Extracting Quantum Power by Using Algorithms and Their Verification

Seiichiro Tani, Seiseki Akibue, and Yuki Takeuchi

Abstract

Quantum computers are expected to increase computational speed compared with current and future computers that work in accordance with the conventional computational principle and to revolutionize information processing. Such an increase in speed is achieved using quantum algorithms that extract computational power from quantum-computer hardware. This article briefly introduces our recent results on quantum algorithms, quantum-circuit optimization to support their efficient implementation, and quantum-circuit verification necessary for their reliable execution.

Keywords: quantum computer, quantum algorithm, quantum information processing

1. Quantum algorithm as core technology

Just as with current computer systems, the performance of quantum-computer systems will heavily depend on what software (i.e., algorithms) we use on quantum-computer hardware. In this sense, quantum algorithms can be considered a core technology for increasing quantum speed. Many studies developed sophisticated quantum-algorithmic techniques and proposed a variety of applications for them with quantum advantage. However, there are still issues regarding quantum algorithms that we need to better understand. To achieve an effective increase in quantum speed, it is also essential to implement quantum algorithms as compact quantum circuits and ensure the reliability of circuit execution.

2. Quantum algorithms for classical problems

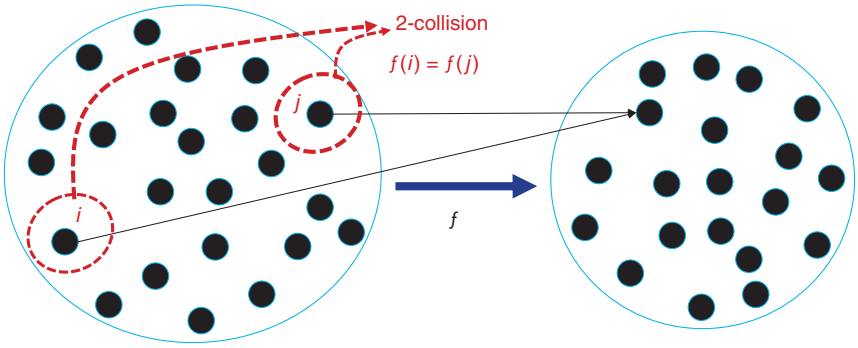
Most problems we encounter in daily life are unrelated to the concept of quantum mechanics, i.e., classical problems. If quantum computers can solve classical problems much faster than any computer based on the conventional computational principle, i.e., any classical computer, they will significantly impact our lives and society.

Previous theoretical studies have revealed many problems that quantum computers can solve much faster [1]. In particular, the collision problem has been central in developing various quantum-algorithmic techniques, such as quantum walk, which increase quantum speed in solving practically important classical problems such as matrix multiplication. We take the collision problem as an example to consider quantum-algorithmic techniques from a different point of view, that is, from the viewpoint of information-communication security.

Quantum computers can be used not only for improving our lives but for malicious attacks such as cracking ciphers. It is thus essential to assess the security of ciphers against strong attacks involving quantum computers (i.e., quantum attacks). For such an assessment, one needs to know how much ability attackers have by devising quantum-attack methods, which are simply quantum algorithms. We devised the fastest quantum-attack method against random hash functions, a core technology in cryptography, on the basis of the knowledge of quantum algorithms that we have accumulated [2].

A hash function takes long data as input and outputs short data. The hash functions used in ciphers are unique in that the input data are hard to infer from the output data. Such hash functions have many applications that require falsification prevention, such as electronic signatures and public-key cryptography.

Let us explain an attack against hash functions



A 2-collision of a hash function f is a pair (i, j) of two distinct elements mapped via f into the same value, i.e., a pair (i, j) such that $f(i) = f(j)$. Similarly, an ℓ-collision is an ℓ-tupple of ℓ distinct elements mapped via f into the same value.

Fig. 1. Collision of hash function.

1. Sample a subset $I \subset [N]$ and compute its image $f(I)$, where $[N] \equiv \{1, \cdots, N\}$.
2. Quantum search the preimage of $f(I)$ for a set $J \subset [N] \setminus I$ and compute its image $f(J)$.
3. Quantum search the preimage of $f(J)$ for an element $k \in [N] \setminus (I \cup J)$.
4. Output 3-collision (i, j, k).

