

QuCT: A Framework for Analyzing Quantum Circuit by Extracting Contextual and Topological Features

Siwei Tan
Zhejiang University
Hang Zhou, China
siweitan@zju.edu.com

Xinghui Jia
Zhejiang University
Hang Zhou, China
jxhhhh@zju.edu.com

Yongheng Shang
Zhejiang University
Hang Zhou, China
yh_shang@zju.edu.com

Congliang Lang
Zhejiang University
Hang Zhou, China
langcongliang@zju.edu.com

Ziqi Tan
Zhejiang University
Hang Zhou, China
tanziqi@zju.edu.com

Andre Python
Zhejiang University
Hang Zhou, China
apython@zju.edu.com

Liang Xiang
Zhejiang University
Hang Zhou, China
xiangliang@zju.edu.com

Tingting Li
Zhejiang University
Hang Zhou, China
litt2020@zju.edu.com

Liqiang Lu^{*}
Zhejiang University
Hang Zhou, China
liqianglu@zju.edu.com

Shudi Wang
Zhejiang University
Hang Zhou, China
shudiwang@zju.edu.com

Jieming Yin
Nanjing University of Posts
and Telecommunications
Nan Jing, China
Jieming.Yin@njupt.edu.cn

Jianwei Yin^{*}
Zhejiang University
Hang Zhou, China
zjuyjw@zju.edu.com

ABSTRACT

In the current Noisy Intermediate-Scale Quantum era, quantum circuit analysis is an essential technique for designing high-performance quantum programs. Current analysis methods exhibit either accuracy limitations or high computational complexity for obtaining precise results. To reduce this tradeoff, we propose QuCT, a unified framework for extracting, analyzing, and optimizing quantum circuits. The main innovation of QuCT is to vectorize each gate with each element, quantitatively describing the degree of the interaction with neighboring gates. Extending from the vectorization model, we propose two representative downstream models for fidelity prediction and unitary decomposition. The fidelity prediction model performs a linear transformation on all gate vectors and aggregates the results to estimate the overall circuit fidelity. By identifying critical weights in the transformation matrix, we propose two optimizations to improve the circuit fidelity. In the unitary decomposition model, we significantly reduce the search space by bridging the gap between unitary and circuit via gate vectors. Experiments show that QuCT improves the accuracy of fidelity prediction by 4.2× on 5-qubit and 18-qubit quantum devices and achieves 2.5× fidelity improvement compared to existing quantum compilers [19, 55]. In unitary decomposition, QuCT achieves 46.3× speedup for 5-qubit unitary and more than hundreds of speedup for 8-qubit unitary, compared to the state-of-the-art method [87].

*Corresponding Author: Jianwei Yin, Liqiang Lu

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

MICRO '23, October 28–November 1, 2023, Toronto, ON, Canada

© 2023 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0329-4/23/10.

<https://doi.org/10.1145/3613424.3614274>

CCS CONCEPTS

- Hardware → Quantum technologies.

KEYWORDS

quantum computing, quantum circuit synthesis, quantum error correction

ACM Reference Format:

Siwei Tan, Congliang Lang, Liang Xiang, Shudi Wang, Xinghui Jia, Ziqi Tan, Tingting Li, Jieming Yin, Yongheng Shang, Andre Python, Liqiang Lu, and Jianwei Yin. 2023. QuCT: A Framework for Analyzing Quantum Circuit by Extracting Contextual and Topological Features. In *56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23), October 28–November 1, 2023, Toronto, ON, Canada*. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3613424.3614274>

1 INTRODUCTION

Quantum computing has developed rapidly over the last few decades, offering polynomial or even exponential speedup in several areas, such as chemistry simulation [9], database search [29], and combinatorial optimization [21]. Quantum circuit is a widely-used quantum programming model that describes the computation by quantum gates. In the current Noisy Intermediate-Scale Quantum (NISQ) era [63], strong motivation exists to develop circuit analysis and optimization techniques to improve the efficiency of quantum circuit design. For example, estimating fidelity is critical in minimizing noise overhead [3, 42, 65, 66], thereby improving the probability of a circuit producing the correct result [12, 49, 57, 79, 84]. Additionally, for applications represented as unitary matrices (unitaries), it is necessary to decompose them into circuits with executable basic gates, such as single-qubit and two-qubit gates [6, 26, 34, 72, 80].

However, the analysis and optimization of quantum circuits still rely on classical computers, which have to face the trade-off between accuracy and computational burden. For example, cross-entropy benchmarking (XEB) [3] and randomized benchmarking (RB) [42] are two widely-used fidelity models. They model gate

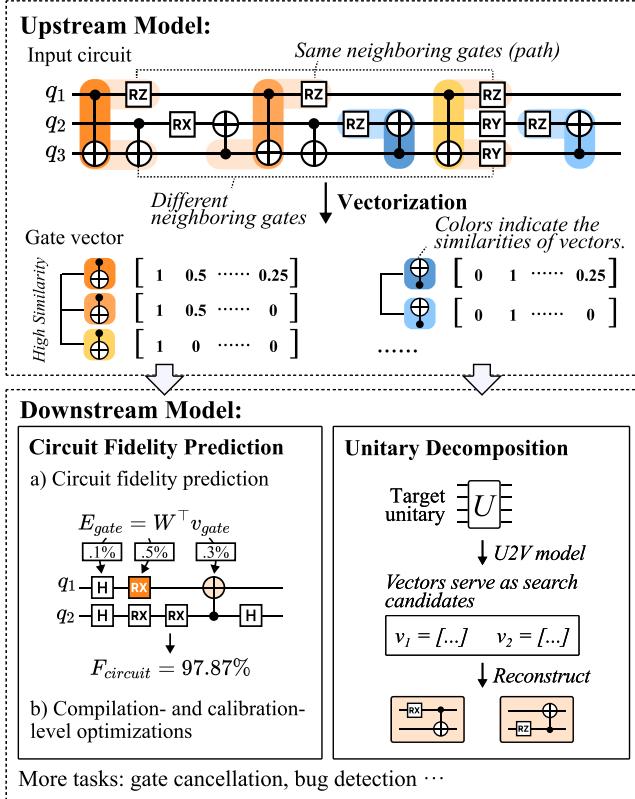


Figure 1: Key components of the QuCT framework. In the upstream model, each gate is transformed into a vector that captures its neighboring circuit features. The downstream models take these vectors as input for various analysis tasks, such as fidelity prediction and unitary decomposition.

fidelities as single values and efficiently estimate the circuit fidelity via a polynomial function, while they exhibit low accuracy. For example, estimating the fidelity of a 5-qubit Grover algorithm [29] on IBM Oslo quantum processor may lead to about 15% difference between the real and predicted fidelity using the RB-based method. An alternative approach is to simulate the Master's equation with density matrix. This is accurate but shows high computational complexity. For example, the number of qubits is limited to 34 when simulating noisy circuits on A100 GPU [33]. Techniques of unitary decomposition also suffer from this dilemma. Mathematical approaches like Column-by-Column Decomposition (CCD) [34] and Quantum Shannon Decomposition (QSD) [72] can decompose a 5-qubit unitary in a few seconds, but generate thousands of gates, leading to poor performance and even failure when deploying to quantum devices. On the other hand, aiming to minimize the number of gates, the recent decomposition approach QFAST [87] adopts a search-based method to approximate the target unitary. However, this advantage comes at the expense of increased time complexity. For example, it takes around 60 hours for QFAST to decompose a 5-qubit unitary.

These limitations fundamentally originate from the absence of an analysis method to extract and preserve circuit features. The inaccuracy of the XEB- and the RB-based approaches mainly comes

from the inability to model errors caused by gate interactions, such as crosstalk [84] and pulse distortion [69]. CCD [34] and QSD [72] inherently follow matrix decomposition theories without considering the features of the circuit structure. On the other hand, because of the lack of preserving circuit features in a formal representation, the accurate approaches [33, 87] have to repeatedly go through the circuit, requiring a large amount of time. For example, to accurately predict the fidelity, it needs to simulate a circuit multiple times [33]. Similarly, to get a decomposition solution with fewer gates, QFAST [87] has to revisit a large number of candidate circuits and calculate the matrix distance to the target unitary. Ideally, an extraction should thoroughly cover the contextual and topological features of the circuit, providing a unified model to enable rigorous analysis and optimization tasks.

In this work, we propose QuCT, a unified framework that comprises an upstream model and multiple downstream models. Figure 1 shows the overview of QuCT. The upstream model is characterized by its ability to vectorize each quantum gate while taking the circuit features into consideration. We first formally define the concept of *path* that describes the relation of a gate with its neighboring gates in terms of types (e.g., CX, RZ) and execution orders (e.g., parallelism, dependency). Each gate is vectorized by enumerating the paths starting from it. In the gate vector, each element represents a path, and its value indicates the degree of correlation between the path and the starting gate. The primary advantage of vectorization is that it transforms the unstructured circuit into a set of one-dimensional vectors, which significantly reduces the arithmetic complexity while retaining the contextual and topological features. The vectorization model is trained offline and only needs to be performed once for a target circuit.

The vectorization serves as a new representation of quantum circuit, allowing various analysis and optimization tasks. In this work, we introduce two downstream models to perform fast and accurate fidelity prediction and unitary decomposition, respectively. In the fidelity prediction model, the circuit fidelity is estimated by applying a linear transformation on the gate vectors, where the transformation matrix is trained using a fidelity dataset obtained from the real execution results of quantum circuits. Moreover, the prediction provides guidance to mitigate circuit error in gate scheduling and hardware calibration. In the unitary decomposition model, the gate vector helps to bridge the gap between the unitary and the circuit. Different from existing methods that exhaustively enumerate all possible gate combinations as search candidates, our model effectively reduces the search space by identifying gate vectors that may be involved in the resulting circuit of the target unitary. Benefiting from the expressive representation, we can easily obtain the decomposition solution by reconstructing the circuit based on the paths recorded in the gate vector.

By extending downstream models, our framework can also be applied to other analysis tasks, such as gate cancellation and bug detection. The main contributions of this paper are summarized as follows:

- We propose QuCT, a unified framework for quantum circuit analysis, which decouples analysis tasks into an upstream vectorization model and multiple downstream models, providing accurate analytical results with low computational costs.

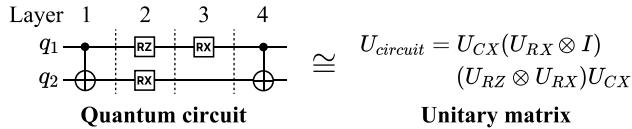


Figure 2: Example of a quantum circuit with four layers and its equivalent unitary.

- We propose an accurate model for fidelity prediction, which is extended from the upstream model. Benefiting from our vectorization representation, our model naturally supports the analysis of the errors caused by gate interactions and offers optimization techniques to mitigate these errors.
- We propose a unitary decomposition model that achieves remarkable speedup compared to the state-of-the-art method [87]. Our approach prunes search space efficiently by capturing the circuit similarity between different unitary matrices.

