The Implementation of Deutsch-Jozsa's algorithm on IBM Quantum by Bidirectional Computation

Hyungseok Chang[1,a)] Hidefumi Hiraishi[1,b)] Hiroshi Imai[1,c)]

Keywords: NISQ, Quantum Computation, Deutsch-Jozsa's algorithm, Bidirectional computation

1. Introduction

The development of quantum computation has led to the discovery of various quantum algorithms that outperform existing classical ones. Deutsch and Jozsa found the first algorithm which achieved quantum exponential speed-up [4]. Deutsche-Jozsa's algorithm solve an artificial computational problem to determine whether a given function is constant or balanced. Then Shor showed that prime integer factoring can be solved by quantum computation in polynomial time [14]. Prime integer factoring cannot be solved in polynomial time by classical computation unless P=NP, which is the ground of modern cryptography. Furthermore, Grover proposed a quadratic speed-up algorithm for database search [6], which is applicable for a wide range of combinatorial problems.

While there have been these fruitful theoretical results, only limited experimental results have been obtained. This is because it is still difficult to control quantum manipulation, and thus to obtain correct computational results. To overcome this completely, it is necessary to construct quantum error correction [15]. However, to realize error correction, a large number of qubits are required, and it is considered to take a long time.

Under these circumstances, however, the development of an imperfect form of quantum computer is underway: a quantum device called Noisy Intermediate-Scale Quantum(NISQ) devices. NISQ devices do not have quantum error correction and it causes noise during the calculation process, making it difficult to obtain correct results. On the other hand, it can execute quantum algorithms in small size and some research showed quantum supremacy using NISQ devices [1], which is counteracted by Edwin et al. [13]. Thus, it is possible to conduct experimental research while previous research is limited in theoretical aspect. For this reason, there is a lot of interest in how to use NISQ devices and expectation that if the execution can be done with reduced noise, it can be used practically.

There are various research to implement more efficiently enough to be able to execute on NISQ devices. Nam et al. reduced T gates, which has relatively high error rate in the implementation of the quantum Fourier Transform [11], which is often used as a subroutine in quantum computing. Amico et al. [9] implemented with classical quantum hybrid style on IBM Quantum. They used semi-classical quantum fourier transform [5] to divide circuits into a few parts reducing error caused by noise. This implementation succeeded to factor 15 and 21, however failed to 35. Although it is not realistic to execute on NISQ devices, there are some research to implement Shor's algorithm with less size circuit, such as [2], [12].

Also, it is necessary to compile the circuit we want to execute into the circuit real devices can execute because some gates can only work when the target qubits are adjacent in devices. Therefore, there are many research such as [8] to propose compile method.

While these studies, we have focused on the reversibility of quantum computation. This is the property that inputs can be recovered by tracing backwards from output because reversible computation has no loss of information. Therefore, when we know the input and output of the circuit, it is possible to determine whether an input and output are in a correct input-output relationship executing partitioned circuits that are smaller than the original circuit. In addition, since such a problem would be useful for deterministic problems, we applied it to Deutsch-Joza's algorithm. Since Deutsch-Jozsa's algorithm is the first quantum speed-up algorithm, sometimes it is used as an benchmark for implementing physical devices [7]. Meanwhile, Nagate et al. [10] suggested possibility of application to quantum key distribution. We also confirmed a slight improvement in performance in experiments with real devices, IBM Quantum.

[1] The University of Tokyo
[a)] hsk1226@is.s.u-tokyo.ac.jp
[b)] hiraishi@is.s.u-tokyo.ac.jp
[c)] imai@is.s.u-tokyo.ac.jp

2. Preliminaries

2.1 Deutsch-Jozsa's Algorithm

This algorithm classified whether a given function f is balanced function of constant function with probability one. Constant function which always returns 0 or 1, or a balanced function which returns 0 for the half inputs and returns 1 for the latter half inputs. While classical needs to calculate f exponential time to assuredly classify, this algorithm needs only one time.

Assumed we have a given unitary operation U_f corresponding f which operates $U_f : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$, the quantum algorithm is described as follows.

Algorithm procedure

Inputs: (1) Given a unitary operator U_f. It is promised that $f(x)$ is either constant or balanced function.

