XEB.jl: Challenging Quantum Supremacy in Julia

Chen Zhao (Harvard University)

Table of contents

Quantum Supremacy and Linear XEB

Spoofing algorithms for linear XEB

Features of XEB.jl

Numerical results

Quantum Speedups

Exponential speedups

- Shor's algorithm for integer factoring
- HHL algorithm for linear equations

Polynomial speedups

- Grover's algorithm for searching

Requires fault-tolerant quantum computing (long terms)

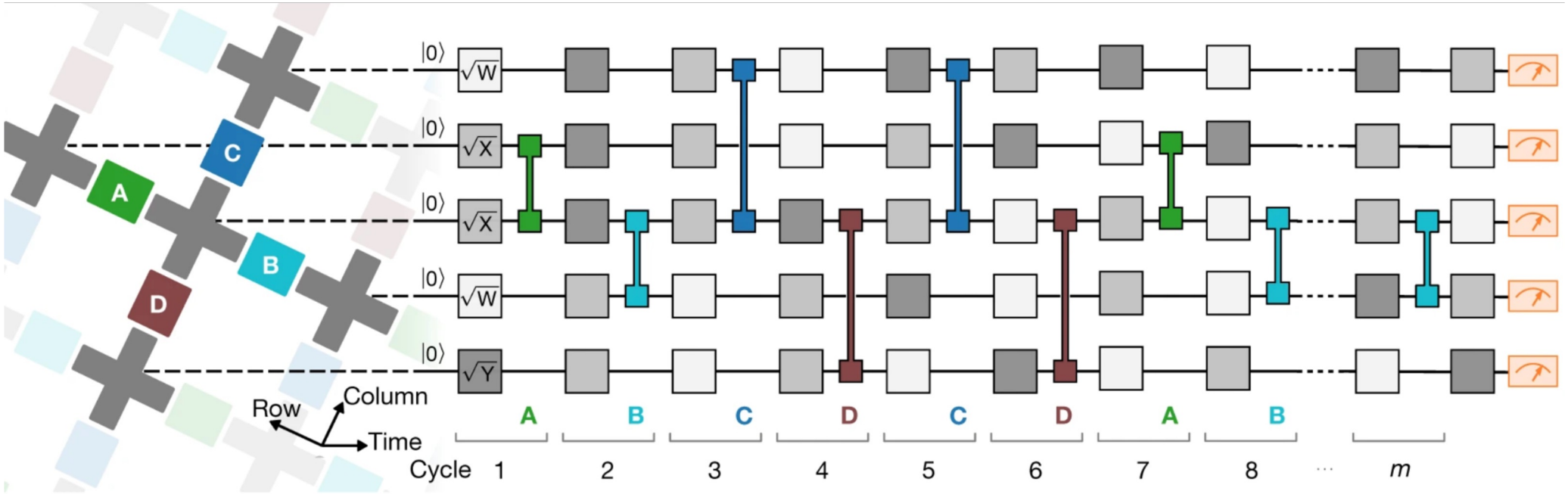
Can we verify quantum advantages on near-term devices?

- Random quantum circuit sampling (Google 2019[1], USTC 2021[2, 3])
- Gaussian Boson sampling (USTC 2021[4, 5])
- IQP circuits
- …

[1] Arute, F. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505–510 (2019).
[2] Wu, Y. *et al.* Strong Quantum Computational Advantage Using a Superconducting Quantum Processor. *Phys. Rev. Lett.* **127**, 180501 (2021).
[3] Zhu, Q. *et al.* Quantum Computational Advantage via 60-Qubit 24-Cycle Random Circuit Sampling. (2021) doi:10.48550/arXiv.2109.03494.
[4] Zhong, H.-S. *et al.* Quantum computational advantage using photons. *Science* **370**, 1460–1463 (2020).
[5] Zhong, H.-S. *et al.* Phase-Programmable Gaussian Boson Sampling Using Stimulated Squeezed Light. 9.

