QRAFT: Reverse Your Quantum Circuit and Know the Correct Program Output*

Tirthak Patel and Devesh Tiwari
Northeastern University
Boston, Massachusetts, USA

ABSTRACT

Current Noisy Intermediate-Scale Quantum (NISQ) computers are useful in developing the quantum computing stack, test quantum algorithms, and establish the feasibility of quantum computing. However, different statistically significant errors permeate NISQ computers. To reduce the effect of these errors, recent research has focused on effective mapping of a quantum algorithm to a quantum computer in an error-and-constraints-aware manner. We propose the first work, QRAFT, to leverage the reversibility property of quantum algorithms to considerably reduce the error beyond the reduction achieved by effective circuit mapping.

CCS CONCEPTS

• **Computer systems organization → Quantum computing**.

KEYWORDS

Quantum Computing, Quantum Error Mitigation, NISQ Computing

ACM Reference Format:
Tirthak Patel and Devesh Tiwari. 2021. QRAFT: Reverse Your Quantum Circuit and Know the Correct Program Output. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '21), April 19–23, 2021, Virtual, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3445814.3446743

1 INTRODUCTION

Current Noisy Intermediate-Scale Quantum (NISQ) computers have demonstrated early potential and have shown forward progress in practical realization of quantum computing. However, one of the major bottlenecks toward the wide adoption of NISQ computers is the high error rate [31, 42]. When a quantum program is executed on current NISQ devices, the program may complete successfully, but the fidelity of individual operations is low and hence, the final output may not be correct. While quantum computing promises orders of magnitude performance improvement for a class of algorithms, such improvements are not useful if the programmer cannot deduce the correct program output, or verify the expected/true program output. The goal of this paper is to *help quantum programmers automatically deduce the correct program output while running on error-prone NISQ machines.* Toward that end, next we discuss the limitations of the existing approaches and the key ideas and contributions of our proposed solution, QRAFT.

Limitations of the State of the Art. NISQ machines suffer from multiple types of errors that can make the final algorithm output erroneous, including qubit coherence errors, quantum gate operation errors, and state preparation and measurement errors. However, error rate on different qubits on a given machine may differ significantly depending on the quantum operation type. Therefore, the impact of error experienced by a quantum program can be reduced by intelligently mapping a program's logical operations to the least erroneous physical components on a given machine (e.g., avoid mapping the program on physical qubits that exhibit higher error rate for a certain type of quantum gate or operation).

Existing approaches reduce the impact of errors experienced by a quantum program by carefully mapping a program's logical operations to the lowest-error qubits of a quantum computer [13, 29, 33, 43–46, 50, 53]. This approach is known as **"optimal circuit mapping"**, where the logical operations of a quantum program are intelligently *mapped* on to a set of physical qubits considering multiple factors including choosing physical qubits with the lowest historic error rate [29, 33, 43, 45, 50, 53], choosing physical qubits with minimal error variance [41], using machine learning models to select qubits based on operation-specific error rates [38], minimizing cross-talk noise [33, 35, 44, 48, 51], the physical connection between qubits, and the sequence of operations. However, the current approaches have two major limitations:

I. Lack of knowledge about a program's true output: Current approaches aim to minimize the error occurrence probability and learn certain characteristics of the program under study (e.g., number of operations, type of operations, etc.) [4, 16, 38, 51, 53]. *However, these approaches do not (and cannot) know the "correct/true program output" of the program irrespective of how many times or on which qubits the program is executed. Hence, the observed output can only be used as the best guess of the correct output.*

All prior studies assume that they know the correct program output apriori and report the difference between the observed output and the apriori-known correct program output as the "error in estimation of the program output" [29, 36, 46, 47, 52]. Unfortunately, this approach is not useful for programmers who may not always know the correct program output. *Such approaches mostly focus on identifying a single dominant output state with non-zero correct probability and have very limited effectiveness for programs having multiple output states with non-zero correct probabilities.*

*A part of this work was performed during an unprecedented time – around the peak of the COVID-19 pandemic. We were able to conduct this scientific study only because first-line responders and essential workers worked tirelessly to keep our community safe and functioning. We are grateful for their effort. This paper is dedicated to the memories of all the first-line responders and essential workers who sacrificed their lives trying to keep ours safe.

ASPLOS '21, April 19–23, 2021, Virtual, USA

ACM ISBN 978-1-4503-8317-2/21/04.
https://doi.org/10.1145/3445814.3446743

II. Everyone has access to the "best" qubits and knows which qubits are the best: Existing approaches over-optimistically assume that each end user has access to all the qubits, especially the most reliable ones, and their historical error rates to run their programs [29, 33, 35, 38, 41, 44, 46, 50]. However, this assumption that will be seriously challenged in future quantum computing systems shared among multiple users concurrently [13]. Moreover, historical error information may become business-sensitive, similar to classical computing systems.

Key Ideas and Contributions. QRAFT is a novel method to accurately estimate the true program output. QRAFT demonstrates for the first time that reversing the circuit can *reveal* what the program actually does. The key insight behind QRAFT is to *reverse the quantum circuit and execute the full forward + reverse circuit* to deduce the correct program output for all the quantum states of the original (forward) circuit.

Quantum operations, unlike classical operations, are reversible. The original input can be restored by applying an inverse operation. QRAFT leverages this property and extends it to the entire quantum circuit. QRAFT appends the reverse circuit at the end of the original forward circuit and executes the forward + reverse circuit. *Reversing a circuit enables QRAFT to "partially" know the inherent correct output of the full circuit – all output states must get reduced to original input states – a piece of "ground truth" information that QRAFT gets access to and exploits. This is a departure from all prior works since they do not have the full or partial knowledge of such ground truth related to the program's true output.* To develop a better understanding, Sec. 3 describes multiple useful insights obtained from characterizing the reverse execution of quantum circuits.

While this approach is encouraging, we discovered that a straight forward rule-based application of this approach does not yield the expected results due to complex interaction of errors with the underlying original circuit. QRAFT designs a learning-based prediction model that generates the true output probability when fed the output from the runs of a circuit.

Unlike prior works, QRAFT accurately estimates the magnitude of probabilities for programs with multiple non-zero probability output states. Our evaluation demonstrates that, compared to the existing approaches based on optimizing the circuit mappings, QRAFT reduces the median state error by up to 7%, dominant state error by up to 30%, and total program error by up to 20% across different algorithms. The state-of-the-art approach has only 20% of the circuit states with 0% error, while QRAFT ensures that over 70% of the states have 0% error, when tested with 200 randomly generated circuits. Further, QRAFT does not suffer from error bias factors such as true state probability and state hamming weight - a key source of weakness in existing approaches.

QRAFT demonstrates that it is possible to deduce the correct program output successfully, even in the absence of an optimal (or a near-optimal) circuit map (relaxing a prerequisite of existing approaches which assume unrestricted access to the "best" qubits and know which qubits are the best based on the historical error information). Our evaluation confirms that QRAFT scales to greater number of qubits, and its effectiveness it not sensitive to the choice of NISQ platform or specific circuit characteristics such as circuit depth and number of operations.

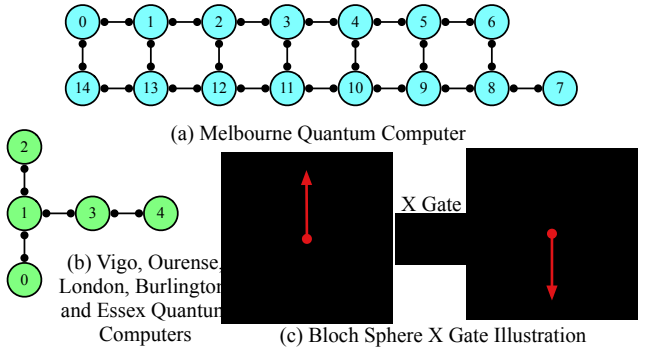


Figure 1: (a) Qubit connectivity layout of the 15-qubit Melbourne quantum computer. (b) Layout of five-qubit quantum computers. (c) Bloch Sphere illustration of qubit state transformation.

Next, we briefly cover the overview of quantum computing and motivation (Sec. 2), then, discuss the design and implementation of QRAFT(Sec. 3), and finally, analyze the evaluation results (Sec. 4).

2 BACKGROUND AND MOTIVATION

2.1 Quantum Computing Overview

Quantum Bit. In contrast to a classical bit, a quantum bit (refereed as *qubit*) can be in a superposition of states 0 and 1 until measurement. Upon measurement, it collapses to either a 0 or a 1. A qubit's state, $|\psi\rangle$, can be represented as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are the complex-valued amplitudes indicating the probability of measuring $|\psi\rangle$ in the respective basis states $|0\rangle$ and $|1\rangle$.