Outputs: $\begin{cases} 0...0 & (f \text{ is constant}) \\ \text{otherwise} & (f \text{ is balanced}) \end{cases}$

Procedure:

(1) $|0\rangle^{\otimes n} |1\rangle$

(2) $\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$

(3) $\rightarrow \sum_x (-1)^{f(x)} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$

(4) $\rightarrow \sum_z \sum_x \frac{(-1)^{x \cdot z + f(x)} |z\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$

(5) $\rightarrow$ output; k

In procedure 1, we prepare $n+1$ qubits, and first n qubits are initialized $|0\rangle$ and last one qubit is initialized $|1\rangle$. First n qubit is called as function register and last one is called as answer register. By procedure 2, applying Hadamard gates we create the superposition of function register and nswer register. Then, by procedure 3, applying U_f we obtain the state where $f(x)$ is on the phase of superposition. This is caused by the fact that the answer register has positive and negative phase. Lastly, in procedure 4, Hadamard transformation to function register change the state as above and measure the function register. By the sum of $(-1)^{x \cdot z + f(x)}$, when $f(x)$ is constant, the measured value is always all 0 n bits.

2.2 IBM Quantum

IBM Quantum is the system of quantum devices developed by IBM. It is realized by supercoducting qubits and the number of qubits of devices ranges from 1 to 65. Some of these devices are open to the public via cloud service.

Since pairs of qubits to which 2-qubit gate can be applied are limited for physical reason, quantum circuit needs to be converted to an executable circuit so as to satisfy the physical restriction. This conversion adds SWAP gates to make distant qubits adjacent in order to perform two qubit gates. The circuit before conversion is sometimes called as a logical circuit and the circuit after conversion is called as a physical circuit.

A simulator is also provided by IBM Quantum. It can solve circuits with up to 32 qubits and it enables us to confirm the correct output.

2.3 The reversibility of quantum computing

We introduce two elementary quantum circuits and see how it proceeds reversible computation. Both circuits are consisted of one Toffoli gate and two CNOT gates. The second circuit is inverse version of the first circuit. We figured them in Figure 1 and 2. As an example of reversibility, we will calculate circuits with an input. Given an input $|1\rangle |1\rangle |1\rangle$ to the first circuit. Then, the state changes as follows

$$\begin{aligned} |1\rangle |1\rangle |1\rangle &\rightarrow |1\rangle |1\rangle |0\rangle && \text{First Toffoli gate} \\ &\rightarrow |1\rangle |1\rangle |1\rangle && \text{Second CNOT gate} \\ &\rightarrow |0\rangle |1\rangle |1\rangle && \text{Third CNOT gate} \end{aligned}$$

Thus, we obtain $|0\rangle |1\rangle |1\rangle$ as output. Let us see, when we give the second circuit to this state as input. It transforms the state as

$$\begin{aligned} |0\rangle |1\rangle |1\rangle &\rightarrow |1\rangle |1\rangle |1\rangle && \text{First CNOT gate} \\ &\rightarrow |1\rangle |1\rangle |0\rangle && \text{Second CNOT gate} \\ &\rightarrow |1\rangle |1\rangle |1\rangle && \text{Third Toffoli gate.} \end{aligned}$$

We obtain $|1\rangle |1\rangle |1\rangle$ as output, which is the input of the first circuit. In this way, reversible circuit can obtain input from output reversing itself.

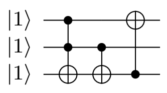

Fig. 1 A circuit with input $|111\rangle$

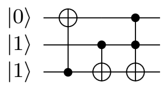

Fig. 2 A circuit with input $|111\rangle$

3. Methods

3.1 Idea of bidirectional computation

Using the reversibility of quantum computing, we considered brand new way to calculate with less noise dividing the circuit into two parts and to confirm the whether an output of the gives circuit is correct for an input or not.

We assumed that we are given a circuit U which outputs ψ_o for an input ψ_i. We first divide the circuit U into two parts U_1 and U_2 satisfying $U = U_1 U_2$ and execute the first half with input ψ_i. Let the output of the first half be a_1. Then we run the circuit $U_2^\dagger$, which is reversed U_2 with ψ_o as an input and We obtain a_2. Here, considered reversibility, a_1 and a_2 should be same. Strictly speaking, it is not the measurement results but the states that should be same.