Random quantum circuit sampling

- Apply a random circuit to an initial state
- Measure the final state and get bitstrings
- Compute the linear cross-entropy benchmarking (XEB)



Linear XEB

$$\chi(p,q) = 2^n \left[\sum_{x \in \{0,1\}^n} p(x)q(x) \right] - 1$$

- Can approximate fidelity under certain assumption
- Experimentally feasible
- There is evidence of hardness[5-8]

[6] Aaronson, S. & Chen, L. Complexity-Theoretic Foundations of Quantum Supremacy Experiments. 66.
[7] Bouland, A., Fefferman, B., Nirkhe, C. & Vazirani, U. On the complexity and verification of quantum random circuit sampling. *Nature Phys* **15**, 159–163 (2019).
[8] Aaronson, S. & Gunn, S. On the Classical Hardness of Spoofing Linear Cross-Entropy Benchmarking. *Theory of Computing* **16**, 1–8 (2020).
[9] Krovi, H. Average-case hardness of estimating probabilities of random quantum circuits with a linear scaling in the error exponent. 26.

Experimental results

	#qubits	#depth	χ
Google 2019[1]	53	20	2.24×10^{-3}
USTC-1[2]	60	20	6.62×10^{-4}
USTC-2[3]	60	24	3.66×10^{-4}

[1] Arute, F. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505–510 (2019).
[2] Wu, Y. *et al.* Strong Quantum Computational Advantage Using a Superconducting Quantum Processor. *Phys. Rev. Lett.* **127**, 180501 (2021).
[3] Zhu, Q. *et al.* Quantum Computational Advantage via 60-Qubit 24-Cycle Random Circuit Sampling. (2021) doi:10.48550/arXiv.2109.03494.

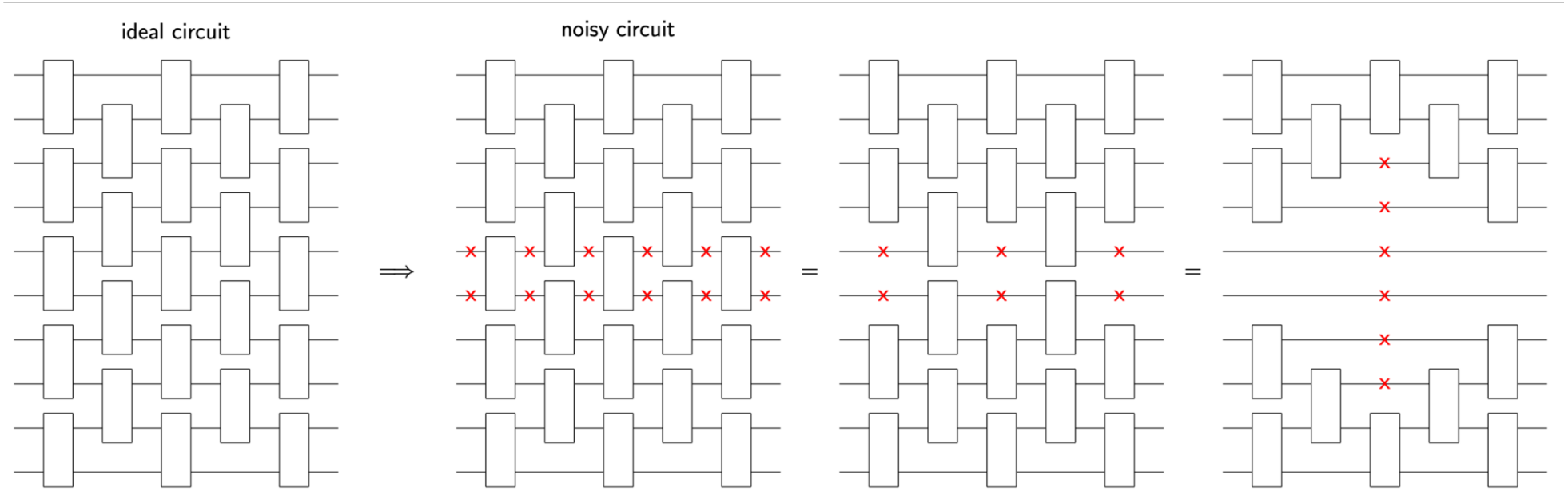
Spoofing algorithms

- Pan et al. 2021[10] (better XEB than Google with 512 GPU in 1 day)
- Gao et al. 2021[11] (~10% XEB of Google's with 1 GPU in a few seconds)

[10] Pan, F., Chen, K. & Zhang, P. Solving the sampling problem of the Sycamore quantum supremacy circuits. *arXiv:2111.03011 [physics, physics:quant-ph]* (2021).
[11] Gao, X. *et al.* Limitations of Linear Cross-Entropy as a Measure for Quantum Advantage. *arXiv:2112.01657 [cond-mat, physics:quant-ph]* (2021).