In the experiment, QuCT reduces the inaccuracy from 24.37% to 5.68% in the fidelity prediction compared to the XEB-based model [3] on an 18-qubit superconducting quantum device. Furthermore, QuCT achieves 2.5 \times fidelity improvement compared to the current compilers [19, 55]. In the decomposition of 5-qubit unitaries, QuCT achieves 46.3 \times speedup and shows 1.3 \times gate reduction compared to QFAST [87]. QuCT can also decompose an 8-qubit unitary in 8.3 hours, whereas alternative approaches would take over a month [87] or generate 26.2 \times more gates [72]. The source code of QuCT is publicly available on (<https://github.com/JanusQ/QuCT-Micro2023>).

2 BACKGROUND

2.1 Quantum Circuit

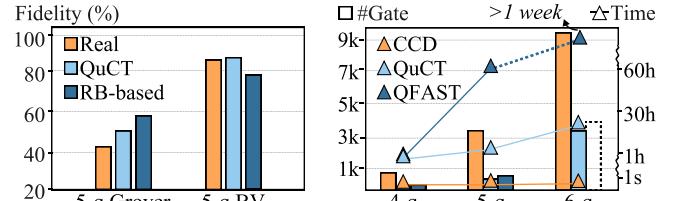
Quantum circuit (circuit) is a widely-used quantum programming model. It consists of a sequential arrangement of *quantum gates* (gates) G that operate on a set of *qubits* Q . Each quantum gate G comprises an operation and operated qubits Q_i :

$$\begin{aligned} G &= \{g_1, g_2, \dots, g_N\} \\ g_i &= (op, \{Q_i\}), Q_i \in Q \end{aligned} \quad (1)$$

Gates that manipulate one qubit refer to single-qubit gates (e.g., RX, RY, RZ, H, and U gates), and gates that manipulate two qubits refer to two-qubit gates (e.g., CX \oplus , and CZ \otimes gates). Not all gates (e.g., 3-qubit unitary gates) can be directly implemented on the target quantum device. Before deployment, they must be transformed into *basic gates* that only include specified types of gates determined by the hardware.

Definition 1. We define *layer* as the basic unit of the circuit timeline. In each layer, a qubit can be operated by at most one gate, and gates within the same layer are executed in parallel.

Each gate is mathematically represented as a unitary matrix according to its operation and parameters. The overall unitary of the circuit is calculated by applying matrix multiplication and tensor-product (\otimes) on the unitaries of gates. Figure 2 provides an example of a circuit with four layers and its unitary, where U_{CX} , U_{RX} , U_{RZ} are unitaries of CX, RX and RZ gates, and I represents the 2×2 identity matrix.



(a) Real and predicted fidelities

(b) Numbers of resulting gates and time cost of unitary decomposition.

Figure 3: Motivational examples.

2.2 Quantum Circuit Analysis

In this paper, we introduce two representative tasks for quantum circuit analysis.

Fidelity prediction aims to estimate the probability of getting correct results of a circuit under noise. In the case of superconducting quantum computers, errors can be categorized into two types [25], including: *a*) gate errors resulting from decoherence and imperfect implementation; and *b*) measurement errors occurring when qubit information is read into classical hardware. Recent works [3, 42, 77] predict the overall circuit fidelity in a polynomial form as follows:

$$F_{circuit} = \prod_{q \in Q}^{N_q} F_q \prod_{q_1, q_2 \in Q}^{N_{q_1, q_2}} F_{q_1, q_2} \prod_{q \in Q}^{N_q} MF_q, \quad (2)$$

where N_q , N_{q_1, q_2} denote the number of single-qubit gates of qubit q and the number of two-qubit gates between qubits q_1 , q_2 . F_q , F_{q_1, q_2} , and MF_q represent the single-qubit gate, the two-qubit gate, and the measurement fidelities, respectively. We define gate error as $(1 - F_q)$ or $(1 - F_{q_1, q_2})$. In addition to the aforementioned types of errors, errors from unexpected gate interactions between gates, e.g., crosstalk [57] and pulse distortion [69].

Unitary decomposition takes a unitary as input and decomposes it into matrices of basic gates, resulting in an equivalent circuit. Early methods, such as CSD [34], QSD [72], and CCD [34] decompose a unitary into a sequence of smaller unitaries following mathematical equations (e.g., cosine-sine decomposition function [78]). In contrast, recent methods [6, 15, 67, 87] apply search-based methods. To approximate the target unitary, they iteratively search and insert gates to the end of the circuit. QFAST [87] is the state-of-the-art search-based approach that aims to minimize the number of gates after decomposition. In this paper, we extend QFAST as one of our downstream models.

2.3 Motivational Examples

Many fidelity optimization frameworks [3, 20, 42, 57, 69] use Equation 2 to predict fidelity, while they fail to capture the noise resulting from gate interactions, such as crosstalk [84] and pulse distortion [69]. The accuracy of the prediction largely determines the performance of the optimization. For example, UREQA [61] prioritizes prediction accuracy. By considering the noise variance in different types of operations, it achieves a fidelity improvement of around 10% in qubit mapping compared to [75]. However, it ignores gate interactions related to the circuit structure. Additional

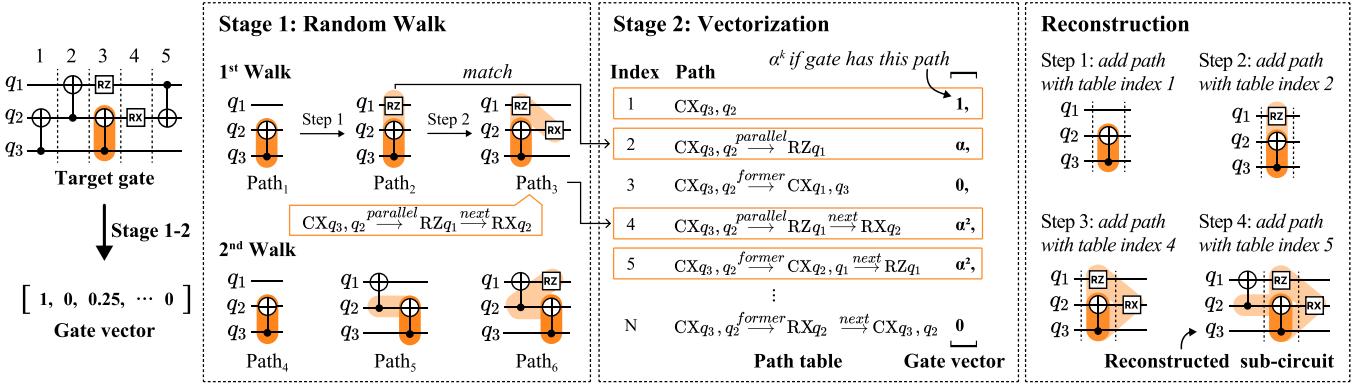


Figure 4: Two-stage process to vectorize a gate in the circuit and the reconstruction process from the gate vector.

experiments and optimizations are required to identify and mitigate specific types of interactions [3, 20, 42, 57, 69]. As their results cannot be integrated into an overall prediction function, the error types they focus on have to be optimized separately. Each optimization may decrease one type of error while increasing another. Figure 3 (a) displays the real fidelities for running the Grover [29] and the BV [5] algorithms on the 5-qubit IBM Oslo quantum device. The fidelities predicted by Equation 2 (with model parameters derived from RB) exhibit differences of 15.54% and 10.40% compared to the real fidelities.

Unitary decomposition plays an important role in circuit optimization [11] and algorithm design [37]. However, early decomposition methods, including QSD [72], CSD [80], and CCD [34], introduce massive redundant two-qubit gates between qubits that actually show no entanglement. Figure 3 (b) presents an example. CCD, the default decomposition method of Qiskit [2], requires more than 9,000 gates to decompose a 6-qubit unitary. Alternatively, searched-based methods may achieve more than 3× gate reduction by searching and inserting the gates that make the circuit closer to the target unitary. However, the search process is very time-consuming. For example, QFAST [87] takes an average of 60.92 hours to decompose a 5-qubit unitary and over a week to decompose a 6-qubit unitary.

The low accuracy and high time complexity of these methods fundamentally come from the lack of a representation to convert the circuit features into mathematical forms. The inaccuracy of prior prediction models [3, 42, 61] mainly results from the unthorough extraction of topological information (the connection and dependency between quantum gates), rendering it impossible to model the complex gate interactions. On the other hand, prior unitary decomposition models [15, 67, 87] exhaustively rely on arithmetic approaches without leveraging the contextual information (the parameter space of quantum gates), incurring massive invalid exploration during the search.

This paper aims to develop an intermediate representation to preserve both contextual and topological features while keeping formulation-friendly. Considering that a quantum circuit is naturally a sparsely-connected graph, we find that vectorization is an effective approach to extract features of such graphs [36, 47]. Our key insight is to leverage random walk [47] to vectorize the circuit

such that gate interactions are captured. Clearly, each element of the vector is assigned a value, which gives the degree to how the gates in a region affect each others. The vector representation enables accurate and fast modeling for various analysis tasks. As shown in Figure 3, QuCT achieves a 1.7× and 2.7× reduction in inaccuracy for these two algorithms by modeling gate interactions on the IBM Oslo quantum processor. Besides, by bridging the gap between the unitary and circuit structure using gate vectors, QuCT achieves remarkable speedup compared to QFAST, meanwhile requiring less number of gates.

3 UPSTREAM MODEL: GATE VECTORIZATION

Before introducing our vectorization model, we define *path* as follows.

Definition 2. A path is a chained relation between multiple gates. A k -step path is formulated as follows,

$$\text{Path} = g_1 \xrightarrow{r_1} g_2 \xrightarrow{r_2} \cdots \xrightarrow{r_k} g_{k+1}, \quad (3)$$

where r_i denotes the relation between two gates, which is categorized into three types: *former*, *next*, and *parallel*. These terms indicate that gate g_{i+1} is in the former, the next, and the same layers of gate g_i , respectively. As shown in Stage 1 in Figure 4, taking Path₃ as an example,

$$\text{Path}_3 = \text{CX}_{q_3, q_2} \xrightarrow{\text{parallel}} \text{RZ}_{q_1} \xrightarrow{\text{next}} \text{RX}_{q_2}$$

is a path starting from the gate CX_{q₃, q₂}. It describes a sub-circuit where the gate RZ_{q₁} is in the same layer as CX_{q₃, q₂}, and the gate RX_{q₂} is in the next layer of RZ_{q₁}.

3.1 Vectorization

To generate paths for each gate, we apply random walk [47] in the circuit. Random walk is a popular algorithm in the graph domain to explore neighboring information of nodes. When extending a path, the next gate is randomly selected from gates that share a relation with the former gate. For the gate that requires vectorization, we collect multiple paths starting from it within a specific number of steps. Figure 4 shows six paths of the CX_{q₃, q₂} gate. The first walk

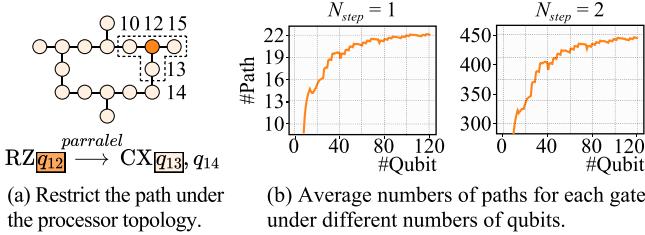


Figure 5: Minimizing the size of the path table. (a) The adjacent qubits of q_{12} include q_{10} , q_{13} , and q_{15} ; (b) Paths are generated under the IBM brick-like topology.

is responsible for the following paths:

Step=0. Path₁ : CXq_3, q_2

Step=1. Path₂ : $CXq_3, q_2 \xrightarrow{\text{parallel}} RZq_1$

Step=2. Path₃ : $CXq_3, q_2 \xrightarrow{\text{parallel}} RZq_1 \xrightarrow{\text{next}} RXq_2$.

