

TensorQC: Towards Scalable Quantum Classical Hybrid Compute

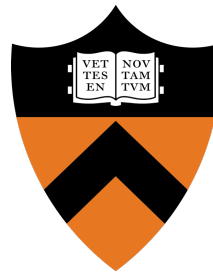
Wei Tang

Department of Computer Science

Princeton University

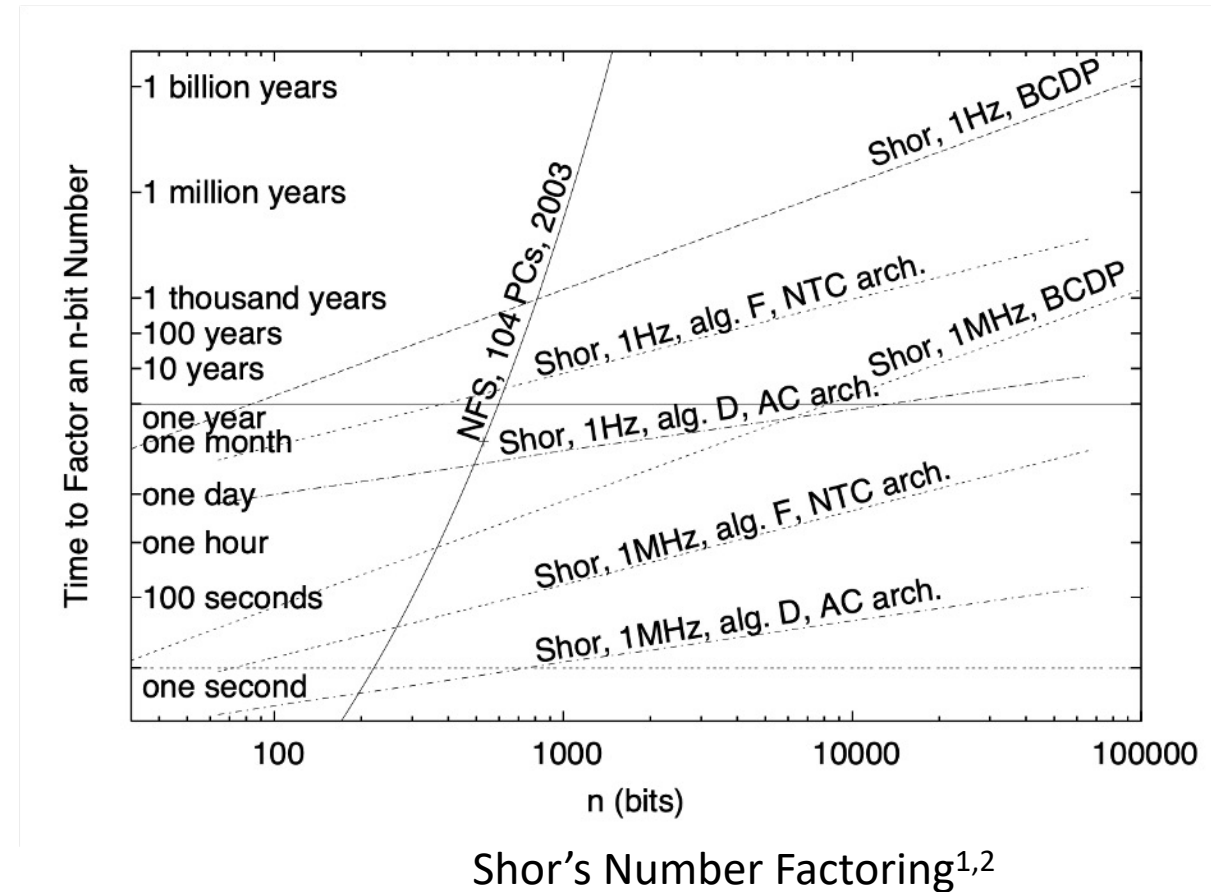


weit@princeton.edu
<https://www.wtang.page>



Quantum Computing (QC): Huge Promise

- With each additional qubit, the size of the ideal computational state space (Hilbert space) available to a QC algorithm doubles.
- Some quantum algorithms offer exponential speedups.



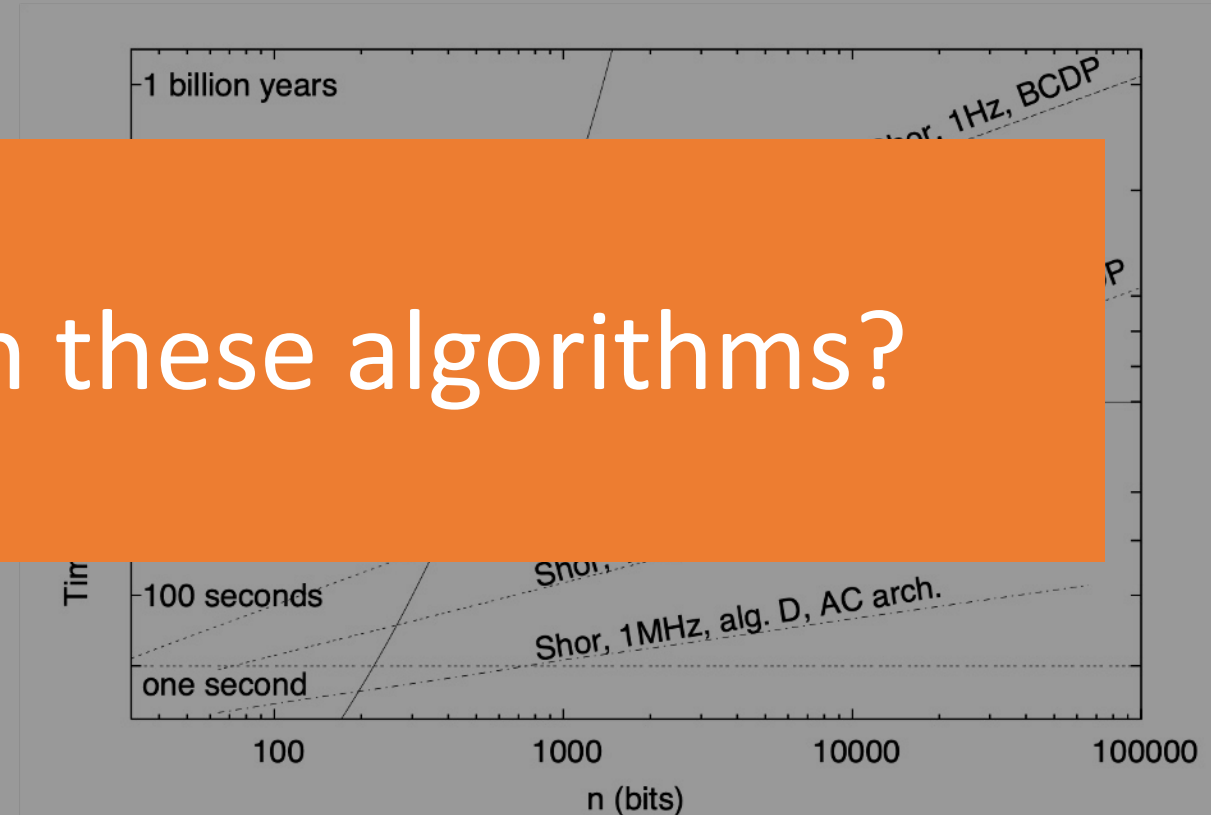
¹Van Meter, Rodney, Kohei M. Itoh, and Thaddeus D. Ladd. "Architecture-dependent execution time of Shor's algorithm." In *Controllable Quantum States: Mesoscopic Superconductivity and Spintronics (MS+ S2006)*, pp. 183-188. 2008.

²P.W.Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM Review, vol. 41, no. 2, pp. 303-332, 1999.

Quantum Computing (QC): Huge Promise

- With each additional qubit, the size of the ideal computational state space

But where do we run these algorithms?



