QUEST: Systematically Approximating Quantum Circuits for Higher Output Fidelity

Tirthak Patel
Northeastern University
Boston, USA

Ed Younis
Lawrence Berkeley National Lab
Berkeley, USA

Costin Iancu
Lawrence Berkeley National Lab
Berkeley, USA

Wibe de Jong
Lawrence Berkeley National Lab
Berkeley, USA

Devesh Tiwari
Northeastern University
Boston, USA

ABSTRACT

We present Quest, a procedure to systematically generate approximations for quantum circuits to reduce their CNOT gate count. Our approach employs circuit partitioning for scalability with procedures to 1) reduce circuit length using approximate synthesis, 2) improve fidelity by running circuits that represent key samples in the approximation space, and 3) reason about approximation upper bound. Our evaluation results indicate that our approach of "dissimilar" approximations provides close fidelity to the original circuit. Overall, the results indicate that Quest can reduce CNOT gate count by 30-80% on ideal systems and decrease the impact of noise on existing and near-future quantum systems.

CCS CONCEPTS

• **Computer systems organization** → **Quantum computing**; • **Theory of computation** → Stochastic approximation.

KEYWORDS

Quantum Computing, NISQ Computing, Approximate Compiling

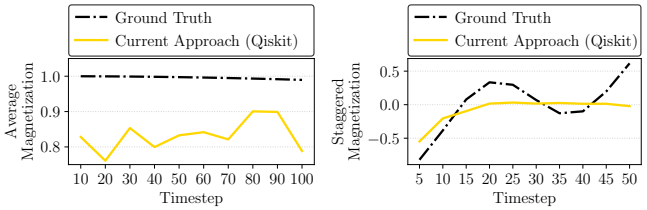
ACM Reference Format:
Tirthak Patel, Ed Younis, Costin Iancu, Wibe de Jong, and Devesh Tiwari. 2022. QUEST: Systematically Approximating Quantum Circuits for Higher Output Fidelity. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '22), February 28 – March 4, 2022, Lausanne, Switzerland.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3503222.3507739

1 INTRODUCTION AND MOTIVATION

We begin by providing a brief relevant background of quantum computing, followed by motivation for Quest, and summarize the approach and contributions of Quest.

ASPLOS '22, February 28 – March 4, 2022, Lausanne, Switzerland

ACM ISBN 978-1-4503-9205-1/22/02…$15.00
https://doi.org/10.1145/3503222.3507739



(a) TFIM **(b) Heisenberg**

Figure 1: The output of the TFIM and Heisenberg quantum algorithms on the IBMQ Manila quantum computer using all Qiskit compiler optimizations (current approach) is far from the expected output (ground truth) at different timesteps of the algorithms.

1.1 Quantum Computing: A Brief Background

The unit of information in quantum computing is the qubit. This is a two-level abstraction, whose state is represented by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, which is a superposition of the $|0\rangle$ and $|1\rangle$ states. When this qubit is measured, its superposition collapses and it can be measured in state $|0\rangle$ with probability $\|\alpha\|^2$ and in state $|1\rangle$ with probability $\|\beta\|^2$, such that $\|\alpha\|^2 + \|\beta\|^2 = 1$. Similarly, an n-qubit entangled quantum system can be represented as a superposition of 2^n states: $|\psi\rangle = \sum_{k=0}^{k=2^n-1} \alpha_k |k\rangle$, such that $\sum_{k=0}^{k=2^n-1} \|\alpha_k\|^2 = 1$.

Under Deutch's computational model, a quantum program can be represented as a circuit of operations [11]. A quantum system can be put into a desired superposition state using quantum gates or operations. For example, a one-qubit NOT operation can be used to rotate the state of qubit ($|0\rangle$ to $|1\rangle$ or vice versa), while a two-qubit controlled-NOT or CNOT operation can be used to entangle two qubits and apply a NOT gate to the "target" qubit if the "control" qubit is in the $|1\rangle$ state. When these operations are applied in succession to one another, they form a "quantum circuit" that represents and executes the corresponding "quantum algorithm." Note that all quantum algorithms can be represented as a sequence of one-qubit rotation gates and two-qubit CNOT gates.

1.2 Motivation for Quest

Existing quantum systems and projected near-term future quantum systems suffer from a pack of different types of errors [22, 34].

These include state-preparation and measurement (SPAM) errors and qubit state decoherence errors. These near-term intermediate-scale quantum (NISQ) devices also suffer from errors related to applying one-qubit and two-qubit gates to a qubit. These errors get compounded over the course of a quantum circuit, as a circuit runs many quantum operations. The two-qubit (CNOT) operation error rate (1-3%) tends to be an order of magnitude above the one-qubit operation error rate.

This makes it difficult to run long quantum circuits with many CNOT gates on near-term quantum computers as the CNOT gates have a high application error, as well as take longer to apply, causing decoherence errors. Previous works have proposed several techniques that focus on circuit compilation and hardware mapping in a manner that reduces the CNOT gate count of the circuit as well as reduces the impact of the errors. These include reducing gate count by collapsing adjacent gates, deleting gate operations using commutativity and unitary laws, reducing the number of CNOT and SWAP operations (implemented using three CNOT gates) by performing layout-aware mapping of the quantum circuit to the hardware, and performing noise-aware mapping to ensure that more noisy qubits and operations are avoided as much as possible [23, 28, 29, 38, 40, 42, 45, 47, 48]. Qiskit [27], IBMQ's python-based quantum compiling package has these state-of-the-art compilation techniques implemented as optimization passes, whereby the optimizations are applied one after another. While these compilation and mapping passes are effective in some cases, they are not able to reduce the gate count to a degree that is able to reduce the noise sufficiently.

As an instance, Fig. 1(a) and (b) show the output of the TFIM (Transverse Field Ising Model) and Heisenberg quantum algorithms, respectively, in the ideal (ground truth) scenario vs. the case in which they are on run on the real IBMQ Manila quantum computer [8]. TFIM outputs the time evolution of the average magnetization (energy) of a four-spin physical system with only the z Hamiltonian interaction component, while Heisenberg outputs the time evolution of a four-spin system with x, y, z Hamiltonian interaction components [4]. Even though the IBMQ Manila computer is a relatively low-error NISQ device and all of the Qiskit compiler optimizations are applied when running the circuit on the computer, the output is far from the expected ideal output. The error is large enough for the output to not provide meaningful insights. For example, with TFIM, the output does not remain consistent across the timesteps (even though it should according to the ground truth output), nor does it have the same magnetization amplitude as the ground truth.

1.3 Approach and Contributions of Quest

Overall, the compilation and mapping passes employed by the state-of-the-art approaches cannot reduce the output error sufficiently. In this work, we present Quest, a technique that focuses on delivering a large reduction in CNOT gate count to reduce the effect of noise.

Quest is built on a few key observations and opportunities. A quantum circuit can be mathematically represented as a unitary matrix. A unitary matrix can have multiple mathematically close "approximations". These approximated unitaries can represent the original circuit's functionality. Quest leverages this property to demonstrate that it is possible to use approximations to our advantage to improve the output fidelity of complex quantum programs on noisy NISQ devices.

While the idea of approximating circuits appears promising, it poses multiple challenges not solved by prior works before it can be realized in practice. First, approximating a complex quantum circuit requires calculating approximate circuits of a large unitary matrix. Unfortunately, the calculation of how mathematically approximate a circuit is to its original circuit is computationally infeasible for quantum programs beyond four qubits due to the need to simulate the circuit on a classical computer. Quest employs circuit partitioning to make the problem tractable: Quest applies approximations on small-size partitions (blocks) and then, combines these approximate blocks to produce an approximate full circuit for the original circuit. While this is the only way to produce approximations for large circuits, it poses two primary challenges.

(1) First, when we combine multiple approximate circuit blocks to produce a full approximate circuit, there is no existing knowledge or mechanism to understand how these approximations interact and affect the overall quality of the full approximate circuit. *Quest overcomes this hurdle by providing a theoretical proof to guarantee that the quality of the approximation of the full circuit can be bounded by controlling the quality of the approximations of blocks. This allows Quest to combine the approximate blocks and guarantee that the overall approximation is of an acceptable quality.*

(2) The second challenge is that the quality of a circuit approximation is expressed as the difference between the unitary matrix of the original circuit and unitary matrix generated after approximation; this measure of approximation does not have strict ordering or direct analytical relationship with the output fidelity. Thus, different "similar" approximate circuits can potentially produce different program outputs (and hence, output fidelities). *Quest overcomes this challenge by devising a novel apriori selection and combination strategy to indirectly control the overall output fidelity by choosing a set of "dissimilar" approximate circuits that have low CNOT gate counts. Quest's method is effective in achieving higher output fidelity by controlling the approximations for individual circuits.*

The main contributions of this work are:

- Quest leverages the approach of circuit synthesis, a technique to generate a mathematically equivalent circuit for a given quantum circuit [10, 15, 16, 25, 37], and approximates it to generate low-CNOT-gate-count circuits that deliver a large reduction in noise.
- Quest defines an apriori approximation selection criterion that enables it to select approximations with fewer CNOT gates in a manner that the effect of the approximations does not manifest in the output of the circuit.
- We provide a theoretical proof to bound the distance of the approximations from the original circuit, which enables Quest to partition the circuit for synthesis and scale up.
- Quest's evaluation demonstrates a 30-80% reduction in the CNOT gate count of circuit while ensuring that the output does not deviate from the output of the original circuit on an ideal quantum system. On a noisy system, Quest reduces the output error by up to 30% points. Quest's open-source

code and data artifacts are available at: https://doi.org/10.5281/zenodo.5747894.

- Using real materials simulation algorithms like TFIM and Heisenberg, Quest demonstrates its ability to track algorithm-specific output even on existing quantum computers due to careful selection of low-error approximations.

2 QUEST-RELEVANT TERMS AND DEFINITIONS

Before we present our technique in Sec. 3, below we introduce some terms and definitions that are used in this paper.

A quantum algorithm (or transformation) is a procedure (or computation) that takes a system from an initial state into a final state, as prescribed by its developer. There are multiple formalism spaces in which we can reason about quantum program behavior: 1) process metrics [20], and 2) domain-specific or standard output metrics [7, 13, 32].

