Is Circuit Cutting Scalable for Practical Quantum Applications?

Songqinghao Yang
University of Cambridge
sqhy2@cam.ac.uk

Prakash Murali
University of Cambridge
pm830@cam.ac.uk

***Abstract*—Circuit cutting allows quantum circuits larger than the available hardware to be executed. Cutting techniques split circuits into smaller subcircuits, run them on the hardware, and recombine results through classical post-processing. Circuit cutting techniques have been extensively researched over the last five years and it been adopted by quantum companies, like IBM, as part of their scaling roadmap. We examine whether current circuit cutting techniques are practical for orchestrating executions on fault-tolerant quantum computers. We conduct a resource estimation-based benchmarking of important quantum applications and different types of circuit cutting techniques. Our applications include practically relevant algorithms such as Hamiltonian simulation, kernels such as quantum Fourier transform and more. To cut these applications, we use IBM's Qiskit cutting tool. We estimate resources for subcircuits using Microsoft's Azure Quantum Resource Estimator and develop models to determine qubit counts and the quantum and classical runtime needs of circuit cutting. We demonstrate that while circuit cutting works for small-scale systems, the exponential growth of the quantum excution runtime and the classical post-processing overhead as the qubit count increases renders it impractical for larger quantum systems with current implementation strategies. As we transition from noisy quantum hardware to fault-tolerance, our work provides useful guidance for the design of quantum software and runtime systems.**

***Index Terms*—circuit cutting, fault-tolerant quantum, resource estimation**

I. Introduction

Quantum computing (QC) holds transformative potential across numerous fields, including chemistry, material science and cryptography [1], [2]. Current quantum hardware includes systems with 50 to 1000 qubits, depending on the physical qubit technology. These systems are popularly referred to as Noisy Intermediate Scale Quantum (NISQ) to indicate their restricted qubit count and limited gate quality [3]. In the last two years, QC has started transitioning from NISQ to fault-tolerance, with demonstrations of systems with a few *logical* qubits that make use of Quantum Error Correction (QEC).

To achieve practical quantum advantage, that is, a speedup over classical computing on a useful problem, we require systems with several thousand logical qubits, potentially with a few million physical qubits [2], [4], [5]. To progress towards this vision, it is important to understand which qubit technologies, software techniques and architectures are likely to scale up to meet the computational needs of practically useful applications and which techniques have scaling limitations. This paper examines the scalability of circuit cutting and knitting techniques for the upcoming fault-tolerant quantum era.

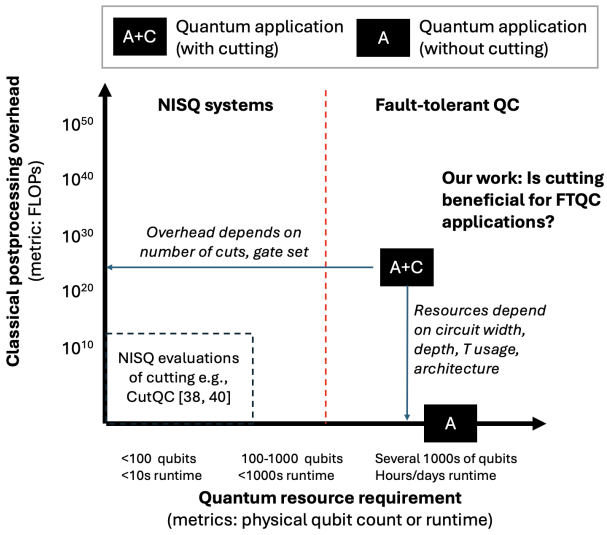

Fig. 1: When circuit cutting is used to run algorithms on a fault-tolerant quantum computer (FTQC), we incur a quantum processing cost and a classical postprocessing overhead. We evaluate these costs for practical-scale benchmarks on a realistic FTQC architecture to determine the scalability of circuit cutting. Prior works are restricted to small NISQ benchmarks, shown in the bottom left of this figure.

Circuit cutting is a popular technique to execute QC algorithms that use more qubits than available in quantum hardware. It partitions large quantum circuits into smaller, manageable subcircuits that are executed on a device. The results from these executions are classically post-processed to reconstruct the result of the original circuit [6]. Circuit cutting has been extensively studied theoretically to explore different types of circuit partitions and optimizations for reducing its classical compute overheads [7]–[11]. It has also been experimentally validated, with demonstrations of circuit executions that go beyond small device sizes, improvements in application fidelity and several tools are available to optimize cut locations and orchestrate executions [12]–[14].

Despite known theoretical exponential scaling, circuit cutting is viewed as a key part of the scalable quantum stack and is a part of industry roadmaps that target million-qubit

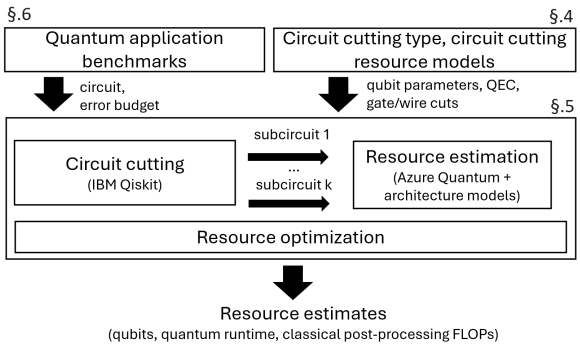

Fig. 2: Our evaluation framework. Inputs are 1) a set of practically important quantum applications and desired error budget and 2) circuit cutting type and models for its resource usage in terms of number of quantum subcircuits, number of samples required and classical reconstruction overheads. Applications are cut and resources are estimated and optimized for subcircuits using models for a fault-tolerant QC system. Outputs include the number of cuts, physical qubits, and the quantum and classical runtime required for the application.

scale devices [7], [15]–[18]. This is primarily due to successful NISQ evaluations of circuit cutting [12]–[14] and attempts to manage exponential scaling overheads through optimized classical postprocessing software [19]. Circuit cutting is also seen as a promising technique to distribute quantum executions across multiple quantum processors in a manner akin to distributed computations in classical computing [18]. However, existing research provides little clarity on the scalability of circuit cutting techniques for future quantum applications.

We ask whether circuit cutting can be used to efficiently orchestrate executions of practically important quantum applications. Figure 1 illustrates our overall research question. We ask whether circuit cutting should be part of the long-term QC stack and whether components in the stack should be designed considering its classical post-processing or communication needs. This is challenging because of a number of factors. First, circuit cutting's resource requirements are closely tied to the structure of the circuits that we wish to partition and vary significantly across problem instances. Second, there are a number of variants of the technique with different classical and quantum overheads, including the basic qubit and gate cuts [6], [8], [20] and more complex types of cuts that depend on the gate set and communication assumptions [7], [10], [21]. Third, practical resource requirements are influenced by the full stack, from the desired fidelity of the application, down to the choice of QEC methods and quality of the physical qubits [22].

We conduct a comprehensive benchmark-driven resource analysis of circuit cutting techniques. Figure 2 illustrates our approach. To address the challenges above, we develop a benchmark suite that consists of a diverse set of *practical-scale* quantum circuits. This is a crucial choice, since in applications like Hamiltonian simulation, only circuits that meet certain depth and connectivity thresholds can serve as meaningful physical models and offer practical QC use cases [18].

To navigate the intricacies of different cutting techniques, we combine the Qiskit add-on circuit cutting framework [23] and the Azure Quantum Resource Estimator to evaluate resource estimates (such as physical qubit counts, runtime) for full applications orchestrated with circuit cutting. In contrast, existing works only provide overhead estimates at the level of individual gates. Further, we develop simple optimization techniques that reduce resource requirements through careful use of distillation factories and resource partitioning across subcircuits. Finally, we augment our benchmarking study with a theoretical analysis of circuit cutting requirements of each benchmark and use this to validate our experimental findings.

