Extracting Success from IBM's 20-Qubit Machines Using Error-Aware Compilation

Shin Nishio[*]
Faculty of Policy Management
Keio University
5322 Endo, Fujisawa City
Kanagawa, Japan
parton@sfc.wide.ad.jp

Yulu Pan[*]
Faculty of Science and Technology
Keio University
3-14-1 Hiyoshi, Kohoku-ku
Yokohama, Japan
pandaman@am.ics.keio.ac.jp

Takahiko Satoh
Quantum Computing Center
Keio University
3-14-1 Hiyoshi, Kohoku-ku
Yokohama, Japan
satoh@sfc.wide.ad.jp

Hideharu Amano
Faculty of Science and Technology
Keio University
3-14-1 Hiyoshi, Kohoku-ku
Yokohama, Japan
hunga@am.ics.keio.ac.jp

Rodney Van Meter
Faculty of Environment and Information Studies
Keio University
5322 Endo, Fujisawa City
Kanagawa, Japan
rdv@sfc.wide.ad.jp

ABSTRACT

NISQ (Noisy, Intermediate-Scale Quantum) computing requires error mitigation to achieve meaningful computation. Our compilation tool development focuses on the fact that the error rates of individual qubits are not equal, with a goal of maximizing the success probability of real-world subroutines such as an adder circuit. We begin by establishing a metric for choosing among possible paths and circuit alternatives for executing gates between variables placed far apart within the processor, and test our approach on two IBM 20-qubit systems named Tokyo and Poughkeepsie. We find that a single-number metric describing the fidelity of individual gates is a useful but imperfect guide.

Our compiler uses this subsystem and maps complete circuits onto the machine using a beam search-based heuristic that will scale as processor and program sizes grow. To evaluate the whole compilation process, we compiled and executed adder circuits, then calculated the KL-divergence (a measure of the distance between two probability distributions). For a circuit within the capabilities of the hardware, our compilation increases estimated success probability and reduces KL-divergence relative to an error-oblivious placement.

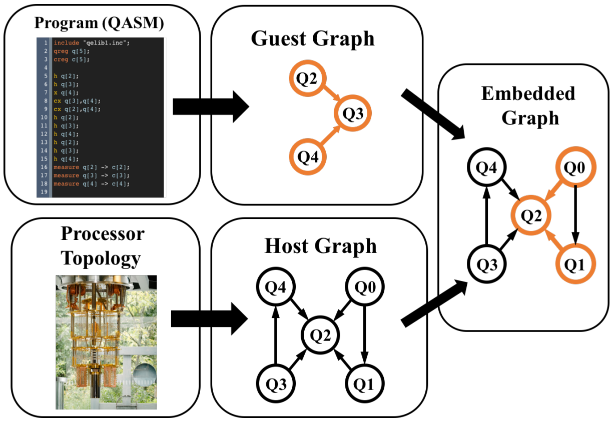
1. INTRODUCTION

Quantum computers exist [1, 2, 3, 4, 5, 6, 7], and once mature, they will surpass classical computers on a range of important problems [8, 9, 10, 11, 12, 13, 14, 15, 16]. Experimental progress in recent years has been rapid, with systems of up to 20 qubits now accessible, and systems ranging from 49 qubits to 128 qubits either undergoing testing in the laboratory or promised for the near future. Quantum machines will overtake classical ones somewhere between 50 and 150 qubits as quantum capability (especially fidelity of gate operations) and improving classical simulation techniques [17, 18] compete, first for demonstration problems then inevitably (we believe) for problems of practical import. This leaves computer engineers with challenges in architecture [19, 20, 21, 22, 23, 24] and programming tools [25, 26, 27, 28, 29, 30]. In particular, because full realization of quantum error correction [31, 32, 33] remains out of reach, in the near term, we must create error-aware compilers for the noisy, intermediate-scale quantum computing era [8].

Compilation varies significantly depending on whether we are compiling for fault-tolerant execution on top of error-corrected logical qubits, or for the "bare metal" machine, and further whether qubits fly (photons), can be moved modest distances (ions), or stay in place (solid-state), and what connectivity constraints are incurred [23, 24, 34]. QEC compilation has been the subject of dramatic advances in recent years (e.g., [35, 36, 37]), but in this paper we focus on bare metal machines, which imposes a different set of goals and constraints. Machine-level compilation involves a series of phases: first, decomposition of higher-level language constructs into a series of one- and two-qubit operations that can be executed on the target system; second, mapping of the variables defined by the programmer to locations in the system, in tandem with generation of appropriate execution of gates between qubits unfortunately placed far apart; third, generation of low-level control for the hardware itself. This paper focuses on the second phase, as shown in Fig. 1. We test our ideas on a specific system, the IBM 20-qubit machine

[*] These authors contributed equally to this work.

named Tokyo [1] and Poughkeepsie, but expect that the ideas will hold for a broad range of solid-state systems.



```
include "qelib1.inc";
qreg q[5];
creg c[5];

h q[2];
h q[3];
x q[4];
cx q[3],q[4];
cx q[2],q[4];
h q[2];
h q[3];
h q[4];
h q[2];
h q[3];
h q[4];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
```

Figure 1: Circuit mapping: Gates between qubits in a program can be represented as edges in a graph. The constraints of the physical system also can be represented in a graph. Compilation embeds the program (guest graph) in the physical topology (host graph) as it assigns qubit variables to locations within the machine.

Tokyo and Poughkeepsie are transmon-style superconducting systems with limited connectivity between qubits, as shown in Fig. 2 [38, 39] and Fig. 3. Each vertex indicates a qubit, and each edge indicates whether or not a multi-quantum bit gate is physically executable. Existing solid-state quantum processors such as this have limitations on the execution of multi-qubit gates such as CNOT because effects such as crosstalk make it undesirable to cross qubit-to-qubit couplers or control wires or to make all-to-all shared buses.

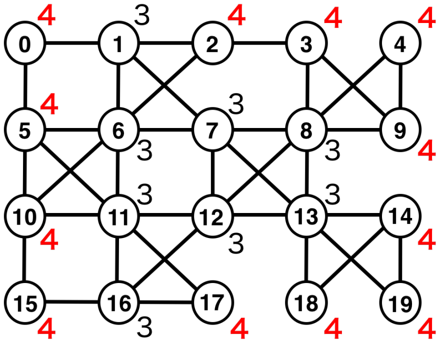


Figure 2: Architecture of IBM's 20-qubit processor named Tokyo. Each vertex represents a qubit, and each edge indicates that a CNOT gate can be executed between the two qubits. The numbers outside the circles indicate the eccentricity, or maximum distance to another qubit.

In existing quantum processors, fidelity degrades due to various errors, affecting the success probability of calculations. The susceptibility to errors of each quantum bit on the quantum processor can vary dramatically. For Tokyo, two-qubit errors dominate. The error rates reported by the Qiskit tools do not distinguish between bit flip errors and phase flip errors (the two primary types of quantum state errors), giving us only a single number to work with. The edges in Fig. 2 have reported error rates ranging from 3% to 12%. (Newer designs for qubit couplers may push those errors to 1% or below [40].) Executing medium-length programs on such a system is challenging, making it imperative not only to minimize the total number of gates, but also to assign program variables to qubits with a careful eye toward which edges will be most used. While a number of projects have focused on gate reduction, we choose to make execution success probability our primary goal.

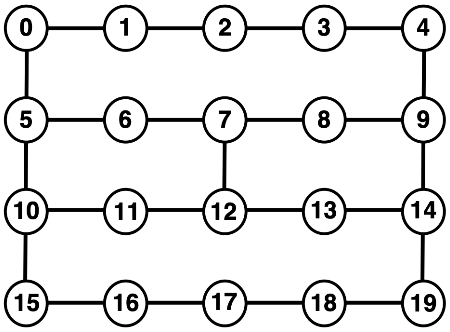


Figure 3: Architecture of IBM's 20-qubit processor named Poughkeepsie. Comparing Tokyo and Poughkeepsie, the number of connections between qubits that can execute CNOT is greatly reduced, but the gate fidelity is improved. as shown in Table 1

To understand the system, we began by conducting a form of system testing known as randomized benchmarking (RB) (described with other background material in Sec. 2). We omit our RB results here because we chose to develop our compilation algorithms using the error rates reported by the Qiskit tools (which are also RB-derived), since extensive testing of the machine itself before every application compilation is inherently impractical.

Our first task is to *assess our ability to accurately predict the success rate* of a given circuit using the product of the individual gate success probabilities as our *estimated success probability* (ESP) (Sec. 3). We find that our exact numbers are off, but we can correctly choose which of two circuits will be *better* on the real machine about two-thirds of the time. We use this to choose how to execute long-distance gates across the chip, selecting from among a group of circuits that are all theoretically equivalent but in practice demonstrate large differences.