First, at the fundamental level, each program is represented by a unitary matrix U. For a n-qubit system in initial state $|\psi\rangle$, the effect of the program is computed as $U|\psi\rangle$, where U is a $2^n \times 2^n$ unitary matrix. While all quantum circuits have a unitary representation, they may also be represented differently. For example, an algebraic representation is better suited for distributed quantum computing [35]. We focus on the quantum gate unitary model in the paper as it is the most pertinent to Quest's mathematical reasoning.

In general, synthesis algorithms use norms to assess the solution quality, and their goal is to minimize $\|U - U'\|$, where U is the unitary that describes the transformation and U' is the computed solution. This norm is referred to as **process distance**. Intuitively, process distance metrics compare how alike are two mathematical representations of a computation. Prior work has defined the process distance as the largest norm when the difference of the unitary transforms is applied to all states in the Hilbert space [5]. This is an impractical metric as its calculation requires scanning the Hilbert space. While early synthesis works employed the diamond norm for process distance [20], more recent works use the Hilbert-Schmidt (HS) inner product due to its lower computational overhead [10, 18, 21].

The Hilbert-Schmidt inner-product is defined as $\mathrm{Tr}(U^\dagger U')$. This metric ranges from 0 to 2^n, and the closer the value is to 0, the higher the process distance of the two unitary matrices. To make the process distance value range from 0 to 1 and to make it so that the closer the value is to 0, the smaller the distance ("zero" distance), typically the HS distance is transformed as $\langle U, U'\rangle_{HS} = \sqrt{1 - \frac{\|\mathrm{Tr}(U^\dagger U')\|^2}{N^2}}$, where $N = 2^n$. We refer to this metric as the **Hilbert-Schmidt or HS distance** and use it for our approximate synthesis technique.

Second, comparing two quantum programs requires examining and reasoning about their output after the circuit is measured. Given two probability amplitudes of the output of two quantum algorithms), there are multiple distances used to asses their similarity depending on the use case of the algorithm. For example, for the TFIM and Heisenberg algorithms shown in Fig. 1 the average and staggered magnetization rates are calculated from their output probabilities. However, in general, a common distance measure across all algorithms is required to compare the probability amplitudes of the produced distribution with the expected distribution.

A probability distribution distance metric that is primarily used to measure the **output distance** is the **Total Variational Distance** (TVD). For an n-qubit circuit, which has $N = 2^n$ output states, the TVD can be calculated as $\frac{1}{2}\sum_{k=1}^{k=N} |p(k) - p'(k)|$, where $p(k)$ is the probability of state k with the original circuit and $p'(k)$ is the probability of state k with the synthesized circuit. Another metric is the **Jensen-Shannon Divergence** (JSD), calculated as $\sqrt{\frac{1}{2}\left[D(p \parallel m) + D(p' \parallel m)\right]}$, where m is the pointwise mean of p and p' and D is the Kullback–Leibler divergence calculated as $\sum_k q(k) log(\frac{q(k)}{r(k)})$ for two probability distributions q and r. Both metrics have a value between 0 and 1, with 0 being the best (lowest distance). We use these two metrics to evaluate the output distance for all algorithms.

Next, we present Quest, an approximate synthesis technique to reduce the CNOT gate count of quantum algorithms.

3 DESIGN AND IMPLEMENTATION OF QUEST

In this section, we provide an in-depth description of how Quest addresses the problem of reducing the number of CNOT gates in a circuit. We begin by providing an overview of the end-to-end design of Quest.

3.1 Overview of Quest Design

As shown in Fig. 2, Quest first partitions the large circuit into multiple smaller circuits called "blocks" to ensure a scalable solution (Sec. 3.3). Next, it generates approximate circuit solutions with low CNOT gate counts for individual blocks using approximate synthesis. Approximate synthesis is a procedure whereby an approximate circuit can be produced by reducing the process distance between the unitary matrices of the approximate and original circuits (Sec. 3.5). Once the synthesized blocks are generated for each block, they are combined to form the approximate full circuit, which now has a lower CNOT gate count than the original circuit.

However, because the circuit is approximate, the output of the circuit can be different from the original circuit. *To overcome this issue, Quest's makes use of its key insight: multiple different approximate full circuits can be generated because the block instances can be combined in different ways.* Quest uses a dual annealing engine to search the block instance search space in a manner that it selects multiple low-CNOT-gate-count full circuit approximations that are mathematically "dissimilar" to each other. Note that not all circuits generated by the approximate synthesis procedure are selected for averaging; the selection is performed using a step-by-step proceduce as described in Sec. 3.6. This ensures that when the outputs of the approximations are averaged, it produces the same output as the original full circuit, but by using circuits that have fewer CNOT gates.We also provide a theoretical proof that bounds the full circuit's approximation distance, without the need to calculate it directly as it is computationally infeasible, to avoid coarse approximations (Sec. 3.8).

Next, we describe each of these steps in detail, starting with a description of circuit synthesis.

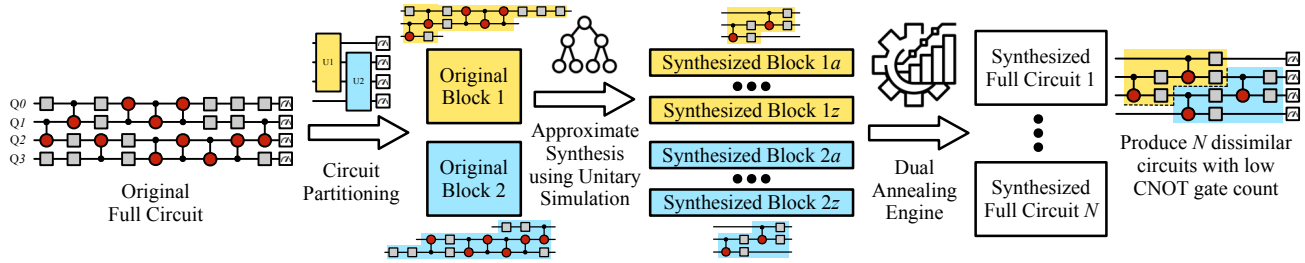


Figure 2: QUEST's methodology of partitioning the circuit into smaller blocks, generating multiple approximate solutions for each block, and selecting "dissimilar" circuits from the block search space while minimizing the CNOT gate count.

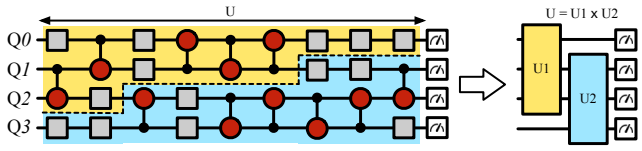


Figure 3: Example of partitioning a circuit in to multiple blocks (represented using unitaries U_1 and U_2). Operations are applied in order from left to right. Isolated squares represent one-qubit rotation gates, circles connected to another qubit represent CNOT gates, and the squares at the end represent measurement gates.

3.2 Overview and Challenges of Circuit Synthesis

Quantum circuit synthesis is the technique used to find a circuit for a quantum algorithm that is mathematically close (referred to in this paper as "exact" synthesis) to the original algorithm circuit. Recall that an n-qubit quantum algorithm can be represented as an $N \times N$ unitary matrix, where $N = 2^n$. This unitary matrix can be calculated by taking a product of all the operations run by a quantum algorithm. For example, if an algorithm runs K operations one after another, and the k^{th} operation is represented using the unitary U_k, the unitary of the entire algorithm can be calculated as $U = U_K U_{K-1} \ldots U_2 U_1$.

Synthesis is the process of finding a circuit with unitary U' (with fewer CNOT gates than the original) for U, such that the process distance between the two is minimized. It is non-trivial to construct an optimal U' (in terms of CNOT gates) in an analytical or rule-based manner for large circuits [16, 43]. For this reason, previous works have proposed numerical optimization solutions [10, 39, 46]. These works attempt to construct the synthesized circuit one layer at a time (a layer typically consists of a combination of one-qubit rotation gates and two-qubit CNOT gates). Once a layer is embedded onto the synthesized circuit, numerical optimization methods are used to optimize the rotation angles such that the process distance between the unitary of the synthesized circuit (U') and the original circuit (U) is minimized. If the distance is within a certain acceptable threshold, the solution is accepted. If an acceptable solution is not found, another layer of gates is added to the synthesized circuit and again the numerical optimization is performed. Note that every time a layer is added, the optimizer has increased degrees of freedom due to the more rotation angles, getting the synthesized circuit's unitary closer to the original circuit's unitary. The Hilbert-Schmidt process distance is widely used to calculate whether the process distance between U' and U is within the acceptable threshold of ϵ [10, 18, 21]: $\sqrt{1 - \frac{\|\mathrm{Tr}(U^\dagger U')\|^2}{N^2}} < \epsilon$.

However, the synthesis technique is not scalable with increase in circuit size as the unitary scales exponentially with the number of qubits. This makes the calculation of the process distance during each iteration of numerical optimization exponentially slower with increase in circuit size. For example, a 20-qubit circuit requires $2^{20\times2}$ inner products to calculate the process distance each time. In fact, it is difficult to calculate the unitary of the entire circuit in the first place because it requires multiplying large quantum operation unitaries. Thus, it is infeasible to perform synthesis on a full circuit unitary for large circuits. The circuit can be divided into manageable chunks before synthesis can be performed. This is the first step of the QUEST procedure.

3.3 STEP 1: Partitioning Large Circuits

A solution to the problem of scaling the circuit synthesis approach is to partition the circuit into blocks of smaller sizes [1]. Fig. 3 shows how this can be achieved for an example circuit of four qubits. Assuming that we can computationally synthesize circuits that are up to three qubits in size, if we partition the four-qubit circuit into two blocks of three-qubits, we can synthesize the two blocks separately to generate circuits equivalent to their corresponding unitaries: U_1 and U_2. The blocks are formed such that there are no connections in terms of two-qubit CNOT gates between the two blocks. Otherwise, they may be entangled and cannot be synthesized separately.

In terms of our example circuit, the unitary U_1 is synthesized such that $1 - \frac{\left\|\mathrm{Tr}(U_1^\dagger U_1')\right\|}{N_1} < \epsilon_1$ and the matrix U_2 is synthesized such that $1 - \frac{\left\|\mathrm{Tr}(U_2^\dagger U_2')\right\|}{N_2} < \epsilon_2$. When the synthesized circuits are obtained for U_1 and U_2, they are put together to form the full synthesized circuit. Note that this approach is scalable because we limit the block size to what is computationally possible. Synthesizing in this manner does not require generating the target unitary for the full circuit, nor does it require calculating the process distance for the entire circuit's unitary. However, this approach also has several challenges, which we discuss in the following section.