To compare measurement values correctly, before measuring them, we should return the basis of the state. In this way, we can confirm the correspondence of output and input reducing the size of the circuit executed at one time.

3.2 Bidirectional implementation of Deutsch-Jozsa's algorithm

Not only is the initial state of the Deutsch-Jozsa's algorithm circuit fixed, but the output of the circuit is always 0 when the given function is a constant function. Therefore, comparing the outputs obtained by executing the divided circuits, we can identify a given oracle is a constant function or a balanced function.

Deutsch-Jozsa'a algorithm can be written as Figure 3. We divide this circuit two parts and the circuits are Figure 4 and Figure 5 where $U_f = U_{f_1} U_{f_2}$. Note that both circuits have an additional Hadamard gates. These are added to return the basis of qubits before measurement. Also, since the bottom qubit was not measured and ended as $|-\rangle$ in the original circuit, the bottom qubit of the latter half is prepared as the same state.

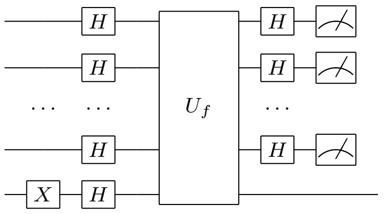

Fig. 3 The circuit of Deutsch-Jozsa's algorithm

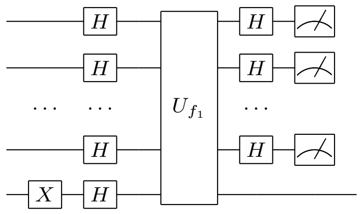

Fig. 4 First half of the divided circuit

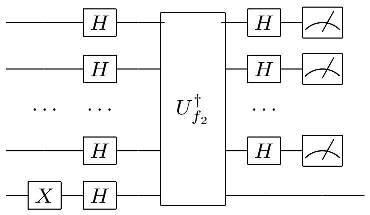

Fig. 5 Latter half of the divided circuit

4. Experiments

We conducted experiments with two oracle functions. Firstly, we will describe the oracles detaily and show results.

4.1 Oracle

We use two functions as oracle. One is a balanced function and the other is a constant function. Both functions are implemented by only CNOT gates and all qubits in function register are connected to answer register so that there are not unused qubits.

The implementation of balanced function is simple. All qubits are connected to answer register once in order. Constant function can be implemented by applying no gates. However, circuits of this implementation is too simple, therefore we use more complex constant functions. We connected all the qubits of function register of constant function to the answer register twice. The arrangement of CNOT gates of constant function follows a certain rule for avoiding too simple implementation. The order of applying CNOT gates follows an array in which the elements of $[1, n/2{+}1, 2, n/2{+}2, ..., n/2, n]$ and $[n, n{-}1, ..., 1]$ are added alternately. For example, when $n = 4$, the order list of qubit which is chosen as a control qubit is [1, 4, 3, 3, 2, 2, 4, 1] as Figure 7.

If we apply one CNOT gate that connects a qubit of function register and the answer register, exactly half of the output will be inverted. Therefore, the first oracle is a balanced function and the second oracle is constant because of the fact that if CNOT gate was applied twice, the output will be returned. The Figure 6 and Figure 7 are respectively the balanced function and the constant function we used.

When we apply bidirectional computing to these circuit, divide them at the center so that the numbers of CNOTS is the same.

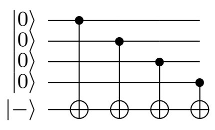

Fig. 6 Oracle1. This is a balanced function when the size of function register is 4.

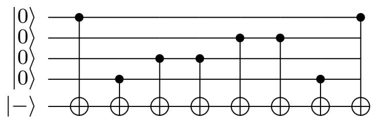

Fig. 7 Oracle2. This is a constant function when the size of function register is 4

4.2 Setting

Using the two oracles described above, we experimented in various size circuit. Here, n means the number of qubits which oracle function has. The device we used is ibmq_toronto, which have 27 qubits and the shots, which is the number of circuit trials, was set to 8192.