XEB.jl: an implementation of the spoofing algorithm in Gao et al.

- Add noises to divide and simplify the circuit
- Simulate each part with tensor networks
- Output bitstring with highest probability



Features of XEB.jl

- Random quantum circuit representation and simplification (Multigraphs.jl)
- Tensor network generation and contraction (OMEinsum.jl)
- Circuit simplification and simulation (Yao.jl)
- Contraction order optimization (OMEinsumContractionOrders.jl)
- Tensor network visualization (Makie.jl)
- GPU support (CUDA.jl)

https://github.com/ChenZhao44/XEB.jl

An Example

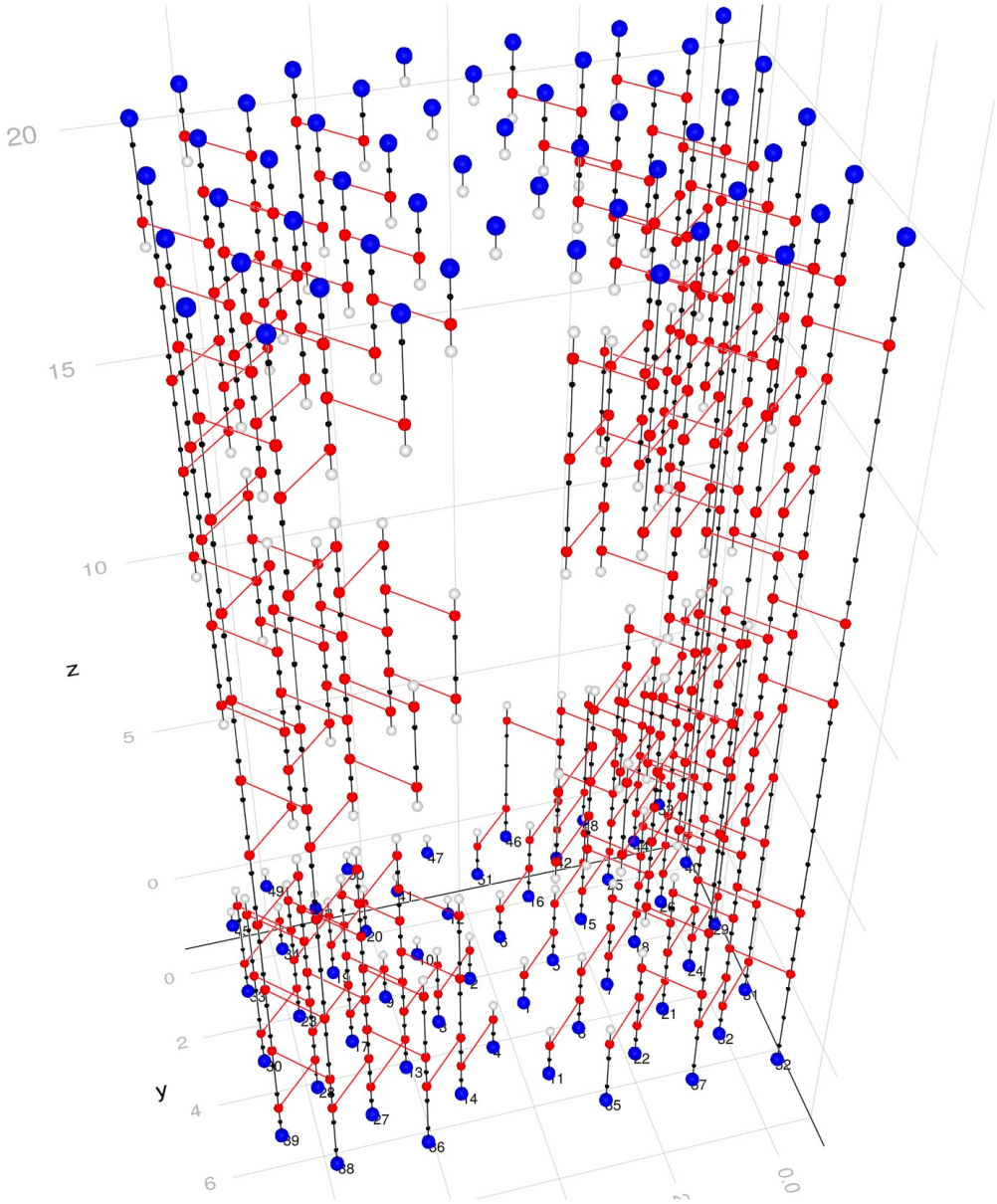
```julia
using XEB

D = 20  # circuit depth
layout = google_layout_53(53, D)

# left part and right part
L = sort!([52, 37, 35, 32, 22, 11, 31, 21, 8, 24, 7, 1,
    29, 18, 5, 26, 15, 6, 40, 25, 16, 44, 42, 51, 53, 48, 46])
R = setdiff(1:53, L)

# add noises and simplify the circuit
cuts = XEB.generate_cut(layout, L, 1, D)
XEB.simplify!(layout, cuts)

# visualize the tensor network
XEB.plot3d(layout)
```