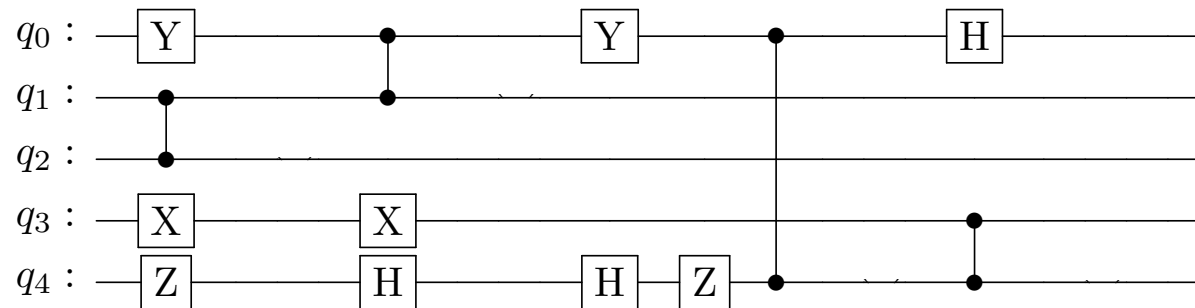
Shor's Number Factoring^{1,2}

¹Van Meter, Rodney, Kohei M. Itoh, and Thaddeus D. Ladd. "Architecture-dependent execution time of Shor's algorithm." In *Controllable Quantum States: Mesoscopic Superconductivity and Spintronics (MS+ S2006)*, pp. 183-188. 2008.

²P.W.Shor,"Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM Review, vol. 41, no. 2, pp. 303-332, 1999.

Existing Solutions

- Given a quantum circuit, the existing options to run it are:
 1. Classical Simulations (Qiskit, Quimb, customized HPC, etc)
 2. Quantum Processing Units (via AWS, IBM, etc)



The Ongoing Battle

Article | [Published: 23 October 2019](#)

Quantum supremacy using a programmable superconducting processor

54-qubit Supremacy circuit in 200 seconds on QPU.

Quantum Physics

[Submitted on 21 Oct 2019 (v1), last revised 22 Oct 2019 (this version, v2)]

Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits

2.5 days (no experimental validation).

[Submitted on 4 Nov 2021 (v1), last revised 28 Aug 2022 (this version, v2)]

Solving the sampling problem of the Sycamore quantum circuits

53-qubits. 15 hours with 512 GPUs.

RESEARCH-ARTICLE **FREE ACCESS**



Closing the "quantum supremacy" gap: achieving real-time simulation of a random quantum circuit using a new Sunway supercomputer

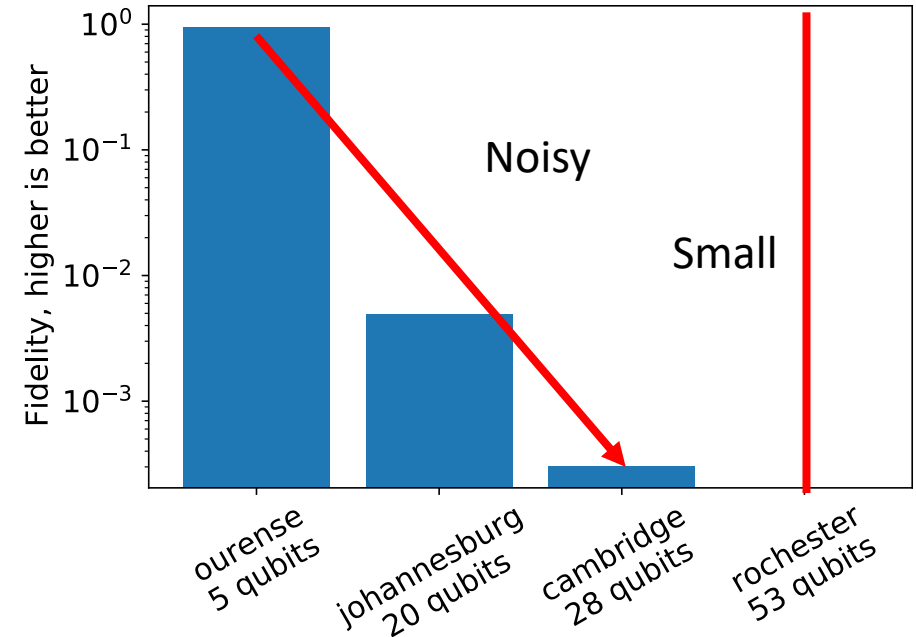
100-qubits. 304 seconds with 42 million cores.



IBM road map to build a 1000 qubit QPU in 2023.

Problems

- Classical Simulations
 - Noiseless, good for small-scale verification and algorithm developments.
 - Tailored methods may exist for specific benchmarks, but ultimately does not scale (unless $P=NP$, then we all need new jobs).
- QPUs
 - Offers insights on QPU performance. Helps hardware and software designs.
 - Need to satisfy both quantity (number of qubits) and quality (fidelity, coherence etc) requirements on qubits to produce useful results.



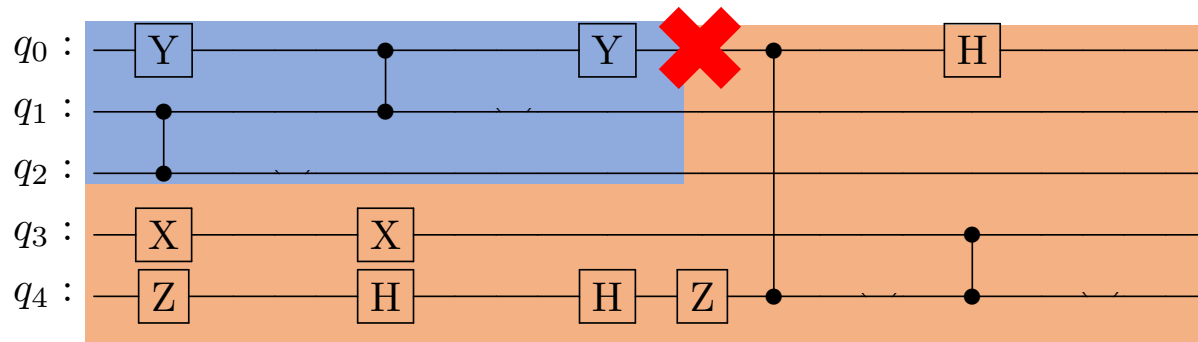
Running Bernstein-Vazirani (BV) algorithm on IBM devices at half capacity. E.g. 26-qubit on the 53-qubit Rochester. 8k shots. Fidelity decreased to effectively 0 beyond 10-qubit benchmarks.

TensorQC: Hybrid Quantum + Classical

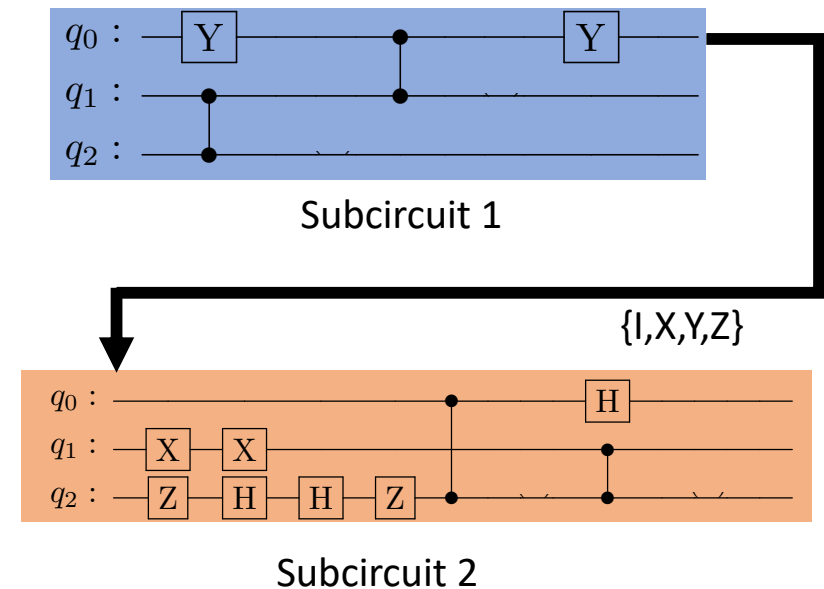
- Cut + QPU + Classical
 1. “Cut” large QC algorithm circuits into small subcircuits runnable on NISQ platforms.
 2. Evaluate the subcircuits on small NISQ platforms.
 3. Use classical reconstruction approaches to sew subcircuits back together.