Also, we set optimization_lavel, which is a compile parameter to specify the defree of optimization to 0 to prevent the compiler from changing the circuit of oracle function. Also,

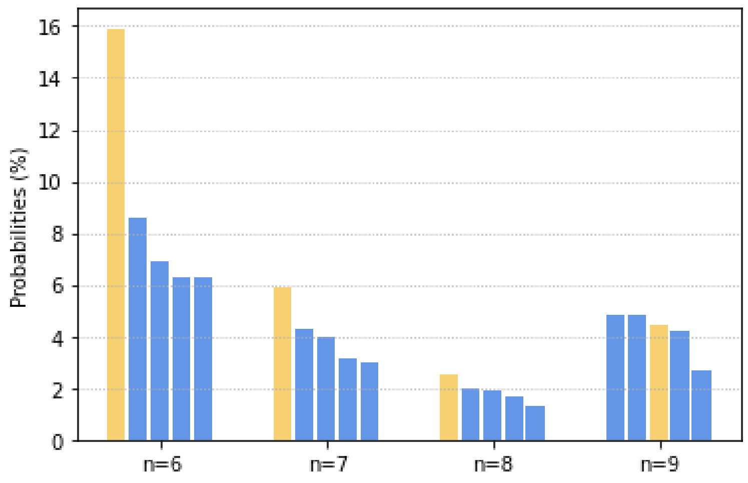

Fig. 8 The results of normal version with oracle 1

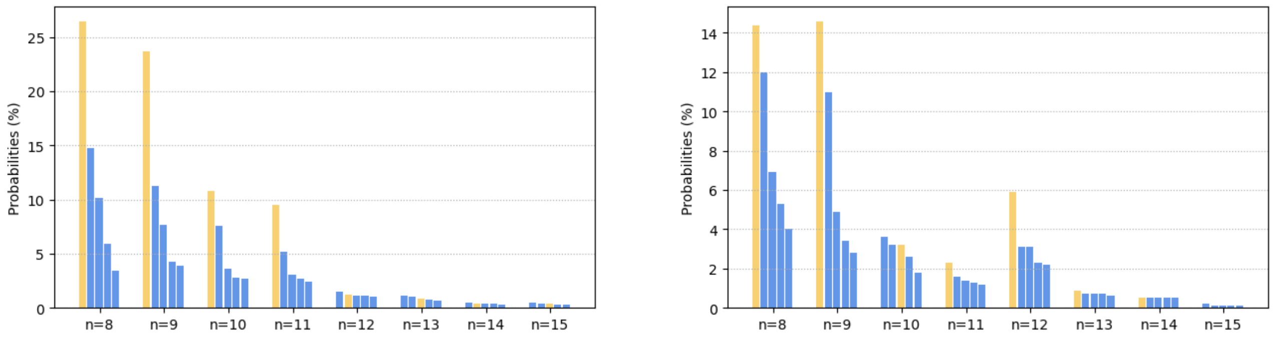

Fig. 9 The results of bidirectional version with oracle 1. The left figure is the result of the first half and the right figure is of the latter half.

since we can choose the physical qubits to be used at compile logical circuit, the logical qubit of the answer register was placed in the physical qubit in the middle of the device. This is because the qubit of answer register is applied CNOT gates with a lot of other qubits. The logical qubits of the function register was assigned to physical qubits as close as possible from the physical qubit of answer register.

4.3 Results

The Figures 8, 9, 10, 11 show the result of each setting. While the theoretical value to be measured was one, other values were also measured because of noise. We selected 5 values in order of probability among the measured values for each n and put their probability on Figure. The probability of the value to be theoretically measured is coloured by yellow. In bidirectional computation case, we have to compare outputs of both circuits to confirm whether both outputs are same or not. To judge the identification correctly, ideal values should be obtained. Here, we assume problems are solved correctly when both circuits outputs theoretical values with the highest probability.

