Parallelizing Simulations of Large Quantum Circuits

Michael A. Perlin
JILA, NIST and University of Colorado at Boulder
Boulder, Colorado, USA
mika.perlin@gmail.com

Teague Tomesh
Princeton University
Princeton, New Jersey, USA
ttomesh@princeton.edu

Bradley Pearlman
JILA, NIST and University of Colorado at Boulder
Boulder, Colorado, USA
brpe4351@colorado.edu

Wei Tang
Princeton University
Princeton, New Jersey, USA
weit@princeton.edu

Yuri Alexeev
Argonne National Laboratory
Lemont, Illinois, USA
yuri@alcf.anl.gov

Martin Suchara
Argonne National Laboratory
Lemont, Illinois, USA
msuchara@anl.gov

ABSTRACT

We present a parallelization scheme for classical simulations of quantum circuits. Our scheme is based on a recent method to "cut" large quantum circuits into smaller sub-circuits that can be simulated independently, and whose simulation results can in turn be re-combined to infer the output of the original circuit. The exponentially smaller classical computing resources needed to simulate smaller circuits are counterbalanced by exponential overhead in terms of classical post-processing costs. We discuss how this overhead can be massively parallelized to reduce classical computing costs.

KEYWORDS

Quantum computing, quantum simulation, circuit cutting, distributed computing, parallel algorithms, HPC

1 INTRODUCTION

Quantum computers can solve certain computational tasks with exponential speedup over their classical counterparts [1]. It should therefore come as no surprise that classical simulations of quantum computations incur exponential cost: the best known techniques for evaluating general quantum circuits with N qubits have $O(2^N)$ classical memory and runtime requirements. Nonetheless, such classical simulations play a major role in developing new quantum algorithms and understanding the behavior of quantum hardware.

Near-term quantum computing devices will be most suitable for executing variational quantum eigensover (VQE) [10, 11] algorithms that solve approximate optimization problems, such as QAOA [4, 5]. The hardware-efficient ansatz (HWEA) is a particular family of VQE circuits that was designed for compatability with near-term quantum hardware [6].

In this work, we implement a recently proposed method to "cut" quantum circuits that are too large to evaluate on available hardware. The process of cutting yields smaller, more tractable circuit fragments, but at the cost of exponentially large classical post-processing overhead [9]. We apply circuit cutting to classical simulations of HWEA circuits that are highly amenable to circuit cutting techniques, and discuss strategies to parallelize the associated post-processing overheads.

SC19, November 17–22, 2019, Denver, CO, USA
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

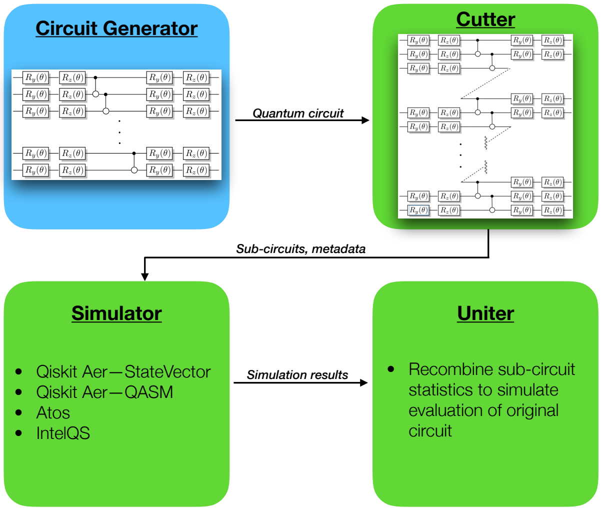

Figure 1: Overview of our Circuit Cutting software. Blue blocks show serial work and green blocks show highly parallel tasks.

2 METHODS

The main contribution of this work is a software tool that implements a variant of the circuit cutting algorithm first described in ref. [9]. Our tool consists four separate parts, which we refer to as the circuit generator, cutter, simulator, and uniter; see Figure 1.