The second walk generates the following paths:

Step=0. Path₄ : CXq_3, q_2

Step=1. Path₅ : $CXq_3, q_2 \xrightarrow{\text{former}} CXq_2, q_1$

Step=2. Path₆ : $CXq_3, q_2 \xrightarrow{\text{former}} CXq_2, q_1 \xrightarrow{\text{next}} RZq_1$.

These two walks generate six paths starting from the same target gate but in different directions.

The total number of paths for a given gate is determined by the number of steps (N_{step}) per walk and the number of walks (N_{walk}), which are configurable parameters. The number of walks determines the number of neighboring sub-circuits sampled as circuit features. The number of steps determines the maximum number of gates in these sub-circuits. Accordingly, the upper-bound number of paths for one gate can be estimated as $(N_{step}N_{walk} + 1)$, where 1 is the 0-step path.

The quantum gate is then vectorized by comparing its paths to a static *path table*. To be specific, the paths in this table are offline generated by enumerating all possible parameters of Equation 3. For each gate, the dimension of the vector equals the size of the path table. If a k -step path in the path table matches a path generated in the random walk, the corresponding element value is set to α^k , where $\alpha \in (0, 1]$ is a decay parameter. For example, in Figure 4, the generated 2-step Path₃ matches the 4th path in the path table. Thus, the 4th element of the gate vector is set to α^2 . Shorter paths are assigned with a larger value, which follows the intuition that the analysis should pay more attention to adjacent gates since they are more likely to exhibit a higher interaction.

3.2 The Size of Path Table

According to Equation 3, the size of the path table depends on the number of steps and the settings of relation r_i and gate g_i . Relation r_i has only three types, while gate g_i includes operated qubits and the gate type. The gate type is hardware-dependent, which consists of the basic gates supported by the target hardware. For example, the gate set of Google Sycamore hardware is composed of \sqrt{X} , \sqrt{Y} , \sqrt{W} and $fSim$ gates [3], while the gate set includes ID, RZ, SX, X, and CX gates for IBM Manila device. In this paper, our gate set

includes RX, RY, RZ, and CZ gates derived from our self-developed superconducting quantum hardware.

Since each path aims to capture the interaction between the starting gate and its neighboring gates, we impose that, for each path in the table, the qubits of gates g_i should be physically connected to the qubits of the starting gate under the device topology. For example, as shown in Figure 5 (a), only qubits q_{10} , q_{15} and q_{13} are connected with qubit q_{12} . Thus, starting from the gate RZq_{12} , paths can only involve gates that operate on q_{10} , q_{15} and q_{13} , e.g., $RZq_{12} \xrightarrow{\text{parallel}} CXq_{13}, q_{14}$. The table size is, therefore, mainly determined by the complexity of the device topology, not the number of qubits. We also remove redundant paths that lead to the equivalent circuit. For example, $RXq_3 \xrightarrow{\text{parallel}} RZq_1 \xrightarrow{\text{next}} RXq_2$ and $RXq_3 \xrightarrow{\text{next}} RXq_2 \xrightarrow{\text{former}} RZq_1$ describe the same circuit features. Figure 5 (b) shows the number of paths for each gate under the brick-like topology, where there are around 22 and 450 paths for 128-qubit IBM Washington processor with $N_{step} = 1$, and $N_{step} = 2$, respectively.

3.3 Expressivity

There have been various representations to extract graph features such as graph kernel [44], graph sampling [83], spectrum analysis [28], and random walk [47]. Among them, random walk is an effective approach for sparsely-connected graphs, e.g., recommendation systems [36] and knowledge inference [47]. The sparsity is also a prominent feature of quantum circuits. However, different from the prior random walk-based methods that capture the similarity between different nodes (e.g., finding similar preferences of two customers) by comparing the paths of each node, our vectorization is expressive to preserve the surrounding information of the target gate for circuit reconstruction. Clearly, the contextual features are recorded in the parameter g_i of Equation 3, including gate types and operated qubits. The topological features are preserved as the relation in the path. By referring to the path table with the nonzero elements of the gate vector, we can identify the paths related to the target gate and partially reconstruct the circuit. The target gate is at the head of the path. Layers of neighboring gates can be retraced according to their relations with their previous gates.

Figure 4 presents an example of reconstruction. According to the gate vector and the path table, the 1st, 2nd, 4th, and 5th paths in the table are identified. The first path contains only the starting gate. Based on the relation in these paths, we can resketch the sub-circuit within the step. The ability of circuit reconstruction offers the opportunity to the downstream models that involve circuit generation from the gate vector, such as the unitary decomposition.

4 DOWNSTREAM MODEL 1: CIRCUIT FIDELITY PREDICTION AND OPTIMIZATION

This section presents the methodology for modeling and optimizing circuit fidelity using our vectorization technique.

4.1 Fidelity Prediction

QuCT revises the prediction in Equation 2 by formulating the error E of each gate as the dot-product between its vector v_i and a weight

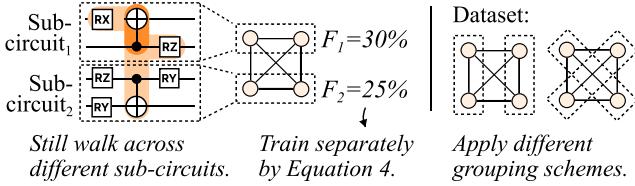


Figure 6: Workflow to use separable circuits to train weight vector W .

vector W :

$$E(v_i) = W^\top v_i,$$

$$F_{\text{circuit}} = \prod_{g_i \in G} (1 - E(v_i)) \prod_{q \in Q} MF_q. \quad (4)$$

The weight vector W is trained by the stochastic gradient descent algorithm [39] based on a fidelity dataset. This dataset is hardware-dependent. The dataset consists of the ground-truth circuit fidelities by executing a set of randomized circuits on the target quantum device. The ground-truth fidelity of each circuit is labeled via the Hellinger fidelity function [51] as follows:

$$F_{\text{circuit}}^{\text{groundtruth}} = 1 - \frac{1}{\sqrt{2}} \|\sqrt{P_{\text{measured}}} - \sqrt{P_{\text{ideal}}} \|_2, \quad (5)$$

where P_{measured} and P_{ideal} are the measured and ideal (noise-free) distributions, respectively.

Here, we briefly introduce the reason for choosing dot-product as the equation to predict the gate fidelity. According to [40], the error of a gate is mathematically equal to the sum of the trace of the Kraus operators, where each Kraus operator formulates the evolution caused by a source of noise. This suggests that the dot-product is consistent with mathematical intuition. In other words, each weight evaluates the effect of a path, which may correspond to the trace of a source of noise. Specifically, the weight element for the 0-step path models noise from the gate itself, while weights for other paths represent noise from the interaction among gates. The weight element of 0 suggests the path is not related to a source of error. We also tried other methods like machine learning and deep learning. They show limited improvement yet are accompanied by a disproportionately high computational complexity.

In the fidelity dataset construction, we allow more randomness when generating circuits to ensure the generality of the model. Clearly, given the hardware constraint of basic gates and topology, we generate circuits with different numbers of gates and proportions of two-qubit gates by randomly inserting gates. Compared to the circuits in RB [42] (only Clifford gates) and XEB [3] (only repeated blocks), our fidelity dataset can cover more complex interactions between gates. In QuCT, our dataset comprises 2000 circuits for each device with 5 qubits or 18 qubits, with the circuit depth ranging from 5 to 160. The time of generating the dataset takes around 1.7 hours per device (including 20.0 minutes of QPU access time).

We also allow generating separable circuits for processors with more than hundreds of qubits in the fidelity dataset construction. Circuits executed on these processors usually have a large number of gates, making the final fidelity vanish to zero. A large amount

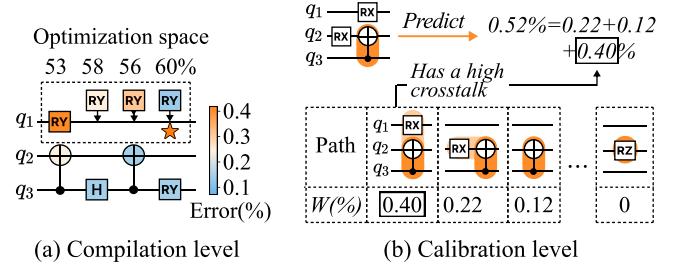


Figure 7: Fidelity optimization techniques.

of zero-valued fidelities means less valid information, which negatively affects the training convergence and the model accuracy. To address this, the separable circuits used in the fidelity dataset restrict the entangled qubits into sub-circuits within a small number of qubits. For example, in Figure 6, the circuit can be partitioned into two independent sub-circuits that execute simultaneously on the target hardware. Each sub-circuit has a relatively smaller number of gates, leading to a higher fidelity. We label the fidelity of each sub-circuit and use it for training. Note that the paths of gate vector v_i still walk across different sub-circuits, thereby capturing the interactions of the entire circuit. To improve generality and accuracy, we apply breadth-first-search to generate different grouping schemes. Since qubits are sparsely connected in real-world quantum hardware, the fidelity dataset is sufficient to cover all grouping schemes. For example, there are 236 grouping schemes under the 128-qubit IBM Washington device topology, while the fidelity dataset contains thousands of circuits.

4.2 Fidelity Optimization

As QuCT provides fine-grained gate fidelity prediction and interpretable weight, it allows various optimization techniques to improve the circuit fidelity.

Compilation-level optimization. A typical compilation flow includes routing and scheduling. The routing pass transforms the circuit to satisfy the processor topology. Clearly, it inserts SWAP gates to change the qubit mapping, ensuring that all two-qubit gates can be implemented by the coupler of the processor. By precisely predicting the fidelity, QuCT can be integrated with existing compilers [12, 61, 87] to find the routing solution with the best fidelity. For instance, the recent SATMAP [55] compiler uses a MAX-SAT solver to find the routing solution that minimizes the number of gates, which leads to different output circuits due to heuristic search. By extending SATMAP with our prediction model, we can guide the compilation to select the output circuit with maximum fidelity (Abbr. *QuCT_route_opt*). Scheduling means adjusting the layer of gates to improve the fidelity under the execution dependency. For example, in Figure 7 (a), the RY gate operated on q_1 can be moved to any of the following three layers with the functionality of this circuit remaining the same. By analyzing the circuit fidelity under each scheduling choice, QuCT helps to find the gate allocation with the highest predicted fidelity (Abbr. *QuCT_sched_opt*).