Figure 8, 9 show the result of normal Deutsch-Jozsa's algorithm with oracle 1 and the results of bidirectional version of Deutsch-Jozsa's algorithm with oracle 1. In the case of normal version, when $n = 6$, the difference between the second and first measurement probabilities was nearly doubled, but when $n \geq 7$, the difference became considerably smaller and at $n = 9$, there were more other values were measured. On the other hand, when bidirectional computation is applied, the most often measured values of the first half circuit and the most measured values of the latter half are equal even at $n = 9$ and $n = 11$. At $n = 15$, it is confirmed that it was impossible to output correct answers in both the first half circuit and the latter half circuit. Regarding $n = 10, 12, 13, 14$, either the first half or the latter half always output the correct value but the other output did not output the correct answer.

Next, the result of experiments with oracle 2 is shown on Figure 10, 11. While CNOT gates of oracle 1 were in a simple arrangement, oracle 2 has twice the number of CNOT gates and the order is a little more complicated. Therefore, the experiments with oracle 2 was conducted with smaller n because of higher error. The normal version circuit succeeded up to $n = 4$ and the probability became low from $n = 5$. When $n = 6$, it was almost random probability distribution. The version applied bidirectional computation succeeded when $n = 5$, but only one circuit obtained a proper value at $n = 6$ and both circuits failed to measure correct value when $n = 7$.

4.4 3-partition experiments

In this subsection, we will explain experiments increas-

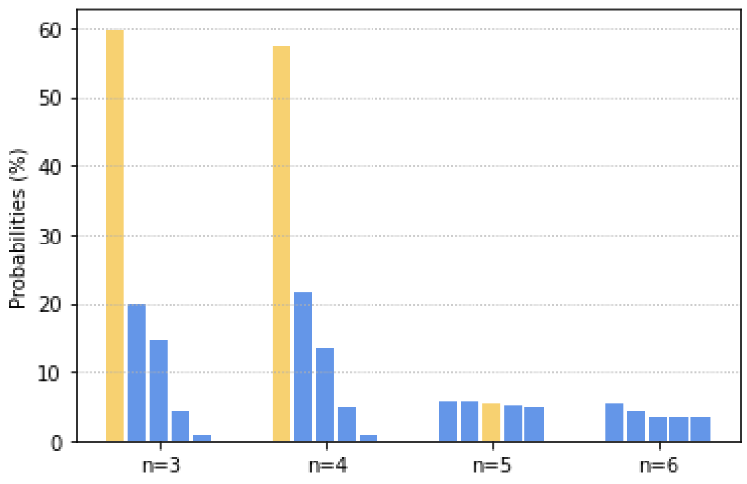

Fig. 10 The results of normal version with oracle 2

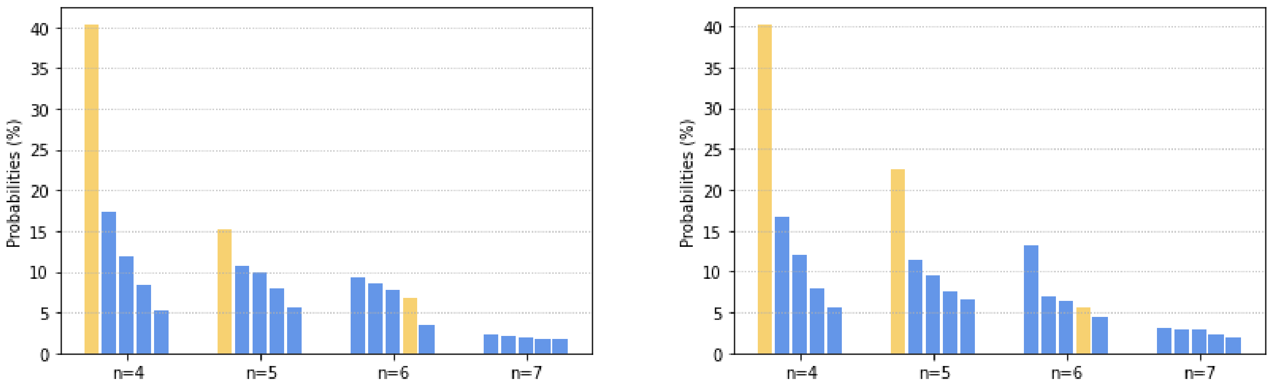

Fig. 11 The results of bidirectional version with oracle 2. The left figure is the result of the first half and the right figure is of the latter half.

ing the number of dividing. We did not apply bidirectional computing to dividing circuits in this time. Instead of bidirectional computing, we recursively use outputs as input. Concretely, when we divide an oracle into three parts, named C_1, C_2, and C_3. Firstly, we execute C_1 circuit and obtain output x_1. Then, we execute C_2 with x_1 as input. Thus, we give the outputs of the previous circuit as input. Due to absence of information loss, we can execute Deutsch-Jozsa's algorithm as above.