Our contributions are as follows:

- Circuit cutting reduces physical qubit requirements on fault-tolerant QC systems by an average 30%. However, obtaining qubit reductions comes at a prohibitive cost in terms of the large number of cuts, the runtime required on the quantum computer and classical post-processing time. Therefore, circuit cutting is not scalable for most practically useful quantum algorithms.
- Most QC applications do not display the connectivity structures required for obtaining a small number of cuts, with typical cases requiring a linear increase in the number of cuts as problem size increases.
- The choice of cutting method and type of gate that is cut have a significant impact on the quantum excution runtime and classical post-processing overhead. Certain benchmarks such as QAOA have manageable classical overhead, but large quantum excution runtime, while others such as QFT have manageable quantum overheads with exponential classical overheads. Therefore, we conclude that in spite of a wide variety of cutting techniques being developed [10], [11], [24], [25], resource requirements are extremely high across different cutting scenarios.

II. Background

Circuit cutting: Circuit cutting involves three phases:

- **Cut**: In this phase, a large quantum circuit is divided into smaller, manageable subcircuits by strategically introducing "cuts" at specific points. This is typically achieved by inserting operations such as mid-circuit measurement or other unitaries that allow the system to be partitioned while preserving essential quantum information.
- **Run**: The subcircuits are then executed independently on quantum hardware or simulators. The choice of the subcircuits to run will depends on two factors: the sampling number N_s and the pool of subexperiments C, constructed from the subcircuits $c_i \in C = \{c_1 \dots c_{N_{\text{pool}}}\}$, with N_{pool} being the total number of subexperiments in the pool. After cutting the circuit, one needs to uniformly sample N_s subcircuits from the pool C to execute and record the measurements.

- **Reconstruct**: The results from the subcircuits are combined mathematically with tensor networks as Kronecker delta product to recreate the outcomes of the original circuit. This phase often involves classical post-processing to integrate the data and correct for any approximations introduced during the cutting phase. Typically, if there are n cuts in a circuit with N qubits, the result tensor would have dimension $4^n \times 2^N$, which has n legs with 4-dimension from decomposition and one leg with 2^N-dimension, which is the output for the original circuit [14], [26].

Fault-tolerant quantum computation: FTQC is distinguished from NISQ devices by its use of QEC to reliably execute arbitrarily long computations, while NISQ systems operate without error correction, limiting their scalability due to noise. By encoding logical qubits into entangled states of multiple physical qubits, QEC schemes can detect and correct errors without directly measuring the logical qubits. Despite the application of QEC, errors still exist from the logical level, the rotational gate synthesis, the magic state distillation and the algorithmic level. We use the term *overall error budget* to define the acceptable level of error, computed as:

$$\epsilon = \epsilon_{\text{log}} + \epsilon_{\text{dis}} + \epsilon_{\text{syn}} + \epsilon_{\text{alg}} \quad (1)$$

in the final results of a quantum computation. The error terms here are additive because it is a first approximation of the logical error when the individual gate logical error rates are low. This is a standard error model used in prior quantum resource estimation works [22], [27].

III. Resource Requirements and Tradeoffs

A. Types of cutting techniques

There are several ways of cutting a quantum circuit into subcircuits, with varying resource requirements. Circuit cutting can be primarily classified into two methods: 'wire cutting' and 'gate cutting' [24], [28]. As the name suggests, qubits are cut in the former and two-qubit gates are cut in the latter. The total samples required for reconstruction with error ε_{rct} are $\mathcal{O}(16^{n_{\text{wire}}}/\varepsilon_{\text{rct}}^2)$ and $\mathcal{O}(9^{n_{\text{gate}}}/\varepsilon_{\text{rct}}^2)$, respectively, for the wire cut and gate cut (CX) [6], [20]. For gate cuts, the overhead is dependent on the type of gate that is cut and even the rotation angles that may parameterize it. Table I summarizes these overheads [23]. Therefore, two applications with the same patterns of two-qubit gates may have entirely different circuit cutting overheads owing to differences in the exact type of two-qubit gates they use.

B. Full-stack resource influence and tradeoffs

Circuit cutting resource requirements are also heavily influenced by an application's error requirements and the architecture of the quantum computer. As the required error for an application decreases, the resource requirements tend to increase, both in terms of physical qubit counts and runtime. Such behavior is typically understood for single quantum circuits, but when circuit cutting is used, we have several subcircuits each with its own error. If we have k subcircuits, a naive approach is to run each subcircuits with error ϵ/k. However, this may be suboptimal in terms of resource requirements since subcircuits may have different qubit counts, gate depths and magic state requirements. Our explores this optimization opportunity and its impact on the practicality of circuit cutting.

TABLE I: The table above provides the sampling overhead factor for various two-qubit instructions, provided that only a single instruction is cut. Note that gates like CYGate, RYYGate, and CRZGate have the same overhead as their X counterparts. For simplicity, we do not list them all here. Collected from [23].

Instruction	Sampling overhead factor
CSGate, CSXGate	$3 + 2\sqrt{2} \approx 5.828$
CXGate, CHGate	$3^2 = 9$
iSwapGate, SwapGate	$7^2 = 49$
RXXGate, RZXGate	$[1 + 2\,\lvert\sin(\theta)\rvert]^2$
CRXGate, CPhaseGate	$[1 + 2\,\lvert\sin(\theta/2)\rvert]^2$

The physical resource requirements with cutting depend on the type of error correction code and distillation factories. We assume surface codes for this work, a standard choice for QEC on superconducting qubits and other solid state platforms [29], [30]. For distillation, we consider two aspects. First, the type of distillation circuit that is used has an impact on the error of magic states, which in turn influences the error of the circuit that uses it and its resource requirements. Second, the number of distillation factories influences qubits and runtime: more factories mean that subcircuits run faster, but at the expense of higher physical qubit counts.

In summary, we ask: *"What are the qubit and runtime demands when circuit cutting is employed to run applications? Do practical-scale quantum applications have the necessary clustering structure to benefit from circuit cutting? How does the instruction mix of the application and cutting method influence the scalability of circuit cutting? How much do other factors such as the subcircuit errors and distillation choices affect circuit cutting performance?"*

C. Overview of our approach

To answer the design questions, we develop the toolflow shown in Figure 2. Using a benchmark that consists of practically-relevant quantum applications and a model for a surface code-based quantum computer, we conduct resource estimation studies to determine the impact of circuit cutting on application qubit counts and runtime needs.

To enable this study, we first develop models to quantify the resource needs of circuit cutting. To apply these models for applications, we use IBM's Qiskit circuit cutting tools to find optimized cuts. We used Microsoft's Azure Quantum Resource Estimator to determine resources for executing the subcircuits from the cut circuit. We also develop simple optimizations to reduce resource needs through error and distillation factory optimizations.

IV. Modelling Circuit Cutting Executions

Given a quantum circuit and a cutting method such as wire or gate cutting, we develop models that capture the logical qubit requirements, number of subcircuits, quantum excution runtime and classical post-processing overhead. These models are based on known theoretical foundations of circuit cutting.

Circuit cutting uses the quasi-probabilistic decomposition (QPD) of a quantum channel, which evolves ρ_0 to ρ as $\mathcal{E}(\rho_0)$. Then using the Kraus decomposition [31], $\mathcal{E}(\rho) = \sum_{j=1}^{m} E_j \rho E_j^\dagger$, this representation allows flexibility in choosing local Kraus operators E_j. For a bipartite system $\rho = \rho^{(1)} \otimes \rho^{(2)}$, a channel $\mathcal{E}$ acting non-locally can be decomposed into local operations:

$$\mathcal{E}(\rho) = \sum_{i=1}^{m} q_i \mathcal{E}_i^{(1)}(\rho^{(1)}) \otimes \mathcal{E}_i^{(2)}(\rho^{(2)}), \tag{2}$$

where q_i are quasi-probabilities ($\sum q_i = 1$), allowing the separation of qubits for circuit cutting.