Example: Making One Cut

- Example: Cut one edge to split a 5-qubit circuit into two smaller (3-qubit each) subcircuits.
- TensorQC performs vertical cuts on qubit wires, in other words, timewise cuts.



$$P = \sum_{k=\{I,X,Y,Z\}} p_1(k) \otimes p_2(k)$$



Example: Making One Cut

- Can cut any qubit line.
- However, need to select a proper set of edges to fully separate the subcircuits.
- High-level intuition: performing state tomography at the cut points.

$$\begin{array}{c} u \end{array} \xrightarrow{\quad} \begin{array}{c} v \end{array} \Rightarrow \frac{1}{2} \sum \left\{ \begin{array}{l} \begin{array}{c} u \end{array} \xrightarrow{\quad} \boxed{I} \otimes \left(\begin{array}{c} \boxed{|0\rangle} \xrightarrow{\quad} v + \begin{array}{c} \boxed{|1\rangle} \xrightarrow{\quad} v \end{array} \right) \\ \begin{array}{c} u \end{array} \xrightarrow{\quad} \boxed{X} \otimes \left(2 \begin{array}{c} \boxed{|+\rangle} \xrightarrow{\quad} v - \begin{array}{c} \boxed{|0\rangle} \xrightarrow{\quad} v - \begin{array}{c} \boxed{|1\rangle} \xrightarrow{\quad} v \end{array} \right) \\ \begin{array}{c} u \end{array} \xrightarrow{\quad} \boxed{Y} \otimes \left(2 \begin{array}{c} \boxed{|i\rangle} \xrightarrow{\quad} v - \begin{array}{c} \boxed{|0\rangle} \xrightarrow{\quad} v - \begin{array}{c} \boxed{|1\rangle} \xrightarrow{\quad} v \end{array} \right) \\ \begin{array}{c} u \end{array} \xrightarrow{\quad} \boxed{Z} \otimes \left(\begin{array}{c} \boxed{|0\rangle} \xrightarrow{\quad} v - \begin{array}{c} \boxed{|1\rangle} \xrightarrow{\quad} v \end{array} \right) \end{array} \right.
 \end{array}$$

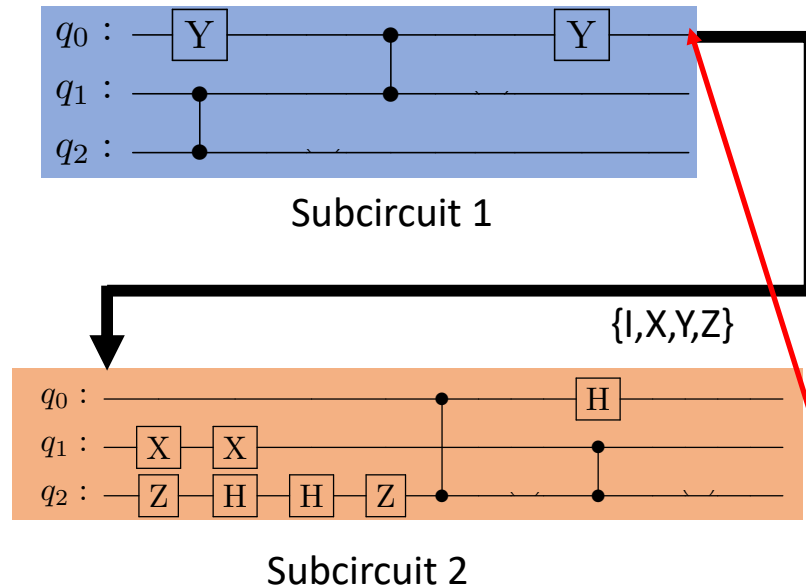
Example: Making One Cut

- Can cut any qubit line.
- However, need to select a proper set of edges to fully separate the subcircuits.
- High-level intuition: performing state tomography at the cut points.

$$\begin{array}{c}
 \xrightarrow{u} \xrightarrow{v} \Rightarrow \frac{1}{2} \sum \left\{ \begin{array}{l}
 \xrightarrow{u} [I] \otimes \left([0] \xrightarrow{v} + [1] \xrightarrow{v} \right) \\
 \xrightarrow{u} [X] \otimes \left(2 [+] \xrightarrow{v} - [0] \xrightarrow{v} - [1] \xrightarrow{v} \right) \\
 \xrightarrow{u} [Y] \otimes \left(2 [i] \xrightarrow{v} - [0] \xrightarrow{v} - [1] \xrightarrow{v} \right) \\
 \xrightarrow{u} [Z] \otimes \left([0] \xrightarrow{v} - [1] \xrightarrow{v} \right)
 \end{array} \right\}
 \end{array}
 \Rightarrow \begin{array}{l}
 \text{Need to measure in } I, X, Y, Z. \\
 \text{Need to initialize in } |0\rangle, |1\rangle, |+\rangle, |i\rangle.
 \end{array}$$

Example: Making One Cut

- Initializations:
 - $|0\rangle$: ordinary initialization, no modifications.
 - $|1\rangle$: $+X$
 - $|+\rangle$: $+H$
 - $|i\rangle$: $+SH$
- Measurements:
 - I : computation basis, no modifications.
 - X : $+H$
 - Y : $+S^\dagger H$
 - Z : computation basis, no modifications.



If I :

$$\overline{XX0}, \overline{XX1} \rightarrow +\overline{XX}$$

Else:

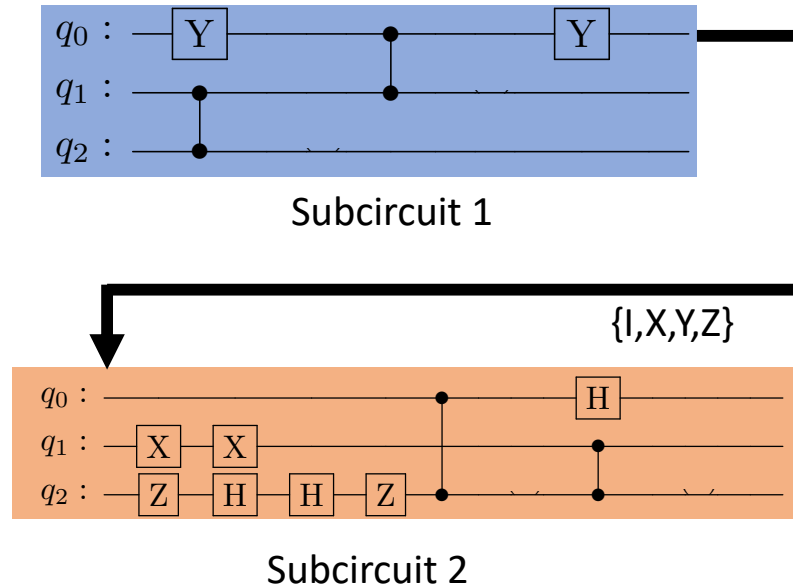
$$\overline{XX0} \rightarrow +\overline{XX}$$

$$\overline{XX1} \rightarrow -\overline{XX}$$

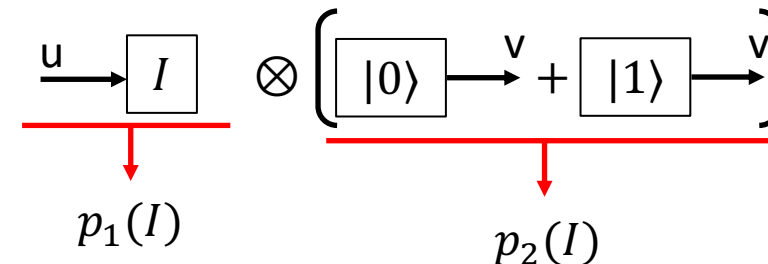
Not in the
uncut circuit
output

Example: Making One Cut

- Initializations:
 - $|0\rangle$: ordinary initialization, no modifications.
 - $|1\rangle$: $+ X$
 - $|+\rangle$: $+ H$
 - $|i\rangle$: $+ SH$
- Measurements:
 - I : computation basis, no modifications.
 - X : $+ H$
 - Y : $+ S^\dagger H$
 - Z : computation basis, no modifications.