Calibration-level optimization. Calibration tries to locate the error in the circuit and tune the amplitudes and phases of pulses to improve the fidelity [38, 41]. By setting the decay parameter

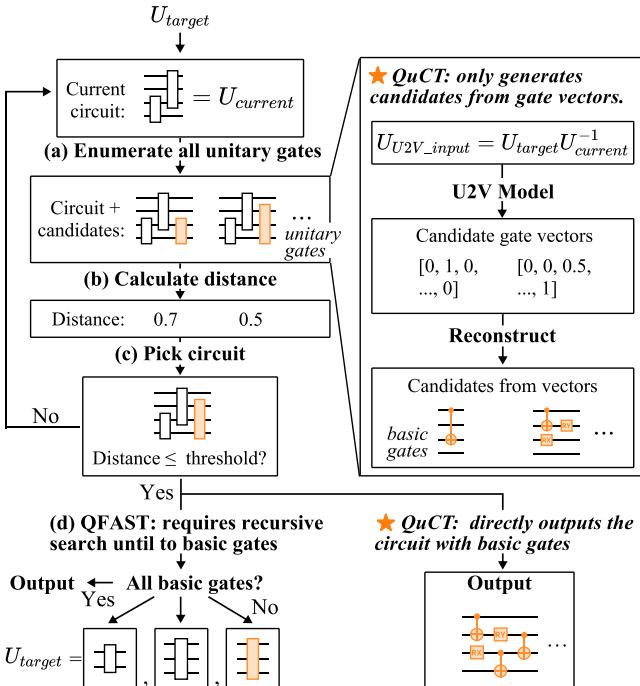


Figure 8: Workflow of the unitary decomposition.

to 1 during vectorization, QuCT provides an interpretable weight (Equation 4). As there is a one-to-one correspondence between weights and paths, by identifying the path with a large weight, critical source of noise are efficiently located. For example, calibrating crosstalk is a necessary step to mitigate error at the pulse level [45], which requires $\mathcal{O}(N^2)$ execution on quantum hardware [57]. Using QuCT, we can easily find such crosstalk by identifying large weights with 1-step paths. Figure 7 (b) provides an example. In the table, according to the value of each weight element, the RX gate on q_1 increases the error of its parallel CX gate on q_2 and q_3 by 0.4%. As a 0.4% reduction is significant compared to average gate error (~0.1%), this suggests high crosstalk that requires more attention to minimize this error.

5 DOWNSTREAM MODEL 2: UNITARY DECOMPOSITION

The second application of our vectorization model is used to decompose unitaries. We integrate our vectorization method with QFAST [87], which is a state-of-the-art method for achieving the minimum number of decomposed gates. It adopts an A* recursive algorithm, which iteratively approximates the target unitary by inserting unitary gates until the matrix distance is within the threshold. As shown in the left part of Figure 8, a typical QFAST iteration consists of four steps:

- (a) For the current circuit (e.g., a 4-qubit circuit that has been inserted with two unitary gates), enumerate all possible small unitary gates as candidates (e.g., 2-qubit and 3-qubit unitary gates) that have fewer qubits than the current circuit.

- Insert these candidate gates at the end of the current circuit, search their parameters, and calculate the updated matrix distance with the target unitary.
- Select the updated circuit with the minimum distance and check whether the distance is less than the threshold. If not, return to step (a).
- If the distance is within the threshold, check whether all unitary gates are in the basic gate set. If not, decompose them following steps (a) to (c).

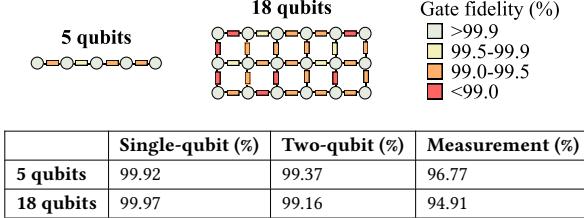
QFAST suffers from a long decomposition time due to exhaustive searching among various candidates. On the one hand, each unitary gate has numerous parameters to search before calculating the distance. The number of parameters increases exponentially with the number of qubits, e.g., it takes more than 30 minutes to search among 10 candidate gates for an 8-qubit circuit in one iteration. On the other hand, the search process also requires decomposing all unitary gates into basic gates, which dramatically increases the number of iterations.

The right part of Figure 8 illustrates how QuCT accelerates the decomposition process. Instead of exhaustively enumerating a large number of candidate gates, our approach is to consider a gate vector as a search candidate. The gate vector serves as an intermediate representation between the unitary and the circuit. By identifying the vectors that may be involved in the circuit of the target unitary, we can prune the candidate space. More importantly, we can easily reconstruct the circuit with basic gates according to the paths recorded in the gate vector, eliminating the additional overhead of the decomposition to basic gates.

5.1 Unitary-to-Vector Model

Since each gate vector implies the features of a sub-circuit (see Section 3.3), the purpose of the U2V model is to find the candidate vectors that tend to be part of the resulting circuit of the target unitary. The U2V model serves as the bridge between unitaries and gate vectors, where the sub-circuits reconstructed from these candidate vectors will replace the search space of QFAST. To build such a model, we obtain a U2V dataset composed of $\langle \text{unitary}, \{\text{vectors}\} \rangle$ pairs, derived from a set of random circuits generated with the same scheme mentioned in Section 4.1. To obtain high-quality decomposition results, these circuits are optimized using Qiskit transpiler [2] to minimize the number of gates. We then run our upstream vectorization model to obtain the vectors and calculate the unitaries of these circuits. Note that the circuit is not necessarily optimal because the U2V model aims to capture the potential gate vectors of the target unitary rather than the entire circuit. In other words, by combining these vectors, we might get an alternative circuit with a more compact representation. Based on this dataset, we train a random forest model [7] with k decision trees. Given a unitary as input, each tree will predict a gate vector that shows the maximum probability of appearing in this unitary.

As the data size and the unitary input size can be very large, we develop two pruning strategies to reduce the search space and accelerate the overall decomposition process. First, considering that the decomposition is an iterative insertion process starting from the beginning of the circuit, we only choose the vectors from the gate in the first layer when generating unitary-vectors pairs.

**Table 1: Two quantum devices involved in the experiment.**

Second, to accelerate the inference of the U2V model, we apply eigen-decomposition [23] to project the original unitary to a low-dimensional matrix, where the Eigen matrix is determined by selecting the most significant eigenvectors of the unitaries in the U2V dataset.

Our vector-based approach is different from the template-based approach that selects candidates from a limited-size template library. The vector-based approach searches the circuit by learning the mapping from unitary to path features via the U2V model, resulting in a high-quality solution. In contrast, the template-based approach shows smaller design space due to its coarse-grained construction of circuits, which leads to more gates and search time.

5.2 QuCT Decomposition Flow

For the current circuit and its unitary $U_{current}$ in each iteration, the objective is to find the rest circuit. To approximate the target unitary U_{target} , the input unitary U_{U2V_input} of the U2V model of this iteration equals the unitary of the rest circuit, which should satisfy the following equation:

$$U_{target} = U_{U2V_input} U_{current}. \quad (6)$$

Note that U_{U2V_input} is on the left side of $U_{current}$. Thus, the input of the U2V model can be calculated as follows:

$$U_{U2V_input} = U_{target} U_{current}^{-1}. \quad (7)$$

The U2V model then outputs gate vectors that may be involved in the circuit. We reconstruct the circuits from these vectors, which serve as the search candidates. Mathematically, the candidate space of QFAST includes unitaries with all combinations of qubits, where the size of the space is $\sum_{i=2}^{N_q-1} C_{N_q}^i$. For example, when $N_q = 8$, the candidate space of QFAST is 246. The candidate space of our method equals the number of the trees (k) in the model. Empirically, $k = 2N_q$ is adequate to find the appropriate candidate, which leads to a 15.4× space reduction when $N_q = 8$.

6 EVALUATION

6.1 Methodology

Quantum hardware. We use two superconducting quantum devices to evaluate our fidelity prediction model: a) a custom device with 5 Xmon qubits in a chained topology; b) a custom device with 18 Xmon qubits arranged in a 6×3 grid qubit topology; Both devices use RX, RY, RZ, and CZ gates as basis gates, with gate times of 30 ns and 60 ns for single-qubit and two-qubit gates, respectively. The single-qubit gate fidelity and two-qubit fidelity of each device are benchmarked by isolated RB [42]. For simultaneous RB[24], the

Config.	Fidelity prediction						
	Upstream model			Downstream model			
	Qubit	N_{step}	Path table	W	Fidelity dataset	Training time	Test dataset
Config-0	18	0	45	45	2,000	1.2 min	2,000
Config-1	18	1	3,420	3,420	2,000	10.5 min	2,000
Config-2	18	2	11.5k	11.5k	2,000	95.6 min	2,000
Config-3	5	2	342	342	2,000	5.4 min	2,000
Config-4	350	1	99.2k	99.2k	2,000	175.0 min	2,000

Config.	Unitary decomposition						
	Upstream model			Downstream model			
	Qubit	N_{step}	Path table	W	Fidelity dataset	Training time	Test dataset
Config-5	4	4	1,033	8	2,000	18.4s	110
Config-6	5	4	2,206	10	2×10^4	354.3 s	110
Config-7	8	4	28.2k	16	4×10^4	81.7 h	110

Table 2: Setup of QuCT models.

Abbreviation	Benchmark
HS	Hamiltonian simulation [9]
ISING	Linear Ising model [37]
QSVM	Quantum support vector machine [68]
QFT	Quantum Fourier transformation [74]
GHZ	Greenberger–Horne–Zeilinger state [27]
BV	Bernstein–Varzirani algorithm [5]
QEC	Quantum error correction code [8]
MUL	Quantum multiplier
QNN	Quantum neural network [4]
QGAN	Quantum generative adversarial network [50]

Table 3: 10 benchmarks used in the experiments

single-qubit and two-qubit fidelities of both devices are above 99% and 98%, respectively.

Quantum simulator. To demonstrate the scalability of QuCT, we design 7 simulators. We perform simulations on the Qiskit Aer QASM simulator (version 0.39.0) using 50-, 100-, 150-, 200-, 250-, 300-, and 350-qubit circuits. To increase efficiency, the simulation is based on the grouping scheme in Section 4.1 that avoids entanglement across the groups. The error of the gate itself is modeled as bit flip, phase flip, and depolarization. The error from the interaction between gates is modeled by applying an RX operator with a random angle ($[-\pi/20, \pi/20]$) to a 1-step path. In other words, the two gates of a 1-step path will be added with the RX operator if this path is injected with a noise. Under these settings, the fidelity of these 7 simulators is 99.88% - 99.97% for single-qubit gates and 99.21% - 99.68% for two-qubit gates, benchmarked by isolated RB.