The effectiveness of circuit cutting techniques is significantly limited by the exponential increase in runtime required to stitch together the results from subcircuits. Circuit cutting entails classically monitoring all quantum degrees of freedom at each cut point, which naturally leads to an exponential growth in computational demand as the quantum state size expands. This exponential overhead in circuit execution remains a dominant cost of the QPD decomposition process, regardless of the goal of the circuit cutting application [32].

Specifically, we categorize the overhead of circuit cutting into 1) the *runtime sampling overhead* and (2) the *classical recombination post-processing overhead*.

Runtime sampling overhead: This term refers to the total runtime required on the quantum computer when circuit cutting is used to run an application. This is dependent on two factors. First, when a circuit is cut, the number and type of cuts determine the number of degrees of freedom that must be tracked at each cut, leading to a set of subcircuits. From this set, a subset is sampled based on the reconstruction error required in the expectation value of the observable of interest. Second, each subcircuit that is sampled is run on the quantum computer and the architecture of the device influences its runtime and qubit needs.

It means our total quantum excution runtime equals the number of samples times the quantum excution runtime of each sample. Formally, the first term can be expressed as:

$$\text{Sampling overhead} := \gamma(\mathcal{E})^2, \quad \gamma(\mathcal{E}) := \sum_{i=1}^{m} |q_i|. \tag{3}$$

In practice, the subcircuits are executed with the total number of samples N_s to achieve a desired error ε_{rct}:

$$N_s = O\left(\gamma^2 \times \frac{1}{\varepsilon_{\text{rct}}^2}\right), \tag{4}$$

which comes from the Hoeffding's inequality of the Monte-Carlo sampling. Furthermore, if we apply the cutting at n places in a quantum circuit, the total number of measurements with the error ε_{rct} results in $O(\gamma^{2n}/\varepsilon_{\text{rct}}^2)$. The total samples required for reconstruction with error ε_{rct} were first shown in [6], [20] to be $\mathcal{O}(9^{n_{\text{gate}}}/\varepsilon_{\text{rct}}^2)$ and $\mathcal{O}(16^{n_{\text{wire}}}/\varepsilon_{\text{rct}}^2)$, respectively, for the gate cut (CX) and wire cut.

More specifically, our task is to estimate the expectation $\mathbb{E}[f(y)]$ for a random bitstring $y \in \{0,1\}^N$ sampled from the original circuit [33]. The input state ρ is initialized in $|0\rangle\langle 0|^{\otimes N}$ and the observable is calculated from some output function $f : \{0,1\}^N \to [1,-1]$. At each space-like cut, we get 6 different sets of single-qubit operations, so n_{gate} cuts induce $6^{n_{\text{gate}}}$ terms. Likewise, n_{wire} time-like cuts induce $8^{n_{\text{wire}}}$ terms. We call each of these partition-induced circuit instances a subexperiment. The total number of subexperiments are then $6^{n_{\text{gate}}} 8^{n_{\text{wire}}}$. we can then take a Monte-Carlo approach to estimate the sum, that is, we randomly choose N_s circuits with a uniform distribution in the pool of the subexperiments to run and average them.

We assume a space-efficient model for the subcircuit executions. That is, when we have a collection of subcircuits to execute, we execute them one after another on the QC system. This leads to the minimum number of logical and physical qubits, at the expense of runtime. This approach has two benefits. One, while it increases the overall runtime due to the sequential execution of subcircuits, it significantly reduces the total qubit resources necessary for synthesis and distillation factories. Two, this model allows us to determine the best case qubit reductions possible from circuit cutting. Another approach would be to consider a time-efficient model where subcircuits are run in parallel on a large number of small QC systems, leading to high qubit usage and reduced runtime. The maximum benefits from such an execution are at best a linear speedup in the quantum excution runtime, but classical post-processing costs will not change.

In the space-efficient model, with each subexperiment having Q_c logical qubits, the number of logical qubits required is

$$Q_{\text{max}} = \max_{c \in \{\text{subexperiments}\}} Q_c. \tag{5}$$

The quantum excution runtime is upper bounded by

$$T_{\text{max}} = \max_{c \in \{\text{subexperiments}\}} T_c \times N_s. \tag{6}$$

where T_c is the quantum excution runtime for the subexperiment c. We determine T_c using resource estimation.

Classical recombination post-processing overhead: This term refers to the amount of classical post processing required to reconstruct results after circuit cutting. The process of post-processing, the contraction of two subcircuits, is equivalent to a pairwise tensor contraction. The memory requirement for such post-processing arises from two main components [14]:

1) Storage of input subcircuit tensors: The tensor for each subcircuit has dimensions $4^n \times 2^N$ where n is the number of cuts and N is the total number of qubits in the original circuit.
2) Storage of intermediate tensors: These arise from products computed at each contraction step.

The compute cost is determined by the number of floating-point multiplications (FLOPs) in each contraction. In [14] the authors have reduced the number of multiplications one needs to do by contracting the subcircuit sequentially instead of doing the contraction at one go, removing the redundancy of those unaffected subcircuits for wire cuts. Ref. [26] explores similar tensor contraction optimizations for gate cuts. However, both the memory and compute requirements scale with the number of cuts per subcircuit in each contraction step, FLOPs required $\sim O\left(4^n\right)$.

V. Resource Estimation and Optimization for Circuit Cutting

The input to our toolflow in Figure 2 is the circuit for a quantum application and the required error ϵ. We cut this circuit into a number of subcircuits. Logical circuit cutting requirements (such as Q_{max}, T_{max}) that were described in the last section are translated into physical resource estimates using a resource estimator and resource optimization techniques.

A. Circuit Cutting Method

We require a method that can partition a circuit into smaller subcircuits. Several past works have explored cutting techniques that reduce the sampling overhead. Optimization based techniques such as CutQC [13] do not scale to the problem sizes that we use, therefore we require heuristics. We use IBM's Qiskit circuit cutting add-on tool [23]. This tool implements gate and wire cuts, with Dijkstra's best-first search. This tool allows us to specify the *maximum number of qubits per subcircuit*, to control the number and sizes of subcircuits. Although a heuristic, our theoretical analysis of cut counts of applications in Section VII-C shows that this tool produces cuts that are near-optimal, justifying its use for benchmarking.

B. Resource Estimation of Subcircuits

For each subcircuit that is generated from the cutting, we wish to estimate the number of physical qubits required to run it and the expected runtime on a quantum processor. To accomplish this, we require 1) a resource estimation tool and models for a fault-tolerant quantum stack and 2) the error budget with which each subcircuit must be executed.

Several resource estimation tools have been developed to provide estimates with different levels of granularity [22], [34]–[36]. Among these tools, we use Microsoft's Azure Quantum Resource Estimator (AQRE) [22] since it works with real algorithm implementations, models the full-stack of a fault-tolerant system and provides state-of-the-art estimates.

For an input circuit, AQRE counts the number of logical qubits, logical operations, and depth determines the required number of magic states. Based on the required error of the circuit, it computes the desired logical error rates. We model a superconducting qubit system with surface code error correction, using the physical qubit and error correction models provided in AQRE. AQRE converts the logical estimates into physical estimates by applying the resource models for error correction and distillation. It outputs the number of physical qubits and required quantum excution runtime.

To determine the total resources needed for a collection of subcircuits, we assume the space-efficient execution model described in the previous section. Therefore, the number of physical qubits is determined as the maximum number of physical qubits needed for any subcircuit. That is, the total qubit count is driven by the peak qubit demand of individual subcircuits rather than their cumulative requirements. The total runtime is obtained by equation 6, where T_c is the physical runtime estimated from AQRE.