The second task is to build on this capability and compile and test complete circuits (Sec. 4). Unfortunately, we know that the circuit mapping phase, similar to place-and-route in hardware design, is NP-complete [41]. Therefore, we use a beam search-based heuristic that includes some stochastic behavior. We compile an adder circuit [42] for input register sizes of one, two, and four qubits, consisting of tens to well over a hundred two-qubit gates. Compilations are

[1] Tokyo is only the name of the machine, it actually resides in Yorktown Heights.

repeated with different random number seeds to assess the performance. For the smaller circuits, our compiler results in executions on the quantum hardware with substantially less divergence from the expected output distributions than simple random placement of variables on the chip. The largest circuits exceed the current capabilities of the system.

We conclude from this data from a real quantum computer that the simple ESP can be used to improve the success probability of quantum computations for a reasonable (classical) compilation cost, and we expect that this will scale to the largest processors projected to exist in the next few years. We suggest that further work on more nuanced metrics will allow still further improvements (Sec. 5).

2. BACKGROUND

Quantum computing research has been appearing in computer architecture venues for over fifteen years, so we dispense with a complete introduction here [19, 20, 21, 22]. Instead, we wish to focus on the key problems with measuring and modeling errors to achieve high success rates in execution, and their relationship to compilation that will lead to using quantum computers to solve problems that classical systems cannot.

2.1 Error model

Our model classifies errors in quantum circuits into three groups. Fig. 4 shows where such errors occur. Rather than a formal quantum mechanical model [43], here we need only the error probabilities.

1. Single-Qubit Gate errors (G) may be unitary bit flips ($|0\rangle \rightarrow |1\rangle$ and $|1\rangle \rightarrow |0\rangle$) or phase flips ($(\alpha|0\rangle + \beta|1\rangle) \rightarrow (\alpha|0\rangle - \beta|1\rangle)$), where $\{\alpha, \beta\} \in \mathbb{C}$ are the complex amplitudes of a qubit's state, or they may be non-unitary errors such as relaxation, in which $|1\rangle$ tends to decay to the lower-energy state $|0\rangle$;

2. Bi-Qubit gate errors (CNOT error) (B) may also flip the value or phase of one or both qubits, and are particularly insidious because they *propagate* errors from one qubit to another; they are critical to creating quantum entanglement and appear in algorithms in large numbers; and

3. State Preparation And Measurement (SPAM) errors (S) are important but can appear only once per qubit in a program execution on the IBM machines, and so have less cumulative impact on success.

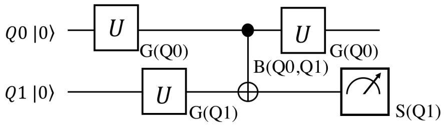


Figure 4: Errors and the gates with which they occur

IBM publishes the value of each of these errors on a per-qubit or per-coupler basis as a backend property of each machine, retrievable using a Qiskit call available to IBM Q Network members. Tokyo is calibrated once a day, using a procedure called *randomized benchmarking*, below.

Beyond these straightforward errors, various crosstalk and resonances within the system cause the state of qubits to affect one another. A complete characterization of this would require extensive *tomography*.

2.2 Tomography

Quantum tomography comes in two primary forms: state tomography and process tomography. This tomography tells us how well we have done at creating our desired state, or how well a particular process (gate or set of gates) works, respectively. It can be used to characterize errors. However, a practical problem arises: the number of possible states naturally grows exponentially with the number of qubits; moreover, we need to test not just for bit flip errors, but also for phase flip errors. Ultimately, we may need to execute the creation of the state or the gate sequence $k3^n$ times for n qubits, where k is a constant determined by the precision we require for the reconstruction and may be thousands. Although only tomography or a similarly rigorous (and heavyweight) procedure can tell us about the state-dependent crosstalk and other factors, it is impractical even at 20 qubits and altogether beyond reach for larger systems. This need led to the creation of randomized benchmarking.

2.3 Randomized Benchmarking

We can assess the fidelity of a set of possible gates (e.g., a commonly used set of gates known as the *Clifford group*) under a broad range of conditions with dense coverage of input states by using the following randomized benchmarking (RB) procedure [44, 45]. Fig. 5 sketches the outline of an RB circuit.

1. Randomly select m gates from the Clifford group and arrange them in any order.

2. Select the Clifford gate (or gates, if performing RB on more than one qubit) that will reverse the operation of the entire preceding sequence of m Clifford gates,

$$C_{m+1} = \left(\prod_{i=1}^{m} C_i \right)^{\dagger}; \tag{1}$$

execute this as the $m+1$th gate.

3. Measure the qubit(s). If the output state is not equal to the input state, then an error has occurred somewhere in the whole circuit.

4. Change the number m and perform steps 1 to 3 again.

By performing the above operation for various values of m, the attenuation of the fidelity as a function of the circuit length can be fitted. In this way, it is possible to estimate the average value of errors per gate included in the gate set. Of course, SPAM (state preparation and measurement) errors are included in all the quantum circuits, but this effect can be compensated for by calculating the attenuation.

The gate error rates reported via Qiskit are derived using a procedure similar to this, executed daily on Tokyo. These values are not a complete description of the behavior of the

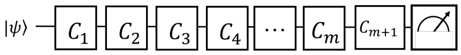


Figure 5: Randomized Benchmarking

system under all circumstances, but they are valuable and derived at reasonable cost. One of our primary research questions, then, is whether such numbers are *good enough* to enable us to extract the maximum success probability from the machine.

2.4 Architecture-Aware Compilation

Adapting to the topology of the processor using architecture-aware circuit design and compilation has been a research topic since the early proposals for large-scale systems and applications [20, 21, 46, 47]. Those early studies focused on the impact on execution time.

Recently, researchers have begun paying attention to fidelity improvements. For example, in the study by Zulehner *et al.*, the cost function is computed by assigning a cost of 10 to a two-qubit gate, versus a cost of one for a single qubit gate corresponding roughly to difference in error penalty [48]. However, in that work, no consideration is given to the qubit-to-qubit variance in error rate. Tannu *et al.* and Finigan *et al.* maximize ESP considering errors per quantum bit [49, 50]. Finigan's research showed that ESP was improved by verifying the optimized circuit with IBM's 16-qubit machine.

3. LONG-DISTANCE CNOTS AND MAKING CHOICES

Each G, B, and S error above will have an error rate ε dependent on type and location. Once assigned locations, we can compose our Estimated Success Probability for a sequence of gates to be

$$ESP = \prod_i (1-\varepsilon_i). \tag{2}$$

To evaluate this score function, we conducted the following two experiments.

3.1 Path selection for remote CNOT

Most quantum algorithms use many CNOT gates. In an architecture such as Fig. 2, it is not always possible to arrange the control qubit and the target qubit close to each other. If not, it is necessary to connect the qubits with the remote CNOT gate, or to move via SWAP to the vicinity. We define the following problem and assess our ability to select the optimal solution using ESP by experiment on Tokyo.

Problem 1: CNOT Path Selection

The programmer wishes to execute the circuit shown in Fig. 6. When the starting point (control qubit) and the end point (target qubit) for the Bi-Qubit gate (CNOT) are not neighbors, which path is the highest fidelity?

We conducted the following experiments.

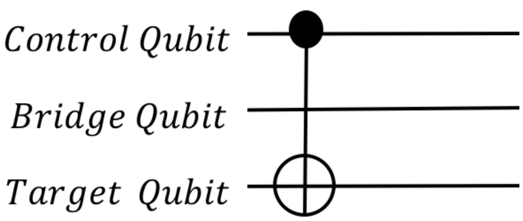


Figure 6: Circuit A, a CNOT gate skipping across another qubit

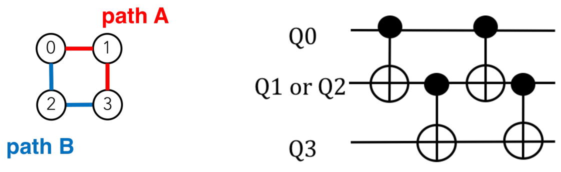
1. Select the path predicted to complete with the highest fidelity using our proposed score function (ESP). Fig. 7a, Fig. 8a, and Fig. 9a show options for 2, 3, and 4 hops path selection. The corresponding circuits are in Fig. 7b, Fig. 8b, and Fig. 9b.

2. Execute both paths on the actual machine for 1000 shots. For simplicity, $|000\rangle$ is used as the input state. If the state of the target qubit is not equal to the state of the control qubit($|0\rangle$), the path can be regarded as including errors. As a result, the optimum path (success probability is the highest) is determined.