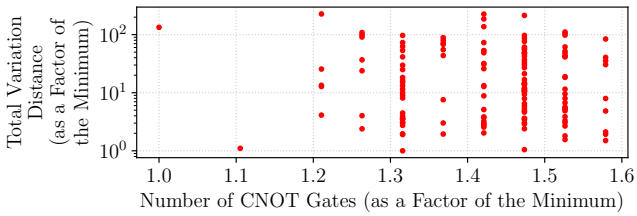


Figure 4: Relationship between the number of CNOT gates and the TVD for several exactly synthesized solutions of a four-qubit Variational Quantum Eigensolver (VQE) circuit.

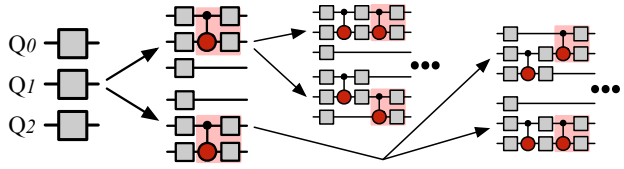


Figure 5: Leap compiler builds the circuit layer-by-layer.

3.4 Challenges with Exact Synthesis of Blocks

An exact solution that minimizes the process distance might not necessarily minimize: 1) the CNOT gate count, and 2) the output distance: how far the output of the synthesized circuit is from the output of the original circuit. As for 1), exact synthesis solutions provide a small reduction in the number of CNOT gates due to the strict process distance threshold requirements (more layers have to be added to the circuit during synthesis to allow for more degrees of freedom).

With regard to 2), the output distance cannot be used as a metric of mathematical exactness during synthesis because the state of a quantum system can only be represented using its unitary. Moreover, when the circuit is partitioned, each block cannot be optimized using an output distance metric because individual blocks have no output of their own. Only the full circuit has an output and it can only be measured after execution. Thus, while the process distance is required for synthesis, the output distance comes into play when the output of the algorithm needs to be interpreted.

However, circuits with similar process distances can have different output distances and CNOT gate counts. Fig. 4 demonstrates the relationship between the number of CNOT gates and the output distance (TVD) for several exactly synthesized solutions of a four-qubit Variational Quantum Eigensolver (VQE) circuit. All solutions have a similar process distance of less than 10^{-5} (exact solution threshold) and yet have TVDs in a large range. The solution with the minimum number of CNOT gates has one of the highest TVDs, while the solution with only 10% more CNOT gates than the minimum (1.1 factor) has a lower TVD. This small-scale example shows that it is not always advisable to select the exact solution with the fewest number of CNOT gates. Thus, QUEST breaks away from the notion of having an exact synthesized solution by designing an approximate synthesis approach.

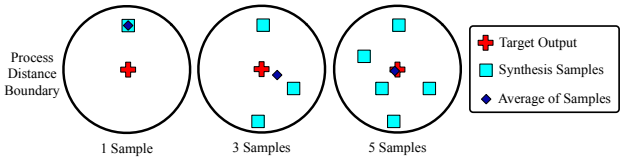


Figure 6: Visual representation of how averaging over multiple approximations of a full circuit can produce a similar output as the original circuit (with the average having a low output distance).

3.5 STEP 2: Generate Approximate Circuits for Blocks

QUEST employs an approximate synthesis procedure that generates multiple low-CNOT-gate-count circuit approximations in a manner that minimizes the output distance between the synthesized circuits and the original circuit of a block.

The key to achieving this reduction is realizing that different types of circuits can be synthesized such that they have process distance below a certain threshold. Recall that synthesis is performed using numerical optimization building layer-by-layer. The qubits that these layers are placed on and the rotation angles that are assigned to them during the process distance minimization procedure affects the final synthesized circuit that is produced. Multiple different approximate circuits can be produced by varying these factors.

To achieve this, QUEST modifies the Leap compiler [39] to return the best M circuits with the lowest process distance at each layer of the compiler tree. The compiler constructs a circuit tree one layer at a time as shown in Fig. 5. Each layer consists of a CNOT gate between two qubits followed by two rotation gates on both the qubits. It attempts this layer on all allowed two-qubit combinations and optimizes the rotation angles to minimize the unitary process distance. As more layers are added, the tree expands due to the increased number of layer permutations. Every few layers, it picks the branch with the least process distance and starts reconstructing the tree from there to reduce the number of optimization evaluations. QUEST uses the compiler to generate multiple approximations at each layer of the tree (a tree layer roughly corresponds to one CNOT gate). This enables the generation of approximate circuits of different qualities in terms of process distances for each block at different CNOT gate counts. All approximate solutions are generated until the tree exceeds the CNOT gate count of the original circuit. As more layers are constructed and the tree becomes deeper, the process distance decreases. However, because we would like to have multiple solutions for a block, including the ones with higher CNOT gate counts than the minimum CNOT gate count solution, we collect all solutions of different CNOT gate counts.

3.6 STEP 3: Putting Together Approximate Blocks

How can these low-CNOT-gate-count approximations be used to reduce the output distance? Many different approximations are generated to approximate a full circuit using QUEST's methodology.

For example, say a circuit has 8 blocks and each block has 10 approximations. Then, 10^8 approximations are possible for the full circuit even in this small case. The quantum-specific challenge is that it is not possible to simply take an average of all of these approximations in the exponentially large space. Random sampling in this space generates poor quality outcomes (> 0.1 or 10% total variation distance to the original circuit) because different approximations have different fidelities and CNOT-gate counts.

As a solution to this problem, Fig. 6 demonstrates a visual example of how selecting multiple synthesized circuits can help ensure that the output distance is reduced. The figure is read from left to right, with the first circle showing when one approximate circuit sample is used and the last circle showing when six samples are used. The red cross shows the output of the original circuit in Hilbert space, the boundary circle around it demarcates the boundary of the process distance threshold used for approximate synthesis (this threshold is larger than used with exact synthesis so as to generate low-CNOT-gate-count solutions). The blue squares show the output samples of the approximate synthesized circuit and the dark blue diamond shows the average of the samples. We make two points:

1) If only the circuit with the fewest number of CNOT gates is selected, it can result in a high output distance as shown in the first circle. While the circuit with the lowest CNOT gate count might also coincidentally have a low output distance for some algorithms, this cannot be established analytically as it assumes knowing the ground truth output. *Instead, averaging over M low-CNOT-gate-count circuits can help us regulate and control the output distance with more robustness than simply choosing one circuit.* The trade-off here is between the CNOT gate count and the output distance. If multiple circuits are not selected, we do not have a way to control and reduce the output error. On the other hand, if the CNOT gate count is high, it defeats the purpose of synthesis.

2) It is also not sound to just have many random approximations. If the approximations are mathematically similar (e.g., if the six samples in the third circle were in the same region of the circle), their output cannot average out to reduce the output distance. Thus, Quest must ensure that the approximations are "dissimilar," while also having a low CNOT count. Many dissimilar full circuit approximations are available due to (1) different possible permutations when blocks are put together and (2) dissimilar block approximations from multiple branches of the synthesis tree.

Quest achieves this balance by using a dual-annealing-based minimization algorithm [36] shown in Algorithm 1 to minimize an objective function that places equal weight on CNOT-gate count and the dissimilarity of the approximations: $\min f = \frac{1}{2} \times$CNOT Count$+ \frac{1}{2} \times$ Approx. Dissimilarity

The CNOT count in this objective function is simply the normalized CNOT gate count of the approximation compared to the original circuit. The approximation dissimilarity is calculated as the fraction of already selected circuit samples with similarity to the new sample. Consider two approximate circuit samples, S_1 and S_2. If the process distance between the two samples ($\langle S_1, S_2 \rangle_{HS}$) is less than the maximum of their process distances to the original circuit ($\max\{\langle S_1, O \rangle_{HS}, \langle O, S_2 \rangle_{HS}\}$), then the two samples are considered similar: $\langle S_1, S_2 \rangle_{HS} \leq \max\{\langle S_1, O \rangle_{HS}, \langle O, S_2 \rangle_{HS}\}$. Intuitively, in the visualization shown in Fig. 6, this means that both samples are in the same region of the circle. If the process distance between the two is greater than the maximum of their process distances to the original circuit, then the two circuits are on the opposite side of the circuit; thus, their output can be averaged out. The first sample has an approximation dissimilarity of zero (since there are no already selected circuits), and the objective function will select the approximate circuit with the lowest CNOT-gate count. As more samples are selected, the approximation dissimilarity gains more significance as it becomes increasingly difficult to find dissimilar approximations. However, the equal weight on CNOT count ensures that approximations that have too many CNOTs are not selected.

Algorithm 1 Dual annealing engine's objective function.

1: $O \leftarrow$ Original circuit
2: $A \leftarrow$ Approximate circuit to score
3: $S \leftarrow$ Already selected approximations
4: $c_{norm} \leftarrow$ (CNOT count of A) / (CNOT count of O)
5: $\epsilon \leftarrow$ Process distance threshold
6: **if** $\langle A, O \rangle_{HS} > \epsilon$ **then** ▷ If the threshold is breached
7: **return** 1.0
8: **else if** $S == \{\varnothing\}$ **then** ▷ If this is the first sample
9: **return** c_{norm}
10: **else**
11: $m = 0$ ▷ Fraction of similar samples
12: **for** $s \in S$ **do**
13: $m = m + \langle A, s \rangle_{HS} \leq \max\{\langle A, O \rangle_{HS}, \langle O, s \rangle_{HS}\}$
14: **end for**
15: $m = m/\text{size}(S)$
16: **return** $\frac{1}{2} \times m + \frac{1}{2} \times c_{norm}$
17: **end if**

While this objective function works well for non-partitioned circuits, it requires a tweak to accommodate large partitioned circuits. Calculating if two full circuit approximations (constructed by putting together block approximations) are similar should not require the computationally infeasible use of the full circuit unitaries. Instead, Quest uses the metric "fraction of all circuit blocks that are similar" in the objective function as it is a scalable alternative that works well in practice. As an instance, for two approximations of a full circuit consisting of ten blocks, if three of the blocks are mathematically similar, the two approximations receive a similarity score of 0.3. Next, we discuss a major challenge when approximating large partitioned circuits.