QuCT model. Table 2 shows the detailed configuration of QuCT models. The parameters of the upstream model include the number of qubits and the number of steps of random walks. The decay (α in Figure 4) is set to 0.4, which will be evaluated in the following sections. The upstream model has 8 configurations; the first five are used for fidelity prediction and optimization, and the other three are used for unitary decomposition. Config-0 to config-3 are evaluated on real-world hardware. Config-4 is the configuration for the 350-qubit simulator. We do not list the configurations for the other six simulators for simplicity. We write a Python program to implement

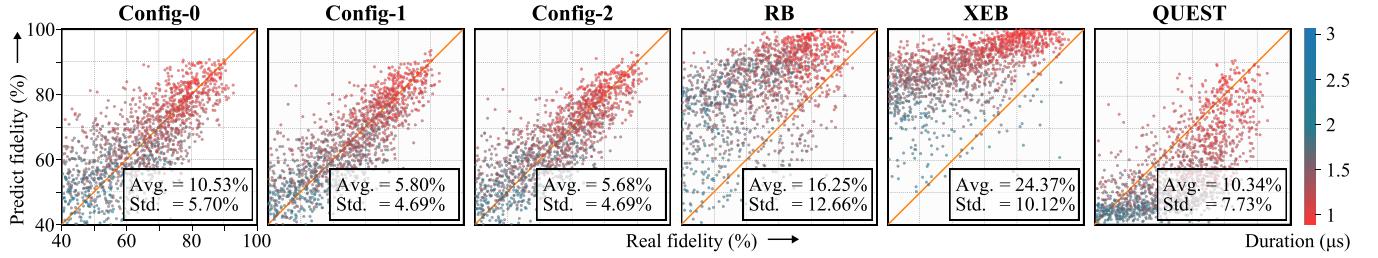


Figure 9: Comparison between the real and predicted circuit fidelities using QuCT (config-0 to config-2), the RB-based model [42] and the XEB-based model [3] and QUEST [82] on the 18-qubit device. Avg. means the average prediction inaccuracy. Std. means the standard deviation.

random walks in our upstream model. In our fidelity prediction model, we adopt Adam optimizer [39] to implement stochastic gradient descent. Before training the prediction model, we set the learning rate to 0.01, the batch size to 100, the split ratio to 0.8, and the maximum number of epochs to 100. The training stops when reaching the maximum number of epochs. We use Scikit-learn [62] to implement random forest in our unitary decomposition model. The unitary is reduced to the matrices with the top-10 Eigenvalues when generating the U2V model.

Dataset. The fidelity dataset is a hardware-dependent dataset, which is built by collecting the real results on the target quantum device. The circuits are generated by the method introduced in Section 4.1, where the maximum size of the groups is 5. We randomly divide the dataset into training and testing sets. For each device, both the training and testing datasets contain results of 2000 circuits. The circuit depth ranges from 5 to 100. Each circuit is sampled 2000 times on the target device. For the config-3 model, we also evaluate it using the benchmarks from Table 3. In the unitary decomposition, we set the number of steps to 4 so that the gate vector can be reconstructed to a larger sub-circuit.

Baselines. We compare our fidelity prediction model with the RB-based model [42], the XEB-based model [3] and QUEST [82]. We set the learning rate of QUEST to 5×10^{-4} , the batch size to 100, the split ratio to 0.8, and the number of epochs to 1000. We compare our fidelity optimization techniques with SATMAP compiler [55] and crosstalk-aware scheduling compiler [19]. For the unitary decomposition, we use three baselines, including QFAST [87], Squander [67], CCD [34], and QSD [72].

6.2 Fidelity Prediction

Figure 9 shows the fidelity predicted by QuCT, the RB-based model [42], and the XEB-based model [3] on the 18-qubit device. The x-axis and y-axis represent the real fidelity and the predicted fidelity, respectively. The color indicates the different duration times of circuits. The prediction inaccuracy is defined as $\Delta\text{fidelity} = |\text{real fidelity} - \text{predicted fidelity}|$.

Evaluation on 18-qubit device using randomized circuits. As shown in Figure 9, the point above the diagonal line indicates the overestimation of the fidelity, and the point below means underestimation. Overall, QuCT with config-2 achieves the lowest prediction inaccuracy (5.68%), leading to 2.8 \times , 4.2 \times , and 1.8 \times reduction compared to the RB-based model (16.25%) [42], the XEB-based model (24.37%) [3] and QUEST [82] (10.34%), respectively. The high

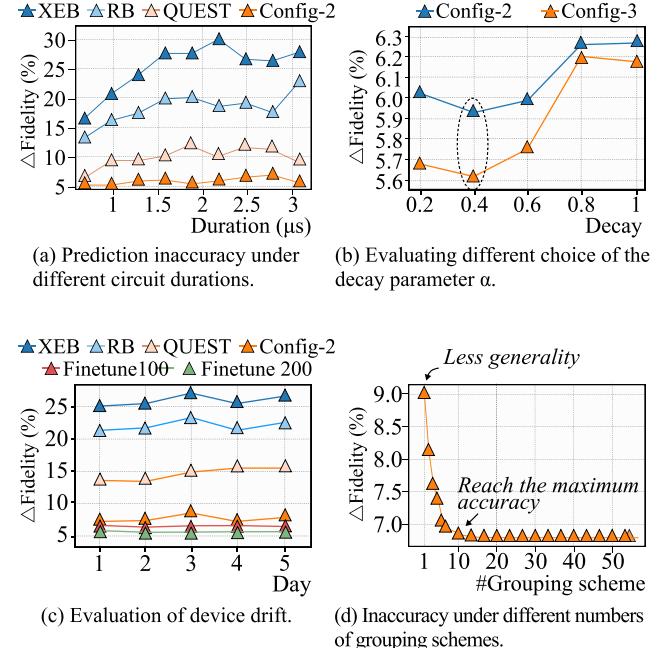


Figure 10: Detailed analysis of fidelity prediction model.

accuracy of QuCT results from the unified extraction of circuit features, which covers various sources of noise. In addition, the dataset of RB and XEB shows less generality, which further reduces accuracy. Though QUEST tries to model the gate interaction using a graph neural network, it is a coarse-grained approach that is less accurate when estimating the fidelity of each individual gate. And it uses a neural network with numerous parameters, taking a lot of time for convergence during training.

The prediction of QuCT is also more stable, which reduces the standard deviation from 10.12% to 4.69%. Improved stability is obtained by our vectorization, which effectively models the interactions between gates, whereas RB and XEB only consider the noise from individual gates and hence tend to overestimate the fidelity. Config-0 has the highest inaccuracy among all downstream models as this configuration sets the number of steps to 0, which also means considering each gate an individual unit, but it is more accurate than RB and XEB as it is operation-aware. When comparing config-1 to config-2, there is little accuracy improvement (0.12%). It implies that the interaction between gates mainly happens within two steps

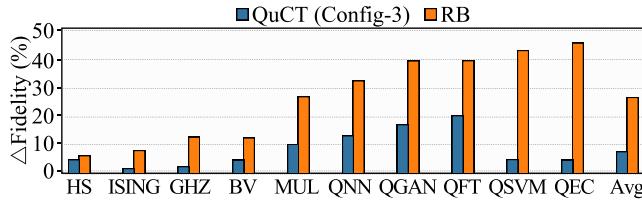


Figure 11: Fidelity prediction on the 5-qubit device.

(among three gates). Thus, we can speculate that 2 steps are sufficient to extract circuit features. This also matches the theory and empirical observations in many works [32, 69, 71], as current hardware implementation applies sparse signal lines between qubits to enable the interactions. The signal transmitted in these lines exponentially decreases in both temporal dimension (the duration of the circuit) and spatial dimension (the length of the signal lines), making the noise local.

In Figure 9, we can observe that the inaccuracy of RB and XEB increases over the duration of the circuit. To further investigate this trend, Figure 10 (a) illustrates the predicted fidelity under different duration times, which matches the aforementioned observation. In a $2\mu s$ duration period, the maximum inaccuracy of XEB is 27.34%. In contrast, it is 5.55% of QuCT with Config-2, which achieves 21.79% inaccuracy reduction.

In Figure 10 (b), we evaluate the different choices of the decay parameter α used during vectorization. For both 5-qubit and 18-qubit devices, the best prediction accuracy is achieved when the decay reaches 0.4. A larger decay makes the model pay more attention to longer paths, while these paths result in little noise.

Figure 10 (c) presents the prediction accuracy under the device drift across 5 days, where the device is calibrated every two days. The ground-truth fidelity is collected every day and compared with different prediction models. We can see that QuCT with config-2 outperforms all other models, which achieves 6.30% - 17.78% inaccuracy reduction. To further improve the accuracy, we propose to fine-tune the downstream model using the updated fidelity of 100 and 200 circuits (requiring less than 5 minutes), which shows 8.27% and 20.12% inaccuracy reduction, respectively.

Figure 10 (d) explores the prediction accuracy with different numbers of grouping schemes. For our 18-qubit device with grid topology, there are 59 grouping schemes in total. When applying 10 grouping schemes, the inaccuracy converges to 6.14%. With only one grouping scheme, the inaccuracy is increased to 9.13%.

Evaluation on 5-qubit device. Implementing these benchmarks on the 18-qubit device requires a large number of gates, which could lead to near-zero fidelity. Thus, we deploy them on the 5-qubit device with config-3. The results are shown in Figure 11. On average, QuCT reduces the prediction inaccuracy from 27.52% to 7.73% over 10 benchmarks. The prediction from the RB-based method is more inaccurate on these benchmarks compared to the results on the 18-qubit device with randomized circuits. One reason for this phenomenon is that these benchmarks exhibit a higher proportion of two-qubit gates, resulting in more than 30% inaccuracy. For example, RB is 39.47% and 43.70% inaccurate on QGAN and QSVM benchmarks as they have 36.85% and 37.32% two-qubit gates, respectively. Another reason is that these benchmarks are computationally expensive, which reduces the prediction accuracy

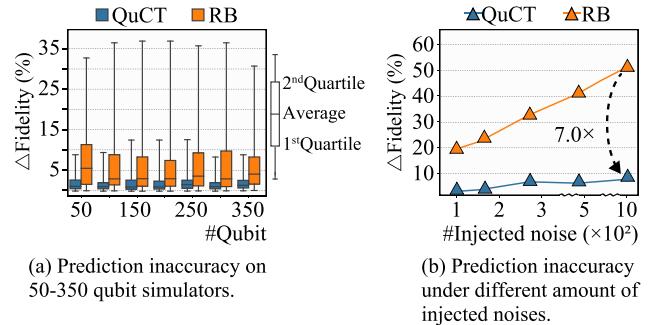


Figure 12: Prediction inaccuracies on the simulator with more qubits. The results of (b) are obtained from the 100-qubit simulator.

as analyzed before. For example, QFT ($7.3\mu s$) and QEC ($6.7\mu s$) are the top-2 circuits with the longest duration time, making the inaccuracy reach nearly 40%. In contrast, QuCT shows less than 10% inaccuracy in 7 out of 10 benchmarks. The relatively large inaccuracy of QuCT occurs in the QGAN and QFT benchmarks because their qubit measurement is associated with a uniform probability distribution, leading to insensitivity to noise when calculating fidelity.

Evaluation on 50-qubit to 350-qubit simulators using randomized circuits. To demonstrate the scalability, we evaluate QuCT on multiple simulators with different numbers of qubits, as shown in Figure 12 (a). Compared to the RB-based method, QuCT shows 4.3× inaccuracy reduction with a much lower standard deviation (13.72% that of RB). The prediction on the 350-qubit simulator (config-4) is more accurate compared to the prediction on the real-world quantum device (config-1 in Figure 9), although both these two configurations set the number of steps to 1. This may result from the fact that real-world device is affected by additional complex noise that may stem from the interactions of the environment and the defect of classical hardware, which is hard to model by QuCT.