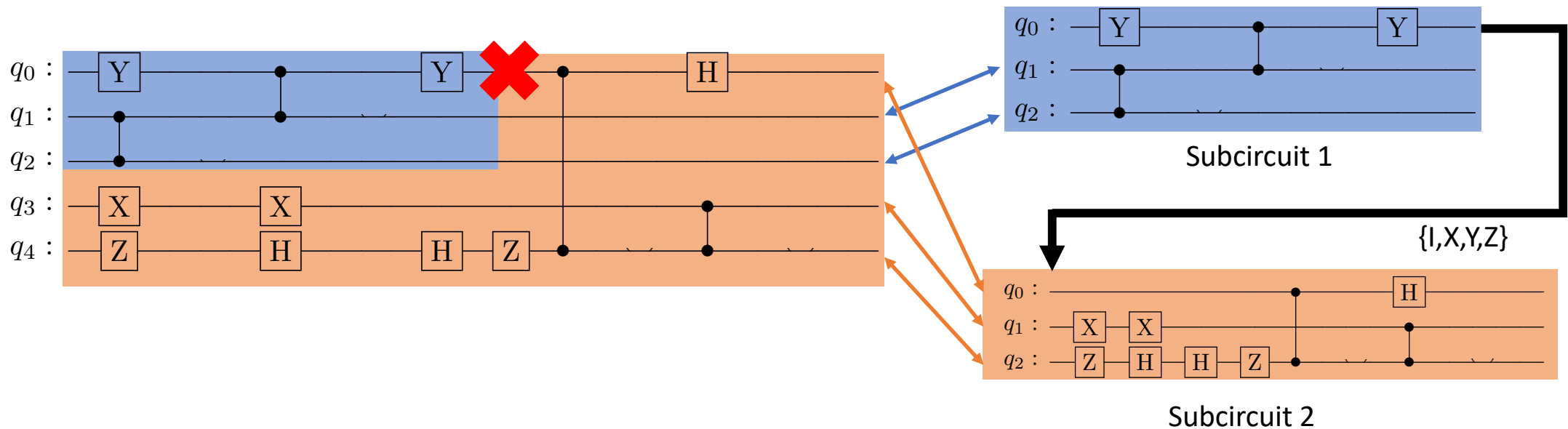


$$P = \sum_{k=\{I, X, Y, Z\}} p_1(k) \otimes p_2(k)$$



Example: Making One Cut


- What is the reconstructed probability of e.g. $|01010\rangle$?
- The relevant state of subcircuit 1 is $|10\rangle$, subcircuit 2 is $|010\rangle$.




TensorQC Challenges

- For an n qubit circuit, circuit cutting makes K cuts and produces n_c subcircuits.

- $$P = \sum_{k=1}^{4^K} \bigotimes_{i=1}^{n_c} p_i(k) \in \mathbb{R}^{2^n}$$


Classical


QPU

TensorQC Challenges

- For an n qubit circuit, circuit cutting makes K cuts and produces n_c subcircuits.

• $P = \sum_{k=1}^{4^K} \bigotimes_{i=1}^{n_c} p_i(k) \in \mathbb{R}^{2^n} \longrightarrow \text{Exponentially long probability output}$

Exponentially many terms

Finding optimal cuts is NP-hard

TensorQC Challenges

- For an n qubit circuit, circuit cutting makes K cuts and produces n_c subcircuits.

- $P = \sum_{k=1}^{4^K} \bigotimes_{i=1}^{n_c} p_i(k) \in \mathbb{R}^{2^n} \longrightarrow$ Exponentially long probability output
Solution: State Merging

Exponentially many terms
Solution: Tensor Network

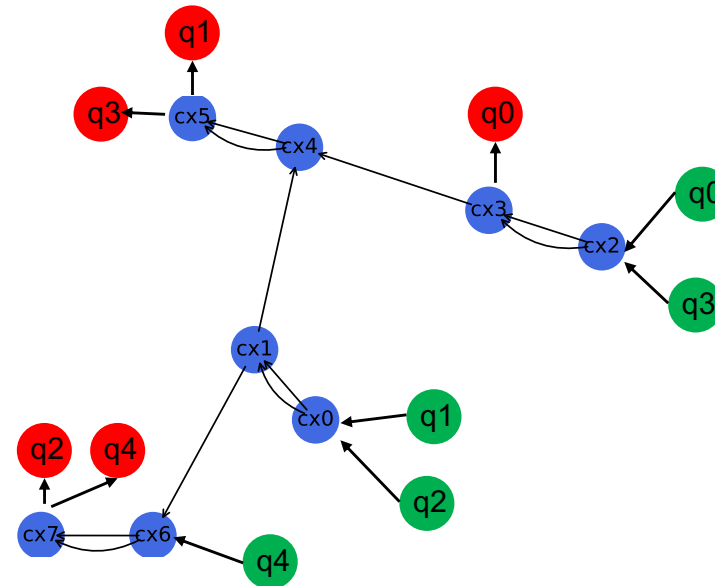
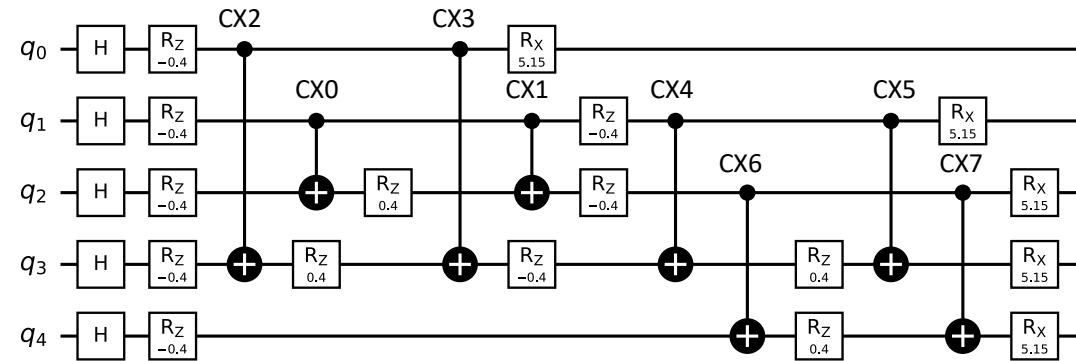
Finding optimal cuts is NP-hard
Solution: multilevel graph partition

- Prior work in CutQC uses simple parallelization, Mixed Integer Programming solver and Dynamic Definition.

Tang, Wei, et al. "Cutqc: using small quantum computers for large quantum circuit evaluations." *Proceedings of the 26th ACM International conference on architectural support for programming languages and operating systems*. 2021.

Tensor Graph Abstraction

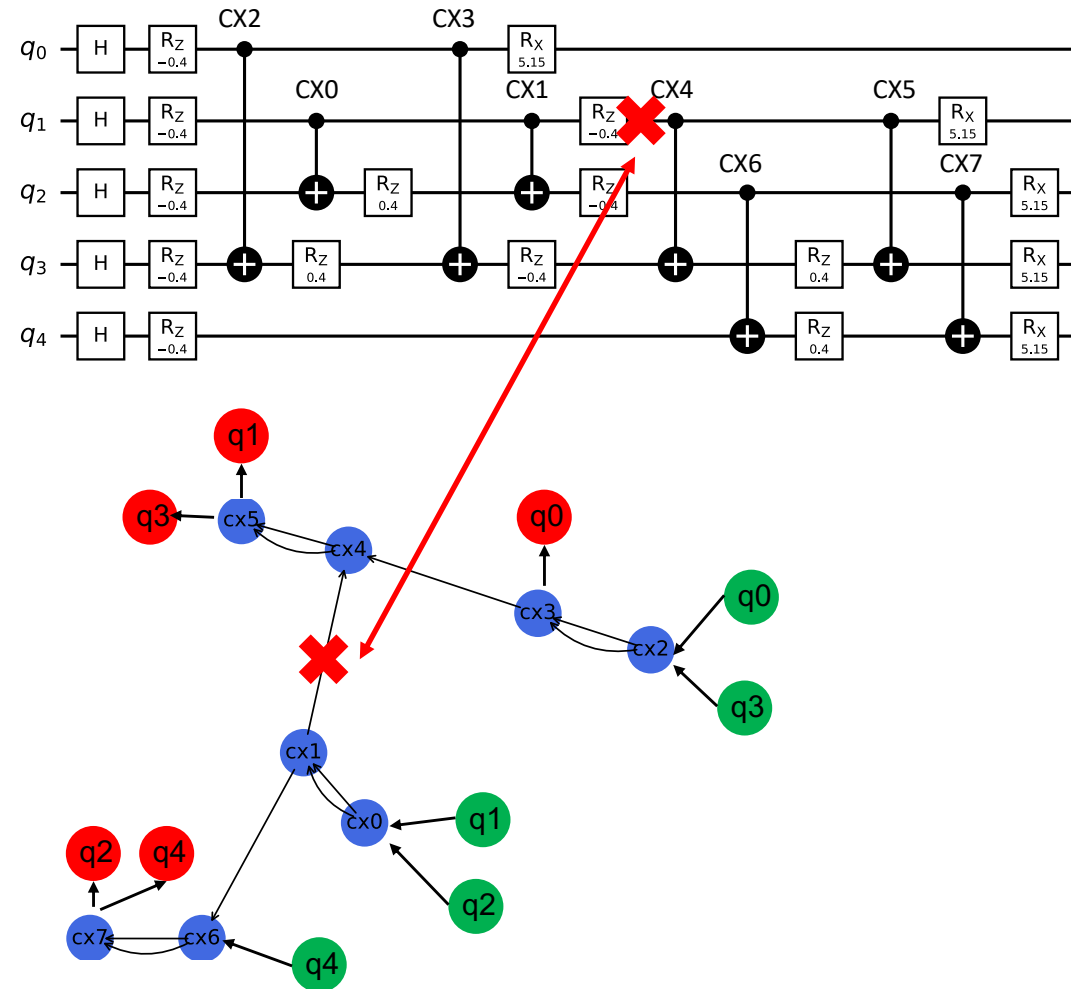
- A quantum circuit can be represented as a Directed Acyclic Graph.
- Gates are the vertices. Qubit lines connecting the gates are the edges.
- Also contains input (green) and output (red) qubits.
- Single qubit gates are omitted since they do not affect the connectivity.