C. Error budget and Distillation Optimizations

Error budget: To set the error budget with which each subcircuit must execute, a naive option is set it as ϵ/k where k is the number of subcircuits. However, when the maximum number of qubits per circuit is high, we may have subcircuits where qubit counts are highly imbalanced. For example, a 100 qubit circuit could be cut into two subcircuits with 90 and 10 qubits each. In this setting, the error budget can be partitioned such that the 90 qubit circuit can use more error budget. This allows the 90 qubit circuit to run with a lower QEC code distance, since code distance is inversely proportional to the available error margin. A surface code logical qubit with distance d uses $2d^2$ physical qubits. Therefore, this optimization reduces physical qubit usage, further favoring circuit cutting resource estimates. This optimization also reduces runtime of the largest subcircuits since the logical cycle time of the surface code is linearly proportional to the code distance. We implement this optimization by proportionally dividing the error budget across subcircuits, in the ratio of their qubit counts. Across our benchmarks, we observed 0.5%-10% reductions in the physical qubit counts compared to equal error budget splitting.

Distillation factories: The number of distillation factories determines the rate at which magic states are produced and supplied for running non-Clifford gates. Higher values accelerate magic state production, thereby reducing runtime but increasing the demand for physical qubits. AQRE optimizes runtime, therefore it uses a large number of distillation factories to achieve minimum runtime. Experimenting with different values for number of factories, from a baseline of one factory, we observed a gradual runtime reduction of five fold from 167% down to 33% as the factories increased, where the percentages were both compared to the baseline runtime. Here we only consider a single instance's execution time, T_c of each of the subcircuits c compared to the original circuit's run time, $\frac{\sum T_c}{T_{\text{basline}}}$) However, we also found that the physical qubit requirements almost doubled at a T factory value of count for Hamiltonian simulation, increasing from 56% to 87%, compared to the baseline. Since this does not favor circuit cutting, we use a small number of factories in our experiments, from one to five, depending on the benchmark. Further runtime improvements from increasing distillation are at best one order of magnitude; our experiments show that this is not sufficient to make circuit cutting competitive for large benchmarks.

TABLE II: Summary of the benchmarks we used in this work. The middle column illustrates the logical qubit requirement for each algorithm depending on the problem size.

Algorithm (problem size)	Logical qubits	Logical depth
Heisenberg $D = \{3, 4, 5, 6\}$	$N = D \times D$	$10^{4.7}, \cdots, 10^{5.1}$
Ising $D = \{3, 4, 5, 6\}$	$N = D \times D$	$10^{4.2}, \cdots, 10^{5.2}$
Fermi-Hubbard $D = \{2, 3, 4\}$	$N = 2 \times D \times D$	$10^{4.1}, \cdots, 10^{4.8}$
QFT $N = \{5, \cdots, 60\}$	N	$10^{2.3}, \cdots, 10^{4.0}$
FABLE $2^N, N = \{2, 3, 4, 5\}$	$N + 1$	$10^{2.4}, \cdots, 10^{4.3}$
QPE $N = \{2, 4, 6, 8\}$	$2 \times N$	$10^{2.2}, \cdots, 10^{4.7}$
QAOA $N = \{10, \cdots, 100\}$	N	$10^{2.9}, \cdots, 10^{3.5}$
Random $N = \{5, \cdots, 20\}$	N	$10^{2.7}, \cdots, 10^{4.4}$

VI. Experimental Setup

A. Benchmarks

Table II summarizes our benchmarks are relevant for practical use cases of QC systems. Hamiltonian simulation with Ising, Heisenberg, Fermi-Hubbard models are expected to be one of the first scientific applications where we see practical quantum advantage over classical computing [4], [22]. QFT is a core component of Shor's factoring algorithm [37]. Quantum Phase Estimation and block encoding are important for quantum chemistry computations. Although quantum advantage prospects are not theoretically clear for QAOA, we include it in our benchmarking. While prior works have studied circuit cutting on some of these benchmarks, we choose benchmark instances that reflect practical computations. For example, for Hamiltonian simulation, we choose sufficient Trotter steps to allow highly accurate simulations, based on parameters from [4], [22]. In contrast, [26] uses one or two Trotter steps, which is insufficient for practically useful results from the benchmark. In several cases (examples: QFT beyond 60 qubits, Ising models with 10x10 qubits with 80 Trotter steps), IBM's cutting tool fails to determine cuts because of large circuit size. In these cases, our benchmark includes the largest sizes we could cut. We will denote N as the total number of qubits and m as the maximum number of qubits allowed in a subcircuit.

B. Implementations and tools

Benchmark implementation and cutting: Our benchmarks were implemented in IBM Qiskit [38] with Python (version 3.12.5). Circuit cutting and manipulation were performed using the Qiskit add-on `circuit-cutting` (version 0.9.0). The circuits developed for Hamiltonian simulations utilize native two-qubit gate sets, including the R_{zz}, R_{xx}, R_{yy} for entangling operations. Additionally, for the QFT circuits, controlled-phase gate is used, while other general circuits relied on controlled-not gate for entanglement. The chosen gate set for each circuit was selected to reflect common implementations and could be further improved to facilitate the overhead reduction.

Resource Estimation: For resource estimation, we used Microsoft's Azure Quantum Resource Estimator (AQRE) with server access. We assumed a superconducting qubit, with a gate speed of 50ns, readout latency of 100ns and gate and readout fidelity of 10^{-4} [22]. We use surface code as the choice of QEC and 15-to-1 distillation factories. These choices are available as models in AQRE. We also experimented with trapped ion-like configurations. While it provides different physical error rates as well as the overall physical requirements, our experiments found that such a choice does not make much difference since the total number of cuts is too high to be compromised by the hardware choice. We will not discuss this in our main results.

Metrics: We assume that circuit cutting is implemented with one quantum computer and subcircuits are executed sequentially on that device. For the resource estimation on the reduction of number of physical qubits, we will use the metric, Percent Reduction $= 100\% \times (N^{\text{phy}}{}_{\text{baseline}} - \max\{N^{\text{phy}}_{c_1} \dots N^{\text{phy}}_{c_{N_{\text{pool}}}}\})/N^{\text{phy}}{}_{\text{baseline}}$, where $N^{\text{phy}}_{\text{baseline}}$ is the original circuit's physical qubits requirement and $N^{\text{phy}}_{c_i}, c_i \in C$ is the number of physical qubits count for each subexperiments c_i in the pool C. Other metrics that we are report are the number of cuts obtained after circuit cutting and the quantum excution runtime after circuit cutting computed with equation 6.

VII. Results

A. Number of cuts vs. physical qubit reduction

How to Interpret Figure 3: The x-axis represents the parameter *Max number of qubits per subcircuit*, which determines the maximum number of qubits that any subcircuit can contain during the cut-searching stage. For example, a value of 40% in a 9-qubit quantum algorithm implies that the maximum number of qubits per subcircuit is bounded by $9 \times 40\% \approx 4$. The y-axis represents either the number of cuts required for the circuits or the reduction in physical qubits, as defined in the experimental setup. Finally, the legend in each plot provides information about the problem size. Specifically, for block-encoding benchmarks, 2^N indicates that the goal is to encode a unitary of size $2^N \times 2^N$. For QPE benchmarks, the problem size corresponds to the size of the phase of the unitary being estimated.

Figure 3 (i-p) shows the number of cuts required. These plots contain two variables, the problem size and the maximum number of qubits allowed in a subcircuit (maximal circuit width after applying cuts). There are three clear trends. First, as the problem size, that is, the number of qubits and circuit depth increases, the number of cuts increases. This is because larger circuits require more partitioning to obtain subcircuit qubit counts that are within the limits imposed by the maximum qubits per subcircuit parameter. Second, as the maximum qubits per subcircuit increase, the number of required cuts reduces because a small number of large subcirucits are sufficient to partition the original circuit.