We also test the robustness of QuCT with different numbers of injected noises, as shown in Figure 12 (b). The inaccuracy of the RB-based method linearly increases with the number of injected noises. However, QuCT shows only a little drop in prediction accuracy for the circuit with more noise. When the number of noise increases to 1K, the RB-based method fails to predict the fidelity (53.14% inaccuracy). By contrast, the inaccuracy of QuCT is only 7.61%, which effectively reduces the inaccuracy by a factor of 7.0×. The noise simulated by injecting random RX gates to 1-step paths represents the gate interactions. The strength of QuCT, therefore, lies in its ability to model this complex noise.

6.3 Fidelity Optimization

Compilation-level optimization. We integrate our fidelity prediction model with SATMAP compiler [55] to optimize the fidelity during circuit routing, abbreviated as *QuCT_route_opt*. For gate scheduling, we compare to the technique proposed by Ding et al. [19], which applies a crosstalk-aware scheduling scheme. The scheduling optimization of QuCT is abbreviated as *QuCT_sched_opt*.

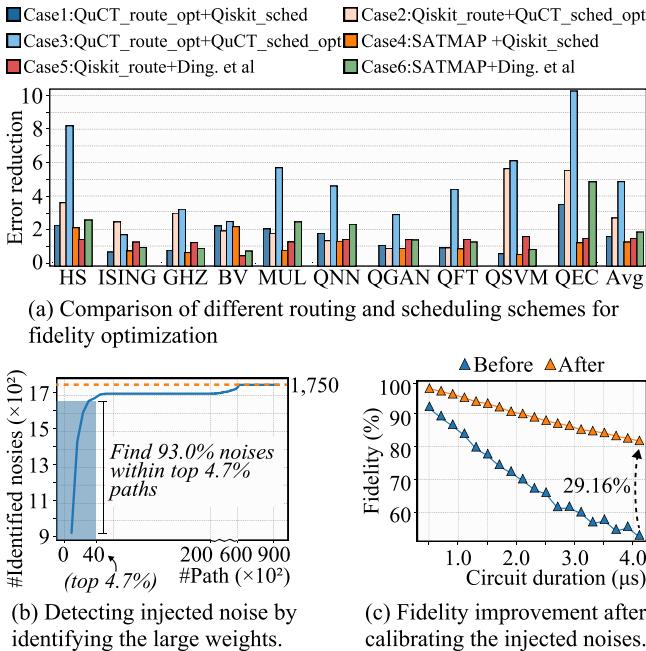


Figure 13: Evaluation of different fidelity optimization techniques. The results of (a) are obtained from the 5-qubit device. The results of (b) and (c) are obtained from the 350-qubit simulator.

The comparison baseline is set as the default routing and scheduling strategy of Qiskit with the optimization level 3. To quantitatively evaluate different optimizations, we define the error reduction as:

$$\frac{\text{Error of (Qiskit_route+Qiskit_sched)}}{\text{Error after optimization}}$$

Figure 13 (a) presents the comparison of different routing and scheduling schemes for fidelity optimization. Compared to the Qiskit baseline, QuCT shows an average error reduction of 5.0× (see case 3). We also compare to the combination of SATMAP [55] and Ding et al. [19] (see case 3 and case 6), where QuCT outperforms in 10 benchmarks and achieves 2.5× improvement of error reduction.

The routing optimization of QuCT improves error reduction from 1.2× to 1.4× compared to SATMAP [55] (case 1 and case 4). The reductions are marginal because the optimization space of the 5-qubit circuits is relatively small, where the default routing scheme of Qiskit [48] also works well. SATMAP aims to minimize the number of gates. Although a small number of gates is usually associated with a high fidelity, SATMAP still lacks a model to quantitatively analyze the fidelity.

As for the scheduling-level optimization, QuCT shows 1.7× improvement compared to [19] (see case 2 and case 5). [19] mainly targets to mitigate the noise from crosstalk. However, instead of modeling a certain type of noise, QuCT optimizes the circuit in a global view by finding the allocation of each gate with maximum fidelity. [19] provides higher error reduction in the benchmarks that have a large number of two-qubit gates, such as the QGAN, QSVM, and QEC, since they involve higher crosstalk noise. While for other benchmarks, it even fails to outperform Qiskit. For example, [19]

suggests a negative effect (0.8× reduction) in the BV benchmark, but QuCT still provides 2.4× reduction (case 5).

Calibration-level optimization. As mentioned in Section 4.2, we can locate the critical path that involves noise by identifying the weight with a large value in Equation 4 and improve the fidelity by calibrating these noisy paths. To evaluate the effectiveness of this technique, in config-4, there is a total number of 99,176 paths with 1,750 of them injected with noise. In Figure 13 (b), paths are sorted according to their corresponding weight element. QuCT can find 93.0% of noise-injected paths (1627 paths) from the top 4.7% paths in the path table. By calibrating these detected paths, QuCT leads to a longer qubit coherence time with 29.16% fidelity improvement, as shown in Figure 13 (c).

6.4 Unitary Decomposition

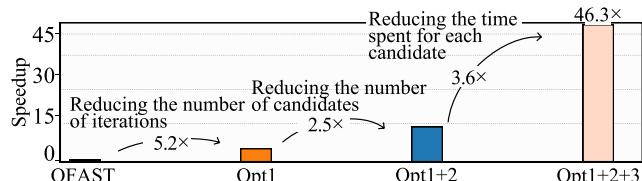
Config-5, config-6, and config-7 are models for 4-qubit, 5-qubit, and 8-qubit unitary decomposition, respectively. Both training and decomposition are performed on the server with two AMD EPYC 64-core CPUs. The test data for each model are 110 unitaries with 100 random-generated unitaries and 10 unitaries of various benchmarks in Table 3. The threshold is set to 0.01, which means the decomposition is completed when the distance between the target unitary and the current unitary of the circuit is within 0.01. To make a fair comparison, all programs can only use a single thread for decomposition. Table 4 summarizes the decomposition results, including the number of gates, the depth of the circuit, and the time of the decomposition process. For the approach that takes more than three weeks, we terminate the decomposition process and give a rough estimation of the possible required time.

Evaluation of time cost. For 4-qubit and 5-qubit random unitaries, QuCT achieves 4.6× and 46.3× speedup, respectively. For benchmark unitaries, it is 2.2× and 5.8×, which drops slightly since the benchmark unitary is less complicated compared to random unitaries. Compared to Squander [67] that aims to minimize the number of CNOT gates, QuCT achieves 8.1× and 55.6× for decomposing 4-qubit and 5-qubit random unitaries. Similar to QFAST, the long decomposition time of Squander mainly comes from the fact that it applies sequential optimization of gate parameters and inserts fewer gates in each iteration of the searching. When the number of qubits goes to 8, QFAST and Squander may require several months or years to find the decomposition solution. QuCT successfully decomposes all 8-qubit random unitaries and benchmark unitaries in 144.4 hours and 26.1 hours, respectively. Our speedup significantly increases when the number of qubits increases because the search space of QFAST increases exponentially with the number of qubits. Our approach can effectively prune the search space by identifying suitable gate vectors as candidates.

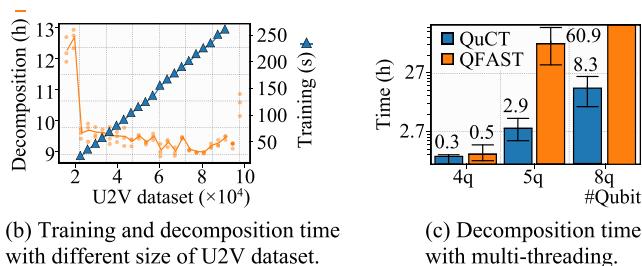
Evaluation of the number of gates. Compared to mathematical methods like CCD [34] and QSD [72]—although they are faster than QuCT—the decomposition solution of QuCT requires fewer gates, resulting in higher reliability of circuits. For example, for 8-qubit benchmark unitaries, QuCT reduces the average number of gates from 8.9×10^4 to 3,392.2 compared to QSD. Compared to QFAST, QuCT also shows some improvement in the quality of the resulting circuit, e.g., 1.3× gate reduction on 5-qubit random unitaries. This improvement is mainly attributed to the reconstruction

		CCD [34]	QSD [72]	Squander [67]	QFAST [87]	QuCT	Impro.
Random	4q	#Gate	1,080.5	859.4	298.2	226.3	206.3
		Depth	170.8	120.1	157.9	74.1	67.4
		Time	1.5 s	0.5 s	14.5 h	8.2 h	1.8h
	5q	#Gate	3,592.9	3,817.2	806.3	887.5	688.1
		Depth	465.5	528.1	428.2	294.1	227.7
	8q	Time	3.6 s	2.1 s	511.2 h	426.2 h	9.2 h
Benchmark	4q	#Gate	7.8×10^4	9.3×10^4	-	2.1×10^4	-
		Depth	3.1×10^4	3.6×10^4	-	1.3×10^4	-
		Time	111.5 s	33.0 s	> 1 y	> 1 y	144.4 h
	5q	#Gate	236.9	270.5	44.3	41.0	40.6
		Depth	169.5	193.5	19.1	17.0	18.2
		Time	0.9 s	0.7 s	1.7 h	1,341.8 s	619.9 s
	8q	#Gate	405.0	472.0	197.2	181.3	165.2
		Depth	302.3	348.7	100.1	93.6	91.1
		Time	0.8 s	0.5 s	94.5 h	3.3 h	2043.0 s
	10q	#Gate	8.1×10^4	8.9×10^4	-	-	3,392.2
		Depth	6.4×10^4	7.6×10^4	-	-	1,948.2
		Time	158.2 s	262.5 s	> 6 mons.	> 6 mons.	26.1 h

Table 4: Comparison of unitary decomposition methods. The improvement of QuCT is calculated by comparing to QFAST [87].



(a) Breakdown of the speedup in the decomposition of 5-qubit unitary.



(b) Training and decomposition time with different size of U2V dataset.

(c) Decomposition time with multi-threading.

Figure 14: Detailed analysis on unitary decomposition.

of the candidate vector. Clearly, according to Figure 8, the paths of the gate vector have already contained information on how to use basis gates to construct the circuit. However, QFAST has to apply a recursive approach to decompose unitary gates into basis gates, which may fall into a local optimal. We also observe that Squander [67] outperforms QFAST in decomposing 5-qubit random unitaries, but it still requires 1.2× gates compared to QuCT. This is because Squander only updates part of gate parameters in each iteration, which may lead to the local optimal.

Decomposition time breakdown. As mentioned in Section 5, the speedup of QuCT is achieved by the following optimization processes.

- Opt1: the total number of search iterations is reduced for two reasons. First, our U2V model helps to identify the gate vectors that share similarities with the target unitary. Second, each gate vector can directly construct the circuit by basis gates.
- Opt2: in each iteration, the number of candidates is reduced thanks to the U2V model.
- Opt3: for each candidate, QuCT requires less time to search gate parameters since the reconstructed circuits are composed of basis gates with fewer parameters. For example, a two-qubit unitary gate used in QFAST includes 16 parameters, while a CZ gate used in QuCT has no parameter to search.