3. If the optimal path matches the path selected in step 1, path selection can be regarded as successful.

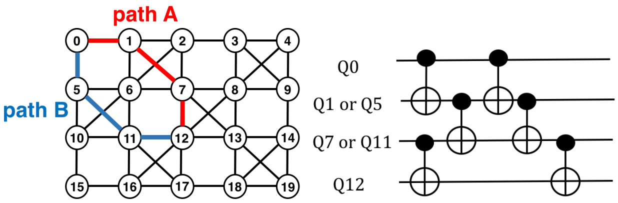
As shown in Fig. 2, the maximum eccentricity of each qubit is 4 on the largest processor made available by IBM (as of December 5, 2018). The eccentricity is the distance from a certain vertex to the furthest vertex on the same graph. Since it is sufficient to connect CNOT using routes of the number of hops, experiments were conducted up to 4 hops.

Some hardware Limitation



(a) 2-hop path selection (b) 2-hop path circuit

Figure 7: (a) is two examples of CNOT path selection for 2 hops. (b) is circuit for (a). Both ends are the same qubits, and only the bridge qubit is different.



(a) 3-hop path selection (b) 3-hop path circuit

Figure 8: As in the case of 2 hops, (a) shows two examples of 3-hop paths, and (b) shows a circuit for this path.

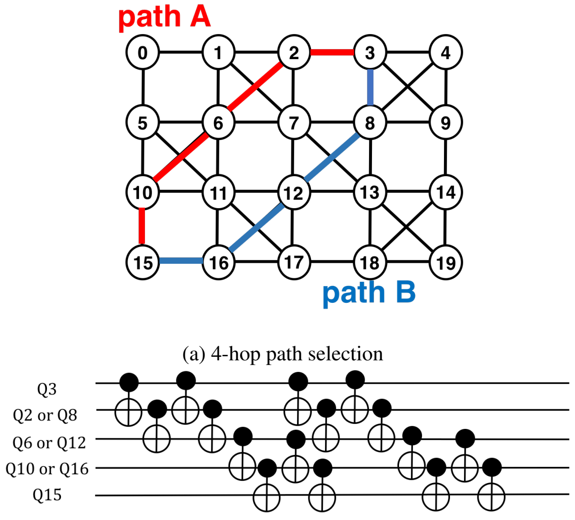
Result 1

ESP was able to select the better of two routes with accuracy of 70% at 2 hops, 66.6% at 3 hops and 62.5% at 4 hops.

Of course, this leaves up to 37.5% of cases in which ESP leads us to the wrong choice. Possible reasons include:

1. The reported error value used to calculate ESP contains both bit-flip and phase-flip errors, but our experiments reveal only bit-flip errors.
2. Long-distance resonances and cross-talk within the system mean that a series of gates doesn't behave the same as an isolated gate.
3. The actual state of the machine drifts faster than the calibration (Randomized Benchmarking) data is updated, so the gate error rates we use may be out of date.

We don't believe that reason 3 is an important effect. Our future plans include tests that will distinguish the relative importance of reasons 1 and 2.



(a) 4-hop path selection

(b) 4-hop path circuit

Figure 9: (a) shows two examples of 4-hop paths for a CNOT between Q3 and Q15, and (b) shows a circuit for this path.

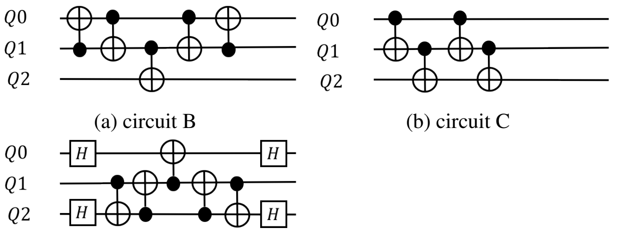
3.2 Circuit Selection for Remote CNOT

Problem 2: Circuit Selection

The programmer wishes to execute the circuit shown in Fig. 6. When the CNOT gate is executed with remote qubits, there are multiple theoretically equivalent circuits as shown in Fig. 10.
However, considering errors, these are not necessarily equivalent. Which circuit has the highest fidelity?

We conducted the following experiments on Tokyo.

1. Select the circuit predicted to have the highest fidelity from B, C, and D in Fig. 10 using our proposed score function (ESP).



(a) circuit B

(b) circuit C

(c) circuit D

Figure 10: Circuits equivalent to the circuit shown in Fig. 6

2. Execute all three circuits on IBM Q20 Tokyo for 1000 shots. For simplicity, $|000\rangle$ is used as the input state.

   If the state of the target qubit is not equal to the state of the control qubit ($|0\rangle$), the circuit has incurred an error.

   As a result, the optimum path (one with highest success probability) is determined.

3. If the optimal circuit matches the circuit selected in step 1, path selection can be regarded as successful.

Result 2

ESP was able to select the optimal circuit among three candidates with accuracy of 40%.

ESP selected Circuit B 35% of the time, circuit C 10% of the time, and circuit D the remaining 55%. Experimentally, the optimal circuit was B for 50%, C for 15%, and D for 35%. ESP is better than random, which would select the optimal circuit only $\frac{1}{3}$ of the time.

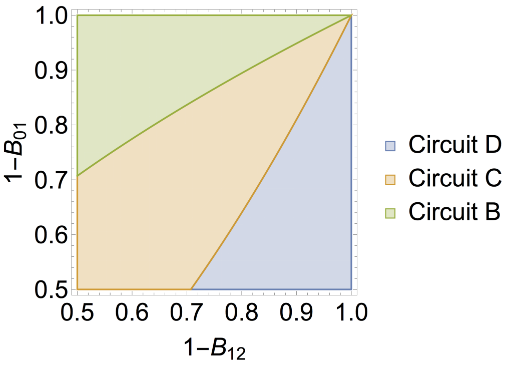


Figure 11: Estimated Best 2-hop Circuit
If the error rates on the two edges are balanced, circuit C will be the best choice. More often on Tokyo, B or D is best.

When ESP is used, which circuit is selected based on the magnitude of Bi-Qubit gate error (B) as shown in Fig.11. B_{01} is the Bi-Qubit gate error for Q0 and Q1. B_{12} is the Bi-Qubit gate error for Q1 and Q2. Intuitively, because C is one fewer two-qubit gate, we would expect it to be the best most of the time. In our experimental results, circuit C is chosen less often than B and D. This is because the difference between B_{01} and B_{12} is large in the current processor.

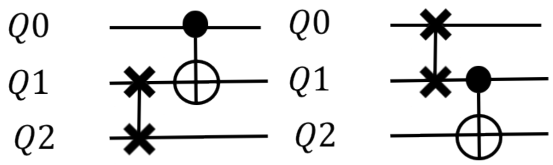
If only the number of CNOT gates is used to predict fidelity of the quantum circuit, this will be the best choice only 15 %

of the time.

Problem 2′: Circuit Selection (including SWAP)

Perform the same circuit as Problem 2 with additional circuit options, including permitting the relocation of a qubit, using the SWAP gate shown in Fig. 12. Which circuit is the highest fidelity?

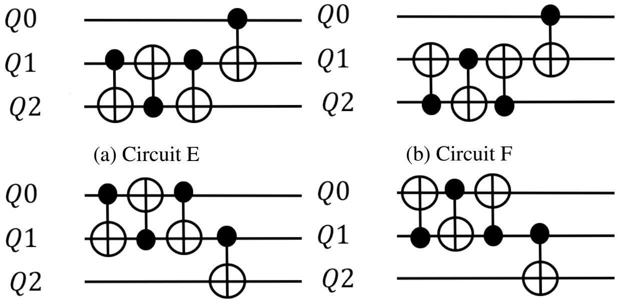
Because Tokyo doesn't implement a SWAP gate natively, we decompose a SWAP into three CNOT gates. Considering the direction of CNOT gates, we have two possible decompositions. Circuits shown in Fig. 12a can be implemented into circuits shown in Fig. 13a and Fig. 13b. Circuits shown in Fig. 12b can be implemented into circuits shown in Fig. 13c and Fig. 13d. We conducted the experiments in the same way as problem 2, using circuits B to H.



(a) SWAP Q1, Q2

(b) SWAP Q0, Q1

Figure 12: Circuits including SWAP gates



(a) Circuit E

(b) Circuit F

(c) Circuit G

(d) Circuit H

Figure 13: Circuits equivalent to Fig. 12. SWAP gate can be implemented with three CNOT gates.

Result 2′

ESP was able to select the optimum circuit from among seven candidates 25% of the time. Fig.14 shows the fidelity of selected circuit relative to the other candidate circuits.

The average of the ESP of the circuit selected was 0.8255. On the other hand, the fidelity obtained by executing the circuit selected by ESP on Tokyo was 0.8609. With seven candidates to choose from, ESP's 25% is better than selecting randomly, which will choose the optimal circuit only $\frac{1}{7}$th of the time. In most cases, we chose an above-average circuit.