Third, the number of cuts is influenced by algorithm structure. Ising, Heisenberg and Fermi-Hubbard models typically require over a thousand cuts since they have nearest-neighbor gates and large depth. For these circuits, gate cuts are primarily used instead of wire cuts because of the highly entangling structure of the circuit. QFT and QPE circuits also require a

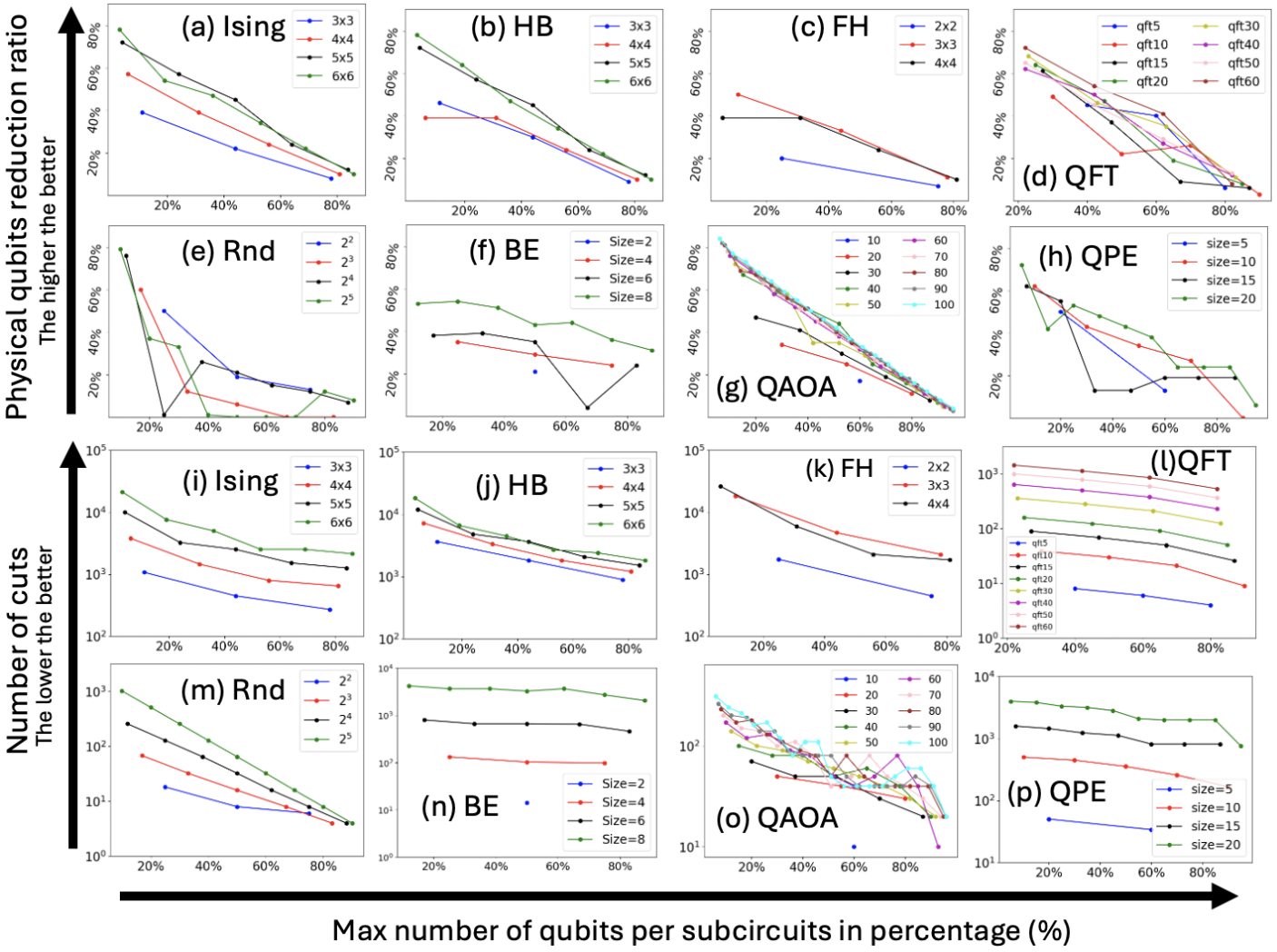

Fig. 3: Data plots for our experiments. The x-axis denotes the maximum number of qubits per subcircuit. The first two rows summarizes the results for the physical qubit reduction and total number of cuts is summarized in row 3 and row 4 for each of the benchmarks. In the inset on the corner in each plot, the total number of logical qubits are displayed.

lot of cuts since they have all-to-all gate patterns. Random circuits exhibit comparable behavior to Hamiltonian simulations, which is expected given their similar entangling structure.

Figure 3 (a-h) shows the percentage reduction in the number of physical qubits needed after circuit cutting, compared to the baseline which doesn't use circuit cutting. The number of physical qubits reduces across algorithms and problem sizes when circuit cutting is used, with reduction ratios ranging from 5 to 80%. Similar to the trends in number of cuts, the percentage reduction in physical qubit usage improves with problem size. This is because when large problems are broken down into very small subcircuits, we get significant reductions in the number of logical qubits, physical qubits and distillation resources required to implement it fault-tolerantly. When the maximum number of qubits per subcircuit is low, the physical qubit reductions are very high, but with increasing subcircuit size, the reductions diminsh. When subcircuits' physical resources are about 80-90% of the original problem size, circuit cutting gives approximately a 10% qubit reduction.

This situation presents a trade-off: achieving a greater reduction in physical qubits necessitates more cuts, thus increasing the overhead, while fewer cuts allow a tolerable overhead at the cost of less reduction in qubit usage. This tradeoff is a challenge for circuit cutting as problem sizes, depth and complexity of gate structure increase.

For instance, for an Ising model 5×5, the subcircuits after cuts have at most 6 qubits per subcircuit at a data point of 24%. The number of cuts required is around $10^{3.5}$ with a reduction of physical qubits around 60%. The number of cuts can be decreased by 25% and each subcircuit has at most 11 qubits. This manner decreases the physical qubits benefits by 10%.

B. *Quantum and classical runtime overheads*

Quantum excution runtime overhead: Table III compares the quantum excution runtime overheads of circuit cutting

TABLE III: The quantum excution runtime is derived from resource estimation data. Values marked as 'N/A' indicate that the wallclock excution runtime is too large to evaluate. The numbers following each algorithm name represent the problem size, measured in the number of logical qubits. Baseline quantum excution runtime and physical qubits refers to metrics of the original full circuit using the resource estimator. The *cutting runtime* represents the execution time for the subcircuits after partitioning the circuit. For this analysis, we only consider cuts that divide the circuit approximately in half by the number of qubits. The final column shows the reduction ratio of physical qubits after cutting. Note that the quantum excution runtime after partitioning provides only a lower bound. The factor ε_{rct}, which accounts for reconstruction error is not included in the calculations. Incorporating ε_{rct} would further increase the estimated runtime. The points in the column of physical qubits reduction labelled 'N/A' means the program failed to find a suitable set of cutting locations. The 'N/A' in the last two overhead columns means that the value has passed the boundary of 10^{65} (s) for the quantum excution runtime or 10^{90} (FLOPs) for the classical overhead. Note here the quantum overhead means the same as the quantum execution wall-clock time, but we interchange these terms here for generalization.