Circuit Generator. In this work, we primarily focus on cutting and simulating HWEA circuits [6]. These circuits are characterized by layers of two-qubit gates sandwiched by single-qubit gates that are parameterized by angles; see Figure 1 for a sketch of a single-layer HWEA circuit. Although applications of HWEA circuits involve classical optimization over single-qubit gate parameters, in this work we are only concerned with application-agnostic features of HWEA circuits, and therefore assign random angles to these gates. The local connectivity of HWEA circuits makes them easy to partition into multiple fragments using only a few cuts, in contrast to e.g. the quantum fourier transform circuit with all-to-all qubit connectivity [8]. As a first step in our software pipeline, we generate a Qiskit `QuantumCircuit` object [2] representing a HWEA circuit with a chosen number of qubits and two-qubit gate layers.

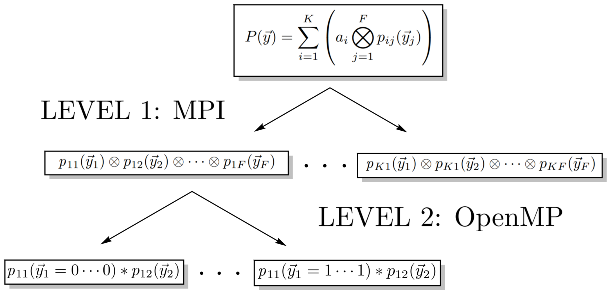
$P(\vec{y}) = \sum_{i=1}^{K} \left(a_i \bigotimes_{j=1}^{F} p_{ij}(\vec{y}_j) \right)$
LEVEL 1: MPI
$p_{11}(\vec{y}_1) \otimes p_{12}(\vec{y}_2) \otimes \cdots \otimes p_{1F}(\vec{y}_F)$
. . .
$p_{K1}(\vec{y}_1) \otimes p_{K1}(\vec{y}_2) \otimes \cdots \otimes p_{KF}(\vec{y}_F)$
LEVEL 2: OpenMP
$p_{11}(\vec{y}_1 = 0 \cdots 0) * p_{12}(\vec{y}_2)$
. . .
$p_{11}(\vec{y}_1 = 1 \cdots 1) * p_{12}(\vec{y}_2)$

Figure 2: Two-level parallelization of the uniter, which must evaluate a sum of outer products of vectors acquired from circuit fragment simulations. Here P is the probability distribution over measurement outcomes $\vec{y}$ on the original circuit; $\vec{y}_j$ is the restriction of $\vec{y}$ to the output qubits of fragment j; p_{ij} is a probability distribution acquired from fragment simulations; and a_i is a scalar coefficient. The sum can be parallelized using MPI by assigning each term its own task (level 1), and each individual task computes an outer product that can be further parallelized using OpenMP and standard numerical techniques (level 2).

Cutter. After generating a HWEA circuit, we use a Monte Carlo variant of Karger's `MIN-CUT` algorithm [7] to search for a set of cuts that partition the circuit into multiple fragments. We place constraints on our search to enforce a maximum number of qubits per fragment, and ensure that the resulting fragments have a balanced numer of qubits. While the HWEA circuit is straightforward to cut by hand, a general-purpose cutting algorithm such as `MIN-CUT` will be necessary for future efforts to cut a variety of circuits, such as the quantum supremacy circuits in ref. [3]. After the search algorithm identifies a satisfactory set of cuts, our cutter returns a Qiskit `QuantumCircuit` object [2] for each fragment, as well as metadata describing how these fragments fit together to recover the original circuit. The Monte Carlo nature of the cut-searching algorithm makes it straightforward to parallelize by running a large number of independent search tasks.

Simulator. Once a circuit has been cut into several fragments, we must simulate several variants (i.e. slightly modified versions) of each fragment. These variants correspond to different sets of measurement bases and state preparations at the locations of cuts, and are necessary to classically reproduce the quantum correlations present in the original circuit. Although the number of variants for each fragment is exponential in the number of incident cuts ($4^{C_{\text{in}}} \times 3^{C_{\text{out}}}$, where C_{in} and C_{out} are respectively the number of cuts at the input/output of the fragment), all variants of all fragments can be simulated independently at this stage of our pipeline. The simulation of these variants is therefore trivially parallelizable by assigning each variant a separate task. Individual variant (circuit) simulations are further natively parallelized in Qiskit's `Aer` simulator to speed up the computation of fragment outputs.