3.7 A Challenge of Approximating Full Circuits

While for individual blocks it can be ensured that the approximations are not too coarse by eliminating ones with a high output distance (Lines 6-7 in Algorithm 1), a major challenge of partitioned synthesis is to ensure that when the approximate blocks are put back together, the process distance of the full circuit approximation is not violated. Blocks cannot be combined without the knowledge of how their process distances accumulate. For example, the process distance may compound multiplicatively when the blocks are combined to form the full circuit. Without having a theoretical

bound on the process distance of the full circuit, the process distance thresholds of its blocks may have to be kept unnecessarily small to be on the safe side. This means that the synthesized blocks likely end up being longer than they need to be as more layers are typically required during the numerical optimization process if the distance threshold is very small. Overcoming this problem can help us synthesize shorter circuits with fewer CNOT operations. To this end, next we provide a theoretical proof to bound the full circuit process distance based on the process distances of its blocks without the need to directly calculate the process distance of the full circuit. This will help us ensure that the full circuit approximations that are too coarse can be eliminated during the dual annealing minimization procedure.

3.8 Theoretical Upper Bound on Process Distance

We prove the theoretical upper bound for a circuit partitioned into two blocks without any loss of generality (e.g., the one shown in Fig. 3), and it can then be extended to a circuit partitioned into K blocks.

We want the process distance of the unitary U of the full circuit to be bounded without performing synthesis on the full circuit due to the lack of scalability: $\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U^\dagger U')\right\|^2}{N^2}} < \epsilon$. U is an $N \times N$ matrix, where N is 2^n, where n is the number of qubits in the full circuit. We choose the $\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U^\dagger U')\right\|^2}{N^2}}$ metric for process distance as it is a reasonable metric to measure unitary equivalency, it is computationally efficient, and it gives us the ability to prove a bound on it without calculating it directly. The ϵ bound needs to be derived based on the process distances of its circuit blocks. Recall that the partitioned blocks are small enough to be efficiently synthesizable and therefore, have known process distances. The example circuit has two blocks and these blocks have the below two process distance bounds by construction (synthesis is performed in a manner that ensures that these bounds are met).

$$\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_1^\dagger U_1')\right\|^2}{N_1^2}} \le \epsilon_1 \quad , \quad \sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_2^\dagger U_2')\right\|^2}{N_2^2}} \le \epsilon_2 \tag{1}$$

Here, N_1 and N_2 are the dimensions of U_1 and U_2, respectively. For the example circuit, we have $U = (I \otimes U_2)(U_1 \otimes I) = U_{I2}U_{1I}$. The notation U_{1I} refers to the unitary $(U_1 \otimes I)$, representing the Kronecker product of the U_1 operation with the identity operation on the remaining qubits (no operation can be represented as the identity operation). Similarly, U_{I2} refers to the unitary $(I \otimes U_2)$. The post-synthesis approximations of these unitaries can be represented as $U' = (I \otimes U_2')(U_1' \otimes I) = U_{I2}'U_{1I}'$. Also, $N = N_I \times N_1 = N_2 \times N_I$.

As a first step, we prove the process distance bound when a unitary representing a partitioned block is extended to the remaining qubits in the full circuit, i.e., we determine the process distance of $U_1 \otimes I$ matrix given the process distance of the U_1 matrix. We begin by rearranging the terms in Eq. 1 to isolate for $\left\|\mathrm{Tr}\,(U_1^\dagger U_1')\right\|$, as is shown below.

$$\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_1^\dagger U_1')\right\|^2}{N_1^2}} \le \epsilon_1 \Rightarrow \frac{\left\|\mathrm{Tr}\,(U_1^\dagger U_1')\right\|^2}{N_1^2} \ge 1-\epsilon_1^2$$
$$\Rightarrow \left\|\mathrm{Tr}\,(U_1^\dagger U_1')\right\|^2 \ge N_1^2(1-\epsilon_1^2) \Rightarrow \left\|\mathrm{Tr}\,(U_1^\dagger U_1')\right\| \ge N_1\sqrt{1-\epsilon_1^2} \tag{2}$$

Next, we show how $\mathrm{Tr}\,(U_{1I}^\dagger U_{1I}')$ is related to $\mathrm{Tr}\,(U_1^\dagger U_1')$:

$$\begin{aligned}\mathrm{Tr}\,(U_{1I}^\dagger U_{1I}') &= \mathrm{Tr}\,[(U_1 \otimes I)^\dagger (U_1' \otimes I)] = \mathrm{Tr}\,[(U_1^\dagger \otimes I^\dagger)(U_1' \otimes I)] \\ &= \mathrm{Tr}\,[(U_1^\dagger \otimes I)(U_1' \otimes I)] = \mathrm{Tr}\,(U_1^\dagger U_1' \otimes II) = \mathrm{Tr}\,(U_1^\dagger U_1' \otimes I) \\ &= \mathrm{Tr}\,(U_1^\dagger U_1')\mathrm{tr}(\mathrm{I}) = \mathrm{Tr}\,(U_1^\dagger U_1')N_I\end{aligned} \tag{3}$$

Substituting Eq. 2 into Eq. 3, we get that $\left\|\mathrm{Tr}\,(U_{1I}^\dagger U_{1I}')\right\| \ge N\sqrt{1-\epsilon_1^2}$, as shown below.

$$\begin{aligned}\left\|\mathrm{Tr}\,(U_{1I}^\dagger U_{1I}')\right\| &= \left\|\mathrm{Tr}\,((U_1^\dagger U_1')N_I\right\| = \left\|\mathrm{Tr}\,((U_1^\dagger U_1')\right\|N_I \\ &\ge N_1\sqrt{1-\epsilon_1^2}N_I = N_1N_I\sqrt{1-\epsilon_1^2} = N\sqrt{1-\epsilon_1^2}\end{aligned} \tag{4}$$

If we rearrange the terms in Eq. 4 (similar to the rearranging in Eq. 2, but in reverse order), we get $\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_{1I}^\dagger U_{1I}')\right\|^2}{N^2}} \le \epsilon_1$. Thus, the process distance of a block unitary that does not span the full size (number of qubits) of a circuit remains bounded by the same threshold when the unitary is extended to the size of the circuit. Therefore, using a similar procedure, we can also show that for the second block, U_2, $\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_{I2}^\dagger U_{I2}')\right\|^2}{N^2}} \le \epsilon_2$.

We now have the tools to bound the process distance of the full circuit. Recall that the process distance for the full circuit is $\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U^\dagger U')\right\|^2}{N^2}}$. Substituting $U = U_{I2}U_{1I}$, we obtain the following:

$$\begin{aligned}\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U^\dagger U')\right\|^2}{N^2}} &= \sqrt{1-\frac{\left\|\mathrm{Tr}\,[(U_{I2}U_{1I})^\dagger (U_{I2}'U_{1I}')]\right\|^2}{N^2}} \\ = \sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_{1I}^\dagger U_{I2}^\dagger U_{I2}'U_{1I}')\right\|^2}{N^2}} &= \sqrt{1-\frac{\left\|\mathrm{Tr}\,[(U_{1I}'U_{1I}^\dagger)(U_{I2}^\dagger U_{I2}')]\right\|^2}{N^2}}\end{aligned} \tag{5}$$

Using the inequality proven by Wang and Zhang [44] and using the above derived bounds for process distances of U_{1I} and U_{I2}, the following relationship is obtained:

$$\begin{aligned}&\sqrt{1-\frac{\left\|\mathrm{Tr}\,[(U_{1I}'U_{1I}^\dagger)(U_{I2}^\dagger U_{I2}')]\right\|^2}{N^2}} \\ &\le \sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_{1I}'U_{1I}^\dagger)\right\|^2}{N^2}} + \sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_{2I}^\dagger U_{2I}')\right\|^2}{N^2}} \\ &\le \sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_{1I}^\dagger U_{1I}')\right\|^2}{N^2}} + \sqrt{1-\frac{\left\|\mathrm{Tr}\,(U_{2I}^\dagger U_{2I}')\right\|^2}{N^2}} \\ &\le \epsilon_1 + \epsilon_2\end{aligned} \tag{6}$$

Combining Eq. 5 and Eq. 6, we get $\sqrt{1-\frac{\left\|\mathrm{Tr}\,(U^\dagger U')\right\|^2}{N^2}} \le \epsilon_1 + \epsilon_2$. This proof can be extended to circuits partitioned into K blocks by providing the proof for two blocks at a time, combining the

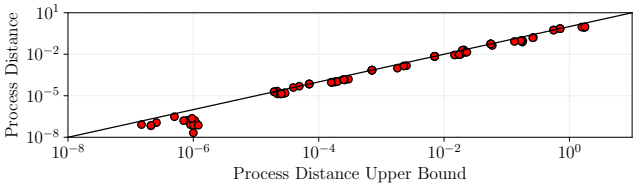


Figure 7: The theoretical process distance upper bound successfully bounds the process distance in practice.

two into a single unitary and providing the proof again for the two unitaries (combined unitary and the third unitary), and so on and so forth, iteratively. *Therefore, for a circuit partitioned into K blocks, we have that* $\sqrt{1 - \frac{\|\mathrm{Tr}(U^\dagger U')\|^2}{N^2}} \leq \sum_{k=1}^{k=K} \epsilon_k$. *The process distance of the full circuit is theoretically upper bounded by the sum of the process distances of all of its partitioned blocks.*

While the relationship between the derived process distance upper bound and the actual process distance cannot be directly/theoretically proven as it algorithm specific and varies depending on how the algorithm circuit is constructed and partitioned, we demonstrate this relationship using real algorithm examples. Fig. 7 shows the relationship between the process distance upper bound and the actual process distance for different algorithms. The results indicate that the derived upper bound is respected across all samples and a relatively tight bound is obtained for different algorithms and process distance values. Therefore, the upper bound enables us to confidently bound the process distance of the full circuit (without the need to calculate it directly) simply by ensuring that its partitioned blocks are combined by the dual annealing engine in a manner that the sum of their process distances is within an acceptable threshold.