A 5-qubit QAOA circuit solving the Maximum Independent Set problem for a random 3-regular graph.

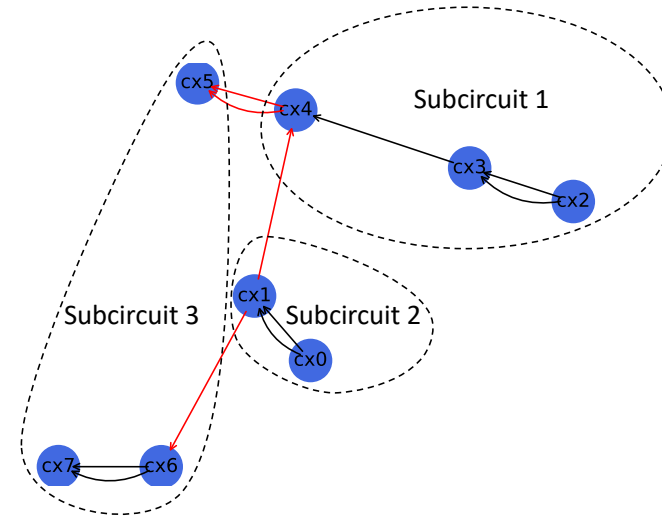
Tensor Graph Abstraction

- Cutting a quantum circuit means cutting an edge in the DAG.



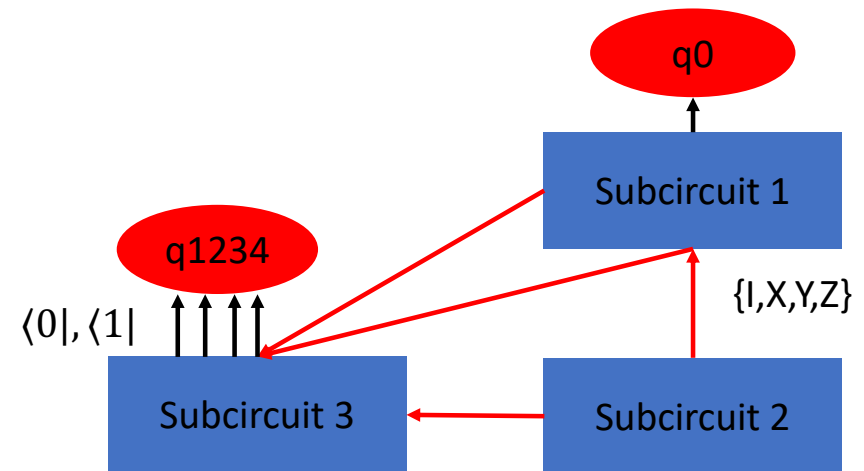
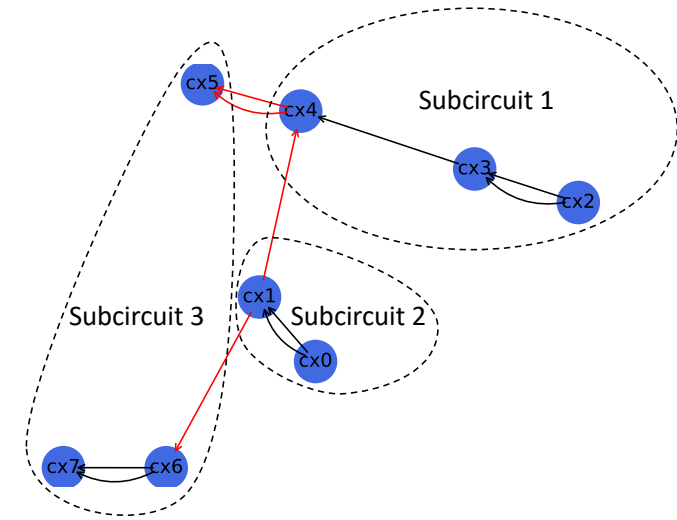
Tensor Graph Abstraction

- Cutting a quantum circuit means cutting an edge in the DAG.
- Make multiple cuts to produce separated subcircuits.



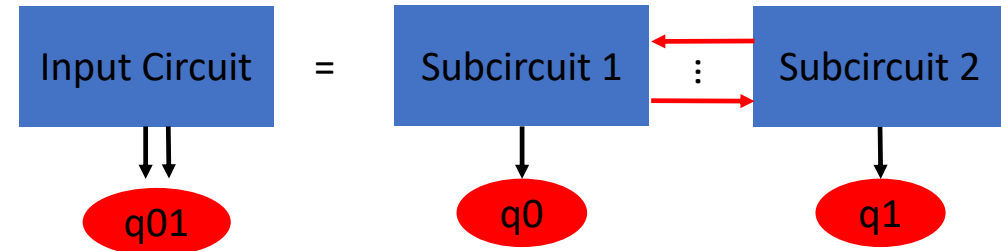
Tensor Graph Abstraction

- Cutting a quantum circuit means cutting an edge in the DAG.
- Make multiple cuts to produce separated subcircuits.
- Abstract the resulting subcircuits as a tensor graph.



Tensor Graph Contraction

- Reconstructing a pair of subcircuits is computing the tensor product of their probability distributions and sum.

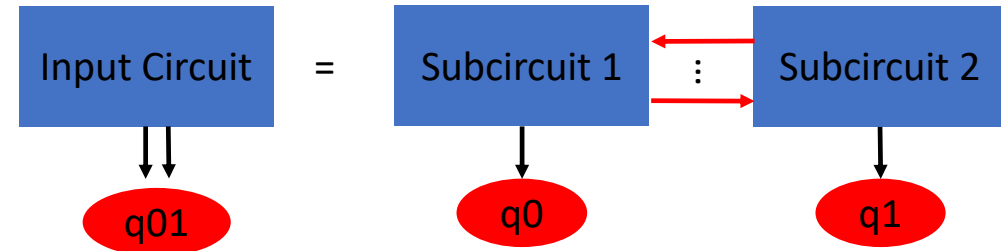


$$P = \sum_{k=1}^{4^{\#cuts}} p_1(k) \otimes p_2(k)$$

$$\rightarrow P_{q_0, q_1} = \sum_k p_1(k, q_0) p_2(k, q_1)$$

Tensor Graph Contraction

- Reconstructing a pair of subcircuits is computing the tensor product of their probability distributions and sum.
- This is mathematically equivalent to tensor contraction.
- Compute cost = $\dim(i) \times \dim(j) \times \dim(k)$ multiplications.