Engineering Qubits. A qubit can be physically realized by carefully engineering the technology. Our experimental study is based on IBM's superconducting-qubit technology – one of the more promising quantum computing technologies. The superconducting-qubit technology consists of a capacitor and a Josephson Junction which form a non-linear LC oscillator. If the circuit parameters are tuned correctly, this oscillator can behave as a quantum particle, i.e., a qubit. The quantum processing unit (QPU) is placed in a dilution refrigeration to maintain a low temperature of 15mK. Two energy levels of a qubit are used as the $|0\rangle$ state (ground state) and the $|1\rangle$ state (excited state). This method of creating a qubit, while promising, has its challenges. One issue is that the qubit can lose its state coherence in a matter of a few milliseconds. Therefore, as the quantum circuit progresses, the qubit's state can lose coherence and the final output can be erroneous.

Assembling Multiple Qubits. Multiple qubits are assembled to realize a quantum computer. Fig. 1(a) and (b) show the topology of all the IBM QX quantum machines used in this study. The topologies demonstrate how different qubits are connected together. The layout and connectivity of the qubits is designed to minimize the cross-talk (undesired noisy cross-qubit interference that can introduce errors) among the qubits.

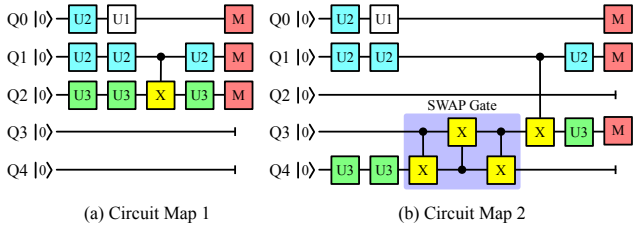


Figure 2: (a) Example of an optimal circuit map. (b) Sub-optimal circuit map that requires an extra SWAP operation.

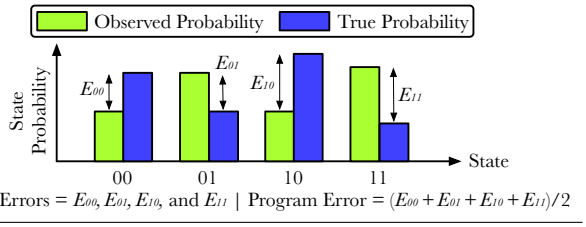


Figure 3: Illustration of the state error and the program error.

Manipulating Qubits. Quantum operations are performed by manipulating the states of the qubits. Different types of quantum operations can be performed on a single qubit. To demonstrate a one-qubit quantum operation, we represent a qubit's superposition on a Bloch Sphere. A Bloch Sphere is a unit sphere where the positive z-axis represents the $|0\rangle$ state and the negative z-axis represents the $|1\rangle$ state. The other two axes represent the qubit phase and superpositions. As shown in Fig. 1(c), initially the qubit is in the $|0\rangle$ state (vector pointing up). Then, an X gate is applied which rotates the qubit about the x-axis by 90° and puts it in the $|1\rangle$ state. Similarly, arbitrary rotations (about the x, y, or z axes) can be applied to create qubit superpositions. However, when the qubit is measured, its state vector collapses to the z-axis ($|0\rangle$ or $|1\rangle$).

Further, multiple qubits can be *entangled* together by performing a two-qubit quantum operation: one qubit acts as the control qubit and the other qubit acts as the target qubit. The superposition of the target qubit is manipulated based on the superposition of the control qubit. For example, a CX gate applies the X gate (rotation of 90° about the x-axis on the Bloch Sphere) to the target qubit if the control qubit is in state $|1\rangle$; if it is in state $|0\rangle$, no change is made to the target qubit. The connectivity between the qubits determine which qubits can have two-qubit gates applied directly.

One-qubit quantum gates are performed by applying Microwave pulses at the resonant frequency of the qubit. Similarly, two-qubit gates require applying Microwave pulses to the coupling bus connecting the two qubits. These operations are subject to a variety of errors. First, the Microwave pulses used to apply quantum gates are also prone to being applied in an erroneous manner. Second, measuring a qubit's final state can also be error-prone: the qubit's energy can be misclassified.

Finally, many different types of gate operations are theoretically possible, but a particular setup typically supports a small set of universal basis gate operations that can be used to express any other gate operation. In the case of IBM's technology, these are the U1, U2, U3, and CX gates. U1, U2, and U3 are one-qubit gates. The U1 gate consists of one frame-of-reference change. The U2 gate consists of one 90° rotation pulse with pre- and post- frame changes. Lastly, the U3 gate consists of three frame changes and two 90° rotation pulses. All one-qubit gates can be applied using the U3 gate, however, the U1 and U2 operations are used for compatible gates as they have a lower error rate because of fewer rotation pulses. The CX gate is a two-qubit gate where one qubit acts as the control qubit and the X gate is applied to the second qubit.

Expressing Quantum Algorithms. A quantum algorithm is expressed as a sequence of quantum gate operations on a set of qubits. Fig. 2(a) shows a simple example of a three-qubit algorithm mapped on to the Vigo computer. We note that a quantum algorithm can be *mapped* to a physical quantum machine in multiple ways. We refer to each such mapping as a **"circuit map"**. Note that all circuit maps for a given quantum algorithm are expected to produce the same outcome, but may differ in terms of the physical qubits used, and the number, order and type of quantum gate operations. For example, Fig. 2(b) shows the same algorithm as shown in Fig. 2(a), but mapped on to the Vigo computer in a sub-optimal manner. Because, the third logical qubit is mapped to physical qubit Q4 and there is no direct connection between Q1 and Q4 (as shown in Fig. 1(b)), an additional SWAP gate (which is made up of three CX gates as shown) has to be used. Thus, because this mapping has more operations, it suffers from more errors.

Upon execution of a quantum algorithm, it produces the **"observed probability"** for each output state. For example, a two-qubit program has four output states and the observed probability can be $p(|00\rangle) = 0.$, $p(|10\rangle) = 0.2$, $p(|01\rangle) = 0.3$, and $p(|11\rangle) = 0.4$. Due to the probabilistic nature of quantum computing, each program is executed multiple times before estimating the "observed probability" for each output state. However, because of the aforementioned coherence, gate, and readout errors, the observed probability may not be the same as the **"true probability"** of states (not known for general quantum algorithms).

Metrics to Quantify the Impact of Errors on Quantum Program Execution. The hardware errors manifest as errors in the program (algorithm) output. For example, consider a two-qubit algorithm with true probabilities $p_t(|00\rangle) = 0.1$, $p_t(|10\rangle) = 0.2$, $p_t(|01\rangle) = 0.3$, and $p_t(|11\rangle) = 0.4$. When the algorithm is run using a circuit map, the observed probabilities are $p_o(|00\rangle) = 0.15$, $p_o(|10\rangle) = 0.15$, $p_o(|01\rangle) = 0.5$, and $p_o(|11\rangle) = 0.2$. We define the **state error** as the absolute difference between the observed probability and the true probability. In this example, the state errors are $e(|00\rangle) = |0.1 - 0.15| = 0.05$ or 5%, $e(|10\rangle) = 5\%$, $p_o(|01\rangle) = 20\%$, and $p_o(|11\rangle) = 20\%$. As illustrated in Fig. 3, the **program error** (also known as the total variation distance [7]) is the sum of the state errors of all output states of an algorithm divided by two. In this case, the program error is 50%. Note that this metric is chosen to represent the overall error due to its ease of interpretibility and simplicity. In our experimentation with other metrics, such as the Hellinger fidelity [22], we found that the relative improvement

Table 1: Quantum algorithms studied in this paper.

Quantum Algorithm	Abbreviation
Bernstein-Vazirani Algorithm [10]	BV
Deutsch-Jozsa Algorithm [11]	DJ
Grover's Algorithm [18]	GRV
Quantum Fourier Transform [49]	QFT
Quantum Phase Estimation [14]	QPE
Simon's Algorithm [27]	SMN

ofQraft over the state-of-art mirrors the same trend as what is captured by program error (Sec. 4).

We also define the **dominant state error** as the state error of the states with maximum true probability. This metric is useful because for many quantum algorithms, the aim is to identify the maximum true probability state. In this example, state $|11\rangle$ has maximum probability and has an error of 20%. Therefore, the dominant state error is 20%. Note that for some algorithms, multiple states can have equal maximum true probability. In that case, the median error of all dominant states is quoted. A circuit map that results in least error from the true probability is said to be the **optimal circuit map**. However, the optimal circuit map is still prone to errors. Thus, even if the true probability is estimated by the observed probability of the optimal circuit map, that estimate can still be erroneous.

2.2 Motivation for Qraft

In this section, first, we describe the state-of-the-art circuit mapping techniques and their limitations.