Putting it together. QUEST enables reduction in the CNOT gate count by partitioning the circuit into smaller blocks and generating multiple approximate circuits for the blocks. It then puts back together the block approximations in a manner that reduces the CNOT gate count as well as generates dissimilar approximations to reduce the output distance using its dual annealing engine. *First, the full circuit approximation with the lowest CNOT count is selected using the dual annealer. Then, the second circuit is selected with half weight on CNOT count and half weight on the distance to the first circuit. The third circuit is selected with half weight on CNOT count and half weight on the average distance to the first two circuits. In this manner, up to* $M = 16$ *full circuit samples are selected for the evaluated algorithms (selection procedure is terminated when the engine returns an already selected circuit).* It is aided by the theoretical upper bound to eliminate coarse approximations in a scalable manner.

Overall, QUEST's time complexity can be analyzed in three parts: (1) **Partitioning.** If we consider the number of gates in the circuit g (upper bounded by $n \times d$, but often much lower, where n is the number of qubits and d is the circuit depth), the complexity of partitioning is $O(g)$ as it scans all circuit gates once to form blocks. (2) **Synthesis.** If b blocks are formed (upper bounded by g, but often much lower), then all b blocks can be synthesized in parallel; complexity is expressed as $O(b)$ if synthesized serially. Synthesis complexity does not depend on the number of approximations generated during synthesis as that number is the outcome of synthesis and not an input condition. (3) **Annealing.** If the annealer is run for i iterations (i approximations are sampled), its time complexity is $O(ibs)$ to sample the s^{th} approximation (s^{th} sample requires distance calculation to previous $s-1$ samples). On the other hand, the space footprint is dominated by block approximations: the space complexity is $O(ab)$, where b is the number of blocks and a is the average number of approximations generated per block.

In the next section, we evaluate the effectiveness of QUEST for different algorithms after providing details about the experimental setup and methodology.

4 EVALUATION

4.1 Experimental Setup and Methodology

Comparative Techniques. We study the CNOT gate count, ground truth output, as well as the error observed in a noisy environment of several techniques. We compare against the original baseline circuits of different algorithms exactly synthesized with the Leap compiler [39] (referred to as the **Baseline**). We also compare to the circuit generated when all the Qiskit compiler optimizations are applied to this Baseline circuit. These compiler optimizations are simply referred to as **Qiskit**. When these compiler passes are applied to the approximate circuits produced by QUEST, the results generated by the produced circuits are referred to as **QUEST + Qiskit**.

Experimental Setup. The partitioning, approximate synthesis, and dual annealing components of QUEST are run on our local compute cluster consisting of 2.4 GHz Intel E5-2680 v4 CPUs. We note that performing optimal partitioning (such that we get the lowest number of blocks) is NP-hard, and hence, heuristic-based methods must be used for efficiency. For example, blocks can be formed greedily starting with the largest block by iteratively scanning the entire circuit multiple times until all blocks are formed. However, this method does not scale efficiently for larger circuits. Therefore, we opted to use a more scalable partitioner that only scans the circuit once from front to back and forms blocks on its way and yields effective results. The circuits are thus partitioned using the scan partitioner available as part of the open-source BQSKit package [1]. A maximum block size of four qubits is used as it synthesizes efficiently and yields good results. Note that some blocks may be of a smaller size if need be. When an algorithm has multiple blocks, the approximate synthesis step is run on different blocks in parallel on up to ten compute nodes. Leap compiler [39], which is also a part of the BQSKit package, is modified and used for synthesis as described in Sec. 3. The Python-based SciPy package is used to run the dual annealing engine [17].

As circuits get larger, they typically generate more blocks, and maintaining a constant approximation distance threshold for the full circuit would require selecting block approximations with lower distances. However, this would result in deeper circuits with more than required CNOT gates. Instead, we increase the distance threshold for the full circuit proportional to the number of blocks. This

Table 1: Algorithms and benchmarks used to evaluate the performance of QUEST.

Algorithm	Description
Adder	Quantum adder circuit [9]
Heisenberg	Time-independent Heisenberg Hamiltonian [4]
HLF	Hidden linear function [6]
QFT	Quantum Fourier transform [30]
QAOA	Quantum alternating operator ansatz [12]
Multiplier	Quantum multiplier circuit [14]
TFIM	Transverse field ising model [4]
VQE	Variational quantum eigensolver [26]
XY	XY quantum Heisenberg model [4]

allows for the approach to be scalable and select blocks with low-CNOT gate count, while the dissimilar approximation selection technique remains robust even as the number of blocks increases. Up to 16 approximations are generated for each algorithm. A balanced weight of 0.5 is used for CNOT gate count and approximation dissimilarity in the objective function that the dual annealing engine minimizes.

The ground truth results are obtained by running the Baseline circuit in an ideal quantum simulation environment using the Qiskit unitary simulator [27], which is part of the Aer package. We perform noisy simulations on circuits up to 16 qubits (it was not possible to run noisy simulations on larger circuits) using the IBMQ QASM simulator available via the IBM quantum experience cloud. A Pauli noise model is used for all the qubits with noise levels of 1%, 0.5%, and 0.1% to simulate how QUEST will perform for future NISQ computers as the noise level decreases. We run circuits up to five qubits on the IBMQ Manila quantum computer in order to demonstrate how the technique performs on existing quantum computers. We use 8192 experimental trials (maximum allowed) per experiment. Each circuit run on a quantum computer takes 10-12 seconds for all trials.

Algorithms and Benchmarks. We use the algorithms and benchmarks listed in Table 1 to evaluate QUEST. Adder and Multiplier are standard quantum arithmetic circuits, while QAOA and VQE are quantum variational algorithms. Heisenberg, TFIM, and XY are time-evolving Hamiltonian algorithms for material simulations. The Heisenberg model has non-zero strength for the coupling interaction between nearest neighbor spins for all three axes (x, y, z), TFIM does for z, and XY does for x and y. We evaluate circuits of size 4-32 qubits.

Evaluation Metrics. In terms of output distance, we use the Total Variation Distance (TVD) and the Jensen-Shannon Divergence (JSD) as defined in Sec. 2. These are general metrics that are typically used to define the output distance across all algorithms. To calculate the output distance from the Baseline for QUEST, the output probability distributions of all of its approximate circuits are averaged to generate one probability distribution. In addition to these metrics, we also study algorithm-specific output distances as necessary. For example, for TFIM and Heisenberg algorithms, we study the differences in the magnetization at different time steps.

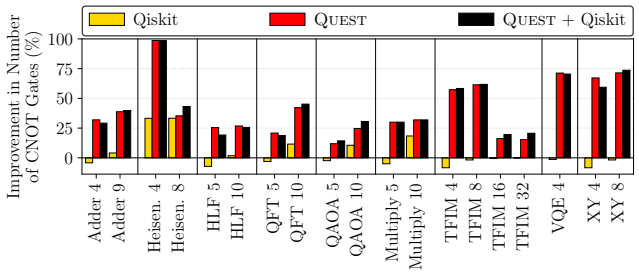


Figure 8: QUEST reduces the CNOT gate count across all algorithms compared to the Baseline circuits as well as the Qiskit optimizations applied on the Baseline circuits. The number next to the algorithm name indicates the number of qubits.

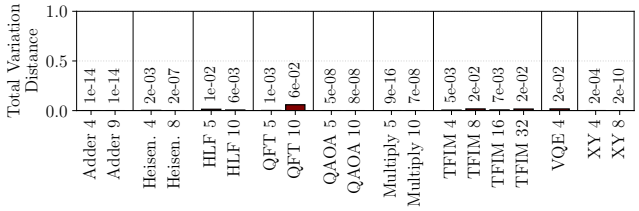
4.2 Results and Analysis

QUEST considerably reduces the CNOT gate count over the Baseline circuit compared to the Qiskit compiler optimizations. Fig. 8 shows the percent improvement, i.e., reduction, in CNOT gate count over the Baseline circuit with Qiskit, QUEST and QUEST + Qiskit for different algorithms. QUEST delivers a reduction in CNOT gate count of 30-80% for most algorithms, even greater than 80% for some algorithms such as Heisenberg, which has many CNOT gates. Circuit characteristics such as a qubit having many CNOTs with a rotating set of other qubits make partitioning more challenging as it may limit the opportunities to create larger blocks. This is seen in algorithms like Multiply and QAOA. Nonetheless, even for algorithms with these challenging characteristics, QUEST demonstrates over 10% CNOT reduction (e.g., QAOA 5) because its approximation methodology performs well even if smaller blocks are formed.

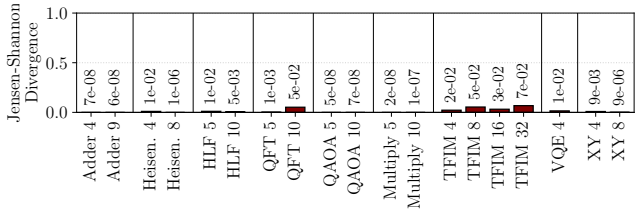
In comparison, the Qiskit compiler optimizations can prove to be less effective depending on the algorithm. For the Heisenberg circuit, Qiskit gives over a 30% reduction in the CNOT gate count. But for most other circuits the reduction is negligible, even resulting in a slight increase in the CNOT gate count for some algorithms such as HLF and Multiply. In contrast, QUEST always performs better than Qiskit and never performs worse than the Baseline.

When the Qiskit compiler optimizations are added on top of the approximate circuits produced by QUEST, there can be a slight improvement or degradation in CNOT count depending on the algorithm. For example, QUEST + Qiskit performs better than QUEST for the eight-qubit Heisenberg circuit, but it performs worse for the four-qubit XY circuit. Nonetheless, for most algorithms we observe that it does not diminish the gains of QUEST and so we use the QUEST + Qiskit configuration for evaluation results going forward.

QUEST's approximations result in a low output distance even in the ideal quantum computing scenario. We now evaluate if the approximate circuits produced by QUEST are resulting in the correct output (close to ground truth results) even in the absence of noise. This can help us understand if the approximate circuits are closely emulating the expected output of the Baseline circuit. Fig. 9(a) and (b) show the TVD and JSD, respectively, between the ground truth output of the Baseline circuit and the approximated noiseless output of QUEST. The figure shows that both the output distance metrics have low values across all algorithms. Given the corresponding reduction



(a) Total Variation Distance



(b) Jensen-Shannon Divergence

Figure 9: Quest ensures low output distance from the ideal output while delivering a reduction in CNOT gate count.