The fidelity of the selected circuit greatly exceeds ESP. This may be because we used only the 0 state at the input of this experiment, whereas the single-qubit gate error used to compute ESP is for a dense gate set. Alternatively, this may be because our experiments are more limited than full tomography and do not reveal phase flip errors.

In order to show this, we experimented with $|+++\rangle$ input state in Problem 3′.

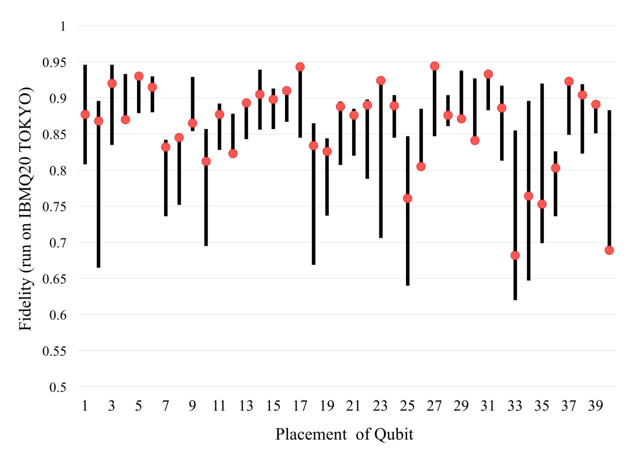


Figure 14: Fidelity of Selected Circuit
Seven candidate circuits (circuits B to H) are evaluated at forty different locations across the surface of Tokyo. Red points represent the fidelity of the circuit selected using ESP. The top of the black bar shows the highest success probability achieved by any of the seven and the bottom of the bar shows the lowest one.

Problem 3: Circuit Selection (on Poughkeepsie)

Perform the same circuit as Problem2âĂŹ on the other 20-qubit processor called Poughkeepsie. Is there any change in the reliability of circuit selection by ESP?

IBM released a quantum processor called Poughkeepsie to IBMQ network members in the winter of 2018. Table 1 shows the performance specifications of Poughkeepsie.

Result 3

ESP was able to select the optimum circuit from among seven candidates 43% of the time. Fig.15 shows the fidelity of selected circuit relative to the other candidate circuits.

Among the options, the average of the Fidelity of the circuit with the best result was 0.8492. On the other hand, the average of Fidelity of the execution result of the circuit selected by ESP was 0.8405, and the difference between these was 0.87%. This is extremely small compared to 3.5% in tokyo, and it can be said that the reliability is improved.

The main factor is considered to be the improvement in the reliability of fitting function of the Randomized Benchmarking. That is, there is a possibility that the value of the error is less likely to fluctuate.

Table 1: the performance specifications of two IBMQ 20-qubit systems named Tokyo and Poughkeepsie

	tokyo (2nd gen 20-qubit system)	poughkeepsie (3rd gen 20-qubit system)
Mean of Two-qubit (CNOT) error rates $\times 10^{-2}$	2.84	2.25
best	1.47	1.11
worst	7.12	6.11
Mean of Single-qubit error rates $\times 10^{-3}$	1.99	1.07
best	0.64	0.52
worst	6.09	2.77

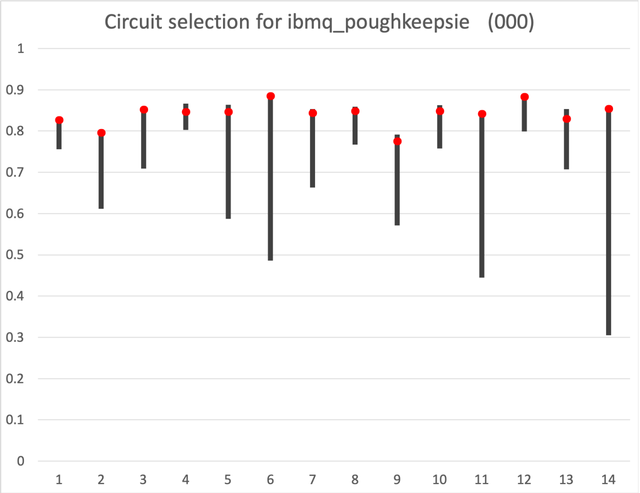


Figure 15: Fidelity of Selected Circuit on Poughkeepsie The experimental results were plotted in the same manner as in Fig.14. The upper limit of the bar and the red dot is close.

Problem 3′: Circuit Selection ($|{+}{+}{+}\rangle$)

Perform the same circuit as Problem 3 but with different initial input states on Poughkeepsie. Is there any change in the reliability of circuit selection by ESP?

The same circuit selection is performed by changing the quantum state before CNOT execution to a state other than 000. The state was used for simplicity. As an alternative to the X-axis measurement, the Z measurement was performed after the H gate was performed after the CNOT. The processor used Poughkeepsie to facilitate comparison with result 3

Result 3′

ESP was able to select the optimum circuit from among seven candidates 36% of the time. Fig.16 shows the fidelity of selected circuit relative to the other candidate circuits.

Among the options, the average of the Fidelity of the circuit with the best result was 0.8077. On the other hand, the average of Fidelity of the execution result of the circuit selected by ESP was 0.7966, and the difference between these was 1.10%.

A significant improvement in accuracy is seen over result 2 ’, but a little less accurate than result 3. Also, the overall fidelity itself is low.

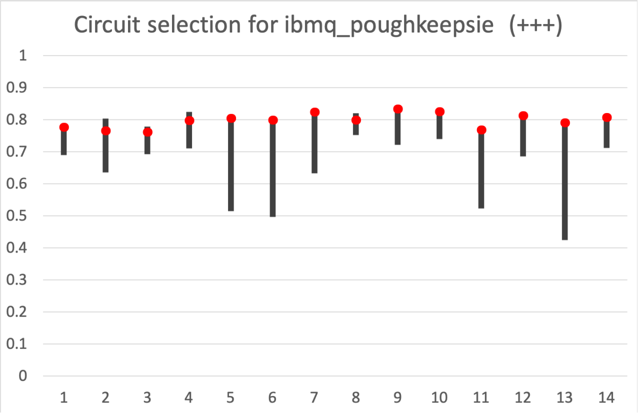


Figure 16: Fidelity of Selected Circuit on Poughkeepsie The experimental results were plotted in the same manner as in Fig.14. The upper limit of the bar and the red dot is close.

Since the number of gates is increasing, it is estimated that the prediction accuracy has fallen.

4. COMPILING COMPLETE PROGRAMS

In the previous section, we discuss several possible realizations of a single CNOT gate on IBM’s QX architecture and compare estimates and observational results of their probability of success. A general purpose compiler, however, is required to combine such realizations in order to deal with much more complex quantum circuits with many CNOT gates. Since there are several possible combinations of realizing the circuit, the compiler must be able to distinguish more reliable realizations from the others. To give a metric of several realizations, we extend the definition of Eq. 2’s Estimated Success Probability; we define the Estimated Success Probability of a circuit C as the product of ESPs of its components:

$$ESP(C) = \prod_{g \in C} ESP(g) = \prod_{g \in C} (1 - \varepsilon_g). \qquad (3)$$

In this model, we assumed that

1. each gate and measurement in a circuit either succeeds completely or fails to execute and stops the whole execution; and
2. the probabilities of such failures are independent of each other and depend only on the physical qubits on which the operations act

With this formalization, the compiler's task is defined to be the maximization of $ESP(C)$, and we apply a combinatorial optimization method described in the following subsections to this end.

Although Equation 3 is simple and easy to calculate for the optimization process, we cannot use it when evaluating optimized circuits on real NISQ computers. The problem is that, in general, the output of a quantum circuit is not a single value but a probability distribution. Imagine a quantum circuit with Hadamard gates and measurements for each qubit. Since this circuit produces a uniform distribution over all of the possible measurements, we cannot figure out whether a single-shot measurement of the circuit on a quantum device is a successful execution or not. Instead, we have to sample multiple shots from the circuit and compare the resulting empirical distribution against the ideal distribution a noiseless quantum computer would produce. We propose using the KL divergence between the empirical and ideal distributions as a distribution-based measure of a compiled circuit and studied the relationship between ESP and KL divergence on IBM Q20 Tokyo machine.

4.1 Search Space

In this section, we scrutinize the search space of the ESP maximization problem.

Each state we visit in the optimization process is parameterized by two parts: the execution state of gates and the qubit mapping. A gate execution state describes which gates have already been executed and which have not. Let N be the number of gates, then the number of execution states is 2^N. Moreover, for each gate execution state, there exists ${}_QP_V = Q \times (Q-1) \times \cdots \times (Q-V+1)$ arrangements of quantum variables onto the physical qubits [51].

The search space of this problem can be visualized as a collection of hypercubes. Each hypercube consists of a complete set of execution states, and there is one corresponding hypercube for each qubit mapping. There are two types of transitions between the states: gate execution and swap insertion.

