Challenging Claims of Quantum Supremacy Using Reinforcement Learning

Samuel Garnett [1], Darwin Vargas [1], Zeliang Zhang [2], and Yanglet Xiao-Yang Liu [1,2]

[1]Rensselaer Polytechnic Institute, [2]Columbia University

Background

Introduction: We challenge Google's claim of quantum supremacy. The quantum circuits are represented as a tensor network, reinforcement learning algorithm finds an efficient tensor network contraction ordering (TNCO).

Qubits: Qubit in a superposition state, $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where $\alpha_0, \alpha_1 \in \mathbb{C}$ and $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

Quantum Gates: Google's Sycamore Circuit uses single-qubit gates (1) and double-qubit gate (2):

$$\sqrt{\mathbf{X}} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}, \sqrt{\mathbf{Y}} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \sqrt{\mathbf{W}} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -\sqrt{-i} \\ \sqrt{-i} & 1 \end{bmatrix} \quad (1) \qquad fSim(\theta, \phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -i\sin(\theta) & 0 \\ 0 & -i\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{bmatrix} \quad (2)$$

Sycamore Circuit: Google's Sycamore Circuit in Fig. 1, $\mathbf{R} \in \{\sqrt{\mathbf{X}}, \sqrt{\mathbf{Y}}, \sqrt{\mathbf{W}}\}$ is chosen at random. $\mathbf{U}$ represents a double-qubit gate.

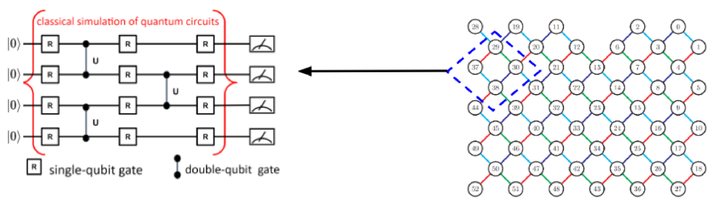
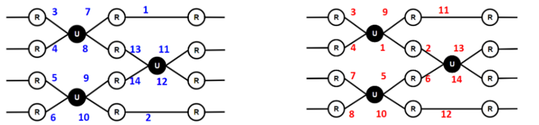


Figure 1. Example of Google's Sycamore circuit.

A quantum circuit as a tensor network. where nodes are tensors and lines are contractions.



The ordering in blue use 976 multiplications

The ordering in red use 5056 multiplications

TNCO Problem

Problem: We formulate the TNCO problem as follows

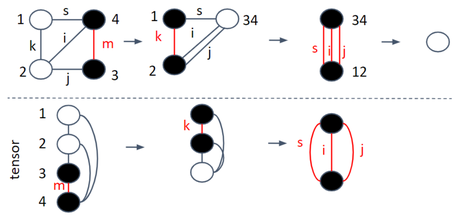


Figure 2. Example contraction of a tensor network.

The cost function measures the computational cost of each configuration:

$$C(x) = \sum_{i=1}^{N-1} \left\{ (2 - \sum_{u=1}^{N-i} x_{u,i})^2 + \sum_{u=1}^{N} \sum_{v=1}^{N} J_{u,v}^{i} x_{u,i} x_{v,i} \right\} \quad (5)$$

where $J_{u,v}^{i}$ is the number of multiplications when contracting a pair of tensors.

Implementation

Policy Network: Represent the tensor network as a symmetric matrix $\mathbf{M}$. E_t represents the set of remaining edges. The policy network, transformer network, generates a contraction ordering $P = (e_1 ... e_n - 1)$, where $e_t \in E_t$. Several implementation tricks:

- **Massively Parallel Gym Environment**: Run multiple gym environments for data sampling.
- **Learn To Optimize (L2O)** [3]: utilize LSTM or transformer networks [3].
- **Dual Replay Buffers**: one for all TNCOs, and one for high-quality TNCOs.
- **Swarm optimization** [4]: in order to converge on a high-quality optimum [4].
- **Curriculum Learning** [5]: sequentially solves a series of more complex TNCO problems [5].

Results

Synthetic tensor networks: the same setting as [2], our RL-Ising algorithm outperforms all baselines. Table 1 for tensor-train network and Table 2 for tensor-tree network.

Table 1: Results on synthetic tensor-train networks.

Tensors	400	600	800	1000	1500	2000
Scale	$\times 10^{120}$	$\times 10^{180}$	$\times 10^{241}$	$\times 10^{301}$	$\times 10^{451}$	$\times 10^{602}$
OE-Greedy [13]	17.22	27.67	4.44	3.83	-	-
CTG-Greedy [18]	10.33	16.60	2.67	4.28	-	-
CTG-Kahypar [18]	10.23	16.60	4.67	4.26	14.12	4.57
RL-Ising	**5.16**	**8.28**	**2.14**	**2.14**	**7.01**	**2.29**

Table 2: Number of multiplications for synthetic random tensor networks.