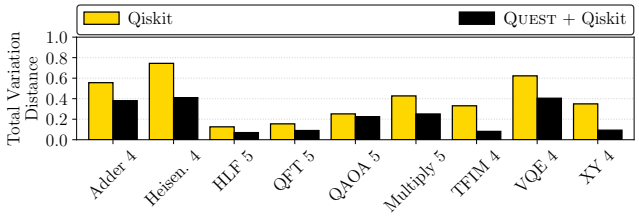


Figure 10: Quest + Qiskit reduces the TVD compared to just Qiskit when the algorithm circuits are run on the IBMQ Manila computer.

in CNOT gate count, these results signify the usefulness of circuit approximations even in a fault-tolerant environment. As both metrics have similar trends, we use only the TVD going forward for brevity.

Quest delivers a significant reduction in TVD on a real NISQ machine. Next, we evaluate the performance of Quest on IBMQ Manila, one of the most recent and least error quantum computers available via IBMQ open access for algorithms which were possible to run. Fig. 10 shows the TVD from the ground truth when the algorithms are run with just the Qiskit optimizations vs. when they are run with Quest + Qiskit. While the raw TVD numbers are large for most algorithms due to the high noise of the current state-of-the-art quantum computers, Quest + Qiskit reduces the TVD by over 0.3 or 30% points in some cases. For example, for the four-qubit TFIM circuit, the TVD drops from 0.35 to 0.08.

Quest delivers a significant reduction in TVD even for larger circuits and as hardware noise is decreased in a quantum simulation. We now present larger circuits run in a noisy simulation environment with noise levels of 1%, 0.5%, and 1% in Fig. 11(a), (b), and (c), respectively. The figures show the percentage reduction in TVD compared to when the Baseline circuit is run with a noisy simulation for Qiskit and Quest + Qiskit. We see that across the board, Quest + Qiskit reduces the TVD even as the hardware noise is reduced. This demonstrates the usefulness of circuit approximations even when projected on to the future when the noise is reduced by 10× (0.1% compared to the current average noise of over 1%).

Quest incurs one-time cost for building approximate circuits to yield meaningful output quality for circuits with a large number of CNOT gates. Quest's approximate circuit building process consists of three steps: (1) partitioning, (2) synthesis, and (3) dual-annealing engine. Fig. 12(a) and (b) show absolute time required for different circuit and relative contribution from each step. Overall, for most circuits Quest can be completed within a few hours. The only exception is TFIM 32 which takes almost a full day. Partitioning takes up most of the time due to TFIM circuit structure. Synthesis and dual-annealing engines are not major contributers. The exact amount of time that each block's synthesis takes depends on the "complexity" of the unitary (e.g., how many gates are in the block, how entangled it is, and how the gates are arranged), which is not possible to quantify analytically. While we do not cap this time manually, empirically we noticed that every four-qubit block across all algorithms synthesized within two minutes.

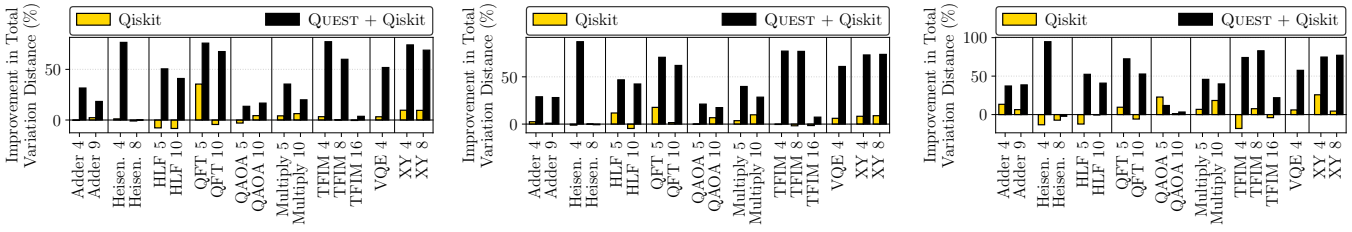
While this one-time cost is non-negligible, it has potential for significant reduction (e.g., all blocks can be synthesized in parallel). But, we did not need to focus on reducing this overhead because of the feedback provided by the domain scientists and physicists who are actively working and improving the two major target applications. They assessed this cost is hidden in the code development and improvement cycle, and hence, wanted this effort to focus on obtaining meaningful output quality. For example, the magnetization curve for the Heisenberg application should match the ground truth curve. Quest achieved these algorithmic and science goals, as confirmed by our case study on TFIM and Heisenberg over their entire time evolution landscape (discussed below).

Also, we note that this overhead is not incurred each time the program needs to be compiled with Qiskit and executed. This is because Quest produces full approximate circuits as one-time output. This one-time output can be compiled with Qiskit (in the order of seconds) whenever needed for optimally mapping the approximate circuit on physical qubits.

4.3 TFIM and Heisenberg: A Case Study

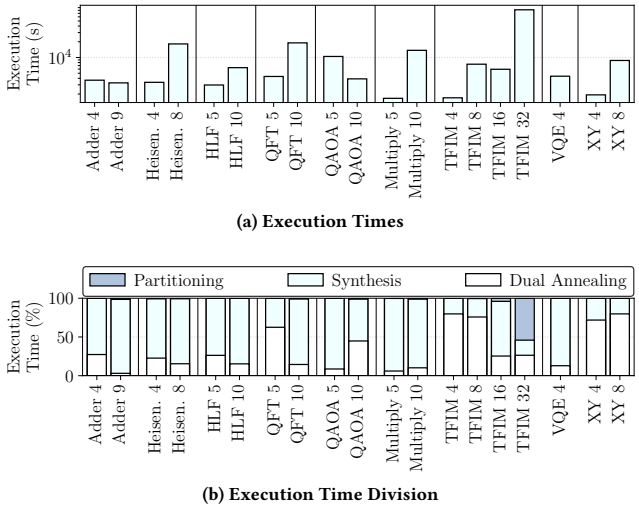
Quest + Qiskit is able to more closely track the ground truth output than just Qiskit optimizations due to the larger reduction in the number of CNOT gates. Fig. 13 shows the time evolutions of the four-spin TFIM and Heisenberg circuits with ground truth, and the Qiskit circuit, and Quest + Qiskit approximate circuits when run on the IBMQ Manila computer. Each step in the time evolution is a different circuit that is separately run with Quest. For the TFIM algorithm, the Quest + Qiskit magnetization line is more stable and closer in magnitude to the ground truth than is the Qiskit magnetization line. For Heisenberg, Quest very closely tracks the ground truth magnetization, while Qiskit produces meaningless results.

In fact, Fig. 14 plots the simulation results with noise levels 1%, 0.5%, and 0.1%, and the figure shows that projected reduction in



(a) 1% Noise (b) 0.5% Noise (c) 0.1% Noise

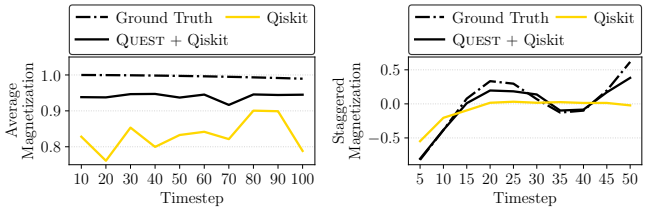
Figure 11: Noisy simulations with different levels of noise indicate that QUEST can reduce TVD even for future NISQ devices.



(a) Execution Times

(b) Execution Time Division

Figure 12: The execution overhead of QUEST and its division among the different steps varies for different algorithms.

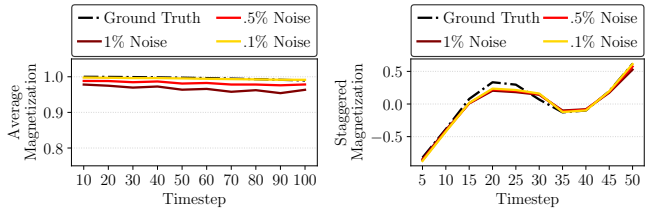


(a) TFIM 4 (b) Heisenberg 4

Figure 13: QUEST achieves closer to the ground truth output on the IBMQ Manila machine than the Baseline.

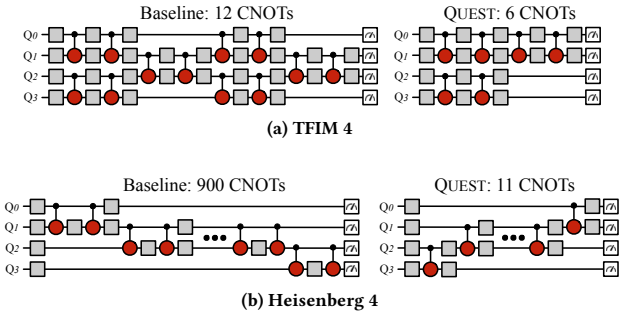
hardware errors can further reduce the output distance for TFIM and Heisenberg. This is due to the large reduction in the CNOT gate count of both the algorithms.

Fig. 15(a) shows that circuit structure of the TFIM algorithm at the 100^{th} timestep with Baseline and one of the approximations generated using QUEST. Similarly, Fig. 15(b) shows the circuit structure of the Heisenberg algorithm at the 50^{th} timestep. The figures



(a) TFIM 4 (b) Heisenberg 4

Figure 14: For TFIM, QUEST performs better as the hardware noise decreases. For Heisenberg, QUEST's output is close to ground truth even in a high noise environment of 1%.



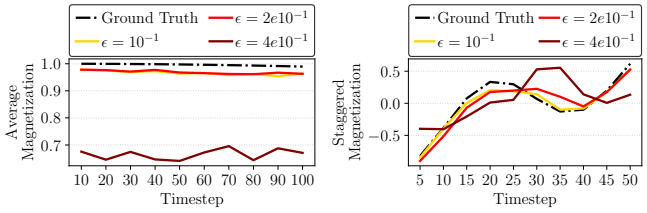
(a) TFIM 4

(b) Heisenberg 4

Figure 15: Illustration of the reduction in the CNOT gate count with QUEST for one of the TFIM and Heisenberg approximate circuits.

illustrate the large reduction in CNOT gate count. For example, for the Heisenberg algorithm, the gate count reduced from 900 CNOTs to just 11 CNOTs with approximate circuits. This reduction enables fewer operations errors and lower decoherence errors due to faster execution.