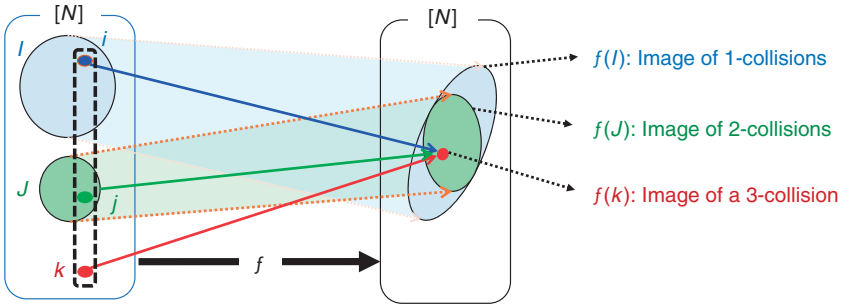


Fig. 2. Quantum algorithm for finding a 3-collision.

using the concept of collisions. We say that multiple elements are a collision of a hash function f if they are mapped via f to the same value. More concretely, we say that ℓ elements mapped via f into the same value are an ℓ-collision. We also say that ℓ is the multiplicity of the collision. In **Fig. 1**, since two elements i, j are mapped into the same value by f, the pair (i, j) is a collision.

One can falsify electronic documents even with message authentication codes if they find a collision of a hash function. It is thus required to assess the computational hardness of finding a collision (i.e., how long it takes to find a collision) before using a hash function as part of a cryptographic system. This assessment requires concrete algorithms for finding a collision. We designed a quantum algorithm that quickly finds an ℓ-collision for any given ℓ (**Fig. 2**). The computation time of this algorithm is optimal in the sense that it achieves the theoretical limit, which makes it possible to rigorously assess the security of cryptographic systems that include hash functions.

Figure 3 compares our algorithm with the previously best algorithm [3] in terms of the number of evaluations of a hash function to find an ℓ-collision of the function, approximating the time taken to execute our algorithm. For every $\ell \geq 3$, our algorithm is superior to the previous one. When $\ell = 2$, both algorithms have the same number of evaluations

ℓ (multiplicity)	2	3	4	5	…	ℓ
Previously best algorithm [3]	$N^{\frac{1}{3}}$	$N^{\frac{4}{9}}$	$N^{\frac{13}{27}}$	$N^{\frac{40}{81}}$	…	$N^{\frac{1}{2}\left(1-\frac{1}{3^{\ell}-1}\right)}$
Our algorithm	$N^{\frac{1}{3}}$	$N^{\frac{3}{7}}$	$N^{\frac{7}{15}}$	$N^{\frac{15}{31}}$	…	$N^{\frac{1}{2}\left(1-\frac{1}{2^{\ell}-1}\right)}$

Fig. 3. Comparison of our algorithm with the previously best algorithm provided in [3] in the number of evaluations of a hash function, approximating the time taken to find an ℓ-collision for every ℓ, where N is the image size of the hash function.

since the previous algorithm achieves the theoretical limit. As a numerical example, let us consider when N is 2000 bits, in which case our algorithm is a billion times faster than the previous one.

To execute a quantum algorithm efficiently, we must consider optimizing the quantum circuit that implements the algorithm. In the next section, we introduce some of our results on quantum-circuit optimization.

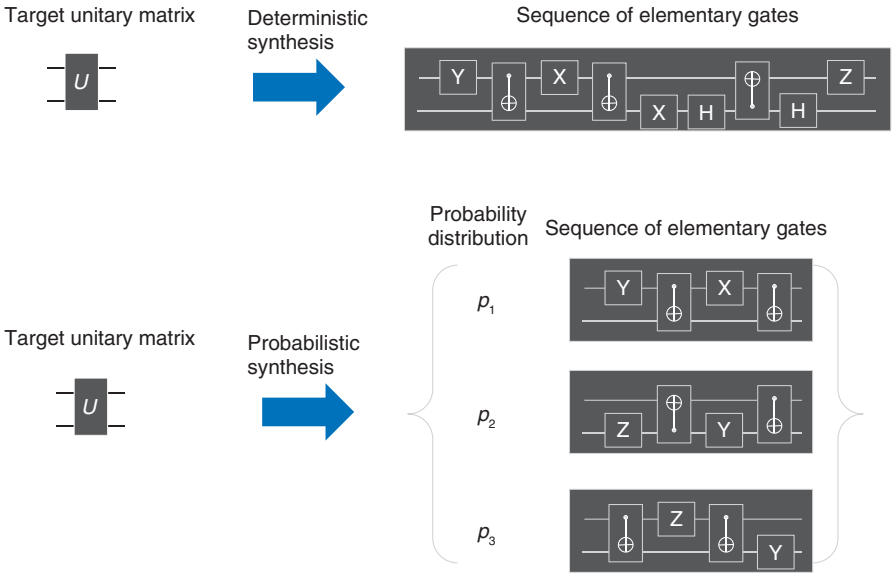
3. Quantum-circuit optimization

We can represent the dynamics of any quantum system, such as photons and electrons, as a unitary matrix. Thus, a 'command' sent by a quantum algorithm to a quantum computer is also essentially represented by a unitary matrix. For example, Shor's algorithm, which efficiently solves the factoring problem, uses the unitary matrix called the quantum Fourier transformation as an essential command. However, we cannot accurately implement such unitary matrices as the corresponding dynamics since quantum systems are fragile against noise. We thus implement a desired unitary matrix by sequentially implementing unitary matrices selected from a finite set (called an elementary gate set), the elements of which are implementable with negligible error.

