous method in terms of the depth. Our method can improve the depth in all the cases.