Optimal circuit mapping and its use toward estimating the correct program output. Considering that different qubits have different error rates, a significant amount of research effort has been geared toward the problem of determining the optimal circuit map to run an algorithm [3, 8, 13, 33, 35, 38, 44–48, 51], given the coherence times and error rates of different qubits.This state-of-the-art research in optimal circuit mapping has been integrated into the Qiskit compiler (IBM's quantum computing language) [2]. These optimizations include selection of densely-connected qubits to minimize swapping and two-qubit operations, gate synthesis and gate cancellation using commutativity properties to reduce physical operations, and error-and-fidelity-aware qubit mapping to select qubits with the lowest error rates [29, 33, 34, 50]. Once a best-effort optimal circuit is generated, multiple runs are performed to get the observed probability of all output states (because the quantum output is probabilistic). This observed probability serves as an estimate for the true state probabilities. The true program output may not be known for an arbitrary quantum algorithm, therefore, the observed program output of an optimized circuit map is the best estimate.

Next, we quantify the error in the observed probability of the output states of a quantum algorithm using optimized circuit maps generated using all of the above mentioned optimizations available with the Qiskit compiler. Then, we discuss why optimized circuit maps alone are not sufficient to accurately estimate the true state probabilities of quantum algorithms.

Methodology. We run six quantum algorithms shown in Table. 1 on the quantum computers shown in Fig. 1. Each algorithm is run with multiple optimal circuit maps across three days. Note that

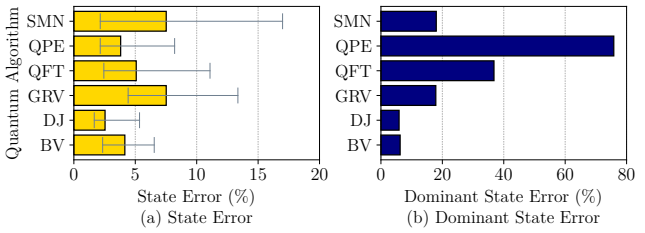


Figure 4: (a) The median error in state probability estimation across all states. The range indicators show the 25^{th} percentile to 75^{th} percentile range. (b) The dominant state error.

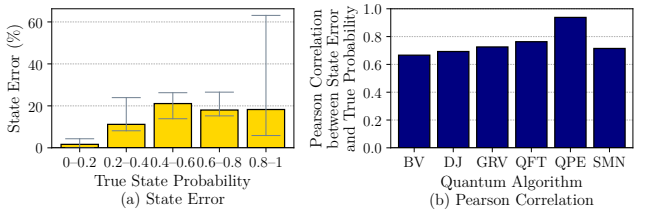


Figure 5: (a) More probable states are likely to experience higher state errors. (b) The Pearson Correlation shows that the true state probability is highly correlated with the state error.

an optimal circuit map for a given algorithm on a given computer varies at different times as the coherence times and operation errors vary temporally (these errors are noted when the computers are calibrated daily and this information is used when generating an optimal circuit map by the Qiskit compiler). Each run consists of 1024 trials (i.e., the circuit is executed 1024 times and each trial gives one output state) and the observed state probability is the fraction of trials for which the state is the output.

Fig. 4(a) shows the median state error across all states and multiple runs of six quantum algorithms. The median state error can be as high as 8% (e.g., Grover's algorithm and Simon's algorithm), while the 75^{th} percentile error can be higher than 15% (e.g., Simon's algorithm). While many quantum algorithms require estimation of the true probability of all states (e.g., QFT, QPE, and SIA), some quantum algorithms specifically require identification of states with maximum true probability (e.g., BV, DJ, and GRV). For example, Grover's algorithm (GRV) is designed to search for input states which produce a particular output given a black-box function. Therefore, the algorithm amplifies the probability of only those states which result in the particular output. Fig. 5(b) shows that the dominant states of all algorithms have a much higher error rate compared to the median. For example, the two dominant states of Grover's algorithm have a median state error of almost 20%.

> **Takeaway.** The error in estimating the true probability of output states can be as high as 8% median error and 15% 75^{th} percentile error for optimized circuit maps. The state error of the dominant state can be higher than 20% in many cases. That is, the optimal circuit map is effective at reducing the impact of errors on the program output, but it cannot deduce the true output of a program.

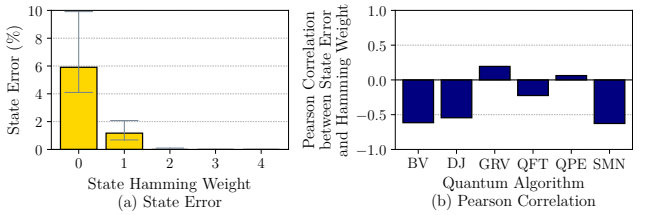


Figure 6: (a) States with a lower hamming weight have higher state errors for the DJ algorithm. (b) The correlation of the hamming weight and the error varies for different algorithms.

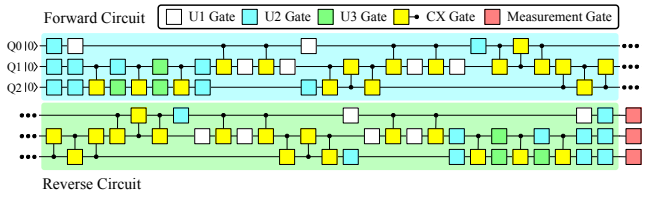


Figure 7: QFT's forward circuit is appended with the reversed version of it to construct the forward + reverse circuit.

Next, we dig deeper to understand the magnitude of error estimation by quantifying its correlation with different state-specific properties. Based on the insight that the dominant states have much higher error than the median of all states, we plot the state error as a function of the true state probabilities in Fig. 5(a). The figure shows that generally, the state error increases as the true state probability increases. The median state error for states with true probability between 0 and 0.2 is 2%, while the median state error for states with true probability between 0.8 and 1 is 18%. Fig. 5(b) shows the Pearson Correlation, which quantifies the linear relationship between the true state probability and the state error. Pearson Correlation calculates the strength of the linear relationship between two random variables: a value of 1 indicates a strong positive fit, 0 indicates no linear relationship, and -1 indicates a strong negative fit. The figure shows that the Pearson Correlation is over 0.85 across all algorithms, indicating that the state error increases as the true state probability increases. This is because a state having a higher true output probability translates to more trails ending up in that state. However, quantum operation and hardware errors can shift some or many of those trails to other states. A state that has a close-to-zero probability has fewer expected trails and is affected by this phenomenon at a lower rate.

Another state property that might affect its error is the state hamming weight (number of qubits in the $|1\rangle$ state, which are more likely to relax). Fig. 6(a) shows that for DJ algorithm, states with lower hamming weight are likely to observe a higher state error (6%) than states with higher hamming weight (≈0%). Fig. 6(b) shows that this is true for BV and SMN algorithms as well, as they have a strong negative correlation between the state's hamming weight and its error. *This is because states with higher hamming weights have more number of qubits in the excited state. Thus, they are likely to relax into states with lower hamming weights, which end up accumulating these errors.* Again, we observe that an optimal circuit map cannot mitigate this issue.

> **Takeaway.** Optimal circuit maps cannot mitigate error bias caused by factors that are inherent to the program such as true state probability and state hamming weight. These properties of a program and a state can neither be avoided nor altered; they need to be considered in order to reduce the error in true output probabilities of states.

Summary. Overall, the current approaches have the following limitations and avenues for improvements. Optimal circuit mapping aims to minimize the error occurrence probability. However, it does not know the true program output of the program irrespective of how many times or where the program is executed. Hence, the observed output can only be used as the best guess of the expected correct program output. This is further substantiated by our observation that optimal circuit maps cannot mitigate error bias caused by factors that are inherent to the program. Finally, this approach inherently assumes that the program has access to all qubits and can evaluate which best qubits can be employed for circuit mapping. This paper's goal is to deduce the true program output of any arbitrary quantum program by running them on NISQ machines, while addressing the above limitations and assumptions.

3 QRAFT: DESIGN AND IMPLEMENTATION

How does QRAFT reverse a circuit and use it for deducing the correct program output? QRAFT takes a quantum circuit and removes the measurement operations at the end and reverses all the operations one-by-one. Then, it adds the measurement operation at the end. A visual presentation of a reversed circuit corresponding to the QFT algorithm is provided in Fig. 7. The top shows the original circuit (referred to as the **forward circuit** or FC), while the bottom shows the operations in the forward circuit reversed (referred to as the reverse circuit). The entire circuit combined is referred to as the **forward + reverse circuit** or FRC. At the completion of the forward + reverse circuit, the qubit states are measured.

However, simply running a FRC and measuring the output state probability does not directly enable us to estimate the error of the FC (Sec. 3.1). Instead, as detailed later, both types of circuits are run multiple times with 1024 trials and their output state probabilities are recorded. Note that it is possible to accurately estimate the error in the observed state probabilities of the FRC since the final output is known. QRAFT leverages this insight to learn about the state errors of individual output states of the forward circuit. QRAFT accomplishes this by using the observed state probabilities of these circuits as input to a true state probability prediction model. This prediction model is trained using a large variety of random circuits with different types of static and dynamic features and tuned using Bayesian Optimization. The trained model can be used to predict the true state probabilities of any quantum algorithm (Sec. 3.2).