Can we do better?

Estimate linear XEB without sampling

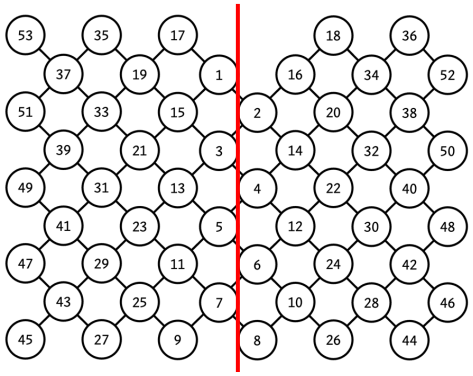
Original method (requires sampling)

1. Sample a random quantum circuit
2. Compute the output probability distribution with a tensor network (closed bound dimension 4 + open bound dimension 2)
3. Compute XEB from the probability distribution
4. Take average over all samples

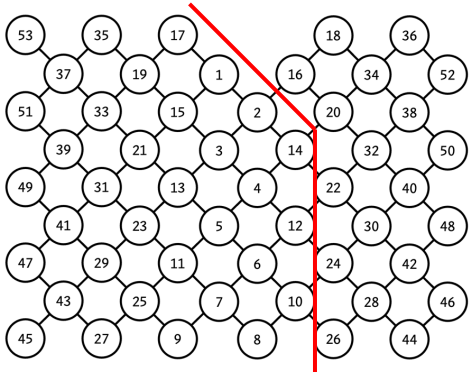
New method (without sampling)

1. Using the ZX-calculus to represent XEB as a tensor network (closed bound dimension 3 + no open indices)
2. Contract this tensor network and get XEB