Execution of a gate sets an execution flag of the gate execution state, moving the state upward from the $000\ldots0$ vertex toward $111\ldots1$ in the same hypercube. On the other hand, insertion of SWAP gates does not change the execution state while it changes the qubit mapping. Thus SWAP insertion moves the state laterally, making a jump to another hypercube. Under this setting, each path from a state at the bottom to a state at the top constitutes a possible compiled circuit, and the compiler's task can be interpreted as finding the best path from the bottom to the top.

As discussed in the prior sections, CNOT gates can be executed if and only if the control qubit and the target qubit are adjacent to each other in the physical topology. This adjacency constraint invalidates some edges. Moreover, to preserve the logical function of the given quantum circuit during the compilation, some gates must be executed before other gates. We call this constraint of order a gate dependency. Gate dependencies forbid some transition edges in hypercubes.

A naive brute-force algorithm will traverse $2^N \times {}_QP_V$ states in the worst case, which is infeasible. Additionally, [41] shows that finding the optimal path is an NP-hard problem when the optimization target is the number of inserted SWAP gates. Thus, finding the optimal solution would be intractable in practice.

4.2 The Optimization Algorithm

Therefore, we propose a beam-search based heuristic optimization algorithm. Beam search is a modified version of breadth-first search where at each depth, instead of adopting all candidates as breadth-first search does, only a fixed number of promising states are searched. The number of candidates kept is called the beam width B. As the beam width grows, the number of states that the algorithm visits increases and better solutions will be found, at the cost of higher time and space complexity. Specifically, if $B = 1$, beam search is identical to greedy search, and to breadth-first search if $B \to \infty$.

In addition, the choice of initial qubit mappings is crucial because it corresponds to the initial states of the beam search. One possible way to generate initial mappings is to assign quantum variables to physical qubits randomly. Besides that, several papers [41, 51] have proposed heuristic methods to generate proper mappings in order to get shorter or more shallow circuits. Our approach combines these methods. In other words, we start the search with an initial qubit mapping computed by a heuristic method in addition to random initial mappings. This approach enjoys the performance of heuristics while also exploring the possibility of nicer configurations through chance. We denote the number of random initial mappings as M.

We show the compilation algorithm in Algorithm 1. In this algorithm, the number of executed gates inside all the states in the next state set (S_{i+1}) is incremented by one from those of the previous state set (S_i). Therefore, after N iterations of the outermost loop, S_N contains states with all gates executed.

Inside the loop, we iterate the state set S_i and update it by the new set S_{i+1}. Since the innermost loop only checks N gates, the inside of the loop runs $N|S_i|$ times for each i. Inside the loop, the dependency check (line 7) can be done in constant time with auxiliary information encoded in $state$. Moreover, the BEST_SWAP function call (line 9) is also a constant time operation with caching. Therefore, the most compute-intensive part of the algorithm is calculating the score of the state at line 12. Since UPDATE_SCORE takes $O(N)$ time (described in Section 4.3.1), the complexity of the update is $O(N^2|S_i|)$ for each i. On the first iteration, $|S_0| = M$ because S_0 is filled with initial states and when $i > 0$, $|S_i| \leq B$ holds because beam search prunes the states. Therefore, the total complexity of this algorithm is $O(N^2M + N^3B)$, where the first term corresponds to the first iteration and the second term to the rest.

We can ignore the complexity of pruning (line 17) as pruning takes $O(|S_{i+1}|) = O(N|S_i|)$ time with the Floyd-Rivest algorithm [52], which has lower complexity than computing the score of each new state.

4.3 Subroutines for Compilation Algorithm

In the following sections, we describe the three subroutines which appear in the compilation algorithm.

4.3.1 *Scoring States*

Algorithm 1 Compilation Algorithm

```
function COMPILE($gates, topology$)
    $N \leftarrow |gates|$
    $S_0$ ← initialize states with INITIAL_MAPPING($gates, topology$)
    for $i = 0$ upto $N$ (exclusive) do
        $S_{i+1} \leftarrow \{\}$
        for all $state \in S_i$ do
            for all $g \in gates$ do
                if dependency of $g$ is satisfied then
                    $qubits \leftarrow state.mapping[g.qubits]$
                    $swap \leftarrow$ BEST_SWAP($qubits, topology$)
                    $state' \leftarrow state$ with $swap$ inserted and $g$ executed
                    $state'.esp \leftarrow state.esp \times ESP(swap) \times ESP(g)$
                    UPDATE_SCORE($state', gates, topology$)
                    $S_{i+1} \leftarrow S_{i+1} \cup \{state'\}$
                end if
            end for
        end for
        $S_{i+1} \leftarrow$ top-$B$ states of $S_{i+1}$
    end for
    return the best $state \in S_N$
end function
```

Algorithm 2 Update Score of a State

```
function UPDATE_SCORE($state, gates, topology$)
    $score \leftarrow state.esp$
    for all $g \in gates$ do
        if $g$ has not been executed then
            $swap \leftarrow$ BEST_SWAP($state.mapping, g, topology$)
            $score \leftarrow score \times ESP(swap) \times ESP(g)$
        end if
    end for
    $state'.score \leftarrow score$
end function
```

The function UPDATE_SCORE computes the score of states. Pruning removes states with lower scores calculated by this function.

The simplest score function is just using the ESP of executed gates. Incorporating information of future gates can further improve the selection of states. In this compiler, we multiply current ESP with the imaginary ESPs of each gate to be executed. The imaginary ESP is the Estimated Success Probability of SWAP gates to satisfy the adjacency constraint under the current mapping times the success probability of the gate. Imaginary ESP can differ from actual ESP because the current mapping can differ from mappings at the time of gate execution due to the insertions of SWAP gates required for the execution of former gates.

We show this computation in Algorithm 2. This algorithm enumerates unexecuted gates, so the complexity of computing the score is $O(N)$.

4.3.2 *Heuristic Initial Mapping: Greatest Connecting Edge Mapping*

As mentioned above, our compiler adopts random and heuristic initial mappings as the starting points of search. The core idea of our initial heuristic mapping is to map more significant edges between the quantum variables to less noisy edges in the physical topology. This idea comes from the fact that error rates of CNOT gates are a magnitude higher than those of single qubit gates on the IBM QX architecture [39].

This heuristic starts with counting the number of CNOT gates executed over for each pair of quantum variables. These numbers constitute a guest graph, a graph whose vertices correspond to each quantum variable and edges to the number of CNOT gates over them.

Next, we scan the edges of the guest graph in a similar manner to Prim's algorithm [53]. However, instead of scanning from the lowest to greatest, this algorithm scans from the greatest to lowest, constructing a maximum spanning tree over the guest graph.

During the first scan, both endpoints of the chosen edge are not mapped yet. Thus, we pick the least noisy edge from the physical topology and map those two variables to this edge.

In the following scans, one endpoint of the chosen edge is already mapped, and the other is not, so the chosen edge is the greatest edge connecting the set of mapped variables and that of unmapped variables. The name of the heuristics comes from this. Since one of the variables is already mapped, it would be a good idea to map the other to the adjacent qubit in the physical topology. So we search for the free qubit adjacent to the qubit the endpoint is mapped to and adopt the qubit with the lowest CNOT error rate between the qubits. We show pseudo-code in Algorithm 3. During the scan, a mapped qubit may not have free qubits adjacent to itself. In that case, this algorithm skips that edge and maps unmapped variables to the remaining qubits randomly at the end.

4.3.3 *Finding Best Swap Sequence*

In our compiler, we consider only the realizations of CNOT gates via the insertion of a sequence of SWAP gates. The compiler needs to know how to insert a SWAP sequence to change the mapping such that the adjacency constraint is satisfied in the new mapping. BEST_SWAP function computes such SWAP sequences for each physical qubit in the topology.

The algorithm is shown in Algorithm 4. The main idea of this algorithm is to find the best "meeting edge" for the given pair of qubits. Each qubit will be moved to the endpoints of the meeting edge, and finally, we execute the CNOT gate over the edge. The ESP of this movement plus the CNOT gate can be calculated as the product of ESP of the SWAP sequences and ESP of the CNOT gate over the meeting edge. We can run the shortest path algorithm over the physical topology to find the SWAP sequence with optimal ESP.