QUEST's design decisions of using process distance upper bound threshold and dissimilar approximations yields good results. Fig. 16(a) and (b) shows the output of TFIM and Heisenberg algorithms for different process distance thresholds. Recall that if the threshold is set too high, the dual annealing engine is likely to select coarse approximations selecting more circuits with fewer CNOT gate count than



(a) TFIM 4 (b) Heisenberg 4

Figure 16: Careful selection of process distance threshold for the dual annealing engine produces good approximations of the TFIM and Heisenberg circuits.

selecting ones with dissimilar characteristics (approximations with low process distances among them). The figures shows that this can lead to a large error in the output distance for both algorithms. Thus, careful selection of the threshold is required to ensure beneficial results. However, it also does not have to be tuned exhaustively because as the figure shows, Quest performs well for a wide range of values. Therefore, we set the threshold to be proportional to the number of blocks in the circuit and it works well across all algorithms in practice.

5 RELATED WORK

Quantum Circuit Compiling and Mapping. There has been a large focus on attempting to leverage compiler-based passes and quantum computing rules to reduce CNOT and SWAP gate counts and perform noise- and layout- aware mapping of the qubits to the hardware [23, 28, 29, 38, 40, 42, 45, 47, 48].

For example, in the circuit mapping space, previous efforts have exploited the diverse error characteristics of different qubits to map the same baseline circuit in different ways expecting the output distances to reduce [33, 41]. But these works do not target reducing the CNOT count considerably by employing approximate synthesis to systematically generate dissimilar circuits and thus, have large output distances.

Quantum Circuit Synthesis. Quest employs synthesis as one of the steps in its procedure to generate approximate circuits. Previous synthesis works have attempted to synthesize circuits with only specific gates (e.g., only CNOTs) or universal circuits as exactly as possible with as few CNOT gates as possible [10, 15, 16, 19, 25, 31, 37, 39, 46]. We use a modified version of the Leap synthesis tool for Quest as it performs better than previous approaches [39].

Despite the potential of approximations, not many procedures to generate resource efficient approximations using synthesis have existed before Quest. Madden et al. [24] describe generative procedures using synthesis, but these procedures are non-scalable and lack any apriori criteria for selecting approximations across different algorithms. Amy et al. [3] describe a more scalable direct synthesis algorithm, but the approach leads to very long circuits, in some case, orders of magnitude longer compared to other synthesis tools. In comparison, Quest defines a clear criterion for selecting dissimilar approximate circuits apriori, provides a theoretical proof to bound process distances, and generates circuits with few CNOTs.

6 CONCLUSION

In this work, we presented Quest, a technique to reduce CNOT gate count of quantum circuits using approximate synthesis and distance-based approximation selection. We provided a theoretical derivation to bound the process distance of approximations as well as a dissimilarity criterion to select approximations in a manner that reduces the output distance. Quest achieves a CNOT gate reduction of 30-80% across algorithms while maintaining a low output distance from the output of the original circuit. While Quest's contributions are beneficial in the NISQ era for minimizing the impact of noise, they are also useful for circuit size reduction on fault-tolerant quantum computers.

ACKNOWLEDGMENTS

We thank the ananoymous reviewers and our shepherd, Professor Phillip Stanley-Marbell, for their constuctive feedback. This work was supported by the Office of Science, Office of Advanced Scientific Computing Research Accelerated Research for Quantum Computing Program of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 and the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract No. DE-AC05-00OR22725. This work was also supported by Northeastern University, NSF Award 1910601, and the Massachusetts Green High Performance Computing Center (MGHPCC) facility. IBM Q was also used 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.

A ARTIFACT APPENDIX

A.1 Abstract

This appendix describes the code and data artifacts related to QUEST. The artifacts are open-source at https://doi.org/10.5281/zenodo.5747894. They include the input files for the executed benchmarks, the code for partitioning, synthesis, dual annealing, and simulation, as well as a docker image set up with the code. Please see the following sections for more details, especially the Experiment Workflow section (Sec. A.5) to read in detail about how the artifact directories and code files are organized.

A.2 Artifact Check-List (Meta-Information)

Artifact meta-information is as following:

- **Algorithm:** All used algorithms are included in the input_qasm_files directory
- **Program:** All used benchmarks are included in the input_qasm_files directory.
- **Compilation:** Compiler depedencies: LEAP [39] (included), BQSKit [1], and Qiskit [2].
- **Data set:** All used input data are included in the input_qasm_files directory
- **Run-time environment:** No specific run-time environment required.
- **Hardware:** Any computer that supports Python and IBMQ quantum cloud access.
- **Run-time state:** Artifact not senstivie to any run-time state.
- **Execution:** No specific execution environment required.
- **Metrics:** Total variation distance (TVD) and the Jensen-Shannon Divergence (JSD)

- **Output:** Code outputs .qasm and .npy files.
- **Experiments:** Figures 8-16 in the paper.
- **How much disk space required (approximately)?:** 20 GB
- **How much time is needed to prepare workflow (approximately)?:** Approximately 1 hour to download and install dependencies.
- **How much time is needed to complete experiments (approximately)?:** 3-5 days
- **Publicly available?:** https://doi.org/10.5281/zenodo.5747894
- **Archived (provide DOI)?:** https://doi.org/10.5281/zenodo.5747894

A.3 Description

A.3.1 How to access. The code and dataset artifacts of QUEST can be obtained at: https://doi.org/10.5281/zenodo.5747894

A.3.2 Hardware dependencies. The partitioning, synthesis, and dual annealing components of QUEST were run on our local compute cluster consisting of 2.4 GHz Intel E5-2680 v4 CPUs. However, any computer that supports Python can be used. Running noisy simulations on the IBMQ QASM simulator and running executions on real IBMQ quantum computers requires access to the IBMQ cloud. A free account can be created on the cloud for open access to the noisy simulator and some of the quantum machines available on the cloud: https://quantum-computing.ibm.com.

A.3.3 Software Dependencies. The software dependencies include Python 3.9 libraries glob, numpy, and scipy. The modified version of the LEAP compiler [39] is provided in the leap_compiler directory. The dependencies also include the open-source packages bqskit [1] and qiskit [2]. No proprietary or third-party software artifacts are required. To run the docker container using the provided image, Docker must also be installed: https://docs.docker.com/get-docker.

A.3.4 Data Sets. The only data artifacts are the input circuits, which have been provided in the input_qasm_files directory. The file names have the format ⟨algorithm_name⟩_⟨number_of_qubits⟩. The time step is also specified in file names for time-evolution-based algorithms. No proprietary or third-party data artifacts are required.

A.4 Installation

Installation simply requires the installation of the software dependencies listed in Sec. A.3.3 and downloading the code files from the provided DOI link: https://doi.org/10.5281/zenodo.5747894.

A.5 Experiment Workflow

A.5.1 Directory Organization. **leap_compiler:** Contains the modified version of the leap compiler from [39]. It is imported in quest_code/generate_post_partitioning_files.py and quest_code/generate_post_synthesis_files.py

input_qasm_files: Contains the input files of all the algorithm circuits used in the evaluation of QUEST in OPENQASM 2.0 format.

quest_code: Contains the Python 3.9 code to run all the steps involved in QUEST: partitioning, synthesis, dual annealing, and simulation result generation.

post_partitioning_files: Contains files generated after running the generate_post_partitioning_files.py file. The code generates individual files containing information about the circuit structure (OPENQASM 2.0 format), qubits involved, and the corresponding unitary matrix for all of the blocks obtained after partitioning the circuit.

post_synthesis_files: Contains files generated after running the generate_post_synthesis_files.py file. The code generates individual files containing information about the circuit structure (OPENQASM 2.0 format) and the corresponding unitary matrix for all of the block approximations generated after synthesis.

dual_annealing_solutions: Contains files generated after running the generate_dual_annealing_solutions.py file. A single file is generated containing information about all of the full circuit approximations selected.

simulation_results: Contains files generated after running the generate_simulation_results.py file. A single file is generated containing information about simulation / real-computer results related to executing the full circuit approximations.

docker_image: This directory contains files listing Python and library version information for QUEST. It also contains the Dockerfile to build a docker image with the Python environment required for QUEST (and the libgfortran5 dependency). A built image with the necessary QUEST code has also been provided in quest_docker_image.tar. This image can be used to run a container:

```
docker load --input quest_docker_image.tar
docker run --name quest_container quest
```

A.5.2 Code File Organization. The section describes the function of the files in the quest_code directory. For all files, the parameters that require to be set by the user are expressed at the top of the file, immediately following the library imports. The description comments of these parameters start with XXX for convenient lookup. All parameters are set according to the adder_4 algorithm by default.

generate_post_partitioning_files.py: This file partitions a given circuit and outputs block information. Its library dependencies include glob, numpy, leap_compiler [39], bqskit [1], and qiskit [2]. It requires setting the algo parameter to one of the algorithms in the input_qasm_files directory. A maximum block size of four qubits is used to partition the circuits using the scan partitioner from bqskit. It forms partitions by traversing the circuit left to right and creating four-qubit blocks. Some blocks may be of a smaller size if need be. A maximum block size of four qubits is used as it synthesizes efficiently and yields good results. Running the code will print to connection map, which needs to be specified in the generate_post_synthesis_files.py and generate_simulation_results.py files. The file writes output to the post_partitioning_files directory. For each block, the following files are written: qasm_block_⟨block_id⟩.qasm (OPENQASM 2.0 file describing the circuit structure), qbit_block_⟨block_id⟩.qasm (qubits involve in the block), and unit_block_⟨block_id⟩.npy (the unitary matrix representation of the block). Note that the blocks ids in the topological order of the blocks in the full circuit.

generate_post_synthesis_files.py: The file runs synthesis on all the blocks generated during the previous step and outputs all approximations for a block that are generated by the compiler. Its library dependencies include glob, numpy, leap_compiler [39] and qiskit [2]. It requires setting the connect_map parameter as outputted by the previous generate_post_partitioning_files.py file. The synthesis step can be run using the LEAP compiler for different blocks in parallel (we run it on up to ten compute nodes in our experiments). The file writes output to the post_synthesis_files directory. For each block, multiple approximations with multiple seeds and depths in the approximation tree can be generated. The OPENQASM 2.0 circuit structures of these approximations are stored as a list for each (block id, seed, depth) combination in qasm_block_⟨block_id⟩_seed_⟨random_seed⟩_depth_⟨depth⟩.npy. The corresponding unitary matrices are stored in unit_block_ ⟨block _id⟩_seed_⟨random_seed⟩_depth_⟨depth⟩.npy.