Is reversibility feasible for all quantum circuits, and at what cost? *Yes.* All quantum operations can be mathematically represented as complex square unitary matrices (U). By definition, the conjugate transpose ($U^\dagger$) of a unitary matrix is also its inverse: $U^\dagger U = I$. Therefore, when conjugate transposes of quantum operations are applied in the reverse order, the initial state is obtained. For example, if three quantum operations are applied to a qubit

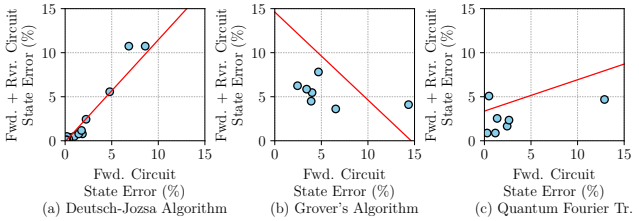


Figure 8: Examples of varying correlation between the state errors of the forward circuit and the forward + reverse circuit.

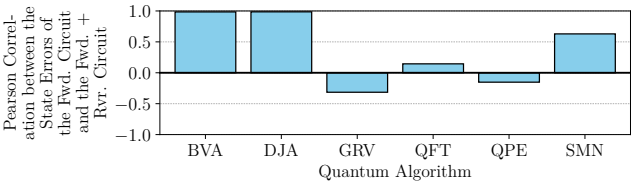


Figure 9: The correlation between the state errors of the forward and forward + reverse circuits varies among algorithms.

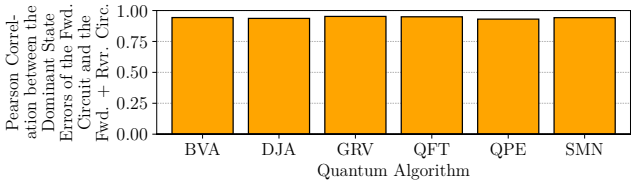


Figure 10: A high correlation exists between the dominant state errors of the forward and the forward + reverse circuits.

initialized to the $|0\rangle$ state, then applying their corresponding complex conjugates in reverse order gives: $|\psi\rangle = U_3^\dagger U_2^\dagger U_1^\dagger U_1 U_2 U_3 |0\rangle = U_3^\dagger U_2^\dagger (U_1^\dagger U_1) U_2 U_3 |0\rangle = U_3^\dagger U_2^\dagger U_2 U_3 |0\rangle = U_3^\dagger U_3 |0\rangle = |0\rangle$. This notation can be extended to multi-qubit circuits. Any quantum circuit can be represented as a sequence of unitary operations. Applying the conjugate transpose of those operations in reverse order gives back the initial state. For an n-qubit circuit with m operations, its reversibility can be represented as $|\psi\rangle = U_m^\dagger U_{m-1}^\dagger \dots U_2^\dagger U_1^\dagger U_1 U_2 \dots U_{m-1} U_m \left|0^{\times n}\right\rangle = \left|0^{\times n}\right\rangle$. This implies that when such a circuit is executed, it should yield the initial ground state $\left|0^{\times n}\right\rangle$ with a probability of one.

Fortunately, the cost of reversibility is rather small and affordable. Reversing any arbitrary quantum circuit is automated and practical as demonstrated by QRAFT's experiments on the IBM QX platform. The runtime of an FRC is negligibly higher than the runtime of an FC. One potential side-effect is the undesired interaction with the coherence time constraints for long quantum circuits (discussed in Sec. 4) since reversing a long circuit runs the risk of exceeding the coherence limit. However, our results indicate that it does not affect the prediction quality of QRAFT even for very long circuits on the IBM QX (QRAFT is not undesirably sensitive to circuit depth).

3.1 Lessons from Characterizing Reverse Circuits

Next, we present our insights and findings from characterizing the executions of FRCs, and how they can guide us toward deducing the true probability estimation of a quantum algorithm. In particular, we test three basic hypotheses in the context of output state probability estimation: *Is the state error of forward + reverse circuit (FRC) correlated with the forward circuit (FC)? Is the dominant state error correlated? Is the program error (sum of all state errors) correlated?*

It is natural to hypothesize that state error of output states in the FC should be half that of the states in the FRC, given that the FC has exactly half of the same gate operations as the FRC. If this is true, the state errors of the FC can be corrected directly to get the true state probabilities. Our study reveals that this is not necessarily true for all algorithms.

However, we note that such an analysis needs to be performed carefully. We cannot simply run FC and FRC back-to-back and attempt to derive correlations in the errors between two circuits. Due to variance among runs, each circuit type needs to be run multiple times and the runs need to be sorted by the error magnitude before testing correlations. If this methodology is not followed, one might pre-maturely conclude that reverse circuit does not reveal additional characteristics.

Fig. 8 shows three examples of when the state error of a state in FRC is plotted against the state error of the same state in the FC. We observe that the behavior varies for different algorithms, and can be quite weak in many cases. The strength of the fit is quantified in Fig. 9, which shows the Pearson Correlation between the state errors of the FC and the FRC.

This finding demonstrates that the state error of the FRC cannot be used to directly correct the state error of the FC in a straightforward manner. The reason for this is that different states get affected differently depending on the algorithm. Different FCs have different true probability distributions. This means the same state (e.g., $|01011\rangle$) of two different algorithms might experience different errors. However, for any given FC, the FRC always has the same distribution: $\left|0^{\times n}\right\rangle$ state has probability one (i.e., all qubits should be $|0\rangle$) and all other states have probability zero. Thus, it is not necessary that state errors of individual states of an FC and FRC must correlate because the FC has a different probability distribution than the FRC, and, as we saw in Sec. 1, state errors are biased by true probability of states. However, we hypothesize that the errors of the dominant states of the FC and the FRC must correlate (even if the dominant states are different for the FC and the FRC). Fig. 10 shows that this is indeed true as the dominant state error is strongly correlated between the FC and the FRC across all algorithms. This insight is useful because we know the dominant state and its error for the FRC.

Next, we ask: what about the program error of the entire circuit? A natural hypothesis is that the program error of the FRC could be approx. twice the program error of the FC simply because it has twice the number of the same operations. Fig. 11 shows that this hypothesis is not true either, however a significant correlation exists across algorithms (Fig. 12).

Our results show that different algorithms can have different slopes in terms of the relationship between the program errors of the FC and

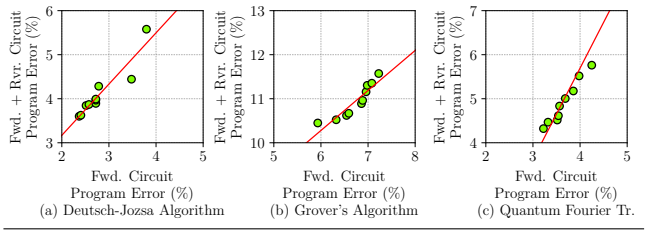


Figure 11: Examples of high Pearson (linear) Correlation between the program error of the forward circuit and the program error of the forward + reverse circuit.

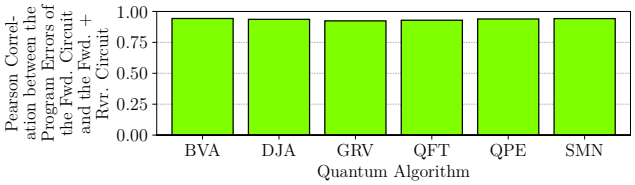


Figure 12: There is high linear correlation between the program errors of the forward and the forward + reverse circuits.

Table 2: Features used for training QRAFT's prediction model.

Features from Circuit and State (Static Features)
(1) Computer used for Execution, (2) Circuit Width, (3) Circuit Depth, Number of (4) U1, (5) U2, (6) U3, & (7) CX Gates in the Circuit, and (8) State Hamming Weight
Features from Forward Circuit Runs
(9) 25^{th} Percentile, (10) 50^{th} Percentile, and (11) 75^{th} Percentile Observed State Probability
Features from Forward + Reverse Circuit Runs
(12) 25^{th} Percentile, (13) 50^{th} Percentile, and (14) 75^{th} Percentile State Error, and (15) 25^{th} Percentile, (16) 50^{th} Percentile, and (17) 75^{th} Percentile Program Error

the FRC. The length (depth) of a circuit contributes to these different slopes. If a FC is very long, its FRC would be even longer and observe a strong loss of coherence across qubits. Thus the program error of the FRC would be higher than twice the program error of the FC. Similarly, for shorter circuits, the qubits might not experience a significant loss of coherence. Hence, the error of the FRC would be low. Different circuit characteristics can contribute to different program-error relationships. The program error of the FRC can be a strong indicator of the FC's program error.