Qbits	25	50	75	100
Scale	$\times 10^{4}$	$\times 10^{7}$	$\times 10^{10}$	$\times 10^{12}$
OE-Greedy [13]	53.7/27.7	75.4/11.3	104/4.5	5296/26.4
CTG-Greedy [18]	40.3/20.3	12.8/ 4.2	8.3/0.9	27.9/ 2.2
CTG-Kahypar [18]	46.4/24.8	13.4/ 4.3	4.1/0.4	54.2/ 1.2
RL-TNCO [34]	13.1/12.5	3.2/ 1.8	1.2/0.2	5.5/ 1.8
RL-Ising	**12.5/11.4**	**2.5/ 1.5**	**0.7/0.1**	**4.9/ 1.1**

Google's Sycamore Circuits: 53 qubits and m cycles. Our RL-Ising algorithm [1] outperforms the RL-TNCO algorithm [2], as shown in Table 3.

Table 3: Number of multiplications for Google's Sycamore circuits.

Cycles	$m = 12$	$m = 14$	$m = 16$	$m = 18$	$m = 20$
Scale	$\times 10^{10}$	$\times 10^{12}$	$\times 10^{13}$	$\times 10^{16}$	$\times 10^{18}$
OE-Greedy [13]	6.23×10^{7}	4.77×10^{7}	7.74×10^{12}	6.21×10^{10}	9.59×10^{8}
CTG-Greedy [18]	1.16×10^{7}	1.91×10^{7}	1.42×10^{10}	3.71×10^{7}	4.19×10^{7}
CTG-Kahypar [18]	2.55×10^{3}	1.41×10^{2}	1.03×10^{4}	48.0	6.69
ACQDP [21]	1.09×10^{3}	71	1.15×10^{4}	25.8	6.65
RL-TNCO [34]	5.44	7.39	-	-	3.49
RL-Ising	**1.31**	**1.07**	**9.27**	**12.98**	**1.23**

We maintain an open-source repository that showcases our research thus far and encourages collaboration. Github repo: `https://github.com/RPIQuantumComputing/RLForQuantumCircuits`

Takehome: The "quantum supremacy" claim still lacks an unequivocal first demonstration, since Google's announcement [3] was under serious questions due to the debatable estimate of 10,000 years' running time for the classical simulation task on the Summit supercomputer.

Ongoing Project

After finding cheaper contraction orderings for the Sycamore Circuit, we are optimizing QFT and FFT algorithms, also formulated as TNCO problems, as shown in Fig. 3.

The QFT uses single and double qubit gates and has a simulation complexity of $O(N \log N)$, where $N = 2^n$ for n qubits.

The FFT can be represented as a branching tensor tree network with a complexity of $O(N \log N)$.

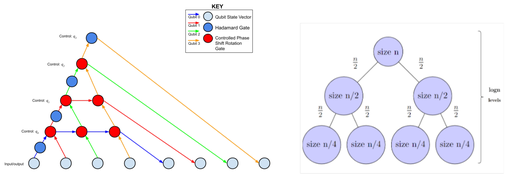


Figure 3. Tensor-tree network for the QFT algorithm (left) and FFT algorithm (right).

In our current experiments, an adapted RL algorithm finds efficient algorithms with less number of multiplications.

Introduction to Quantum Computing at RPI

- This project will be used as a lab task in our Introduction to Quantum Computing class.
- Tensor networks to represent different quantum circuits.
- Efficiency of simulations are a TNCO problem.
- Reinforcement learning algorithm helps find a good contraction ordering.

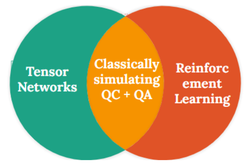


Figure 4. Our methodology: a tensor network + RL approach.

References

1. Xiao-Yang Liu and Zeliang Zhang. Classical Simulation of Quantum Circuits Using Reinforcement Learning: Parallel Environments and Benchmark. Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023), Track on Datasets and Benchmarks, 2023.
2. Eli Meirom, Haggai Maron, Shie Mannor, and Gal Chechik. Optimizing tensor network contraction using reinforcement learning. In International Conference on Machine Learning, pages 15278–15292. PMLR, 2022.
3. Tianlong Chen, Xiaohan Chen, Wuyang Chen, Zhangyang Wang, Howard Heaton, Jialin Liu, and Wotao Yin. Learning to optimize: A primer and a benchmark. The Journal of Machine Learning Research, 23(1):8562–8620, 2022.
4. Christian Blum and Xiaodong Li. Swarm intelligence in optimization. In Swarm intelligence: introduction and applications, pages 43–85. Springer, 2008.
5. Yoshua Bengio, J´ erˆ ome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In International Conference on Machine Learning (ICML), pages 41–48, 2009.