Algorithm	Baseline phy. qubits (s)	Baseline runtime (s)	No. cuts	% reduction phys. qubits	quantum overhead (runtime with cutting) (s)	classical overhead (FLOPs)
QFT 10	$10^{4.43}$	0.07	$10^{1.5}$	20%	2070	$10^{5.56}$
QFT 20	$10^{4.69}$	0.41	$10^{1.9}$	30%	5493	$10^{14.0}$
QFT 30	$10^{4.87}$	1.24	$10^{2.3}$	41%	1.3×10^4	$10^{35.1}$
QFT 40	$10^{4.95}$	2.41	$10^{2.5}$	38%	1.7×10^4	$10^{55.7}$
QFT 50	$10^{5.02}$	3.68	$10^{2.7}$	27%	1.7×10^4	$10^{88.3}$
QFT 60	$10^{5.16}$	4.83	$10^{2.9}$	47%	1.7×10^4	N/A
Ising 3×3	$10^{4.53}$	20.64	$10^{2.7}$	45%	1.0×10^9	N/A
Ising 4×4	$10^{4.70}$	40.04	$10^{2.9}$	30%	1.5×10^9	N/A
Ising 5×5	$10^{4.88}$	69.55	$10^{3.2}$	22%	1.6×10^{11}	N/A
Ising 6×6	$10^{4.99}$	103.70	$10^{3.3}$	34%	1.6×10^{11}	N/A
HB 3×3	$10^{4.58}$	53.65	$10^{3.3}$	9%	3.2×10^{11}	N/A
HB 4×4	$10^{4.70}$	105.98	$10^{3.3}$	10%	1.7×10^{10}	N/A
HB 5×5	$10^{4.88}$	203.01	$10^{3.4}$	12%	1.7×10^{10}	N/A
HB 6×6	$10^{4.99}$	303.37	$10^{3.5}$	10%	1.7×10^{10}	N/A
FH 2×2	$10^{4.43}$	85.03	$10^{2.7}$	13%	3.1×10^{11}	N/A
FH 3×3	$10^{4.64}$	290.58	$10^{3.5}$	28%	N/A	N/A
FH 4×4	$10^{4.70}$	581.22	$10^{3.5}$	27%	N/A	N/A
QAOA 10	$10^{4.43}$	0.15	10	20%	8.0×10^7	$10^{4.77}$
QAOA 20	$10^{4.69}$	0.31	40	18%	4.2×10^{31}	$10^{19.1}$
QAOA 30	$10^{4.80}$	0.52	50	20%	3.4×10^{39}	N/A
QAOA 40	$10^{4.95}$	0.60	50	18%	4.2×10^{31}	N/A
QAOA 50	$10^{5.02}$	1.05	60	18%	1.7×10^{63}	N/A
QAOA 60	$10^{5.08}$	1.27	50	12%	3.4×10^{39}	N/A
QAOA 70	$10^{5.14}$	1.50	80	16%	4.2×10^{31}	N/A
QAOA 80	$10^{5.18}$	1.70	70	18%	1.0×10^{39}	N/A
QAOA 90	$10^{5.23}$	1.94	80	20%	1.8×10^{63}	N/A
QAOA 100	$10^{5.26}$	2.30	50	22%	2.2×10^{55}	N/A
QPE 2	$10^{4.17}$	0.01	$10^{1.1}$	21%	4.3×10^7	$10^{3.79}$
QPE 4	$10^{4.39}$	0.10	$10^{1.9}$	29%	1.2×10^{61}	$10^{23.9}$
QPE 6	$10^{4.58}$	0.66	$10^{2.8}$	30%	N/A	N/A
QPE 8	$10^{4.70}$	5.17	$10^{3.5}$	48%	N/A	N/A
Random 5	$10^{4.20}$	0.02	$10^{1.5}$	13%	8.9×10^{16}	$10^{15.1}$
Random 10	$10^{4.49}$	0.26	$10^{1.7}$	18%	N/A	$10^{23.9}$
Random 15	$10^{4.63}$	1.00	$10^{2.6}$	18%	N/A	N/A
Random 20	$10^{4.76}$	2.70	$10^{2.9}$	N/A	N/A	N/A
BE 2	$10^{3.94}$	0.01	$10^{0.8}$	8%	6.8×10^8	$10^{3.01}$
BE 3	$10^{4.17}$	0.04	$10^{1.2}$	14%	4.7×10^{17}	$10^{7.56}$
BE 4	$10^{4.35}$	0.21	$10^{1.7}$	8%	1.4×10^{34}	$10^{23.9}$
BE 5	$10^{4.51}$	1.04	$10^{1.9}$	18%	7.7×10^{65}	$10^{37.9}$

executions with the baseline which does not use circuit cutting. Across benchmarks, quantum excution runtime is several orders of magnitude higher than the baseline. As a comparison, the original circuit of a 6×6 Ising model only need 103 seconds for execution, but after cutting it requires 5000 years. For all benchmarks, except QFT, the excution runtime is extremely high, typically in the range of years, when circuit cutting is used. This is one of the main reasons why circuit

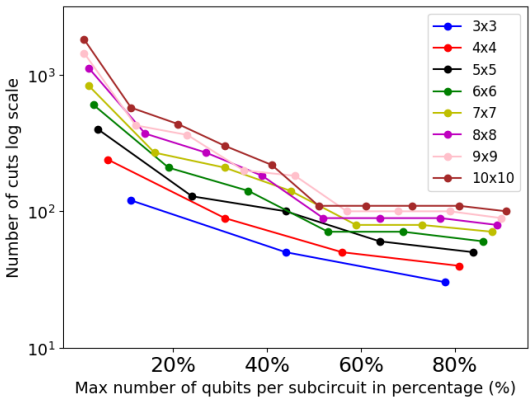

Fig. 4: Single Trotter step 2D Ising model.

cutting is infeasible in practice for fault-tolerant executions. This arises because of the number of cuts required to partition the algorithm and the types of gates used in the algorithm. For circuits like block encoding and random entangled circuits, the native gate is often the CNOT gate which has a large sampling overhead (Table 3). For QFT, the excution runtime is acceptable because QFT's gate type is the controlled phase gate with many small angle rotations which allow a small sampling overhead. However, circuit cutting is still infeasible for QFT, as we discuss next.

Classical post-processing overhead: We consider the classical post-processing overhead, which scales as $O(4^{cuts})$ FLOPs on a classical computer. If we have 100 cuts, this would mean that 10^{60} FLOPs are approximately required, which is beyond the reach of any foreseeable classical supercomputer. Even in the case of QFT which has gate types that are suitable for cutting, the number of cuts is at least 100 for problem sizes beyond 50 qubits that are relevant for quantum advantage experiments. Therefore, across our applications, this exponential classical overhead makes circuit cutting infeasible for practical-scale fault-tolerant executions.

The only benchmark which has acceptable classical overheads is QAOA, with the circuits feasible for cutting up to around 50 qubits. This is closely tied to how the algorithm's circuit structure mirrors the topology of the problem graph. Since QAOA's circuit inherently matches the connectivity and interactions of the graph it operates on, this allows good partitioning if the graph can be clustered well. If the underlying problem graphs are more diverse, it is unclear if circuit cutting will still be feasible. Since QAOA's quantum overheads are very high, we did not explore this further.

Both quantum and classical overheads should be considered while evaluating techniques such as circuit cutting. For certain benchmarks we see that sometimes quantum or classical overheads with circuit cutting are acceptable, but none of our benchmarks show all desirable circuit cutting benefits of reductions in physical qubit usage, low quantum overhead and low classical overhead.

C. Theoretical analysis of number of cuts

To validate the experimental estimates, we developed theoretical models to estimate cut counts for four benchmarks.

Ising, Heisenberg and Fermi-Hubbard models: These Hamiltonian simulations are implemented using a well-known algorithm called Trotterization, an approximation technique that decomposes the time evolution operator e^{-iHt} of a Hamiltonian H into simpler, implementable steps. The Trotter formula approximates the evolution by splitting H into a sum of terms $H = \sum_j H_j$, evolving each term individually, and repeating this sequence over multiple small time steps, N_t. However, the approximation introduces a *Trotter error* (algorithmic error) that scales with the commutators of the Hamiltonian terms. We use the fourth order Trotter approximation based on [22], where the error bound scales as

$$\epsilon_{\text{alg}} = \mathcal{O}\left(\frac{(t\sum_{j<k}\|[H_j, H_k]\|)^{1+p}}{N_t^p}\right), \tag{7}$$

where p is the order of the algorithm. With the scaling relation we can easily invert the expression to find the algorithmic error, depending on the number of Trotter steps.