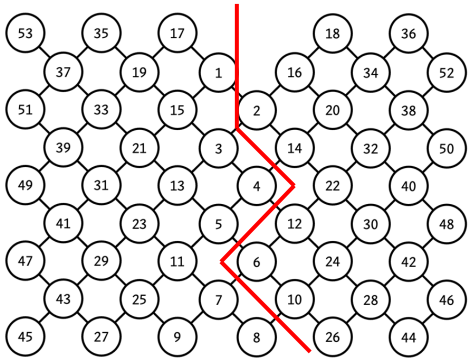
Different cuts



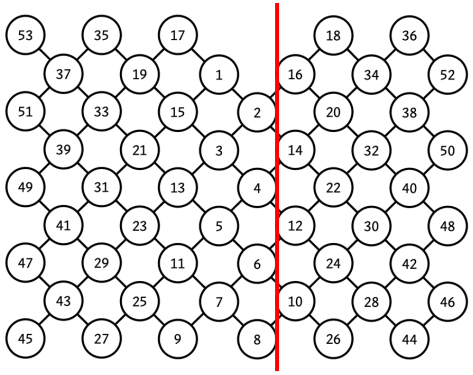
$\chi = 1.523\times10^{-8}$



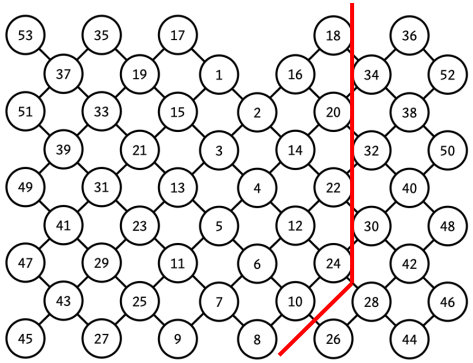
$\chi = 1.554\times10^{-8}$



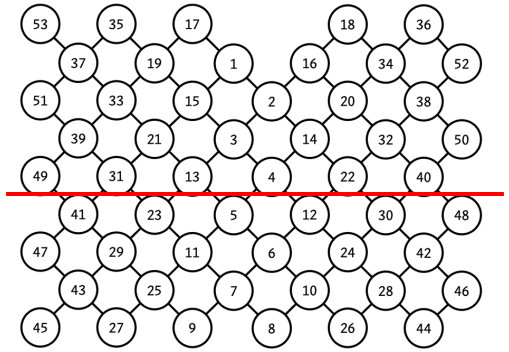