> **Takeaway.** The FRC can be used to generate strong indicators for the dominant state error and the program error of the corresponding FC, but not necessarily the state errors of individual states, which have a complex algorithm-dependent behavior.

These insights motivate the need for a machine-learning-based solution which can absorb these insights, and output an accurate estimate of the true state probabilities of algorithms.

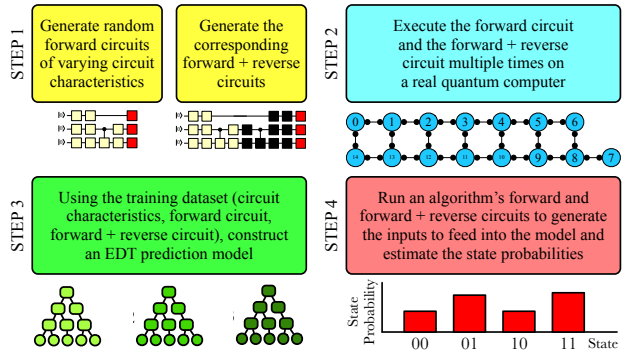


Figure 13: QRAFT's predictive model generation and inference.

3.2 QRAFT's Prediction Model: Build and Use

QRAFT builds a predictive model that takes observed state probabilities of the forward circuit and the forward + reverse circuit from multiple runs and outputs the estimated true output state probabilities of the original (forward) circuit. Next, we describe how QRAFT's prediction model is built.

How is the QRAFT model trained? We build a training dataset by generating ≈1400 random forward circuits of varying widths, depths, and operations across six different IBM quantum computers. Each circuit is run ten times in the forward mode and ten times in the forward + reverse mode to obtain the observed state probabilities and other related features (described next). We experimented with varying number of runs, but found that the prediction quality is not sensitive to the number of runs beyond ten (recall each already consists of 1024 trials). Each circuit has multiple states and because a large fraction of the states tend to have zero probability, they can end up being not recorded in any of the trials of the forward and the forward + reverse circuits. Note that including them in the training can inadvertently boost the accuracy of the prediction model even if it always predicts a true state probability of zero. Therefore, QRAFT eliminates such states from the training as they can be directly assumed to have zero true probability. After the filtering process, we end up with ≈10,000 states for training (also referred as samples). *We do not use real quantum algorithms for training to avoid biasing the results.*

What features does the QRAFT model use? The features used by QRAFT are listed in Table 2 under three categories. The first set of eight features are available directly after circuit compilation without running the circuit. These include the computer for which the circuit is compiled, with width of the circuit (number of qubits), the depth of the circuit (number of sequentially run operations), the number of different types of basis gates, and the state hamming weight. Recall that features such as the number of different types of basis gates, the circuit depth, and the state hamming weight affect the overall error of the circuit and error of the output probabilities. Thus, they are important features to help predict output states correctly and diminish the errors (as we show in our evaluation (Sec. 4)). In fact, we found that these circuit features are enough to generate a high-quality prediction model. Expanding the feature

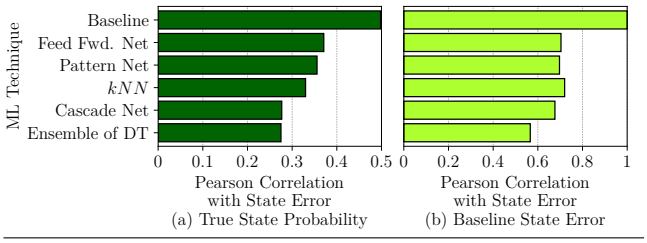


Figure 14: (a) The correlation between the technique's state error and the true state probability, and (b) the correlation between the technique's state error and the Baseline state error is the lowest for QRAFT's Ensemble of Decision Trees (EDT) implementation.

set further by including the structure of the circuit runs the risk of overfitting the model to the random circuits used for training.

The second set of three features are produced from running the forward circuit. Running any circuit just once is not sufficient because of the stochastic nature of the errors. Therefore, each circuit is run ten times (recall that each run has 1024 trials) and the 25^{th} percentile, 50^{th} percentile, and 75^{th} percentile observed state probabilities are used as the input. *Note that the state errors in the forward circuit are not known and cannot be fed as features.* However, during the training process, the state errors of the randomly-generated forward circuits are required to be known to optimize the model. Thus, the ground truth of the random circuits needs to be obtained using a quantum simulator so that their error can be calculated. Note that later in our evaluation (Sec. 4), we show that high-quality a predictor model can be constructed for larger machines and circuits even if the model is only trained using easy-to-simulate, smaller circuits that are less than half the size (in terms of the number of qubits) of the larger circuits. QRAFT can achieve this by ensuring that different smaller circuits (compared to the size of the machine) cover all the different qubits of the computer such that all of their error characteristics can be learned when training.

Lastly, six features are obtained from running the forward + reverse circuit ten times. Because the errors are known for the FRC, the (25^{th}, 50^{th}, and 75^{th} percentile) state error of the state in question and the (25^{th}, 50^{th}, and 75^{th} percentile) program error of the circuit are all used as features.

Which model to select and why? The next step is to determine the type of machine learning model to use for prediction. To determine which ML model performs the best, we compare the biases of five ML models: ensemble of decision trees (EDT), k nearest neighbors (kNN), simple feed-forward neural network (input is fed into the first layer and each subsequent layer has a connection from only the previous layer), cascade-forward neural network (the input is directly fed into also the layers), and pattern-detection neural network (similar to feed-forward network but the output layer gives probability of membership to different classes). The hyperparameters of all models (e.g., number of trees and boosting/bagging method for EDT, number of nearest neighbors for kNN, and number of hidden layers and training function for the neural networks) are optimized using Bayesian optimization.

Selecting a model to mitigate the true state probability bias: Fig. 14(a) shows the Pearson Correlation between the true state probability and the state error of the prediction of different models. *The Baseline is the state error of the optimal circuit maps.* Because the Baseline is inherently biased against high-probability states (as we saw in Fig. 5), it is important for the selected ML model to reduce this bias. Therefore, the model which has the least correlation between true state probability and state error is the most suitable. EDT has the lowest correlation out of all techniques (0.28) which indicates that it is the least biased against high-probability states. Moreover, Fig. 14(b) shows the Pearson Correlation between the state error of the prediction and the Baseline state error (i.e., state error observed for optimal circuit maps). A large correlation indicates that a large Baseline state error may result in a large state error even after prediction. We want to avoid this. The EDT model achieves the lowest correlation out of all ML techniques, which shows that it is the best at correcting the state error irrespective of the Baseline state error.

For the above reasons, QRAFT chooses the EDT model. QRAFT's EDT model determines the true probability of states by constructing multiple decision tree learners and voting among them to estimate the true probability of a state given the input features in Table 2. The dataset is randomly split 85%:15% for training:testing. The model is trained using 5-fold cross validation for robustness against sampling bias. Bayesian Optimization with expected-improvement acquisition function is used for tuning the model hyperparameters (number of tree learners, learning rate, ensemble aggregation method (e.g., bag, AdaBoost, etc.), maximum number of tree splits, and split criterion (e.g., GDI, Towing, etc.)). Note that training has a one-time overhead for each quantum computer and the generated model can be used for all future runs of different algorithms, although periodic training is recommended (the training overhead is less than one hour).

Using QRAFT for Prediction. Once the model is developed, it can be used to predict the true state probabilities of any algorithm. The algorithm can be compiled for any quantum computer and its circuit map can be generated. The circuit should be run ten times in forward mode and ten times in forward + reverse mode. Running 20 runs in total on real quantum computers takes ≈two minutes as each trial of a run finishes within a few milliseconds. The runs can be processed to generate all the input features listed in Table 2. The features can be fed into the model to get an estimate of the true output probabilities of different states of the algorithm. The output probabilities can also be used to find the dominant states.

4 EVALUATION AND ANALYSIS

In this section, we evaluate the effectiveness of QRAFT and attempt to better understand why and how QRAFT works.

Does QRAFT help us deduce the correct program output compared to existing optimal circuit map approach? We answer this question by evaluating QRAFT for (1) the quantum algorithms listed in Table 1 and (2) over 200 randomly generated quantum circuits whose true outputs are known apriori.

Fig. 15 shows the median state error, the dominant state error, and the program error for all tested quantum algorithms, same metrics as discussed in Sec. 1. We make several observations. First, QRAFT is better than Baseline (state error of optimized circuit maps)

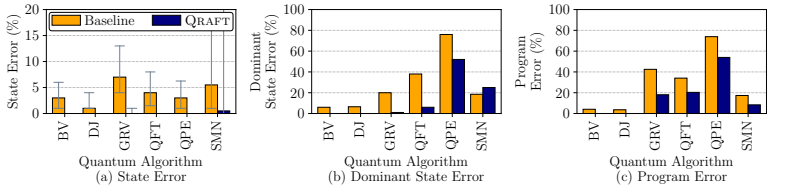


Figure 15: In general, compared to Baseline, QRAFT achieves a lower (a) median state error, (b) dominant state error, and (c) program error.