Hence, it holds that (a) the number of elementary gates = (b) the number of commands × (c) the number of elementary gates used to implement the unitary matrix representing a single command. Since the runtime of a quantum algorithm can be estimated by (a), many studies have been devoted to reducing it. There are two types of such studies: for reducing (b) by exploiting the structure of the algorithm and for reducing (c) without using the structure. It might seem that reducing (c) is unnecessary if we design algorithms by regarding each command as an elementary gate. However, it is necessary to design algorithms independently of the quantum system we use to implement the algorithms since the elementary gate set heavily depends on the system.

To reduce (c), we search for a sequence of elementary gates that approximately implements a target unitary matrix (this procedure is called *synthesis*) since it is impossible to accurately implement an arbitrary unitary matrix by using a sequence of elementary gates chosen from a finite set. We can implement an arbitrary unitary matrix within an arbitrarily small approximation error by using a sufficiently long sequence of appropriate elementary gates. Therefore, traditional synthesis approaches, called *deterministic synthesis*, are used to find the shortest single sequence among all sequences that approximate a target unitary matrix within the desired approximation error. A new synthesis approach, called *probabilistic synthesis*, has been demonstrated to improve the approximation by probabilistically implementing the target unitary matrix. This also indicates that probabilistic synthesis can achieve the desired approximation error with a shorter sequence than the deterministic one (see **Fig. 4**). However, the optimality of current probabilistic-synthesis algorithms was unknown.

We have shown the fundamental limitations on the smallest approximation error achievable with probabilistic synthesis [4]. We have also designed a probabilistic-synthesis algorithm that is efficiently executable and outperforms current probabilistic-synthesis algorithms with respect to approximation error. Numerical simulation showed that our algorithm can reduce (c) by about 50% compared with the best deterministic-synthesis algorithm. The mathematical tools we developed for analyzing optimal probabilistic synthesis are expected to be beneficial to optimizing various types of quantum-classical hybrid information processing.



(Top) Deterministic synthesis is used to find the shortest single sequence among all sequences that achieve a target unitary matrix within the desired approximation error. (Bottom) We can shorten the sequence without increasing the approximation error by probabilistic synthesis.

Fig. 4. Reduction of elementary gates by probabilistic synthesis.

To reliably execute quantum algorithms, it is necessary to verify the operation achieved on a real device by a sequence of elementary gates.

4. Verification methods for quantum computing

Quantum computers are susceptible to errors caused by noise. Two typical techniques for mitigating such errors are *quantum error correction and mitigation* and *verification of quantum computation*. These techniques complement each other. As the name suggests, quantum error correction and mitigation can correct and mitigate the errors, respectively, but they require information such as error models and error probabilities. Verification of quantum computation, however, is applicable to any error, which may be completely unknown, but cannot correct errors and can only determine whether they exist. Verification of quantum computation is a helpful technique for handling errors because only correct (i.e., errorless) computational results can be selected by evaluating the presence or absence of errors.

In cloud quantum computing that enables us to access a remote quantum computer, it should be difficult to characterize errors. Therefore, verification of quantum computation works well for this application. Verification of quantum computation has recently been applied to mitigate errors. The rest of this section describes our recent results on verifying quantum computation.

Although several verification methods have been proposed, almost all are tailored for fault-tolerant universal quantum computers. However, the current or near-term quantum computers called noisy intermediate-scale quantum (NISQ) computers have no error-correction capability. That is why we tried to close this gap by proposing a verification method for NISQ computers [5]. To verify the outputs of NISQ computers, a simple and trivial verification method requires another quantum computer with the same number of qubits. We solved this problem by using the idea of dividing quantum-verification circuits into two small quantum circuits (see **Fig. 5**) and succeeded in efficiently verifying the outputs of NISQ computers with small-scale quantum devices.

5. Outlook

Research on software (i.e., algorithms), as well as hardware for quantum computers, is essential for high-speed quantum computing. With our theoretical expertise, we will explore how to design algorithms that quickly solve fundamental problems on quantum computers and develop theoretical techniques, such