$$P = \sum_{k=1}^{4^{\#cuts}} p_1(k) \otimes p_2(k)$$

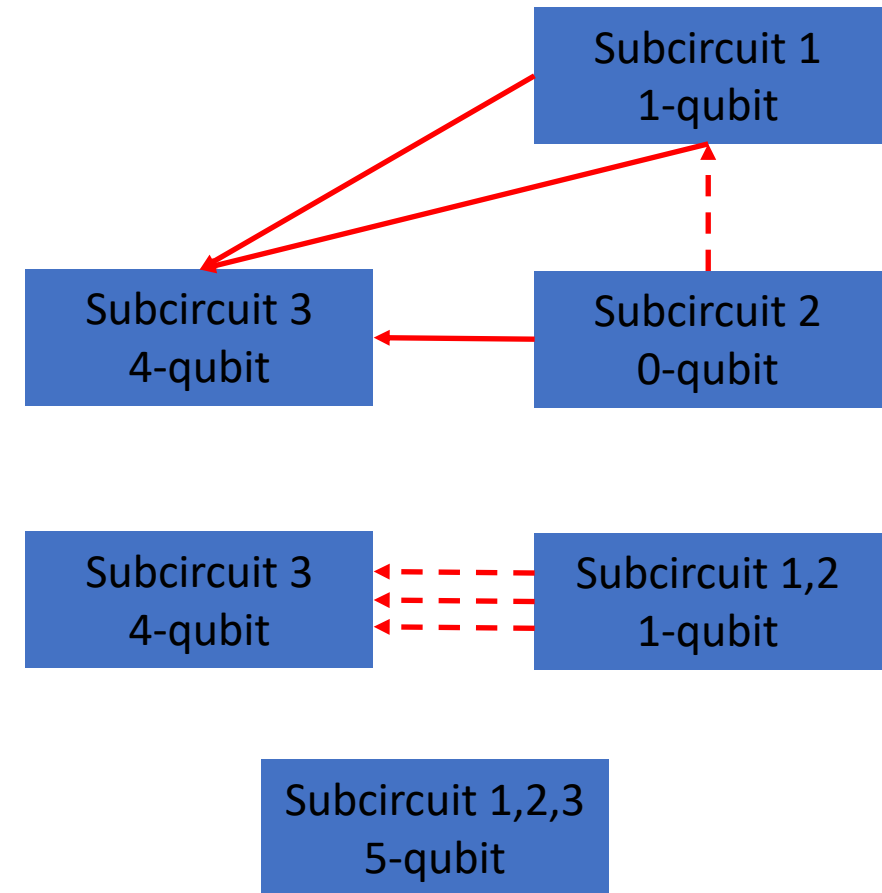
$$\rightarrow P_{q_0, q_1} = \sum_k p_1(k, q_0) p_2(k, q_1)$$

$$\text{---} \overset{i}{\text{---}} \boxed{C} \text{---} \overset{k}{\text{---}} = \text{---} \overset{i}{\text{---}} \boxed{A} \text{---} \overset{j}{\text{---}} \boxed{B} \text{---} \overset{k}{\text{---}}$$

$$C_{i,k} = \sum_j A_{i,j} B_{j,k}$$

Tensor Graph Contraction

- Computing 4^K summations explicitly is equivalent to tensor network contractions.
- Contract the subcircuits one by one, till all are merged.
 - $(4^2 \times 2^1) \times (4^1) \times (4^1 \times 2^0) = 512$
 - $2^4 \times 4^3 \times 2^1 = 2048$
- In contrast, naïve computation requires $4^4 \times (2^1 \times 2^0 + 2^1 \times 2^4) = 8704$ multiplications.



TensorQC Challenges

- For an n qubit circuit, circuit cutting makes K cuts and produces n_c subcircuits.

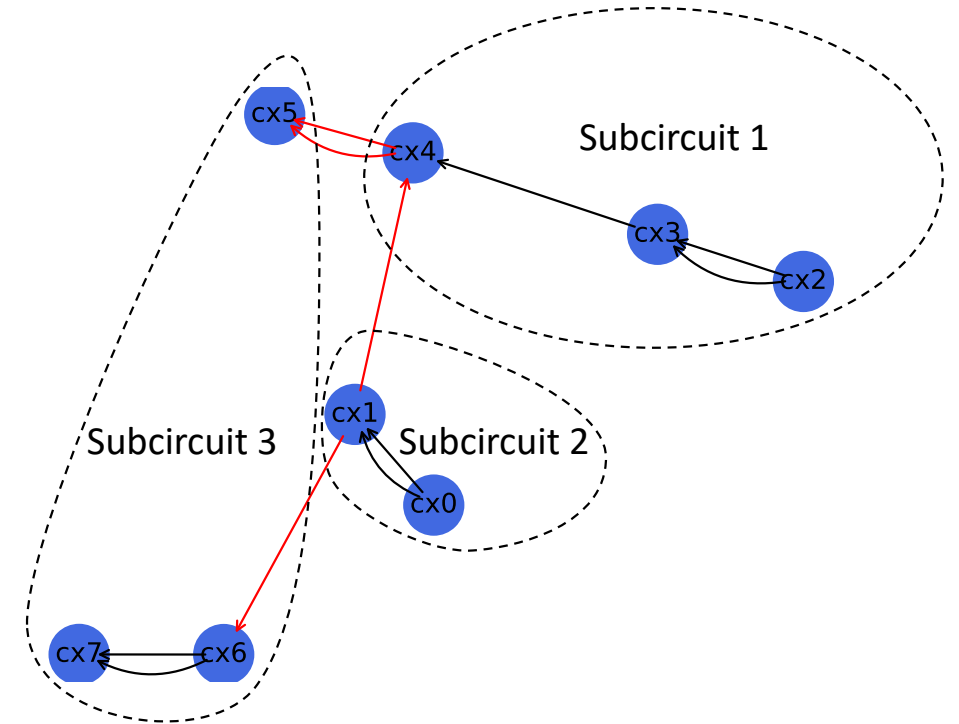
- $P = \sum_{k=1}^{4^K} \bigotimes_{i=1}^{n_c} p_i(k) \in \mathbb{R}^{2^n} \longrightarrow \begin{array}{l} \text{Exponentially long} \\ \text{probability output} \\ \text{Solution: State Merging} \end{array}$

Exponentially many terms
Solution: Tensor Network

Finding optimal cuts is NP-hard
Solution: multilevel graph partition

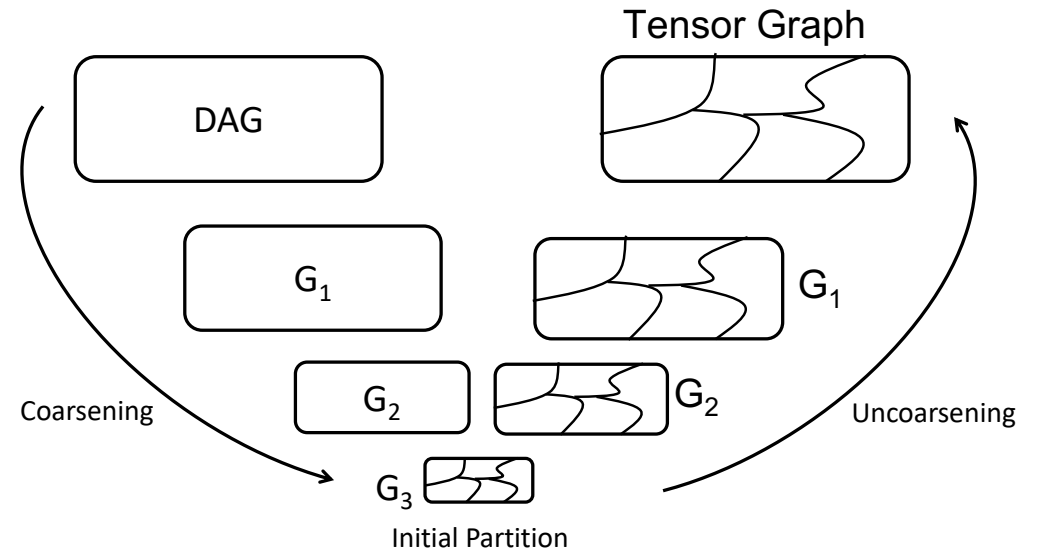
Cut Searching

- Equivalent to a constrained k-way graph partition problem.
- Constrains:
 - #qubits in each subcircuit is limited by available QPU size.
 - #gates in each subcircuit is limited by QPU fidelity.
- Adapt the METIS graph partitioner.
- Much faster than the previous optimal Mixed Integer Programming solver.