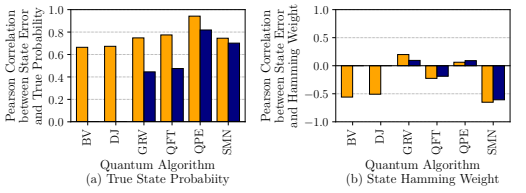


Figure 16: QRAFT reduces the error bias based on (a) true state probability, and (b) hamming weight.

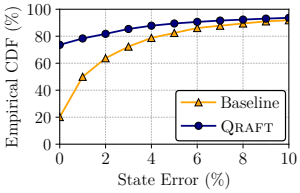


Figure 17: QRAFT achieves an error of 0% for ≈70% of the samples, while Baseline has 0% error for only 20% of the samples.

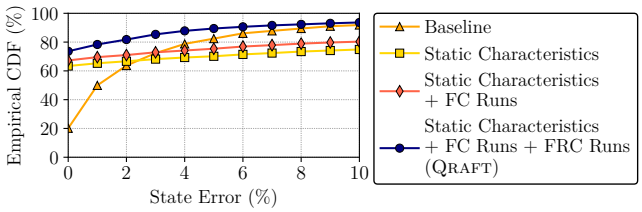


Figure 18: Training with the features generated using the forward + reverse runs (QRAFT) results in lower error than training using the features of just the forward runs or no runs.

in reducing the median state error to 0% across all algorithms except SMN, for which the median state error is reduced from 6% to 1%. Moreover, QRAFT considerably reduces the dominant state error across different algorithms. For example, the dominant state error of GRV is reduced from 20% to 2% and that of QFT is reduced from 37% to 5%. QRAFT also reduces the program error (sum of error of all states divided by two) across all algorithms. For example, the program error of SMN is reduced from 18% to 8%.

In Sec. 1, we saw that the state error of optimal circuit mapping was correlated with true state probability and state hamming weight. Fig. 16 shows that QRAFT reduces the Pearson Correlation of state error with true state probability and state hamming weight across algorithms. For example, the linear correlations of BV and DJ are reduced to 0 for both state properties. Thus, QRAFT also reduces error biases that are dependent on state properties.

While generally effective, QRAFT has further scope for improvement. For example, in case of QPE, the dominant state error is 50% with QRAFT, which even though is a significant improvement over the optimized circuit map (78% error), is still high. This is because QPE is a deep circuit which suffers from qubits losing state coherence. Therefore, the dominant state loses most its observed probability to other states. It is difficult for QRAFT to completely eliminate errors for such high-error algorithms, as the predictor also relies on observed state probabilities of the forward circuit, which are way off from the true state probabilities for QPE.

Fig. 17 shows the CDF of state errors of over 200 random quantum circuits used for testing. Note that different circuits have different number of qubits, depth and number of operations. We observe that over 70% of the samples (each state produces one state error sample) have 0% error with QRAFT (median error is 0%), while only 20% of the samples have 0% error with Baseline (median error is 2%). The 75^{th} percentile error with QRAFT is 1%, while that with Baseline is 4%.

> Overall, QRAFT reduces the median error in estimating the true probability of states by as much as 7%, dominant state error by as much as 30%, and program error by as much as 20% across the wide range of evaluated quantum algorithm. QRAFT also achieves an error rate of 0% for over 70% of circuit states for random circuits.

Does reversing the quantum circuit help toward better estimating the output state probability? A natural inquiry is about what happens if QRAFT is trained without the forward + reverse circuit, i.e., can we reduce the state error by only running the forward circuit and feeding other static circuit information (e.g., circuit depth, number of operations etc.). Toward this end, Fig. 18 shows the CDF of state error when the true state probabilities are predicted using QRAFT (all features listed in Table 2, including errors during the forward and reverse circuit runs), model trained with only the forward circuit runs and static characteristics (only the top two rows of features listed in Table 2), and model trained with just the static characteristics (only the top row of features listed in Table 2). As expected, training using only the static characteristics performs the worst. Adding the features of the FC runs only performs slightly better. In fact, on the tail end, the two methods perform worse than Baseline. The 75^{th} percentile state error with QRAFT is 1%, with Baseline it is 4%, with model trained using FC runs + static characteristics it is 5%, and with model trained using only static characteristics it is 11%. Reversibility helps make better estimate of the output probability because the true probabilities of the output states of the FRC are known and the FRC captures the effects of performing the sequence of operations in the FC.

What makes QRAFT work effectively? To better explain why QRAFT works, we investigate the relative importance of different features used by QRAFT (predictors listed in Table 2)

Importance score of a predictor in an ensemble of decision trees is calculated by averaging its importance score over all tree learners in the ensemble. In a given tree learner, the predictor importance

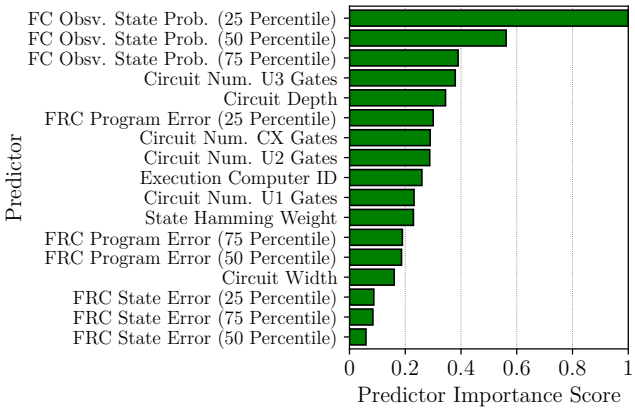


Figure 19: The top features include the program error of the forward + reverse circuit (FRC), along with other features.

score is calculated by summing up the misclassification risk due to all its splits and dividing the sum by the number of branch nodes (nodes that are not leaves).

Fig. 18 shows the features sorted according their importance score. As expected, the three features related to observed state probability of the FC circuit are the most important because the true state probabilities are likely to be in the vicinity of the observed probabilities (although the magnitude and direction of difference between the observed probability and the true probability cannot be known from just the FC circuit). The static circuit characteristics related to number of operations such as number of U3 and CX gates and circuit depth are also important because more operations cause more state error.

Next, we observe that one of the features related to the program error of the FRC ((25th percentile FRC program error) is among the most critical features. The program error of an FRC is an important characteristic because it correlates with the program error of the FC, as we also saw in Sec. 3. On the other hand, features related to the individual state errors of the FRC are in the bottom three because, as discussed in Sec. 3, the state errors do not correlate strongly. These rankings validate our expectation.

> Reversibility and characteristics related to the forward + reverse circuit execution enable QRAFT to reduce the state errors significantly because the forward + reverse circuit indirectly captures the effects of performing the sequence of quantum operations present in the forward circuit and has known true state probabilities.

Is QRAFT's effectiveness dependent on the underlying choice of circuit map? We study QRAFT's ability to reduce the error in the estimation of the true probability of the output state and the choice of circuit map. Executing the optimal circuit map might require having access to all the qubits. But we may not have this access or may not want to incur the compilation overhead of multiple layers of optimizations. We execute "non-optimized" circuit maps for randomly generated circuits. These circuit maps are directly mapped from the logical to the physical qubits without any optimizations (no gate cancellation or synthesis, no error-aware qubit mapping, no circuit depth reduction, etc.). Fig. 20 shows that QRAFT performs well even when applied to non-optimized circuit maps (CMs). QRAFT achieves an error of 0% for over 70% of the samples, while Baseline of non-optimized CMs has an error of 0% for only 15% of the samples. QRAFT applied to non-optimized CMs performs on par with QRAFT applied to optimized CMs.

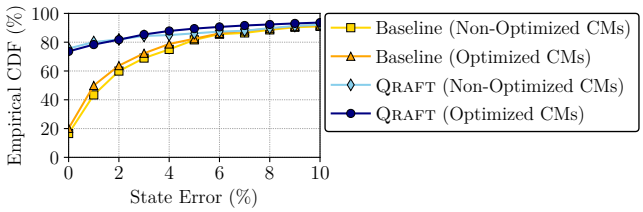


Figure 20: QRAFT achieves a low state error for even non-optimized circuit maps (CMs) as compared to Baseline.