We experimented with oracle 2 whose size is 5. While we cannot obtain correct value in case of bidirectional computing about this circuit, Figure 12, 13 show correct values are obtained with the highest probability in each circuit.

4.5 Dsicussion

Experimental results show that there are some improvements in the size of the problem solved by applying bidirectional computation. Since circuits are divided, it was natural that the size of the problem to be solved would be large, but there was difference about improvements in the size. This was considered to be due to the allocation of physical qubits. The circuit of oracle 2 has larger depth than that of oracle 1 and its structure is also complicated. Therefore, since our qubit allocation is fixed, when the circuit become more complex, the size of physical circuit would tend to increase more. In order to overcome this problem, it is necessary to optimize the allocation to minimize the size of the pysical circuit. This optimization can be considered as a future work.

Also, to find applicable problems is next work. In order to apply bidirectional computation, it is necessary to know outputs of circuits. Although problems with these characteristics are limited, it can be possible to apply to decision problem of prime number using Fermat's little theorem. Fermat's little theorem is the theorem that $a^{p-1} \equiv 1 \pmod{p}$ holds where p is a prime and a is coprime integer with p.df For a given number x, we prepare a circuit to calculate $a^{x-1} \equiv 1 \pmod{x}$. This means input x and output 1 are known. Therefore, it would be possible to apply bidirectional computing.

Moreover, some quantum algorithms has multiple outputs. In this case, it is necessary to prepare superposition state as an input of the latter half circuit. However, in such a case, it would be difficult to apply bidirectional computing because the gates for preparing output superposition states might increase. Also, when middle states to compare are superposition states, we have to compare probability distribution. Because it would cost exponentially for the number of qubits to compare probability distribution, so it should be avoided. As an idea to avoid this, it might be useful to

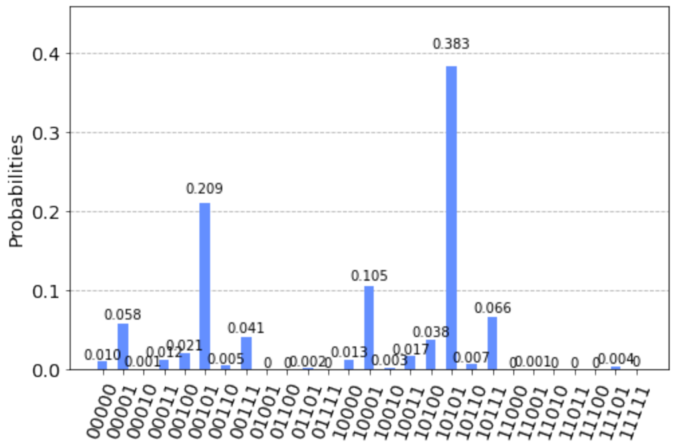

Fig. 12 Outputs of the first circuit. The correct output 1010 is obtained with probability 0.383