For a $D \times D$ square lattice, we have D^2 qubits in the grid for an Ising model. When we divide the $D \times D$ grid into subcircuits, each small circuit contains no more than m qubits. The number of subcircuits, $N_{\text{subcircuits}}$ is approximately:

$$N_{\text{subcircuits}} = \frac{D^2}{m}. \tag{8}$$

The boundary of each subcircuit has qubits that interact with qubits in adjacent subcircuits. If a qubit is located near the boundary of a subcircuit, its interactions with neighbouring qubits in other subcircuits will need to be cut. The number of interactions that span across subcircuits depends on the size and shape of the subcircuits. Assuming a square lattice model, the number of boundary qubits scales roughly as $\sqrt{m}$

$$\text{Boundary qubits} \approx \frac{D^2}{m} \cdot \sqrt{m}. \tag{9}$$

Therefore, the minimum number of cuts needed to divide the 2D Heisenberg model circuit into subcircuits is:

$$\text{Number of cuts} \approx 4 \cdot \text{number of Trotter steps} \cdot \frac{D^2}{\sqrt{m}}. \tag{10}$$

However, for the Heisenberg's model, each qubit interacts with its nearest neighbours through 3 types of controlled-phase gate, $R_{zz}(\theta), R_{xx}(\theta), R_{yy}(\theta)$, in the Trotterized evolution step. So, the constant scaling factor is 12 for the number of cuts comparing to 3 for in Ising model, where only R_{zz} interation is considered. For the Fermi-Hubbard model, the scaling changes again because of the incorporation of both the spin-up and spin-down states, resulting in a total number of qubits of $2 \times D \times D$ in a $D \times D$ lattice.

Our aim to was scale up problems like Ising up to 10×10 — practical sizes, but we found that cutting tools failed beyond 6×6. For example, in Figure 4, the number of cuts is calculated

for each lattice size up to 10×10, considering a **single** Trotter step. The results suggest that post-processing might be feasible due to the manageable number of cuts observed in this case, which aligns with the approach commonly taken by researchers in the literature. However, when scaling to practical scenarios involving many Trotter steps, the number of cuts increases linearly with the number of steps, rendering the problem impractical. A preliminary analysis shows that the average number of cuts across all percentages of subcircuit qubit classifications ranges from $10^{3.2}$ to $10^{4.1}$. Experimental results are consistent with this estimate, yielding a range of values from $10^{3.3}$ to $10^{3.8}$. For the 6×6 lattice model, the predictions from Eqn. (10) are confirmed: as m increases, the rate of reduction in the number of cuts gradually decreases.

QFT: QFT has a structure where each qubit has controlled phase gates with all preceding qubits. By partitioning the circuit into two parts, we can estimate the number of cuts required. For each pair of adjacent subcircuits, there will be controlled-phase gates that need to be cut. The number of such cuts can thus be approximated by:

$$\text{Number of cuts} \approx \sum_{k=1}^{\lceil \frac{n}{m} \rceil - 1} (m \cdot (n - km)), \qquad (11)$$

where n is the number of qubits in the original circuit and m is the size of one of the subcircuits. Intuitively, as the number of cuts increases as n increases for a fixed m since more qubits need to be distributed across subcircuits, leading to more connections between qubits in different subcircuits. As m increases, fewer cuts are required to partition the circuit, because more qubits can remain within a single subcircuit. Experimental data demonstrate an average increase in the number of cuts from $10^{0.75}$ to $10^{3.5}$ as the number of qubits increases from 5 to 60. This trend aligns with the approximation from Eqn. (11), which predicts values ranging from $10^{0.84}$ to $10^{3.5}$, with minimal deviation from the observed results.

These models support our findings by showing that the number of cuts scales with problem size, which translates to increasing exponential classical overheads as we scale up.

These theoretical models are derived from "natural" cuts of an algorithm. For example, in the case of Hamiltonian simulations, the cuts we consider above are the natural partitions of a lattice into smaller lattices. We do not prove that they are theoretically optimal, but we do not expect significant reductions in cut counts for these algorithms, even with advanced graph partitioning strategies. Similarly, GPU acceleration or tensor contraction optimizations such as [12] are effective for small problem sizes, but cannot cope with the exponential compute growth as problem sizes increase.

VIII. Related Work

Types of cut: A variety of circuit cutting techniques have been developed in addition to the qubit and gate cuts [24], [28]. These include cutting techniques that are dependent on the type of gate that is cut [23] and the ability to use classical communication to reduce overhead. [7], [9], [10], [24] assume classical communication and [6], [13], [20] assume there is no classical communication. Local operations (LO) refer to the scenario where the two computers can only perform operations locally and operate entirely independently. In contrast, Local Operations and Classical Communication (LOCC) allow for two-way classical communication between quantum computers, potentially optimizing overhead by improving scaling when multiple places are cut at a single time. However, LOCC provides no advantage when quasi-probabilistically simulating a single gate, as highlighted in [7].

Optimization strategies: Several works have explored how to mitigate the exponential increase in the number of required measurements, with respect to the number of cut locations. Recent strategies include using randomized measurements [9], [39], identifying optimal cut points through golden circuit cutting techniques [40], [41], and applying variational optimization [42]. For the LOCC settings, further studies [9], [11], [24] have reduced the required number of ancilla qubits by using either a random measurement basis or optimized decomposition techniques.

Tensor network methods: Circuit cutting methods are closely related to tensor networks (TN). TN represent quantum states in a compressed form and provide a framework for characterizing entanglement patterns. Prior research [12] introduced the concept of hybrid TN to reduce the overhead of classical post-processing. Adaptive circuit cutting (ACK) [15] demonstrates improvements in the overhead by representing circuits in linear depth [43] with TN.

IX. Conclusions

Circuit cutting is a technique for orchestrating the execution of large quantum programs on small quantum computers.

Since this technique is seen as a promising way of using future distributed quantum hardware [16], our work evaluates whether cutting techniques can indeed scale to the needs of practical applications running on fault-tolerant hardware. Unfortunately, we find that the classical and quantum execution runtime overheads of circuit cutting are too high for practically-relevant benchmarks. Our work aligns with a recent theoretical argument [44] that shows that the sampling overhead of circuit-knitting is exponentially lower bounded by the entanglement cost.

Our results show that classical resources are insufficient for combining multiple quantum processors in a scalable way at the circuit level. However, benefits may arise from constructing a quantum network [18], where individual chips operate using separable algorithms, connected either through circuit knitting or entangled Bell pairs.

Circuit cutting remains a valuable tool for small-scale quantum algorithms, short depth circuits or Hamiltonian simulations with short evolution times. In this setting, further research is needed to reduce classical post-processing costs, such as choosing the optimal gate set for cutting, and mitigate the exponential scaling limitations identified in both practical benchmarks and theoretical analysis. If applications are co-designed with circuit cutting, with highly clustered gate struc-

tures that match the requirements of cutting, circuit cutting could be scaled up in a resource-efficient manner.

REFERENCES

[1] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental quantum cryptography," *Journal of cryptology*, vol. 5, pp. 3–28, 1992.

[2] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, "Encoding electronic spectra in quantum circuits with linear t complexity," *Physical Review X*, vol. 8, no. 4, p. 041015, 2018.

[3] D. Barral, F. J. Cardama, G. Díaz, D. Faílde, I. F. Llovo, M. M. Juane, J. Vázquez-Pérez, J. Villasuso, C. Piñeiro, N. Costas, *et al.*, "Review of distributed quantum computing. from single qpu to high performance quantum computing," *arXiv preprint arXiv:2404.01265*, 2024.

[4] A. J. Daley, I. Bloch, C. Kokail, S. Flannigan, N. Pearson, M. Troyer, and P. Zoller, "Practical quantum advantage in quantum simulation," *Nature*, vol. 607, no. 7920, pp. 667–676, 2022.

[5] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, "Even more efficient quantum computations of chemistry through tensor hypercontraction," *PRX Quantum*, vol. 2, no. 3, p. 030305, 2021.

[6] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, "Simulating large quantum circuits on a small quantum computer," *Physical review letters*, vol. 125, no. 15, p. 150504, 2020.

[7] C. Piveteau and D. Sutter, "Circuit knitting with classical communication," *IEEE Trans. on Information Theory*, pp. 1–1, 2023.