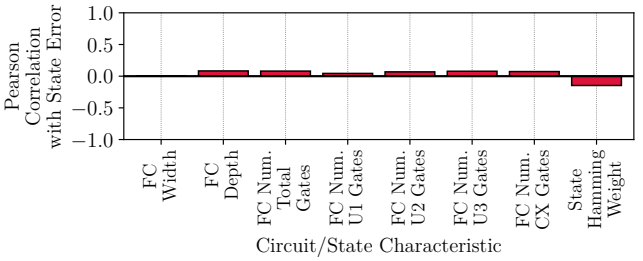
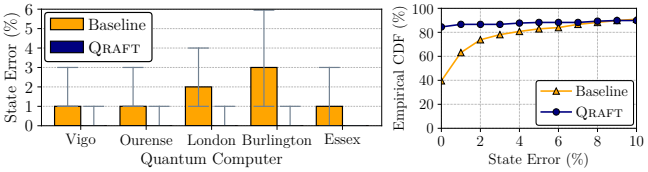


Figure 21: The state error of QRAFT is not significantly correlated with any of the forward circuit (FC) or state characteristics. QRAFT is effective for all types of circuits and states.

> QRAFT provides better estimation of the true output probability of the output states, even when using non-optimized circuit maps. When the availability of reliable qubits is constrained due to high demand, users can leverage QRAFT to estimate the correct output probability of the output states without needing to compile the optimal circuit map.

Is QRAFT's effectiveness sensitive to the circuit characteristics? Fig. 21 shows the Pearson correlation of QRAFT's state error with different forward circuit characteristics covered by the random circuits including width, depth, number of different types of gates, and state hamming weight. This analysis is important to identify if QRAFT is biased against any characteristics. As an example, if the state error of QRAFT's prediction was higher for circuits with a large number of operations, then the corresponding correlation would be high. However, the figure shows that QRAFT's performance is not significantly correlated with any of the circuit or state characteristics (absolute correlation is lower than 0.15 across all characteristics).

> QRAFT is effective at reducing the state error across different circuit and state characteristics including width, depth, number of gates, and state hamming weight.



(a) 5-Qubit Computers **(b)** 15-Qubit Computer

Figure 22: The median state error with QRAFT is 0% on (a) all 5-qubit computers and (b) the 15-qubit Melbourne computer.

Does QRAFT work effectively across different quantum machines? To test this, Fig. 22(a) shows that the median state error of QRAFT is 0% across all 5-qubit computers. While the five computers have the same qubit layout, they have different error rates as shown in the figure due to different qubit and operations fidelity. Burlington has the highest Baseline median state error of 3% and 75^{th} percentile error of 6%. QRAFT reduces the median state error on Burlington to 0% (hence, bars are not visible) and the 75^{th} percentile error of 1%.

Does QRAFT scale to a larger number of qubits? To test this, we evaluated QRAFT on the 15-qubit Melbourne quantum machine. Fig. 22(b) shows the CDF of the state error of QRAFT and Baseline for the 15-qubit Melbourne computer. The figure shows that only 40% of the samples have a state error of 0% using the Baseline method. On the other hand, using QRAFT results in over 80% of the samples having a state error of 0%. More importantly, QRAFT continues to perform well even when trained with smaller circuits only. For example, when QRAFT is trained with circuits of size ≤ 6 qubits and tested on circuits with 12-15 qubits on Melbourne, 80% of the samples have 0% error, compared to Baseline, which achieves 0% error for only 55% of the samples (results not plotted for brevity). Similar results are obtained with other error thresholds (e.g., QRAFT is trained with circuit size ≤ 5 qubits and tested on circuits with 10-15 qubits). This result is important because quantum circuits can only be practically simulated ideally on classical machines (which is needed to know their ground truth for training) if they are small in size. Thus, being able to train using the ground truth of smaller random circuits and achieve high-quality predictions for larger quantum circuits is a significant result. Note that training circuits, even though smaller in size, should collectively span all the physical qubits on the machine to cover the error characteristics and properties of all the qubits.

Overall, our results confirm that QRAFT is scalable to larger quantum computers too. This is expected since QRAFT does not employ any technique whose complexity worsens with the number of qubits – it simply adds the reverse components to the circuit for execution. This does not increase the execution time of circuits because quantum circuits have execution time in the order of milliseconds. Most of this time is taken in initializing and measuring the qubits – this time is the same for FCs and FRCs because both have the same number of qubits.

> Overall, QRAFT is effective across different quantum machines tested on the IBM QX cloud and scales effectively for quantum machines with larger number of qubits (15 qubits on the Melbourne quantum machine).

5 RELATED WORK

In this section, we discuss related NISQ computing works and provide concluding remarks for QRAFT. Current NISQ computing works can be broadly classified as following:

- **Quantum Simulation.** Some works have dedicated effort for developing parallel and noise-aware quantum simulators for a small number of qubits [9, 19, 23, 24, 28, 32].
- **Debugging Algorithms.** Prior works have focused on developing methods of debugging quantum algorithms, which is a difficult problem because, unlike classical bits, qubit states cannot be measured mid-execution [21, 30].
- **Control Pulse Optimization.** Several works have proposed optimizations for the microwave pulses applied at the lowest level to control qubits. These optimizations reduce circuit runtimes, which decreases the state errors [5, 6, 15, 17, 40].
- **Quantum Stack Development.** Considerable effort has been geared toward designing abstraction layers, compilation frameworks, and computation stacks for emerging quantum computing technologies [1, 2, 12, 20, 25, 34, 37].
- **Algorithm Mapping and Error Reduction.** Recent research has focused on best-effort mapping of quantum algorithms to real quantum hardware. These include works that execute layout-constraints-aware circuit mapping [4, 29, 52], characterize and minimize the number of quantum operations [16, 26, 39, 43, 50, 53], and perform error-and-cross-talk-adaptive logical-to-physical qubit mapping [3, 8, 13, 33, 35, 38, 44–48, 51]. For example, PyZX [26] is a python-based framework for circuit optimization that also verifies if the optimized circuit is mathematically equivalent to the original circuit using ZX calculus. Another work by Tannu et al. [46] uses an ensemble of multiple circuit maps with uncorrelated errors to try to reduce the overall output error. A recent work, Veritas [41], goes beyond qubit mapping. Veritas demonstrates the value of minimizing the error variance in qubit selection and applying post-execution statistical error corrections to estimate the correct output. But, Veritas lacks any "ground truth" information about the correct output and relies solely on the observed program output to estimate the correct output.

Note that QRAFT can also be applied to programs that intersperse measurement with computation (e.g., hybrid quantum-classical variational algorithms and repeat-until-success (RUS) circuits) by reversing each phase and handling it separately.

6 CONCLUSION AND FUTURE WORK

In this paper, we introduced QRAFT a method that utilizes the reversibility of quantum algorithms to estimate the correct program

output. We show that Qraft's improvements are not biased against specific algorithm, state, or computer characteristics. We hope that this work opens up opportunities for multiple follow-on works including leveraging reversibility for debugging, benchmarking and program analysis, exploiting reversibility for better circuit mapping, and theoretical modeling of reversibility property.

Acknowledgement. We are thankful to our shepherd, Dan R. K. Ports, and the anonymous reviewers to providing thoughtful feedback despite the pandemic challenges. We are thankful for the support from Northeastern University, NSF Award 1910601, and the Massachusetts Green High Performance Computing Center (MGHPCC) facility. We acknowledge the use of the IBM Q for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.

REFERENCES

[1] Talha Ahmed. 2018. *Q#: A Quantum Programming Language by Microsoft.* Ph.D. Dissertation. Imperial College London.

[2] G Aleksandrowicz, T Alexander, P Barkoutsos, L Bello, Y Ben-Haim, D Bucher, et al. [n.d.]. Qiskit: An Open-source Framework for Quantum Computing.(2019).

[3] Abdullah Ash-Saki, Mahabubul Alam, and Swaroop Ghosh. 2019. QURE: Qubit Re-allocation in Noisy Intermediate-Scale Quantum Computers. In *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 141.

[4] Debjyoti Bhattacharjee, Abdullah Ash Saki, Mahabubul Alam, Anupam Chattopadhyay, and Swaroop Ghosh. 2019. MUQUT: Multi-Constraint Quantum Circuit Mapping on NISQ Computers. In *38th IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2019*. Institute of Electrical and Electronics Engineers Inc., 8942132.

[5] Anastasiia Butko, George Michelogiannakis, Samuel Williams, Costin Iancu, David Donofrio, John Shalf, Jonathan Carter, and Irfan Siddiqi. 2019. Understanding Quantum Control Processor Capabilities and Limitations through Circuit Characterization. *arXiv preprint arXiv:1909.11719* (2019).

[6] Lauren Capelluto and Thomas Alexander. 2020. OpenPulse: Software for Experimental Physicists in Quantum Computing. *Bulletin of the American Physical Society* (2020).

[7] Sourav Chatterjee and Elizabeth Meckes. 2007. Multivariate Normal Approximation using Exchangeable Pairs. *arXiv preprint math/0701464* (2007).