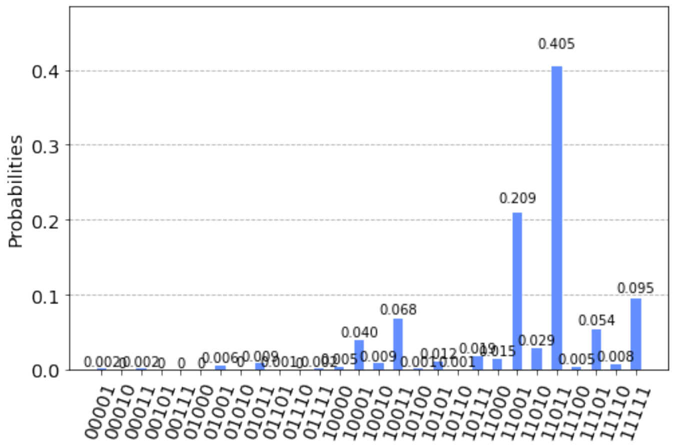
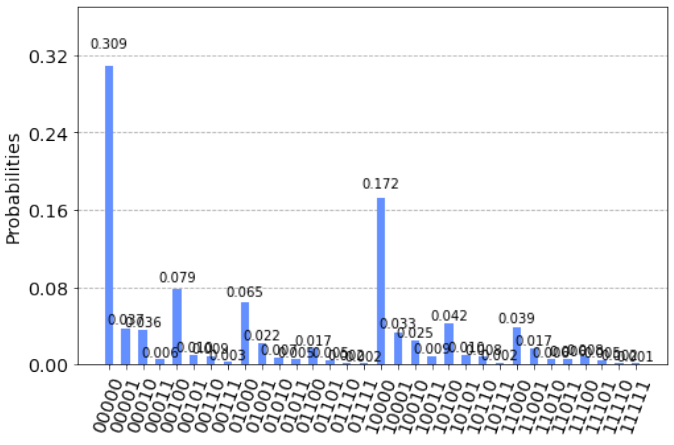

Fig. 13 The left figure is outputs of the second circuit and the right figure is outputs of the third circuit. The correct output of the second circuit 11011 is obtained with probability 0.405 and the correct output of the third circuit 00000 is obtained with probability 0.309

compare the states without measurement such as by SWAP test [3].

5. Conclusion

We proposed bidirectional computation that can divide circuits by taking advantage of the reversibility of quantum computing. Also, we applied bidirectional computation to Deutsch-Jozsa's algorithm and experimented on IBM Quantum, one of NISQ devices. We experimentally confirmed that bidirectional implementation was able to solve slightly larger size of instances.

Acknowledgments Our experiments in this paper were conducted by IBM Quantum computing system as part of the IBM Quantum Hub at University of Tokyo.

References

[1] F. Arute, K. Arya, R. Babbush, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.
[2] S. Beauregard. Circuit for shor's algorithm using 2n+ 3 qubits. *arXiv preprint quant-ph/0205095*, 2002.
[3] H. Buhrman, R. Cleve, T. Watrous, and R. de Wolf. Quantum fingerprint. *Phys. Rev. Lett*, 87:167902, 2001.
[4] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.
[5] R. B. Griffiths and C. Niu. Semiclassical fourier transform for quantum computation. *Physical Review Letters*, 76(17):3228, 1996.
[6] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
[7] S. Gulde, M. Riebe, G. P. T. Lancaster, C. Becher, J. Eschner, H. Häffner, F. Schmidt-Kaler, I. L. Chuang, and R. Blatt. Implementation of the deutsch–jozsa algorithm on an ion-trap quantum computer. *Nature*, 421(6918):48–50, 2003.
[8] A. Hashim, R. K. Naik, A. Morvan, J. Ville, B. Mitchell, J. M. Kreikebaum, M. Davis, E. Smith, C. Iancu, K. P. O'Brien, I. Hincks, J. J. Wallman, J. Emerson, and I. Siddiqi. Randomized compiling for scalable quantum computing on a noisy superconducting quantum processor. *arXiv preprint arXiv:2010.00215*, 2020.
[9] Z. H. Saleem M. Amico and M. Kumph. Experimental study of shor's factoring algorithm using the ibm q experience. *Phys. Rev. A*, 100(1):012305, 2019.
[10] K. Nakata and T. Nakamura. The deutsch-jozsa algorithm can be used for quantum key distribution. *Open Access Library Journal*, 2(08):1, 2015.
[11] Y. Nam, Y. Su, and D. Maslov. Approximate quantum fourier transform with o (n log (n)) t gates. *npj Quantum Information*, 6(1):1–6, 2020.
[12] A. Pavlidis and D. Gizopoulos. Fast quantum modular exponentiation architecture for shor's factorization algorithm.

arXiv preprint arXiv:1207.0511, 2012.

[13] E. Pednault, J. Gunnels, D. Maslov, and J. Gambetta. On "quantum supremacy". Technical report, IBM Rsearch, 2019. https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/, last visited Jan 15, 2021.

[14] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[15] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):R2493, 1995.