$\chi = 1.525\times10^{-8}$



$\chi = 1.508\times10^{-8}$



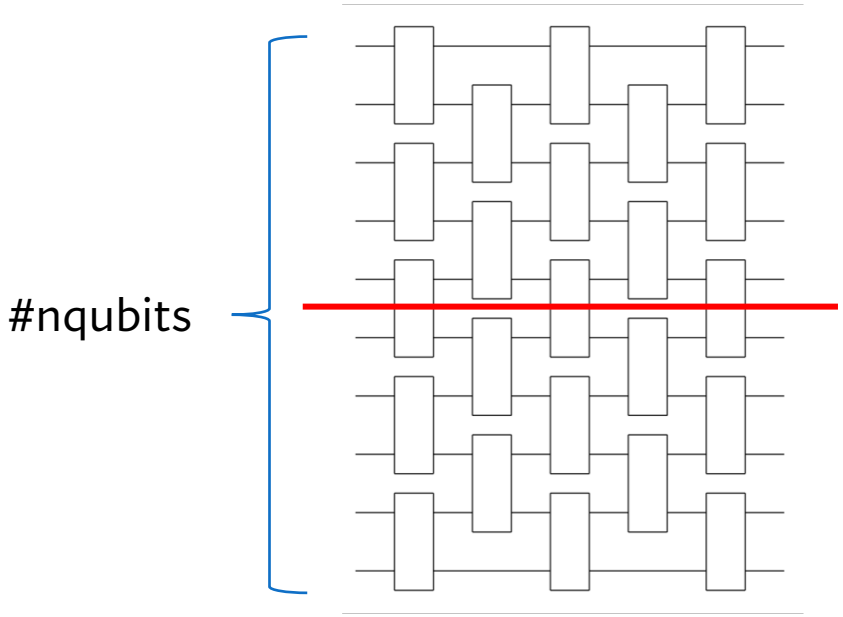
$\chi = 1.229\times10^{-8}$

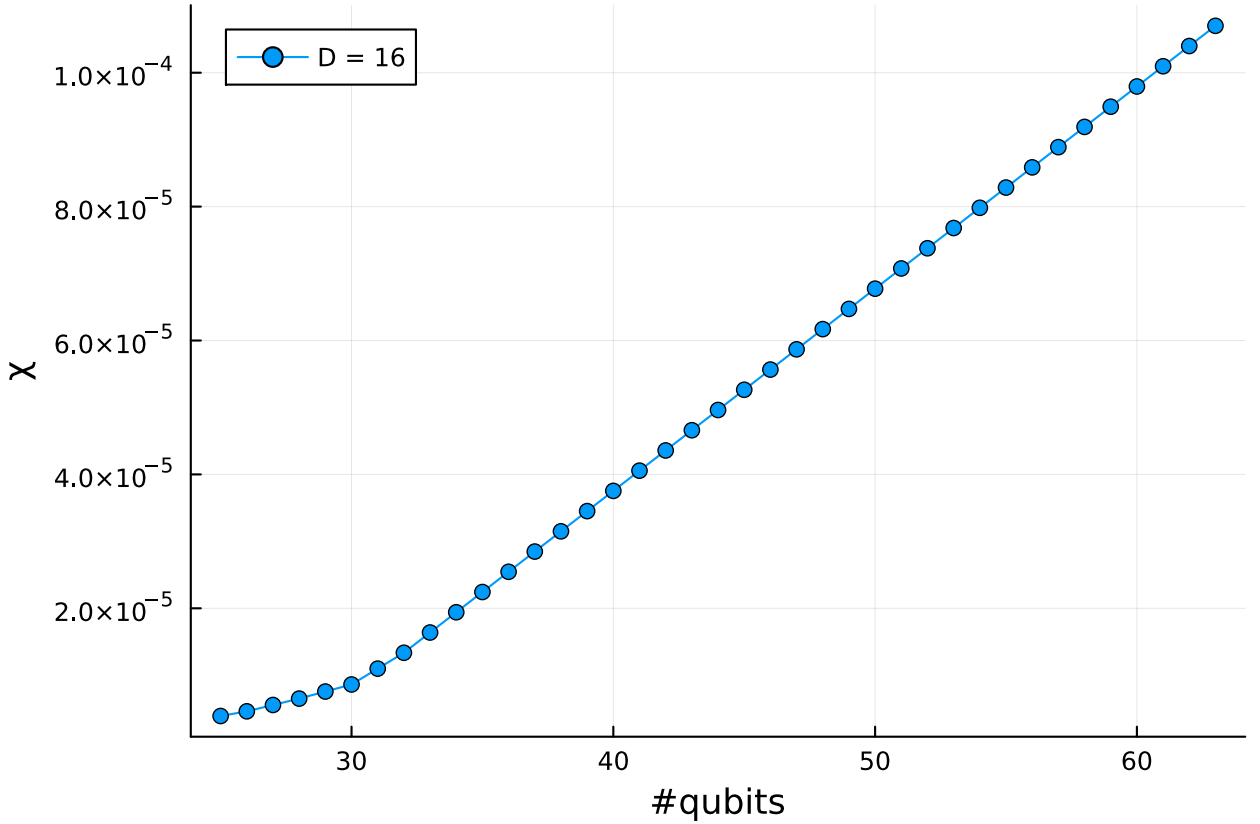


$\chi = 1.684\times10^{-8}$

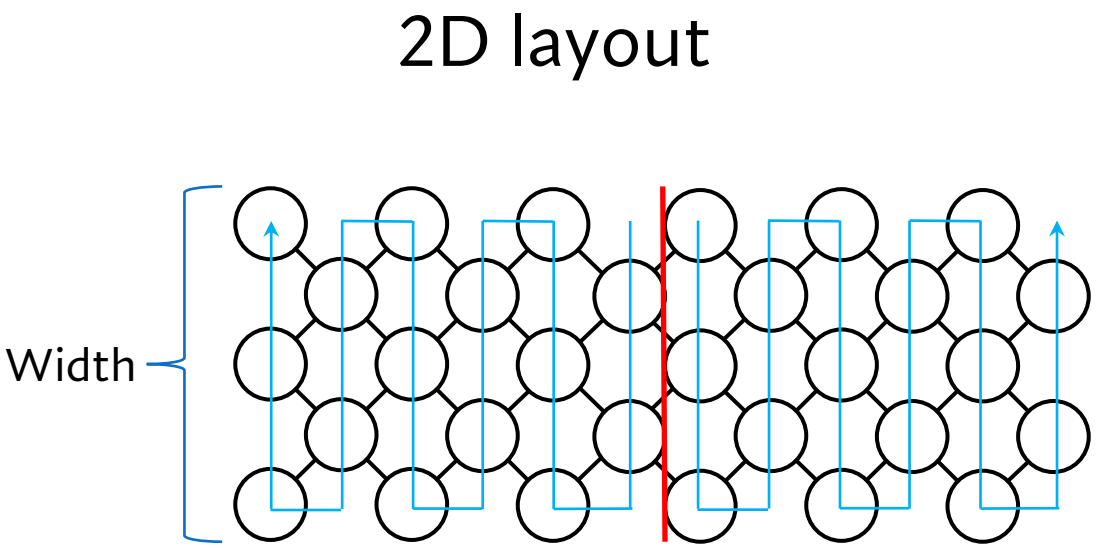