[8] Lukasz Cincio, Kenneth Rudinger, Mohan Sarovar, and Patrick J Coles. 2020. Machine Learning of Noise-Resilient Quantum Circuits. *arXiv preprint arXiv:2007.01210* (2020).

[9] J Ignacio Cirac and Peter Zoller. 2012. Goals and Opportunities in Quantum Simulation. *Nature Physics* 8, 4 (2012), 264.

[10] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. 1998. Quantum Algorithms Revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454, 1969 (Jan 1998), 339–354. https://doi.org/10.1098/rspa.1998.0164

[11] David Collins, KW Kim, and WC Holton. 1998. Deutsch-Jozsa Algorithm as a Test of Quantum Computation. *Physical Review A* 58, 3 (1998), R1633.

[12] Rigetti Computing. 2019. Pyquil documentation. *URL http://pyquil.readthedocs.io/en/latest* (2019).

[13] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. 2019. A Case for Multi-Programming Quantum Computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 291–303.

[14] Miroslav Dobšíček, Göran Johansson, Vitaly Shumeiko, and Göran Wendin. 2007. Arbitrary Accuracy Iterative Quantum Phase Estimation Algorithm using a Single Ancillary Qubit: A Two-Qubit Benchmark. *Physical Review A* 76, 3 (2007), 030306.

[15] Eugen Dumitrescu, Raphael Pooser, and John Garmon. 2020. Benchmarking Noise Extrapolation with OpenPulse. *Bulletin of the American Physical Society* (2020).

[16] Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I Schuster, Henry Hoffmann, and Frederic T Chong. 2019. Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 266–278.

[17] Pranav Gokhale, Ali Javadi-Abhari, Nathan Earnest, Yunong Shi, and Frederic T Chong. 2020. Optimized Quantum Compilation for Near-Term Algorithms with OpenPulse. *arXiv preprint arXiv:2004.11205* (2020).

[18] Lov K Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 212–219.

[19] Gian Giacomo Guerreschi, Justin Hogaboam, Fabio Baruffa, and Nicolas PD Sawaya. 2020. Intel Quantum Simulator: A Cloud-Ready High-Performance Simulator of Quantum Circuits. *Quantum Science and Technology* 5, 3 (2020), 034007.

[20] Andrew Hancock, Austin Garcia, Jacob Shedenhelm, Jordan Cowen, and Calista Carey. 2019. Cirq: A Python Framework for Creating, Editing, and Invoking Quantum Circuits. *URL https://github.com/quantumlib/Cirq* (2019).

[21] Yipeng Huang and Margaret Martonosi. 2019. Statistical Assertions for Validating Patterns and Finding Bugs in Quantum Programs. In *Proceedings of the 46th International Symposium on Computer Architecture*. ACM, 541–553.

[22] Zhi-Xiang Jin and Shao-Ming Fei. 2018. Quantifying Quantum Coherence and Nonclassical Correlation based on Hellinger Distance. *Physical Review A* 97, 6 (2018), 062342.

[23] Tyson Jones, Anna Brown, Ian Bush, and Simon C Benjamin. 2019. Quest and High Performance Simulation of Quantum Computers. *Scientific reports* 9, 1 (2019), 1–11.

[24] Nader Khammassi, Imran Ashraf, Xiang Fu, Carmen G Almudever, and Koen Bertels. 2017. QX: A High-Performance Quantum Computer Simulation Platform. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 464–469.

[25] Nathan Killoran, Josh Izaac, Nicolás Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. 2019. Strawberry fields: A software platform for photonic quantum computing. *Quantum* 3 (2019), 129.

[26] Aleks Kissinger and John van de Wetering. 2019. PyZX: Large Scale Automated Diagrammatic Reasoning. *arXiv preprint arXiv:1904.04735* (2019).

[27] Pascal Koiran, Vincent Nesme, and Natacha Portier. 2005. A Quantum Lower Bound for the Query Complexity of Simon's Problem. In *International Colloquium on Automata, Languages, and Programming*. Springer, 1287–1298.

[28] Gushu Li, Yufei Ding, and Yuan Xie. 2019. SANQ: A Simulation Framework for Architecting Noisy Intermediate-Scale Quantum Computing System. *arXiv preprint arXiv:1904.11590* (2019).

[29] Gushu Li, Yufei Ding, and Yuan Xie. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 1001–1014.

[30] Ji Liu, Gregory T Byrd, and Huiyang Zhou. 2020. Quantum Circuits for Dynamic Runtime Assertions in Quantum Computation. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. 1017–1030.

[31] Margaret Martonosi and Martin Roetteler. 2019. Next Steps in Quantum Computing: Computer Science's Role. *arXiv preprint arXiv:1903.10541* (2019).

[32] David C McKay, Thomas Alexander, Luciano Bello, Michael J Biercuk, Lev Bishop, Jiayin Chen, Jerry M Chow, Antonio D Córcoles, Daniel Egger, Stefan Filipp, et al. 2018. Qiskit Backend Specifications for OpenQASM and OpenPulse Experiments. *arXiv preprint arXiv:1809.03452* (2018).

[33] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. 2019. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 1015–1029.

[34] Prakash Murali, Norbert Matthias Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. 2019. Full-Stack, Real-System Quantum Computer Studies: Architectural Comparisons and Design Insights. *arXiv preprint arXiv:1905.11349* (2019).

[35] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. 2020. Software Mitigation of Crosstalk on Noisy Intermediate-Scale Quantum Computers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. 1001–1016.

[36] Daniel C Murphy and Kenneth R Brown. 2019. Controlling Error Orientation to Improve Quantum Algorithm Success Rates. *Physical Review A* 99, 3 (2019), 032318.

[37] Scott Pakin and Steven P Reinhardt. 2018. A Survey of Programming Tools for D-Wave Quantum-Annealing Processors. In *International Conference on High Performance Computing*. Springer, 103–122.

[38] Tirthak Patel, Baolin Li, Rohan Basu Roy, and Devesh Tiwari. 2020. {UREQA}: Leveraging Operation-Aware Error Rates for Effective Quantum Circuit Mapping on NISQ-Era Quantum Computers. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*. 705–711.

[39] Tirthak Patel, Abhay Potharaju, Baolin Li, Rohan Roy, and Devesh Tiwari. 2020. Experimental evaluation of NISQ quantum computers: error measurement, characterization, and implications. In *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 636–650.

[40] Tirthak Patel and Devesh Tiwari. 2020. DisQ: a novel quantum output state classification method on IBM quantum computers using openpulse. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.

[41] Tirthak Patel and Devesh Tiwari. 2020. Veritas: Accurately Estimating the Correct Output on Noisy Intermediate-Scale Quantum Computers. In *2020 SC20:*

International Conference for High Performance Computing, Networking, Storage and Analysis (SC). IEEE Computer Society, 188–203.

[42] John Preskill. 2018. Quantum Computing in the NISQ Era and Beyond. *Quantum* 2 (2018), 79.

[43] Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I Schuster, Henry Hoffmann, and Frederic T Chong. 2019. Optimized Compilation of Aggregated Instructions for Realistic Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 1031–1044.

[44] Kaitlin N Smith and Mitchell A Thornton. 2019. A Quantum Computational Compiler and Design Tool for Technology-Specific Targets. In *Proceedings of the 46th International Symposium on Computer Architecture*. ACM, 579–588.

[45] Bochen Tan and Jason Cong. 2020. Optimality Study of Existing Quantum Computing Layout Synthesis Tools. *arXiv preprint arXiv:2002.09783* (2020).

[46] Swamit S Tannu and Moinuddin Qureshi. 2019. Ensemble of Diverse Mappings: Improving Reliability of Quantum Computers by Orchestrating Dissimilar Mistakes. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 253–265.

[47] Swamit S Tannu and Moinuddin K Qureshi. 2019. Mitigating Measurement Errors in Quantum Computers by Exploiting State-Dependent Bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 279–290.

[48] Swamit S Tannu and Moinuddin K Qureshi. 2019. Not All Aubits are Created Equal: A Case for Variability-Aware Policies for NISQ-Era Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 987–999.

[49] Yaakov S Weinstein, MA Pravia, EM Fortunato, Seth Lloyd, and David G Cory. 2001. Implementation of the Quantum Fourier Transform. *Physical review letters* 86, 9 (2001), 1889.

[50] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. 2019. Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations. In *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 142.

[51] Ellis Wilson, Sudhakar Singh, and Frank Mueller. 2020. Just-in-time Quantum Circuit Transpilation Reduces Noise. *arXiv preprint arXiv:2005.12820* (2020).

[52] Alwin Zulehner, Alexandru Paler, and Robert Wille. 2018. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018).

[53] Alwin Zulehner and Robert Wille. 2019. Compiling SU (4) Quantum Circuits to IBM QX Architectures. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ACM, 185–190.