Figure 14 (a) shows the speedup breakdown for the decomposition on 5-qubit random unitaries. The three optimizations contribute to 5.2×, 2.5×, and 3.6× speedup compared to QFAST, respectively. QFAST spends 54.5 hours for the decomposition of the QEC unitary, while QuCT takes 728.3 seconds by benefiting from opt3.

Evaluation of different sizes of U2V dataset. Figure 14 (b) shows the training time and decomposition time with different sizes of the U2V dataset. We observe that increasing the dataset size can reduce the decomposition time as a larger dataset helps to capture more similarity between unitaries and vectors. When the dataset size reaches 2×10^4 , the reduction of the decomposition time is not obvious. This may be because this data size is sufficient to extract the features of 5-qubit circuits. The training time mainly consists of the time to build the data set and the random forest model. Both processes have linear time complexity with the dataset size. Thus, the overall training time is linear to the dataset size. Note that the decomposition time is significantly greater than the training time (less than 250 seconds for the 5-qubit model).

Further acceleration using multi-threading. As different candidates can be searched in parallel, we can leverage the multi-threading technique to further accelerate the decomposition. Based on our test, the sweet points for 5-qubit and 8-qubit unitary decomposition are 10 and 16 candidate vectors, respectively. Figure 14 (c) presents the decomposition time after both QuCT and QFAST [87] applying multi-threading. As a result, for 5-qubit random unitaries, QuCT with multi-threading reduces the decomposition time from 9.2 hours to 2.9 hours and shows 20.9× speedup compared to QFAST. For 8-qubit random unitaries, QuCT reduces the decomposition time from 144.4 hours to 8.3 hours. Although QFAST can also benefit from the multi-threading, it would require more than one month to decompose an 8-qubit unitary.

7 RELATED WORK

Fidelity modeling and optimization. Errors of NISQ devices can be modeled by mathematical equations [25, 30] or profiled by various experiments [3, 10, 42, 64–66]. They consider the error of an operation or a qubit as a single value. Additional experiments are required to characterize certain types of errors caused by gate interactions, such as crosstalk [57] and pulse distortion [69] or benchmark the error of a certain circuit [20]. These experiment results are difficult to integrate into a unified prediction model. Graph neural network [70, 82] has been developed to model circuit

fidelity but exhibits a reduced generalization and interpretation power due to high complexities.

Gate cancellation [54, 58] and noise-adaptive qubit mapping [37, 48] have been developed to optimize noise. Some works aim to reduce certain types of noise, such as the crosstalk [57, 84], dephasing [12], and measurement error [13, 59, 79] or optimize noise in certain applications [1, 31]. For example, errors in variational quantum circuits can be mitigated by deleting unimportant gates [49] or distributed execution [77]. Optimization performance highly depends on the accuracy of fidelity modeling [61].

Unitary decomposition. Unitary decomposition can be conducted via some mathematical decomposition equations [34, 46, 72, 80], where [16] and [17] achieve the optimal number of gates in arbitrary single-qubit and two-qubit unitaries, respectively. However, their number of gates is extremely large when decomposing larger unitaries [6, 46, 81]. Search-based methods are more realistic for four to five-qubit unitaries [15, 67, 87]. With regard to the decomposition time and the number of gates, QSEARCH [15] performs best for three-qubit unitaries. QFAST [87] is improved based on QSEARCH for 4 and 5-qubit unitaries. To limit the search space, [6] uses repeat-until-success circuits for approximation. There are also methods to handle specific unitary types [14, 43, 53, 56], such as Clifford unitaries [26, 76] and sparse unitaries [52]. Quartz [86] and Queso [85] are techniques that focus on decomposing parametric unitaries to enable automatic gate transformation. To enable this decomposition, we can construct the path table with parametric gates, which will be the future work of QuCT.

Features extraction on a graph. Extraction of contextual and topological features is important in the analysis of natural language [18], graph [73], and program [60]. They extract features as frequent sub-structures and represent them as vectors. Random walk is applied in graph analysis [22], which puts paths as input of natural networks. Thus, we think that it is reasonable to apply this technique to the analysis of quantum circuits. Some studies leverage pattern matching in analyzing quantum circuits to find sub-circuits for cancellation [35, 54] while suffering from exponential time complexity.

8 CONCLUSION

We propose a unified framework for analyzing quantum circuits, which first utilizes contextual and topological information to improve the accuracy and efficiency of the analysis. Our upstream model extracts gates of circuits into vectors considering their neighboring gates and their dependencies. Our downstream models take gate vectors as input and analyze circuits for specific tasks. We verify our framework with two representative analysis tasks. Our circuit fidelity prediction model shows 4.2× accuracy improvement and achieves 46.3× speedup compared to prior unitary decomposition methods.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant (No. 61825205). This work was also funded by Zhejiang Pioneer (Jianbing) Project (No. 2023C01036).

REFERENCES

- [1] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. 2020. Circuit compilation methodologies for quantum approximate optimization algorithm. In *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 215–228.
- [2] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puenta González, Enrique De La Torre, Delton Ding, Eugene Dumitrescu, Ivan Duran, Pieter Eendebak, Mark Everitt, Ismael Faro Sertage, Albert Frisch, Andreas Fuhrer, Jay Gambetta, Borja Godoy Gago, Juan Gomez-Mosquera, Donny Greenberg, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Lukasz Herok, Hiroshi Horii, Shaohan Hu, Takashi Imamichi, Toshinari Itoko, Ali Javadi-Abhari, Naoki Kanazawa, Anton Karazeev, Kevin Krsulich, Peng Liu, Yang Luh, Yunhe Maeng, Manoel Marques, Francisco Jose Martín-Fernández, Douglas T. McClure, David McKay, Srujan Meesala, Antonio Mezzacapo, Nikolaj Moll, Diego Moreira Rodriguez, Giacomo Nannicini, Paul Nation, Pauline Ollitrault, Lee James O'Riordan, Hanhee Paik, Jesús Pérez, Anna Phan, Marco Pistoia, Viktor Prutyantov, Max Reuter, Julia Rice, Abdón Rodríguez Davila, Raymond Harry Putra Rudy, Minci Ryu, Ninad Sathaye, Chris Schnabel, Eddie Schoute, Kanav Setia, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylor, Kenso Trabing, Matthew Treinish, Wes Turner, Desreee Vogt-Lee, Christophe Vuillot, Jonathan A. Wildstrom, Jessica Wilson, Erick Winston, Christopher Wood, Stephen Wood, Stefan Wörner, Ismail Yunus Akhalwaya, and Christa Zoufal. 2019. Qiskit: An Open-source Framework for Quantum Computing. (2019).
- [3] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P Harrigan, Michael J Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S Humble, Sergei V Isakov, Evan Jeffrey, Zhan Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michelsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neil, Murphy Yuechen Niu, Eric Ostby, Andre Petukhov, John C Platt, Chris Quintana, Eleanor G Rieffel, Pedram Roushan, Nicholas C Rubin, Daniel Sank, Kevin J Satzinger, Vadim Smelyanskiy, Kevin J Sung, Matthew D Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* (2019), 505–510.
- [4] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf. 2020. Training deep quantum neural networks. *Nature communications* (2020), 808.
- [5] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. 1997. Strengths and weaknesses of quantum computing. *SIAM journal on Computing* (1997), 1510–1523.
- [6] Alex Bocharov, Martin Roetteler, and Krysta M Svore. 2015. Efficient synthesis of universal repeat-until-success quantum circuits. *Physical review letters* 114, 8 (2015), 080502.
- [7] Leo Breiman. 2001. Random forests. *Machine learning* (2001), 5–32.
- [8] John Chiaverini, Dietrich Leibfried, Tobias Schaetz, Murray D Barrett, RB Blakestad, Joseph Britton, Wayne M Itano, John D Jost, Emanuel Knill, Christopher Langer, R. Ozeri, and D. J. Wineland. 2004. Realization of quantum error correction. *Nature* (2004), 602–605.
- [9] Laura Clinton, Johannes Bausch, and Toby Cubitt. 2021. Hamiltonian simulation algorithms for near-term quantum hardware. *Nature communications* (2021), 1–10.
- [10] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. 2019. Validating quantum computers using randomized model circuits. *Physical Review A* (2019), 032328.
- [11] Poulam Das, Eric Kessler, and Yunong Shi. 2023. The Imitation Game: Leveraging CopyCats for Robust Native Gate Selection in NISQ Programs. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 787–801.
- [12] Poulam Das, Swamit Tannu, Siddharth Dangwal, and Moinuddin Qureshi. 2021. Adapt: Mitigating idling errors in qubits via adaptive dynamical decoupling. In *Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 950–962.
- [13] Poulam Das, Swamit Tannu, and Moinuddin Qureshi. 2021. Jigsaw: Boosting fidelity of nisq programs via measurement subsetting. In *Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 937–949.