generate_dual_annealing_solutions.py: This file runs the dual annealing engine that generates full circuit approximation samples. Its library dependencies include glob, numpy, and scipy. It requires setting the N_CX parameter, which is the number of CX gates in the original circuit. This parameter is used to normalize the CX count for approximation selection. The SciPy package is used to run the dual annealing engine. The process distance threshold to eliminate coarse approximations in the annealing engine is set proportional to the number of blocks in the circuit and up to 16 approximations are generated for each algorithm. A default weight of 0.5 is used for CNOT gate count and approximation dissimilarity in the objective function that the dual annealing engine minimizes. The file writes output to the dual_annealing_solutions directory. The code generates a single file of the format block_dist_⟨approximation_threshold⟩ _⟨distance_ percentile⟩_⟨weight_on_CX_count⟩_data_n_cxs.npy which contains information about the CX counts, distances, and block ids of selected approximations.

generate_simulation_results.py: This file generates results by running simulations / real-computer executions of the full circuit approximations generated during the previous step. Its library dependencies include glob, numpy, and qiskit [2]. The file requires setting the IBMQ cloud access credentials, as well as the algo parameter to one of the algorithms in the input_qasm_files directory, the NQBT parameter to the number of qubits in the algorithm circuit, the connect_map parameter as outputted by the generate_post_partitioning_files.py file, and the p_gate parameter to set the noise level. The ground truth results are obtained by running the circuit in an ideal quantum simulation environment using the Qiskit unitary simulator, which is part of the Qiskit Aer package. The code supports noisy simulations on circuits up to 16 qubits using the IBMQ QASM simulator available via the IBM quantum cloud. A Pauli noise model is used for all the qubits with noise levels of 1%, 0.5%, and 0.1% to simulate how QUEST will perform for future NISQ computers as the noise level decreases. In the QUEST paper, we run circuits up to five qubits on the IBMQ Manila quantum computer to demonstrate how the technique performs on existing quantum computers. The default setting runs 8192 experimental trials (maximum allowed) per experiment. The file writes output to the simulation_results directory. The code generates a single file of the format block_dist_⟨approximation_threshold⟩_⟨distance_percentile⟩_ ⟨weight_on_CX_count⟩_⟨noise_level⟩_n_cxs.npy which contains information about the ideal and noisy runs: CX counts with just QUEST, unitary matrices, noisy output with just QUEST, CX counts with QUEST + Qiskit optimizations, and noisy output with QUEST + Qiskit optimizations. The unitary matrices can be used to calculate the total variation distance (TVD) and the Jensen-Shannon Divergence (JSD) between the initial circuits and the approximations. The noisy outputs can be used to calculate their TVD to the ideal unitaries. Please check the code for details about the output format.

A.6 Evaluation and Expected Results

The key results in Figure 8 and Figure 9 of the paper should be reproducible using the input algorithm files, partitioning, synthesis, and dual annealing runs on a Python-supported computer and the IBMQ QASM simulator. Figure 10 can be reproduced using the IBMQ Manila quantum computer, which is available under open access account as of December 1, 2021. Figure 11 can be reproduced using noisy simulations at 1%, 0.5% and 0.1% noise levels (setting p_gate parameter in generate_simulation_results.py to 0.01, 0.005, 0.001) using the IBMQ QASM simulator. The timing results in Figure 12 may vary depending on whether 2.4 GHz Intel E5-2680 v4 CPUs are used and the degree of parallelism used for synthesis. Figure 13 requires runs on the IBMQ Manila machine for all time steps of the four-qubit TFIM and Heisenberg circuits (provided in the input_qasm_files directory). Figure 14 and 16 can be generated using the IBMQ QASM simulator using circuits for provided time steps of the four-qubit TFIM and Heisenberg circuits.

REFERENCES

[1] https://github.com/BQSKit. BQSKit: Berkeley Quantum Synthesis Toolkit.
[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] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. 2013. A Meet-in-the-Middle Algorithm for Fast Synthesis of Depth-Optimal Quantum Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 6 (2013), 818–830.
[4] Lindsay Bassman, Connor Powers, and Wibe A de Jong. 2021. ArQTiC: A Full-Stack Software Package for Simulating Materials on Quantum Computers. *arXiv preprint arXiv:2106.04749* (2021).
[5] Ethan Bernstein and Umesh Vazirani. 1997. Quantum Complexity Theory. *SIAM Journal on computing* 26, 5 (1997), 1411–1473.
[6] Sergey Bravyi, David Gosset, and Robert König. 2018. Quantum Advantage with Shallow Circuits. *Science* 362, 6412 (2018), 308–311.
[7] Heinz-Peter Breuer, Elsi-Mari Laine, and Jyrki Piilo. 2009. Measure for the Degree of Non-Markovian Behavior of Quantum Processes in Open Systems. *Physical review letters* 103, 21 (2009), 210401.
[8] Davide Castelvecchi. 2017. IBM's Quantum Cloud Computer Goes Commercial. *Nature News* 543, 7644 (2017), 159.
[9] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. 2004. A New Quantum Ripple-Carry Addition Circuit. *arXiv preprint quant-ph/0410184* (2004).
[10] 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.
[11] David Elieser Deutsch. 1989. Quantum Computational Networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 425, 1868 (1989), 73–90.
[12] Edward Farhi and Aram W Harrow. 2016. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. *arXiv preprint arXiv:1602.07674* (2016).

[13] Alexei Gilchrist, Nathan K Langford, and Michael A Nielsen. 2005. Distance Measures to Compare Real and Ideal Quantum Processes. *Physical Review A* 71, 6 (2005), 062310.

[14] 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).

[15] Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. 2016. Quantum Circuits for Isometries. *Physical Review A* 93, 3 (2016), 032318.

[16] Raban Iten, Oliver Reardon-Smith, Emanuel Malvetti, Luca Mondada, Gabrielle Pauvert, Ethan Redmond, Ravjot Singh Kohli, and Roger Colbeck. 2019. Introduction to UniversalQCompiler. *arXiv preprint arXiv:1904.01072* (2019).

[17] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2016. SciPy: Open Source Scientific Tools for Python, 2001.

[18] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T Sornborger, and Patrick J Coles. 2019. Quantum-Assisted Quantum Compiling. *Quantum* 3 (2019), 140.

[19] Aleks Kissinger and Arianne Meijer-van de Griend. 2019. CNOT Circuit Extraction for Topologically-Constrained Quantum Memories. *arXiv preprint arXiv:1904.00633* (2019).

[20] Alexei Yu Kitaev, Alexander Shen, Mikhail N Vyalyi, and Mikhail N Vyalyi. 2002. *Classical and Quantum Computation*. Number 47. American Mathematical Soc.

[21] Vadym Kliuchnikov, Alex Bocharov, and Krysta M Svore. 2014. Asymptotically Optimal Topological Quantum Compiling. *Physical review letters* 112, 14 (2014), 140504.

[22] Ang Li and Sriram Krishnamoorthy. 2020. QASMBench: A Low-Level qasm Benchmark Suite for NISQ Evaluation and Simulation. *arXiv preprint arXiv:2005.13018* (2020).

[23] 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.

[24] Liam Madden and Andrea Simonetto. 2021. Best Approximate Quantum Compiling Problems. *arXiv preprint arXiv:2106.05649* (2021).

[25] 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* 18, 6 (2016), 063029.

[26] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. 2016. The Theory of Variational Hybrid Quantum-Classical Algorithms. *New Journal of Physics* 18, 2 (2016), 023023.

[27] 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).

[28] 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.

[29] 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.

[30] Victor Namias. 1980. The Fractional order Fourier Transform and its Application to Quantum Mechanics. *IMA Journal of Applied Mathematics* 25, 3 (1980), 241–265.

[31] Beatrice Nash, Vlad Gheorghiu, and Michele Mosca. 2020. Quantum Circuit Optimizations for NISQ Architectures. *Quantum Science and Technology* 5, 2 (2020), 025010.

[32] Michael A Nielsen and Isaac L Chuang. 2010. Frontmatter. , i–viii pages.

[33] 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.

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

[35] Kenneth Regan, Amlan Chakrabarti, and Chaowen Guan. 2018. Algebraic and Logical Emulations of Quantum Circuits. In *Transactions on Computational Science XXXI*. Springer, 41–76.

[36] Kemal H Sahin and Amy R Ciric. 1998. A Dual Temperature Simulated Annealing Approach for Solving Bilevel Programming Problems. *Computers & chemical engineering* 23, 1 (1998), 11–25.

[37] Vivek V Shende, Stephen S Bullock, and Igor L Markov. 2006. Synthesis of Quantum-Logic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 6 (2006), 1000–1010.

[38] 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.

[39] Ethan Smith, Marc G Davis, Jeffrey M Larson, Ed Younis, Costin Iancu, and Wim Lavrijsen. 2021. LEAP: Scaling Numerical Optimization Based Synthesis Using an Incremental Approach. *arXiv preprint arXiv:2106.11246* (2021).

[40] 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.

[41] 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.

[42] 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.

[43] Robert R Tucci. 2005. An introduction to Cartan's KAK decomposition for QC programmers. *arXiv preprint quant-ph/0507171* (2005).

[44] Bo-Ying Wang and Fuzhen Zhang. 1994. A Trace Inequality for Unitary Matrices. *The American Mathematical Monthly* 101, 5 (1994), 453–455.

[45] 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.

[46] Ed Younis, Koushik Sen, Katherine Yelick, and Costin Iancu. 2021. QFAST: Conflating Search and Numerical Optimization for Scalable Quantum Circuit Synthesis. *arXiv preprint arXiv:2103.07093* (2021).

[47] Chi Zhang, Ari B Hayes, Longfei Qiu, Yuwei Jin, Yanhao Chen, and Eddy Z Zhang. 2021. Time-optimal Qubit Mapping. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 360–374.

[48] 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.