In Algorithm 4, we run the shortest path algorithm in the loop; in the implementation, however, we ran the Warshall-Floyd algorithm [54, 55] at the beginning of the compilation to compute the optimal path for all pairs of qubits, and we fetch the result. Moreover, the results of calls to BEST_SWAP themselves can be cached. Therefore, we com-

Algorithm 3 Greatest Connecting Edge Mapping

```
function INITIAL_MAPPING($gates, topology$)
    extract guest graph $G_V = (V_V, E_V)$ from $gates$
    $Mapped = \{\}$
    $(v_1, v_2) \leftarrow$ greatest edge $\in E_V$
    $(q_1, q_2) \leftarrow$ lowest edge $\in topology$
    $M \leftarrow \{v_1 \mapsto q_1, v_2 \mapsto q_2\}$
    $Mapped \leftarrow Mapped \cup \{v_1, v_2\}$
    $E_V \leftarrow E_V \setminus \{(v_1, v_2)\}$
    while not all endpoints in $E_V$ are mapped do
        pick the greatest edge $(v, v') \in E_V$ where $v \in S$ and $v' \notin S$
        if there is a free physical qubit adjacent to $M[v]$ then
            pick the lowest edge $(q, q') \in topology$ where $q = M[v]$ and $q'$ is a free qubit
            $M \leftarrow M \cup \{v' \mapsto q'\}$
            $S \leftarrow S \cup \{v'\}$
        end if
        $E_V \leftarrow E_V \setminus \{(v, v')\}$
    end while
    for all $v \in V_V \setminus Mapped$ do
        pick a random free physical qubit $q \in topology$
        $M \leftarrow M \cup \{v \mapsto q\}$
    end for
    return M
end function
```

Algorithm 4 Find The Best Swap Sequence

```
function BEST_SWAP($qubits, topology$)
    if $qubits.len() == 1$ then
        return []
    end if
    $ESP_{max} \leftarrow 0$
    $swap_{max} \leftarrow []$
    for all $(q_0, q_1) \in topology$ do
        $swap_0 \leftarrow$ find the swap sequence between $qubits[0]$ and $q_0$ with greatest $ESP(swap_0)$
        $swap_1 \leftarrow$ find the swap sequence between $qubits[1]$ and $q_1$ with greatest $ESP(swap_1)$
        $ESP \leftarrow ESP(swap_0) \times ESP(swap_1) \times ESP($CNOT over $q_0$ and $q_1)$
        if $ESP > ESP_{max}$ then
            $ESP_{max} \leftarrow ESP$
            $swap_{max} \leftarrow swap_0 + swap_1$
        end if
    end for
    return $swap_{max}$
end function
```

pute BEST_SWAP for all possible combinations of qubits before Algorithm 1 and we use the cached sequence in the main loop.

4.4 Experimental Evaluation of Compilation

4.4.1 Probability Distribution Based Evaluation

In this section, we explain the experimental evaluation of the compilation on IBM's Q20 Tokyo machine. As described in Equation 3, our compiler uses a simple multiplicative error model to optimize the ESP of quantum circuits. However, ESP has difficulty as an metric of the quality of the compiled circuits when it comes to the evaluation of real quantum hardware.

The difficulty is that the experimental result of a quantum circuit cannot be judged either successful or unsuccessful in general. Imagine a quantum circuit with a Hadamard gate and measurement for each qubit. This circuit will produce a uniform distribution over the all possible measurements. Therefore, we cannot decide whether the circuit succeeded or failed from one shot of measurement. Instead, we have to focus on how close the empirical probability distribution sampled by running the compiled circuit on a NISQ machine multiple times and the ideal distribution an imaginary noiseless quantum computer will produce are.

To deal with this difficulty, we propose a probability distribution based measure of compiled circuits. For each quantum circuit C, let P_{ideal} be the ideal distribution of circuit C and $P_{empirical}$ be the empirical distribution we observed by running compiled circuit on a machine. Then, we define KL divergence $D_{KL}(P_{ideal}||P_{empirical})$ between those two probability distributions as follows:

$$D_{KL}(P_{ideal}||P_{empirical}) = \sum_x P_{ideal}(x) \log \frac{P_{ideal}(x)}{P_{empirical}(x)} \quad (4)$$

where x runs over the possible measurement results. It's known that $P_{ideal} = P_{empirical}\, a.e. \Leftrightarrow D_{KL}(P_{ideal}||P_{empirical}) = 0$ so the lower D_{KL} is, the better the quality of the compiled circuit is.

Boixo *et al.* proposed the cross entropy, which is equal to KL divergence plus a certain offset, as a benchmark for verifying Quantum Supremacy [56] in NISQ devices [57], however, no papers yet use such measures for compiling quantum circuits as far as we know.

4.4.2 Experiment and Result

To test our compiler, we compiled Cucarro's ripple-carry adder circuit [42]. Fig. 17 shows the schematic. In the test bench, we first applied Hadamard gates for each qubit to initialize the states, and then passed the qubits to the adder. Finally, we measured the addition result and the carry-out in the computational basis. We adopted the adder circuit because:

1. the computation is simple to understand, and the ideal distribution is easy to compute; but
2. the addition circuit is made of a complex combination of CNOT gates, and
3. addition is a key component of many quantum algorithms, so its performance is inherently important.

In this experiment, we compared the following two compilation algorithms:

1. The beam search compiler. We used $B = 10000$ and $M = 1000$ as the parameters.

Table 2: The number of gates and compilation time for each adder. Q_{input} denotes the number of input qubits of the adders. g_{ori} is the number of CNOT and single qubit gates in the pre-compiled circuit. $g_{min}, g_{median}, g_{max}$ shows the minimum, median, and max number of gates in the compiled circuits, respectively. $T_{min}, T_{median}, T_{max}$ are the minimum, median and max time in seconds needed to compile the circuit, respectively. We ran the compiler with Intel Core i7-8550U and 16GB RAM. The compiler is implemented in Rust and consists of 1894 lines of code.

		beam search						random selection					
Q_{input}	g_{ori}	g_{min}	g_{median}	g_{max}	T_{min}	T_{median}	T_{max}	g_{min}	g_{median}	g_{max}	T_{min}	T_{median}	T_{max}
1	45	48	66	81	2.96	3.11	3.34	69	96	129	0.022	0.032	0.040
2	82	100	121	145	9.47	10.2	11.6	103	127	211	0.032	0.042	0.053
4	156	255	317	390	31.3	33.6	36.1	239	290	380	0.051	0.063	0.131

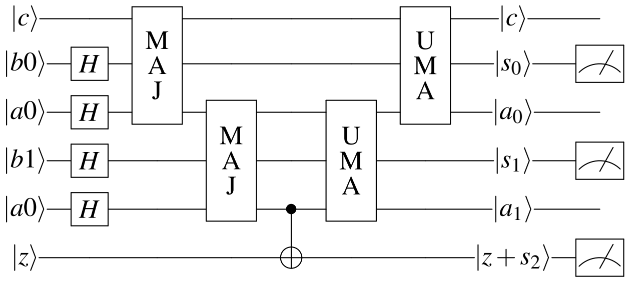


Figure 17: Cucarro's adder circuit of 2-qubit inputs and carry-in ($|c\rangle$) and carry-out ($|z\rangle$). MAJ (MAJority) gate consists of one Toffoli gate and two CNOT gates and computes carry bit by determining the majority of inputs are $|1\rangle$ state. UMA (UnMajority and Add) is also made of one Toffoli and two CNOTs and computes the addition of this digit on $|b\rangle$, uncomputing the other qubits. We denote the i'th bit of the addition as $|s_i\rangle$ in the figure. Toffoli gates are made of 6 CNOT gates and 9 single qubit gates. Since we give $|0\rangle$ for $|c\rangle$ and $|z\rangle$, the measurement of this circuit gives you the distribution of the added value.

2. A random selection compiler. Instead of ranking states via the score function, this compiler randomly picks a state s from S_i and set S_{i+1} as a singleton $\{s\}$ in Algorithm 1. Also, we randomly selected one initial mapping as the starting point.

Since the initial execution state is a superposition state where all values appear with equal probability, the output of this circuit is identical to the distribution of the sum of two rolls of dice. Fig. 18 shows the ideal distribution and observed distributions of two compilers.

We tested the adder circuits for input register sizes of one, two, and four qubits on IBM Q20 Tokyo machine. Table 2 shows the number of gates and the time required to compile.