- [14] Anmer Daskin and Sabre Kais. 2011. Decomposition of unitary matrices for finding quantum circuits: application to molecular Hamiltonians. *The Journal of chemical physics* (2011), 144112.
- [15] Marc G Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi, and Costin Iancu. 2020. Towards optimal topology aware quantum circuit synthesis. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 223–234.
- [16] Christopher M Dawson and Michael A Nielsen. 2005. The solovay-kitaev algorithm. *arXiv preprint quant-ph/0505030* (2005).
- [17] Alexis De Vos and Stijn De Baerdemacker. 2016. Block-Z X Z synthesis of an arbitrary quantum circuit. *Physical Review A* (2016), 052317.
- [18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)* (2018).
- [19] Yongshan Ding, Pranav Gokhale, Sophia Fuhui Lin, Richard Rines, Thomas Propp, and Frederic T Chong. 2020. Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation. In *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 201–214.
- [20] Alexander Erhard, Joel J Wallman, Lukas Postler, Michael Meth, Roman Stricker, Esteban A Martinez, Philipp Schindler, Thomas Monz, Joseph Emerson, and Rainer Blatt. 2019. Characterizing large-scale quantum computers via cycle benchmarking. *Nature communications* (2019), 5347.
- [21] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).
- [22] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering* (2007), 355–369.
- [23] Joel N Franklin. 2012. *Matrix theory*. Courier Corporation.
- [24] Jay M Gambetta, Antonio D Córcoles, Seth T Merkel, Blake R Johnson, John A Smolin, Jerry M Chow, Colm A Ryan, Chad Rigetti, Stefano Poletto, Thomas A Ohki, et al. 2012. Characterization of addressability by simultaneous randomized benchmarking. *Physical review letters* 109, 24 (2012), 240504.
- [25] Konstantinos Georgopoulos, Clive Emery, and Paolo Zuliani. 2021. Modeling and simulating the noisy behavior of near-term quantum computers. *Physical Review A* (2021), 062432.
- [26] Brett Giles and Peter Selinger. 2013. Exact synthesis of multiqubit Clifford+ T circuits. *Physical Review A* (2013), 032332.
- [27] DM Greenberger, MA Horne, and A Zeilinger. 1989. Going beyond Bell's theorem, in "Bell's theorem, quantum theory, and conceptions of the universe," M. Kafatos, editor, Vol. 37 of. *Fundamental Theories of Physics* (1989).
- [28] Robert Grone, Russell Merris, and VS Sunder. 1990. The Laplacian spectrum of a graph. *SIAM Journal on matrix analysis and applications* 11, 2 (1990), 218–238.
- [29] 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* (Philadelphia, Pennsylvania, USA) (*STOC '96*). Association for Computing Machinery, New York, NY, USA, 212–219. <https://doi.org/10.1145/237814.237866>
- [30] Mauricio Gutiérrez, Lukas Svec, Alexander Vargo, and Kenneth R Brown. 2013. Approximation of realistic errors by Clifford channels and Pauli measurements. *Physical Review A* (2013), 030302.
- [31] Adam Holmes, Mohammad Reza Jokar, Ghasem Pasandi, Yongshan Ding, Massoud Pedram, and Frederic T Chong. 2020. NISQ+: Boosting quantum computing power by approximating quantum error correction. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 556–569.
- [32] Sihao Huang, Benjamin Lienhard, Greg Calusine, Antti Vepsäläinen, Jochen Braumüller, David K Kim, Alexander J Melville, Bethany M Niedzielski, Jonilyn L Yoder, Bharath Kannan, et al. 2021. Microwave package design for superconducting quantum processors. *PRX Quantum* 2, 2 (2021), 020306.
- [33] Sergei V Isakov, Dvir Kafri, Orion Martin, Catherine Vollgraff Heidweiller, Wojciech Mruczkiewicz, Matthew P Harrigan, Nicholas C Rubin, Ross Thomson, Michael Broughton, Kevin Kissell, Peters Evan, Gustafson Erik, Andy C. Y. Li, Henry Lamm, Gabriel Perdue, Alan K. Ho, Doug Strain, and Sergio Boixo. 2021. Simulations of quantum circuits with approximate noise using qsim and circq. *arXiv preprint arXiv:2111.02396* (2021).
- [34] Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. 2016. Quantum circuits for isometries. *Physical Review A* (2016), 032318.
- [35] Raban Iten, Romain Moyard, Tony Metger, David Sutter, and Stefan Woerner. 2022. Exact and practical pattern matching for quantum circuit optimization. *ACM Transactions on Quantum Computing* (2022), 1–41.
- [36] Mohsen Jamali and Martin Ester. 2009. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 397–406.
- [37] AliJavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T Chong, and Margaret Martonosi. 2014. ScaffCC: A framework for compilation and analysis of quantum computing programs. In *Proceedings of the 11th ACM Conference on Computing Frontiers*. 1–10.
- [38] Julian Kelly, Peter O’Malley, Matthew Neeley, Hartmut Neven, and John M Martinis. 2018. Physical qubit calibration on a directed acyclic graph. *arXiv preprint arXiv:1803.03226* (2018).
- [39] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [40] Martin Kliesch and Ingo Roth. 2021. Theory of quantum system certification. *PRX quantum* (2021), 010201.
- [41] Paul V Klimon, Julian Kelly, John M Martinis, and Hartmut Neven. 2020. The snake optimizer for learning quantum processor control parameters. *arXiv preprint arXiv:2006.04594* (2020).
- [42] Emanuel Knill, Dietrich Leibfried, Rolf Reichle, Joe Britton, R Brad Blakestad, John D Jost, Chris Langer, Roee Ozeri, Signe Seidelin, and David J Wineland. 2008. Randomized benchmarking of quantum gates. *Physical Review A* (2008), 012307.
- [43] Efekan Kökcü, Thomas Steckmann, Yan Wang, JK Freericks, Eugene F Dumitrescu, and Alexander F Kemper. 2022. Fixed depth Hamiltonian simulation via Cartan decomposition. *Physical Review Letters* (2022), 070501.
- [44] Risi Kondor and Horace Pan. 2016. The multiscale laplacian graph kernel. *Advances in neural information processing systems* (2016).
- [45] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. 2019. A quantum engineer’s guide to superconducting qubits. *Applied Physics Reviews* (2019), 021318.
- [46] Anna M Krol, Aritra Sarkar, Imran Ashraf, Zaid Al-Ars, and Koen Bertels. 2022. Efficient decomposition of unitary matrices in quantum circuit compilers. *Applied Sciences* (2022), 759.
- [47] Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, 529–539.
- [48] 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 (ASPLOS’19)*. 1001–1014.
- [49] Gushu Li, Yunong Shi, and Ali Javadi-Abhari. 2021. Software-hardware co-optimization for computational chemistry on superconducting quantum processors. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 832–845.
- [50] Seth Lloyd and Christian Weedbrook. 2018. Quantum Generative Adversarial Learning. *Physical Review Letters* (2018), 040502.1–040502.5.
- [51] Shunlong Luc and Qiang Zhang. 2004. Informational distance on quantum-state space. *Physical Review A* (2004), 032106.
- [52] Emanuel Malvetti, Raban Iten, and Roger Colbeck. 2021. Quantum circuits for sparse isometries. *Quantum* (2021), 412.
- [53] Esteban A Martinez, Thomas Monz, Daniel Nigg, Philipp Schindler, and Rainer Blatt. 2016. Compiling quantum algorithms for architectures with multi-qubit gates. *New Journal of Physics* (2016), 063029.
- [54] Dmitri Maslov, Gerhard W Dueck, D Michael Miller, and Camille Negrevergne. 2008. Quantum circuit simplification and level compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2008), 436–444.
- [55] Abtin Molavi, Amanda Xu, Martin Diges, Lauren Pick, Swamit Tannu, and Aws Albarghouthi. 2022. Qubit Mapping and Routing via MaxSAT. In *Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1078–1091.
- [56] Mikko Möttönen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. 2004. Quantum circuits for general multiqubit gates. *Physical Review Letters* (2004), 130502.
- [57] 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 (ASPLOS’20)*. 1001–1016.
- [58] Yunseong Nam, Neil J Ross, Yuan Su, Andrew M Childs, and Dmitri Maslov. 2018. Automated optimization of large quantum circuits with continuous parameters. *njp Quantum Information* (2018), 1–12.
- [59] Paul D Nation, Hwajung Kang, Neeraja Sundaresan, and Jay M Gambetta. 2021. Scalable mitigation of measurement errors on quantum computers. *PRX Quantum* 2, 4 (2021), 040326.
- [60] Chris Parnin and Alessandro Orso. 2011. Are automated debugging techniques actually helping programmers?. In *Proceedings of the 2011 international symposium on software testing and analysis*. 199–209.
- [61] 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 (ATC)*. 705–711.
- [62] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* (2011),

- 2825–2830.
- [63] John Preskill. 2018. Quantum computing in the NISQ era and beyond. *Quantum* (2018), 79.
- [64] Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. 2022. Measuring the capabilities of quantum computers. *Nature Physics* (2022), 75–79.
- [65] Timothy Proctor, Stefan Seritan, Kenneth Rudinger, Erik Nielsen, Robin Blume-Kohout, and Kevin Young. 2022. Scalable randomized benchmarking of quantum computers using mirror circuits. *Physical Review Letters* (2022), 150502.
- [66] HT Quan, Zhi Song, Xu F Liu, Paolo Zanardi, and Chang-Pu Sun. 2006. Decay of Loschmidt echo enhanced by quantum criticality. *Physical Review Letters* (2006), 140604.
- [67] Péter Raktya and Zoltán Zimborás. 2022. Approaching the theoretical limit in quantum gate decomposition. *Quantum* 6 (2022), 710.
- [68] P. Rebentrost, M. Mohseni, and S. Lloyd. 2013. Quantum support vector machine for big feature and big data classification. *Physical Review Letters* (2013), 130503.
- [69] Michiel A Rol, Livio Ciocca, Filip K Malinowski, Brian M Tarasinski, Ramiro E Sagastizabal, Cornelis Christiaan Bultink, Yves Salathe, Niels Haandbæk, Jan Sedivy, and Leonardo DiCarlo. 2020. Time-domain characterization and correction of on-chip distortion of control pulses in a quantum processor. *Applied Physics Letters* (2020), 054001.
- [70] Vedika Saravanan and Samah M Saeed. 2022. Data-driven reliability models of quantum circuit: From traditional ml to graph neural network. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).
- [71] Mohan Sarovar, Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. 2020. Detecting crosstalk errors in quantum information processors. *Quantum* 4 (2020), 321.
- [72] Vivek V Shende, Stephen S Bullock, and Igor L Markov. 2005. Synthesis of quantum logic circuits. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. 272–275.
- [73] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*. 488–495.
- [74] Peter W Shor. 1994. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*. IEEE, 124–134.
- [75] 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*. 579–588.
- [76] Mathias Soeken, D Michael Miller, and Rolf Drechsler. 2013. Quantum circuits employing roots of the Pauli matrices. *Physical Review A* (2013), 042322.
- [77] Samuel Stein, Nathan Wiebe, Yufei Ding, Peng Bo, Karol Kowalski, Nathan Baker, James Ang, and Ang Li. 2022. EQC: ensembled quantum computing for variational quantum algorithms. In *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)*. 59–71.
- [78] Brian D Sutton. 2009. Computing the complete CS decomposition. *Numerical Algorithms* (2009), 33–65.
- [79] 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 (MICRO)*. 279–290.
- [80] Robert R Tucci. 1998. A rudimentary quantum compiler. *arXiv preprint quant-ph/9805015* (1998).
- [81] Juha J Vartiainen, Mikko Möttönen, and Martti M Salomaa. 2004. Efficient decomposition of quantum gates. *Physical Review Letters* (2004), 177902.
- [82] Hanrui Wang, Pengyu Liu, Jinglei Cheng, Zhiding Liang, Jiaqi Gu, Zirui Li, Yongshan Ding, Weiwen Jiang, Yiyu Shi, Xuehai Qian, et al. 2022. Graph Transformer for Quantum Circuit Reliability Prediction. (2022), 1–9.
- [83] Tianyi Wang, Yang Chen, Zengbin Zhang, Tianyin Xu, Long Jin, Pan Hui, Beixing Deng, and Xing Li. 2011. Understanding graph sampling algorithms for social network analysis. In *2011 31st international conference on distributed computing systems workshops*. IEEE, 123–128.
- [84] Lei Xie, Jidong Zhai, ZhenXing Zhang, Jonathan Alcock, Shengyu Zhang, and Yi-Cong Zheng. 2022. Suppressing ZZ crosstalk of Quantum computers through pulse and scheduling co-optimization. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'22)*. 499–513.
- [85] Amanda Xu, Abtin Molavi, Lauren Pick, Swamit Tannu, and Aws Albarghouthi. 2023. Synthesizing Quantum-Circuit Optimizers. *Proceedings of the ACM on Programming Languages* 7, PLDI (2023), 835–859.
- [86] Mingkuan Xu, Zikun Li, Oded Padon, Sina Lin, Jessica Pointing, Auguste Hirth, Henry Ma, Jens Palsberg, Alex Aiken, Umut A Acar, et al. 2022. Quartz: superoptimization of quantum circuits. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. 625–640.
- [87] Ed Younis, Koushik Sen, Katherine Yelick, and Costin Iancu. 2021. Qfast: Conflating search and numerical optimization for scalable quantum circuit synthesis. In *2021 IEEE International Conference on Quantum Computing and Engineering*

(QCE). IEEE, 232–243.

Received April 28 2023; accepted July 24 2023