Multilevel Graph Partition

- Coarsening: heuristically samples a max number of edges that do not share any common vertices and merge.
- Initial partition: bisects the coarse graph n_C times to produce an initial partition.
- Uncoarsening: expands the merged edges, refines the partitions using the Global Kernighan-Lin Refinement.



TensorQC Challenges

- For an n qubit circuit, circuit cutting makes K cuts and produces n_c subcircuits.

• $P = \sum_{k=1}^{4^K} \bigotimes_{i=1}^{n_c} p_i(k) \in \mathbb{R}^{2^n} \longrightarrow$

The diagram illustrates the challenges of tensor quantum circuit simulation. The equation $P = \sum_{k=1}^{4^K} \bigotimes_{i=1}^{n_c} p_i(k) \in \mathbb{R}^{2^n}$ is shown. From this equation, three arrows point to different challenges:

- A vertical arrow points from the summation $\sum_{k=1}^{4^K}$ to the text "Exponentially many terms" and "Solution: Tensor Network".
- A diagonal arrow points from the tensor product $\bigotimes_{i=1}^{n_c}$ to the text "Finding optimal cuts is NP-hard" and "Solution: multilevel graph partition".
- A horizontal arrow points from the entire expression to a red-bordered box containing "Exponentially long probability output" and "Solution: State Merging".

Circuits Categorization

	Concentrated Probability Output	Distributed Probability Output
Small Circuits	Locate the solution states (indicated by their high probability).	Obtain the probability distribution.
Large Circuits	Locate the solution states (indicated by their high probability).	Requires exponential shots to converge. Requires exponential memory to write down. Impossible to run with any platform.

State Merging

State Merging

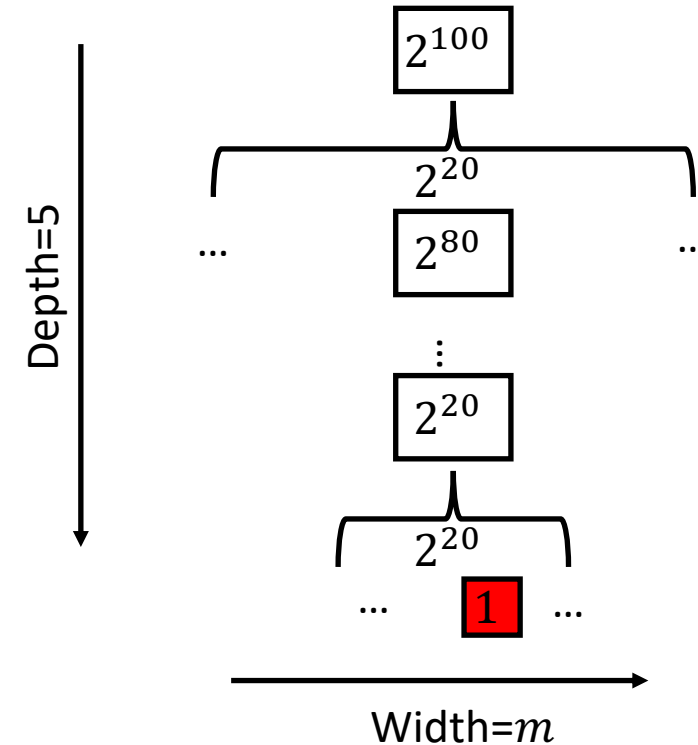
- Insights:
 - We just want to know which are the solution states, not the full probability distribution.
 - The sum of probability for multiple states with 0 probability is still 0.
- SM intuition: reconstruct the sum of probability for the non-solution states, only expand the solution states.

State Merging

- Merge the subcircuit states into bins before classical postprocessing.
 - $p_1(k) = (|0\rangle, |1\rangle), p_2(k) = (|0\rangle, |1\rangle) \rightarrow p_1 \otimes p_2 = (|00\rangle, |01\rangle, |10\rangle, |11\rangle)$
 - $p'_1(k) = (|0\rangle + |1\rangle), p_2(k) = (|0\rangle, |1\rangle) \rightarrow p'_1 \otimes p_2 = (|X0\rangle, |X1\rangle)$
- Less memory and compute overhead.
- Same information if merged states are not solution states.

SM Iterative Search

- An iterative search process to locate the solution states.
- Each recursion expands the bin with the highest sum of probability.
- Expansion runs till the solution states are found.

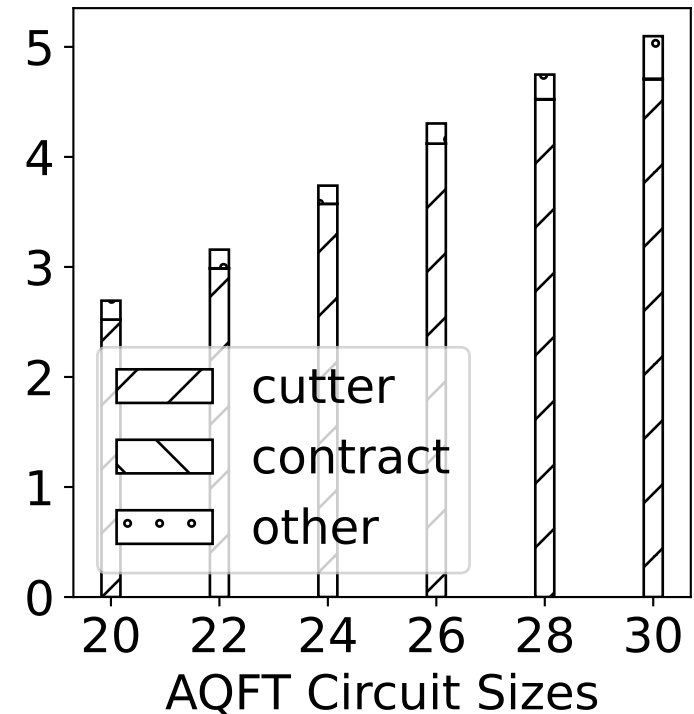
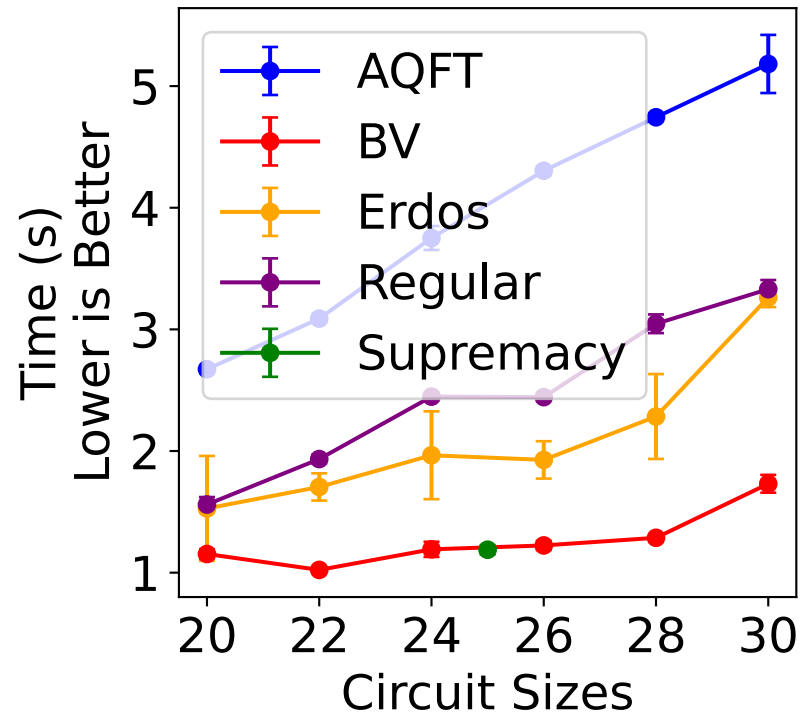


Benchmarks

1. BV : Bernstein-Vazirani circuit solves the hidden sub-string problem. Has 1 solution state.
2. Regular: Quantum Approximate Optimization Algorithm solves the maximum independent set problem for random 3-regular graphs.
3. Erdos: The same algorithm as Regular but for random Erdos-Renyi graphs.
4. Supremacy: Random quantum circuits with depth $(1 + 8 + 1)$.
5. AQFT: Approximate Quantum Fourier Transform that is expected to outperform the standard QFT circuit under noise.