Figures 19a–19c show the results. The horizontal axis is the ESP of the compiled circuits. Note that the horizontal axis is a logarithmic scale. The vertical axis shows the KL divergence between the observed distribution and the ideal distribution. We sampled 5000 shots to compute one KL divergence for each experiment. Moreover, to consider the variation of noises during the experiments, we ran the same circuit several times. Each point in the plot corresponds to the median of KL divergence and the error bar shows the maximum and the minimum for each compiled circuit. We have also included the KL divergence between the ideal

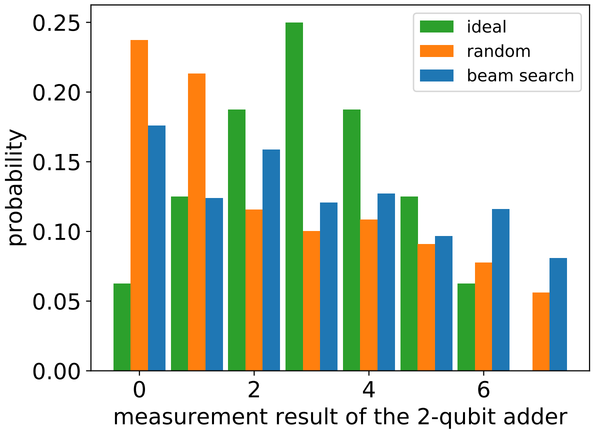


Figure 18: Example distributions of 2-qubit adder experiments. The green bars correspond to the ideal distribution calculated by classical simulation, and the other bars (orange and blue) show the distribution we retrieved by running on IBM Q20 Tokyo the circuits compiled by random selection compilation algorithm and beam search, respectively. In this experiment, we got $ESP = 0.169$ and $D_{KL} = 0.297$ for the random selection compiler, $ESP = 0.091$ and $D_{KL} = 0.216$ for the beam search compiler.

distribution and the uniform distribution as a guide.

First, we can see the points of beam search tend to concentrate in the right half in all plots. This shows beam search can indeed choose realizations of gates such that ESP of the whole circuit improves dramatically. Since the horizontal axis is a logarithmic scale, our technique often results in one magnitude higher ESP.

Next, Fig. 19a shows that most of the 1-qubit adder experiments results in better output distributions than a uniform distribution. As the number of qubits increases and the circuit gets more complicated, however, the KL divergence gets worse. For a 2-qubit adder, only half of the experiments performed better than a uniform distribution, and for a 4-qubit adder, no experiments did. This tendency can be observed regardless of the compilation algorithm.

Finally, we analyzed how KL divergence differs as the ESP of the compiled circuit changes. We draw the linear regression of KL and ESP as a dashed line to each plot. For the 1-qubit adder experiments shown in Figure 19a, there is a clear negative correlation between ESP and KL divergence. Its correlation coefficient was -0.475. Since lower KL divergence means the output of the compiled circuit is closer to the ideal, the negative trend implies that our approach of compiling a quantum circuit with higher ESP actually improves the reliability of compiled circuits for the 1-qubit adder circuit.

When it comes to 2-qubit adder experiments (Fig. 19b), the correlation coefficient of ESP and KL is -0.0279, which means there is almost no correlation between ESP and KL.

Although there is also negative correlation between ESP and KL for the 4-qubit adder experiments, the absolute values of KL divergence are much worse than that of even the uniform distribution. So we can see the improvement of KL divergence here, but the improvement is too small to make the computation reliable.

5. FUTURE WORK AND CONCLUSION

In this paper, we proposed two reliability metrics for quantum gates and circuits: Estimated Success Probability for use during compilation and KL divergence for assessing results. Estimated Success Probability is a composable measure of the quality of quantum operations. The ESPs of qubit initialization, single qubit gates, CNOT gates, and measurements are the error rates given by randomized benchmarking. We defined the ESP of a composite circuit as the product of the ESPs of its components. Due to its composability, compilers can calculate ESP easily, so we adopted full-circuit ESP as the optimization target of our compiler.

conclude entire experiment

Since we cannot experimentally observe the ESP of circuits that end with a superposition state just before measurement, including important building blocks for other algorithms such as adders and the Quantum Fourier Transform, we use KL divergence for comparing the output distributions of experiments across compilation algorithms.

In lieu of full tomography, KL divergence is a useful intermediate tool.

Experiments on path selection and circuit selection showed that using ESP gives lower (better) KL divergence than random choice among shortest-path candidates. However, even in the relatively simple case of selecting a two-hop path, our best success rate is only 70%.

Working
Not so great
success rate

Despite the difficulty of path selection, our experiments showed that our beam search-based compiler improves the ESP of a quantum circuit. Our experiments with 1-qubit adders showed that the improvement of ESP led to lower KL divergence, which means our approach of optimizing ESP experimental can mitigate errors in NISQ devices, at least for smaller circuits. However, as the circuit becomes complex, the relationship between ESP and KL divergence vanished, or KL got much worse than even the uniform distribution. This behavior demonstrates the limit of current NISQ computation, while highlighting the importance of compilation aggressively focused on errors.

Good on simple circuit

No result on complex circuit.

Although ESP is better than random selection, in some cases we can make the optimal choice out of seven candidates 43% of the time. This is due to the fact that ESP and the physical reality are divergent.

One shortcoming of our current approach is that it does not take into account memory errors. Due to the complexity of gate scheduling within Qiskit, augmenting ESP via a gate-by-gate, qubit-by-qubit insertion of Identity (“Wait”) gates complete with T_1 (energy relaxation time) and T_2 (dephasing time) decoherence is a difficult challenge. We are considering methods to incorporate the execution time and apply a blanket decoherence term.

Short coming

We plan to test other circuits such as QFT for more comprehensive benchmarking of the compiler. However, our evaluation will not work for circuits which appear in the context of quantum supremacy because our KL divergence based evaluation requires deriving the ideal distribution with a classical computer beforehand. Therefore, we need a method for estimating KL divergence to better generalize our approach. Additionally, we need a method for extrapolating from KL divergence to the prospects of seeing the expected quantum interference patterns that drive quantum algorithms; this will involve assessing sign errors as well as bit flip errors.

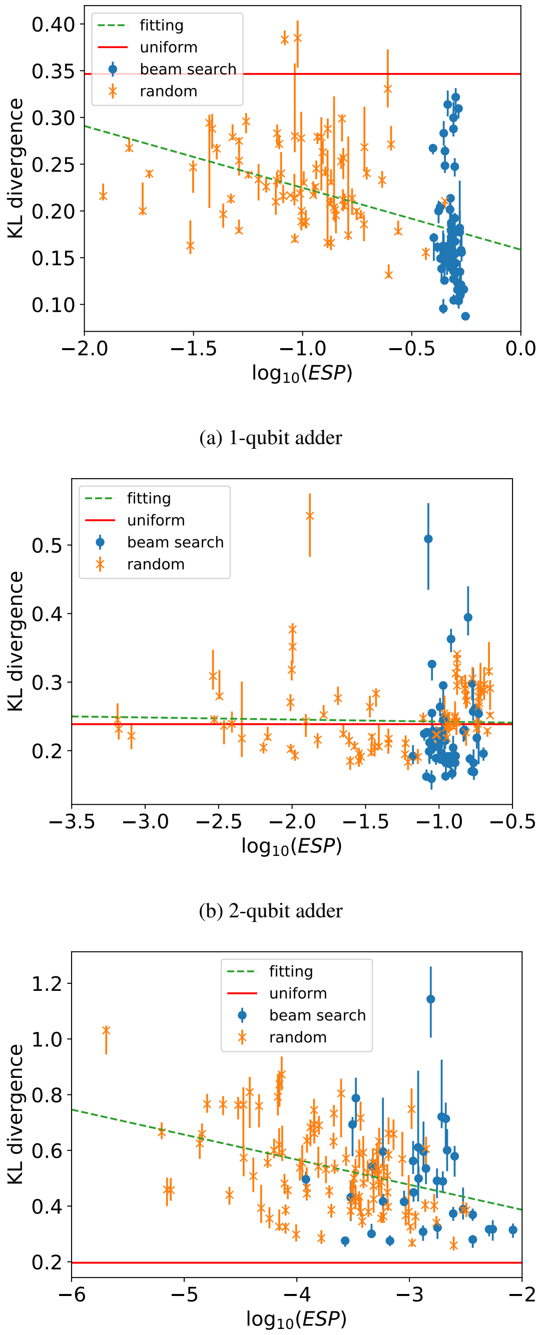
future plans

Finally, now that we have shown the value of error-aware compilation, we hope to influence the design of the Qiskit compiler (and compilers for other languages and systems) or have our code incorporated directly into the standard release. As our ability to correctly predict success improves, the tools can also be used as part of an evaluation of architectural tradeoffs between different types of system couplers and qubit layouts, influencing the design of future generations of quantum computers.

influence

Acknowledgement

The results presented in this paper were obtained in part using an IBM Q quantum computing system as part of the IBM Q Network. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.



(a) 1-qubit adder

(b) 2-qubit adder

(c) 4-qubit adder

Figure 19: The relationship between ESP and KL divergence. In this experiment, we compiled a certain number of copies of the adder circuit with the beam search compiler and the random selection compiler (70 circuits for 1-qubit adder and 4-qubit adder, 100 circuits for 2-qubit). For each circuit, we executed a fixed number of experiments to observe fluctuation of KL divergence (5 experiments for 1-qubit and 2-qubit adder, 10 experiments for 4-qubit). Each experiment ran the circuit 5000 times on IBM Q20 Tokyo and calculated KL divergence from the measurement results. The points in the plot show the median of the KL divergence for each circuit, and the error bars correspond to the maximum and the minimum KL divergence.