Uniter. Once all of the simulation data for circuit fragments has been collected, this data can be used to reconstruct the full probability distribution over measurement outcomes on the original, pre-cut circuit. For a circuit with C cuts and F fragments, reconstruction requires summing over $K \equiv 4^C$ terms that are each an F-fold outer (tensor) product of vectors acquired from fragment simulations. This reconstruction naturally admits two-level parallelization with MPI (level 1) and OpenMP (level 2), as sketched in Fig. 2. The sum of outer products is parallelizable with MPI, as each outer product can be computed independently before adding them all up. Furthermore, each outer product is an instance of the standard BLAS level 3 routine `DGEMM`, which is parallelized using OpenMP threads in standard numerical libraries such as Python's NumPy and SciPy.

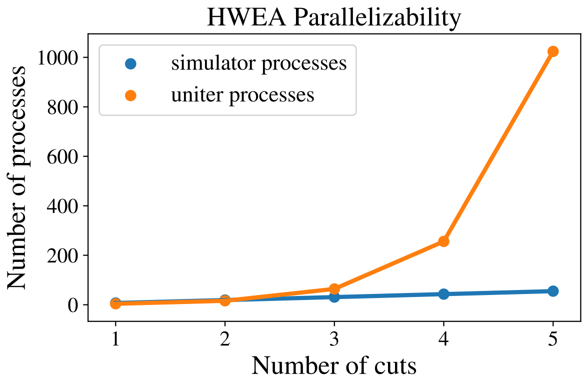

Figure 3: The number of parallelizable computations in the simulator (uniter) scales linearly (exponentially) with the number of cuts to a single-layer HWEA circuit. The linear reduction of fragment size with respect to the number of cuts also implies an exponential reduction in classical resource requirements for each simulator task.

3 RESULTS & CONCLUSIONS

The runtime for naïve classical simulations of a quantum circuit scales exponentially with its number of qubits. Circuit cutting addresses some of this exponential cost by reducing the maximum number of qubits that need to be simulated in an individual quantum circuit. The exponential overhead of classical simulation is then offloaded from a circuit simulator to a fragment uniter, which must do an exponential amount of work to faithfully reproduce the outputs of the original, pre-cut circuit. By pushing exponential overhead to the uniter, circuit cutting enables data and thread level parallelization of this exponential cost through MPI and OpenMP (see Fig. 3).

In this way, our circuit cutting tool enables the evaluation of large quantum circuits by partitioning them into smaller, more manageable pieces. Efficient exploitation of the parallelism present in the uniter is central to the success of our circuit cutting tool.

ACKNOWLEDGMENTS

We thank Jeffrey Larson, James Osborn, and Ruslan Shaydulin for technical contributions to this work. This material is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under contract DE-AC02-06CH11357.

REFERENCES

[1] 2019. Quantum Algorithm Zoo. https://quantumalgorithmzoo.org/.

[2] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, et al. 2019. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2562110

[3] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, et al. 2018. Characterizing quantum supremacy in near-term devices. *Nature Physics* 14, 6 (2018), 595.

[4] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]* (Nov. 2014).

[5] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, et al. 2019. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms* 12, 2 (Feb. 2019), 34. https://doi.org/10/gfx8wp

[6] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, et al. 2017. Hardware-Efficient Variational Quantum Eigensolver for Small Molecules and Quantum Magnets. *Nature* 549, 7671 (Sept. 2017), 242–246. https://doi.org/10/gbwmbw

[7] David R Karger. 1993. Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm. In *SODA*, Vol. 93. 21–30.

[8] Yunseong Nam, Yuan Su, and Dmitri Maslov. 2018. Approximate Quantum Fourier Transform with $O(n \log(n))$ T gates. *arXiv:1803.04933* (2018).

[9] Tianyi Peng, Aram Harrow, Maris Ozols, and Xiaodi Wu. 2019. Simulating Large Quantum Circuits on a Small Quantum Computer. *arXiv:1904.00102 [quant-ph]* (March 2019).

[10] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, et al. 2014. A Variational Eigenvalue Solver on a Photonic Quantum Processor. *Nature Communications* 5 (July 2014), 4213. https://doi.org/10/f6df9g

[11] John Preskill. 2018. Quantum Computing in the NISQ Era and Beyond. *Quantum* 2 (Aug. 2018), 79. https://doi.org/10/gd3xfp