Different number of qubits

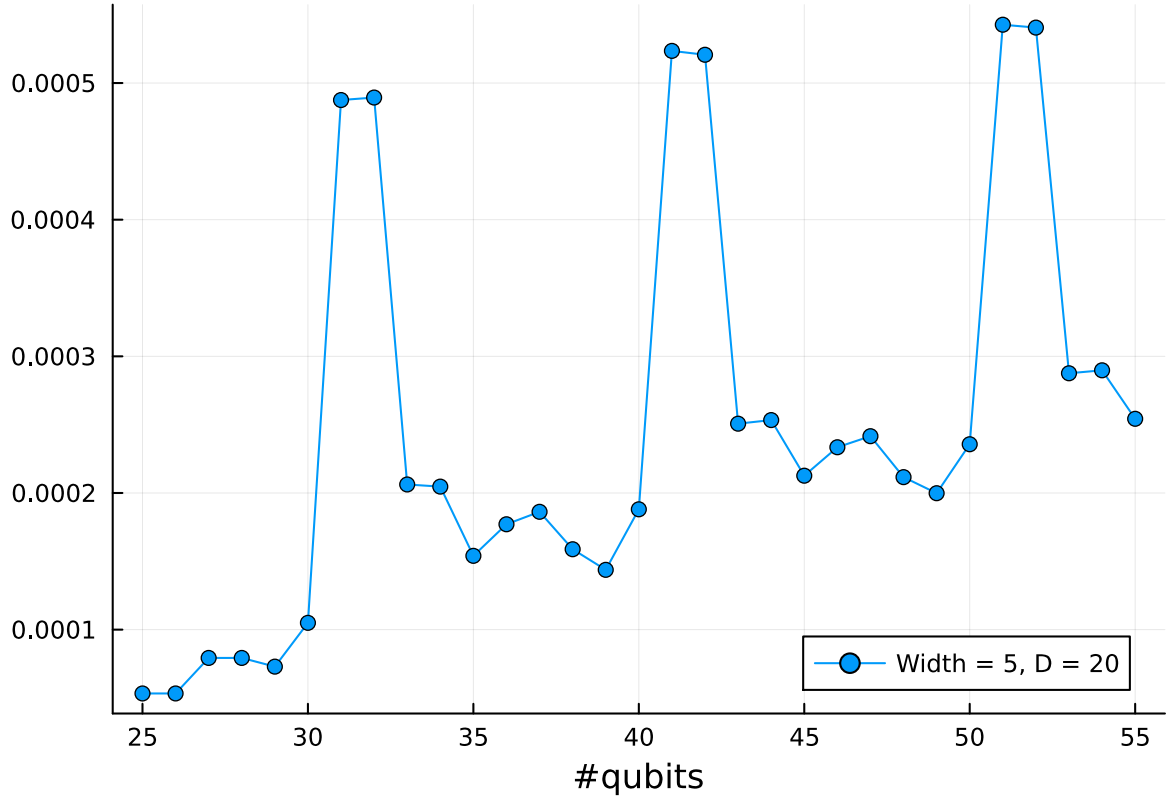
1D layout





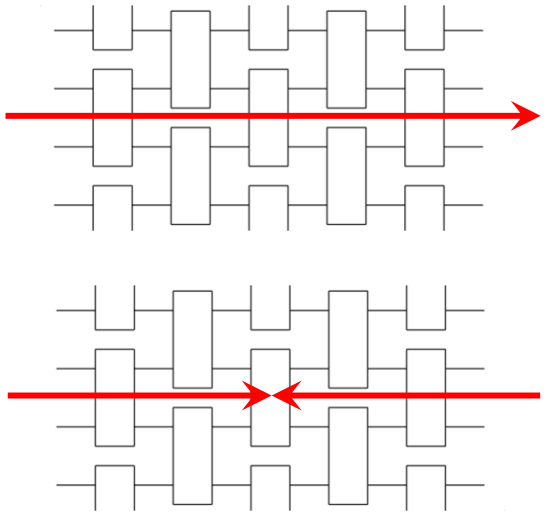
Different number of qubits

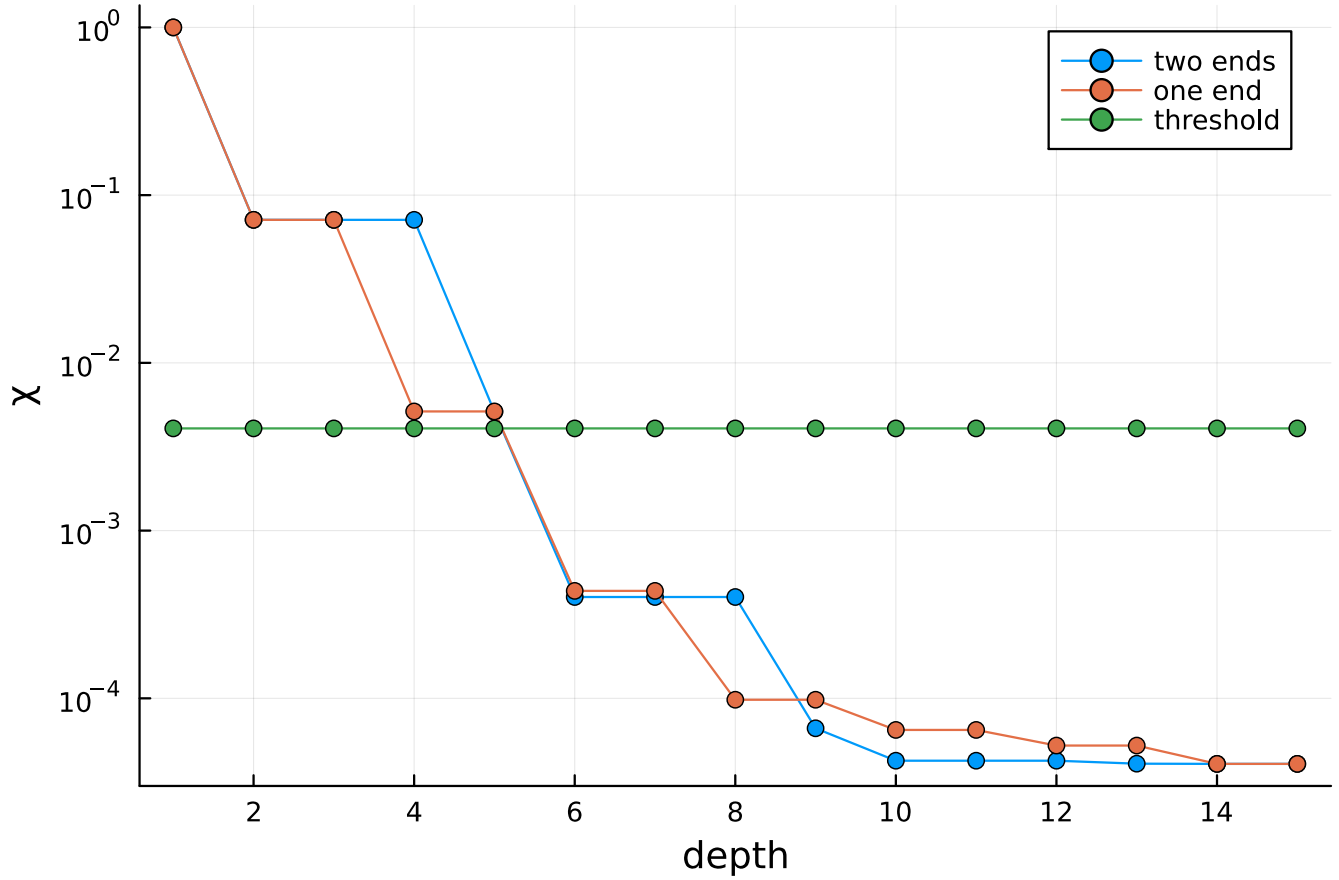




Different depth of cuts

Cutting orders





Conclusion

- Still difficult to get better XEB on a laptop
- Can outdo experiments if qubit number increase
- Julia is a developed platform for tensor networks

Acknowledgment

- Jinguo Liu
- Roger Luo
- Xun Gao
- Xiao-Shan Gao
- Arthur Jaffe

Thank you!