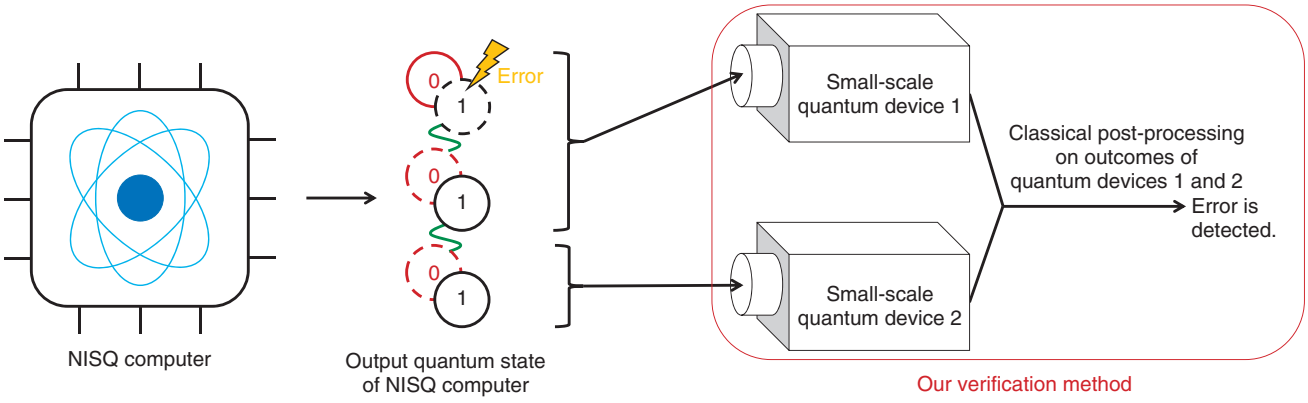


Fig. 5. Our verification method for NISQ computers.

as those presented in this article, to extract the maximum computational power from quantum-computer hardware.

References

[1] A. Montanaro, "Quantum Algorithms: An Overview," npj Quantum Inf., Vol. 2, 15023, 2016. https://doi.org/10.1038/npjqi.2015.23

[2] A. Hosoyamada, Y. Sasaki, S. Tani, and K. Xagawa, "Quantum Algorithm for the Multicollision Problem," Theor. Comput. Sci., Vol. 842, pp. 100–117, 2020. https://doi.org/10.1016/j.tcs.2020.07.039

[3] A. Hosoyamada, Y. Sasaki, and K. Xagawa, "Quantum Multicollision-finding Algorithm," Proc. of the 23rd International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2017), Part II, pp. 179–210, 2017.

[4] S. Akibue, G. Kato, and S. Tani, "Probabilistic Unitary Synthesis with Optimal Accuracy," arXiv:2301.06307, 2023. https://doi.org/10.48550/arXiv.2301.06307

[5] Y. Takeuchi, Y. Takahashi, T. Morimae, and S. Tani, "Divide-and-conquer Verification Method for Noisy Intermediate-scale Quantum Computation," Quantum, Vol. 6, 758, 2022. https://doi.org/10.22331/q-2022-07-07-758

Seiichiro Tani

Distinguished Scientist, Computing Theory Research Group, Media Information Laboratory, NTT Communication Science Laboratories.

He received a B.E. in information science from Kyoto University in 1993 and M.E. and Ph.D. in computer science from the University of Tokyo in 1995 and 2006. He joined NTT LSI Laboratories in 1995 and moved to NTT Network Innovation Laboratories in 1998. Since 2003, he has been studying quantum computing theory at NTT Communication Science Laboratories. He is also an associate member of the Science Council of Japan and visiting professor at the Quantum Computing Unit, International Research Frontiers Initiative, Tokyo Institute of Technology. He was a member of ERATO/SORST Quantum Computing and Information Project, Japan Science and Technology Agency (JST) from 2004 to 2009 and visiting researcher at the Institute for Quantum Computing (IQC), the University of Waterloo, Canada, from 2010 to 2011. He received the Institute of Electronics, Information and Communication Engineers (IEICE) Achievement Award and IEICE Information and Systems Society Best Paper Award. He also received Maejima Hisoka Award. He is a member of Association for Computing Machinery (ACM), the Institute of Electrical and Electronics Engineers (IEEE), IEICE, and Information Processing Society of Japan (IPSJ).

Seiseki Akibue

Researcher, Computing Theory Research Group, Media Information Laboratory, NTT Communication Science Laboratories.

He received a B.E., M.E., and Ph.D. in physics from the University of Tokyo in 2011, 2013, and 2016. He joined NTT Communication Science Laboratories in 2016 and has been studying theoretical topics in distributed quantum computation and classical-quantum hybrid computation. He received Bourses du Gouvernement Français in 2013.

Yuki Takeuchi

Associate Distinguished Researcher, Computing Theory Research Group, Media Information Laboratory, NTT Communication Science Laboratories.

He received a Ph.D. in science from Osaka University in 2018. He joined NTT Communication Science Laboratories as a research associate the same year and was a researcher from 2019 to 2023. Since April 2023, he has been in his current position and engaged in the theoretical investigation of quantum information and is especially interested in the verifiability of quantum computing. He received IPSJ Computer Science Research Award for Young Scientists and Young Scientist Award of the Physical Society of Japan. He is a member of the Physical Society of Japan and IPSJ.