6. REFERENCES

[1] E. Grumbling and M. Horowitz, eds., *Quantum Computing: Progress and Prospects*. National Academies Press, 2018.

[2] T. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. O'Brien, "Quantum computers," *Nature*, vol. 464, pp. 45–53, Mar. 2010.

[3] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, p. 242, 2017.

[4] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe, "Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator," *Nature*, vol. 551, no. 7682, p. 601, 2017.

[5] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, *et al.*, "Probing many-body dynamics on a 51-atom quantum simulator," *Nature*, vol. 551, no. 7682, p. 579, 2017.

[6] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. White, J. Mutus, A. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, no. 7497, pp. 500–503, 2014.

[7] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X. Zhou, and J. O'Brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nature Photonics*, 2012.

[8] J. Preskill, "Quantum computing in the NISQ era and beyond," July 2018. arXiv:1801.00862v3.

[9] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, "Defining and detecting quantum speedup," *Science*, vol. 345, no. 6195, pp. 420–424, 2014.

[10] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1510–1523, 1997. arXiv:quant-ph/9701001.

[11] S. J. Aaronson, *Limits on Efficient Computation in the Physical World*. PhD thesis, U.C. Berkeley, 2004.

[12] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, vol. 549, no. 7671, p. 203, 2017.

[13] M. Mosca, "Quantum algorithms," 2008. arXiv:0808.0369.

[14] D. Bacon and W. van Dam, "Recent progress in quantum algorithms," *Communications of the ACM*, vol. 53, pp. 84–93, Feb. 2010.

[15] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, p. 15023, 2016.

[16] A. P. Lund, M. J. Bremner, and T. C. Ralph, "Quantum sampling problems, BosonSampling and quantum supremacy," *npj Quantum Information*, vol. 3, p. 15, Apr. 2017.

[17] I. L. Markov and Y. Shi, "Simulating quantum computation by contracting tensor networks," *SIAM Journal on Computing*, vol. 38, no. 3, pp. 963–981, 2008.

[18] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, and R. Wisnieff, "Breaking the 49-Qubit Barrier in the Simulation of Quantum Circuits," 2017. arXiv:1710.05867v1.

[19] M. Oskin, F. T. Chong, I. L. Chuang, and J. Kubiatowicz, "Building quantum wires: The long and short of it," in *Computer Architecture News, Proc. 30th Annual International Symposium on Computer Architecture*, ACM, June 2003.

[20] D. D. Thaker, T. Metodi, A. Cross, I. Chuang, and F. T. Chong, "CQLA: Matching density to exploitable parallelism in quantum computing," in *Computer Architecture News, Proc. 33rd Annual International Symposium on Computer Architecture* [58].

[21] N. Isailovic, Y. Patel, M. Whitney, and J. Kubiatowicz, "Interconnection networks for scalable quantum computers," in *Computer Architecture News, Proc. 33rd Annual International Symposium on Computer Architecture* [58].

[22] R. Van Meter, W. J. Munro, K. Nemoto, and K. M. Itoh, "Distributed arithmetic on a quantum multicomputer," in *Computer Architecture News, Proc. 33rd Annual International Symposium on Computer Architecture* [58], pp. 354–365.

[23] R. Van Meter and C. Horsman, "A blueprint for building a quantum computer," *Communications of the ACM*, vol. 53, pp. 84–93, Oct. 2013.

[24] R. Van Meter and S. Devitt, "The path to scalable distributed quantum computing," *IEEE Computer*, vol. 49, pp. 31–42, Sept. 2016.

[25] S. Gay, "Quantum programming languages: Survey and bibliography," *Bulletin of the European Association for Theoretical Computer Science*, June 2005.

[26] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, "Quipper: a scalable quantum programming language," in *ACM SIGPLAN Notices*, vol. 48, pp. 333–342, ACM, 2013.

[27] A. J. Abhari, A. Faruque, M. J. Dousti, L. Svec, O. Catu, A. Chakrabati, C.-F. Chiang, S. Vanderwilt, J. Black, F. Chong, M. Martonosi, M. Suchara, K. Brown, M. Pedram, and T. Brun, "Scaffold: Quantum programming language," Tech. Rep. TR-934-12, Princeton University, July 2012.

[28] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, "ScaffCC: A framework for compilation and analysis of quantum computing programs," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, CF '14, (New York, NY, USA), pp. 1:1–1:10, ACM, 2014.

[29] D. Wecker and K. M. Svore, "LIQUi|>: A software design architecture and domain-specific language for quantum computing," 2014. arXiv:1402.4467.

[30] J. Heckey, S. Patil, A. JavadiAbhari, A. Holmes, D. Kudrow, K. R. Brown, D. Franklin, F. T. Chong, and M. Martonosi, "Compiler management of communication and parallelism for quantum computation," in *ACM SIGARCH Computer Architecture News*, vol. 43, pp. 445–456, ACM, 2015.

[31] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, no. 7, p. 076001, 2013.

[32] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," 2009. arXiv:0904.2557.

[33] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, vol. 87, pp. 307–346, Apr 2015.

[34] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, "Layered architecture for quantum computing," *Phys. Rev. X*, vol. 2, p. 031007, Jul 2012.

[35] P. Selinger, "Efficient Clifford+T approximation of single-qubit operators," *Quantum Information & Computation*, vol. 15, no. 1-2, pp. 159–180, 2015. arXiv:1212.6253.

[36] C. Jones, "Low-overhead constructions for the fault-tolerant Toffoli gate," *Phys. Rev. A*, vol. 87, p. 022328, Feb 2013.

[37] A. G. Fowler and C. Gidney, "Low overhead quantum computation using lattice surgery," 2018. arXiv:1808.06709.

[38] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, "Charge-insensitive qubit design derived from the Cooper pair box," *Phys. Rev. A*, vol. 76, p. 042319, Oct 2007.

[39] IBM, "IBMQ experience device." `https://quantumexperience.ng.bluemix.net/qx/devices`, 2018. (Accessed: 2018-06-07).

[40] M. Roth, M. Ganzhorn, N. Moll, S. Filipp, G. Salis, and S. Schmidt, "Analysis of a parametrically driven exchange-type gate and a two-photon excitation gate between superconducting qubits," *Phys. Rev. A*, vol. 96, p. 062323, Dec 2017.

[41] M. Y. Siraichi, V. F. d. Santos, S. Collange, and F. M. Q. Pereira, "Qubit allocation," in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, CGO 2018, (New York, NY, USA), pp. 113–125, ACM, 2018.

[42] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," 2004. arXiv:quant-ph/0410184.

[43] D. A. Lidar and T. A. Brun, "Introduction to decoherence and noise in open quantum systems," in *Quantum Error Correction*, pp. 3–45, Cambridge Press, 2013.

[44] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland,

"Randomized benchmarking of quantum gates," *Phys. Rev. A*, vol. 77, p. 012307, Jan 2008.

[45] E. Magesan, J. M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking," *Phys. Rev. A*, vol. 85, p. 042311, Apr 2012.

[46] A. G. Fowler, S. J. Devitt, and L. C. Hollenberg, "Implementation of Shor's algorithm on a linear nearest neighbor qubit array," *Quantum Information and Computation*, vol. 4, no. 4, p. 237, 2004.

[47] R. Van Meter and K. M. Itoh, "Fast quantum modular exponentiation," *Physical Review A*, vol. 71, p. 052320, May 2005.

[48] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD)*, 2018.

[49] S. S. Tannu and M. K. Qureshi, "A case for variability-aware policies for NISQ-era quantum computers," 2018. arXiv:1805.10224.

[50] W. Finigan, M. Cubeddu, T. Lively, J. Flick, and P. Narang, "Qubit allocation for noisy intermediate-scale quantum computers," 2018. arXiv:1810.08291.

[51] W. Finigan, M. Cubeddu, T. Lively, J. Flick, and P. Narang, "Qubit allocation for noisy intermediate-scale quantum computers," 2018. arXiv:1810.08291.

[52] R. W. Floyd and R. L. Rivest, "Algorithm 489: The algorithm select – for finding the ith smallest of n elements [m1]," *Commun. ACM*, vol. 18, p. 173, Mar. 1975.

[53] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, pp. 1389–1401, Nov 1957.

[54] S. Warshall, "A theorem on boolean matrices," *J. ACM*, vol. 9, pp. 11–12, Jan. 1962.

[55] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, pp. 345–, June 1962.

[56] J. Preskill, "Quantum computing and the entanglement frontier," 2012. arXiv:1203.5813.

[57] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, pp. 595–600, June 2018.

[58] ACM, *Computer Architecture News, Proc. 33rd Annual International Symposium on Computer Architecture*, June 2006.