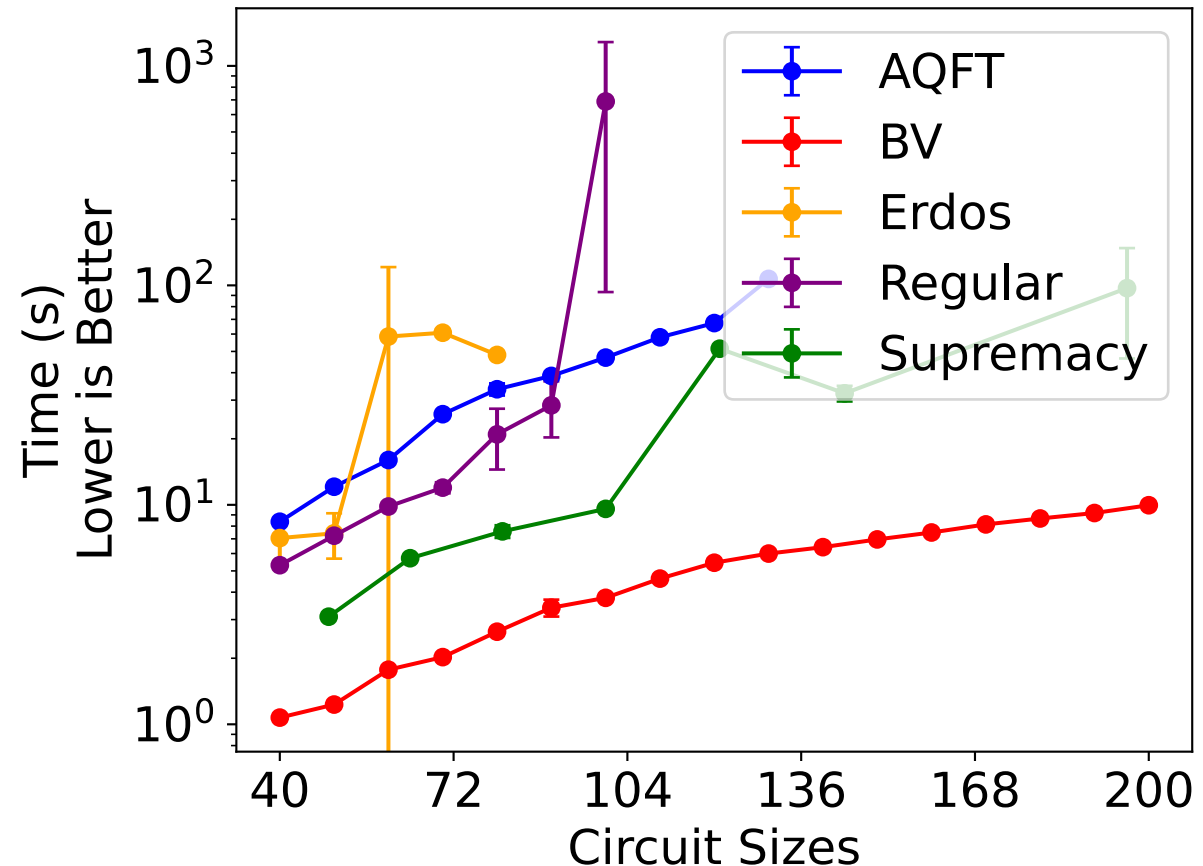
Results: Full State Runtime

- Each data point is the average of 5 trials.
- At 30-qubit, TensorQC is about 10× faster than the Qiskit statevector simulation.
- The runtime breakdown for the AQFT benchmark shows that most of the runtime is from the cut searching.



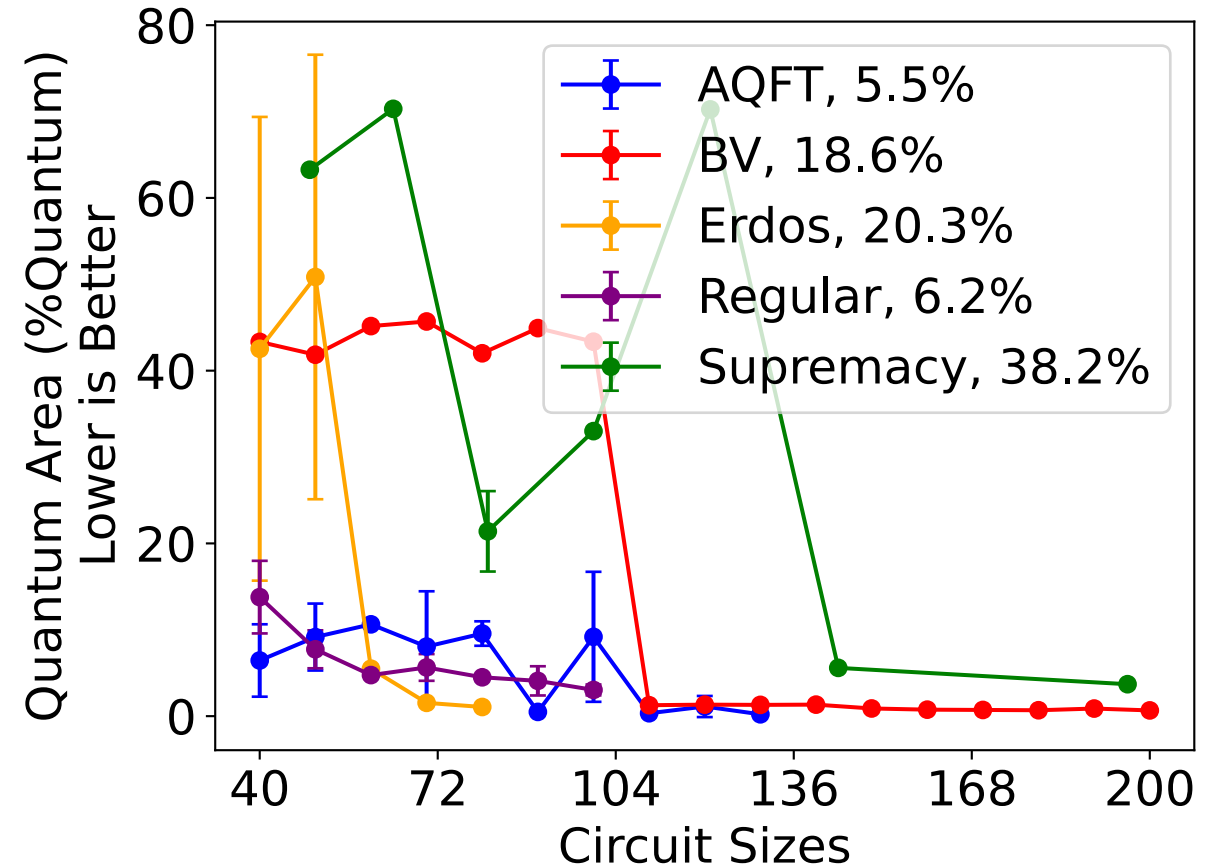
Results: SM Runtime

- Max SM bins $M = 2^{20}$.
Each data point is the average of up to 5 trials.
- Runs $n / \log_2 M$ recursions (i.e. 5 for the 100-qubit benchmarks) to find at least one solution.
- Able to run circuits significantly beyond the classical simulation limit.



Results: Quantum Area Reduction

- Plots the average quantum area (width*depth) of TensorQC as a percentage of uncut.



Results: Compare Against Classical

- We estimate Liu et al. require 1250 cores and 300 seconds for our benchmark.

Method	Supremacy Benchmark	Backend	Runtime	Result
TensorQC	$10 \times 10 \times (1 + 8 + 1)$ #subtasks = 2^6	1 Nvidia A100 GPU on 1 core.	~10 seconds	5 million states at perfect classical postprocessing fidelity. Overall fidelity depends on QPUs.
Liu et al.	$10 \times 10 \times (1 + 40 + 1)$ #subtasks = 32^6	42 million cores on the Sunway supercomputer.	304 seconds	1 million states at $< 1\%$ fidelity.