[8] C. Piveteau, D. Sutter, and S. Woerner, "Quasiprobability decompositions with reduced sampling overhead," *npj Quantum Information*, vol. 8, no. 1, p. 12, 2022.

[9] A. Lowe, M. Medvidović, A. Hayes, L. J. O'Riordan, T. R. Bromley, J. M. Arrazola, and N. Killoran, "Fast quantum circuit cutting with randomized measurements," *Quantum*, vol. 7, p. 934, 3 2023.

[10] L. Brenner, C. Piveteau, and D. Sutter, "Optimal wire cutting with classical communication," *arXiv preprint*, 2 2023.

[11] E. Pednault, "An alternative approach to optimal wire cutting without ancilla qubits," *arXiv preprint arXiv:2303.08287*, 2023.

[12] N. Tornow, C. B. Mendl, and P. Bhatotia, "Quantum-classical computing via tensor networks," *arXiv preprint arXiv:2410.15080*, 2024.

[13] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, "CutQC: Using small quantum computers for large quantum circuit evaluations," in *ASPLOS'21: Proceedings of the 26th ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, pp. 473–486, ACM, 2021.

[14] W. Tang and M. Martonosi, "ScaleQC: A scalable framework for hybrid computation on quantum and classical processors," *arXiv preprint*, 2022.

[15] M. Mohseni, A. Scherer, K. G. Johnson, O. Wertheim, M. Otten, N. A. Aadit, Y. Alexeev, K. M. Bresniker, K. Y. Camsari, B. Chapman, S. Chatterjee, G. A. Dagnew, A. Esposito, F. Fahim, M. Fiorentino, A. Gajjar, A. Khalid, X. Kong, B. Kulchytskyy, E. Kyoseva, R. Li, P. A. Lott, I. L. Markov, R. F. McDermott, G. Pedretti, P. Rao, E. Rieffel, A. Silva, J. Sorebo, P. Spentzouris, Z. Steiner, B. Torosov, D. Venturelli, R. J. Visser, Z. Webb, X. Zhan, Y. Cohen, P. Ronagh, A. Ho, R. G. Beausoleil, and J. M. Martinis, "How to build a quantum supercomputer: Scaling from hundreds to millions of qubits," *arXiv preprint arXiv:2411.10406*, 2024.

[16] "IBM Quantum Roadmap." https://ibm.com/quantum/assets/IBM_Quantum_Developmen_&_Innovation_Roadmap_Explainer_2024-Update.pdf. Accessed: 2024-11-21.

[17] https://www.hpcwire.com/2022/12/05/quantum-circuit-cutting-fills-a-gaping-quantum-computing-hole/.

[18] A. Carrera Vazquez, C. Tornow, D. Ristè, S. Woerner, M. Takita, and D. J. Egger, "Combining quantum processors with real-time classical communication," *Nature*, 2024.

[19] W. Tang and M. Martonosi, "Tensorqc: Towards scalable distributed quantum computing via tensor networks," *arXiv preprint arXiv:2502.03445*, 2025.

[20] K. Mitarai and K. Fujii, "Constructing a virtual two-qubit gate by sampling single-qubit operations," *New Journal of Physics*, vol. 23, no. 2, p. 023021, 2021.

[21] C. Ufrecht, L. S. Herzog, D. D. Scherer, M. Periyasamy, S. Rietsch, A. Plinge, and C. Mutschler, "Optimal joint cutting of two-qubit rotation gates," *Physical Review A*, vol. 109, no. 5, p. 052440, 2024.

[22] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo, "Assessing requirements to scale to practical quantum advantage," *arXiv preprint arXiv:2211.07629*, 2022.

[23] A. M. Brańczyk, A. Carrera Vazquez, D. J. Egger, B. Fuller, J. Gacon, J. R. Garrison, J. R. Glick, C. Johnson, S. Joshi, E. Pednault, C. D. Pemmaraju, P. Rivero, I. Shehzad, and S. Woerner, "Qiskit addon: circuit cutting." https://github.com/Qiskit/qiskit-addon-cutting, 2024.

[24] H. Harada, K. Wada, and N. Yamamoto, "Doubly optimal parallel wire cutting without ancilla qubits," *arXiv preprint arXiv:2303.07340*, 2023.

[25] S. Brandhofer, I. Polian, and K. Krsulich, "Optimal partitioning of quantum circuits using gate cuts and wire cuts," *IEEE Transactions on Quantum Engineering*, 2023.

[26] X. Ren, M. Zhang, and A. Barbalace, "A hardware-aware gate cutting framework for practical quantum circuit knitting," *arXiv preprint arXiv:2409.03870*, 2024.

[27] D. Litinski, "A game of surface codes: Large-scale quantum computing with lattice surgery," *Quantum*, vol. 3, p. 128, 2019.

[28] C. Ufrecht, M. Periyasamy, S. Rietsch, D. D. Scherer, A. Plinge, and C. Mutschler, "Cutting multi-control quantum gates with zx calculus," *Quantum*, vol. 7, p. 1147, 2023.

[29] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of physics*, vol. 303, no. 1, pp. 2–30, 2003.

[30] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 86, no. 3, p. 032324, 2012.

[31] K. Kraus, "General state changes in quantum theory," *Annals of Physics*, vol. 64, no. 2, pp. 311–335, 1971.

[32] S. C. Marshall, J. Tura, and V. Dunjko, "All this for one qubit? bounds on local circuit cutting schemes," *arXiv preprint*, 3 2023.

[33] S. Bravyi, G. Smith, and J. A. Smolin, "Trading classical and quantum computational resources," *Physical Review X*, vol. 6, no. 2, p. 021043, 2016.

[34] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado, "TFermion: A non-Clifford gate cost assessment library of quantum phase estimation algorithms for quantum chemistry," *Quantum*, vol. 6, p. 768, July 2022.

[35] J. R. McClean, N. C. Rubin, K. J. Sung, I. D. Kivlichan, X. Bonet-Monroig, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, *et al.*, "Openfermion: the electronic structure package for quantum computers," *Quantum Science and Technology*, vol. 5, no. 3, p. 034014, 2020.

[36] M. P. Harrigan, T. Khattar, C. Yuan, A. Peduri, N. Yosri, F. D. Malone, R. Babbush, and N. C. Rubin, "Expressing and analyzing quantum algorithms with qualtran," *arXiv preprint arXiv:2409.04643*, 2024.

[37] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, p. 1484–1509, Oct. 1997.

[38] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, *et al.*, "Quantum computing with qiskit," *arXiv preprint arXiv:2405.08810*, 2024.

[39] D. T. Chen, Z. H. Saleem, and M. A. Perlin, "Quantum divide and conquer for classical shadows," *arXiv preprint arXiv:2212.00761*, 2022.

[40] D. T. Chen, E. H. Hansen, X. Li, V. Kulkarni, V. Chaudhary, B. Ren, Q. Guan, S. Kuppannagari, J. Liu, and S. Xu, "Efficient quantum circuit cutting by neglecting basis elements," *arXiv preprint arXiv:2304.04093*, 2023.

[41] D. T. Chen, E. H. Hansen, X. Li, A. Orenstein, V. Kulkarni, V. Chaudhary, Q. Guan, J. Liu, Y. Zhang, and S. Xu, "Online detection of golden circuit cutting points," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1, pp. 26–31, IEEE, 2023.

[42] G. Uchehara, T. M. Aamodt, and O. Di Matteo, "Rotation-inspired circuit cut optimization," *arXiv preprint arXiv:2211.07358*, 2022.

[43] S.-H. Lin, R. Dilip, A. G. Green, A. Smith, and F. Pollmann, "Real- and imaginary-time evolution with compressed quantum circuits," *PRX Quantum*, vol. 2, no. 1, p. 010342, 2021.

[44] M. Jing, C. Zhu, and X. Wang, "Circuit knitting faces exponential sampling overhead scaling bounded by entanglement cost," *arXiv preprint arXiv:2404.03619*, 2024.