Title: Contracting Arbitrary Tensor Networks: Approximate and Exact Approach with Applications in Graphical Models and Quantum Circuit Simulations

Speakers: Feng Pan

Series: Perimeter Institute Quantum Discussions

Date: November 22, 2021 - 9:30 AM

URL: https://pirsa.org/21110038

Abstract: Tensor network algorithms are numerical tools widely used in physical research. But traditionally they are only applied to lattice systems with specific structure. In this talk, tensor network algorithms to deal with physical systems with arbitrary topology will be discussed. Theoretical framework will firstly be constructed to analyze the difficulty of contracting an arbitrary tensor network. Then both approximate and exact contraction approaches will be involved according to computational tasks of interest. Finally two applications, one in graphical models and the other in quantum circuit simulations, will be introduced to demonstrate the performance and potential of arbitrary tensor network algorithms.

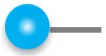
Contracting Arbitrary Tensor Networks: Approximate and Exact Approach with Applications in Graphical Models and Quantum Circuit Simulations

Feng Pan

Institute of Theoretical Physics, Chinese Academy of Sciences

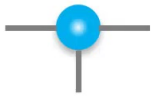
Tensor Network Basics

Scalar



Vector

Matrix

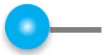


Tensor

Tensor Network Basics

Scalar

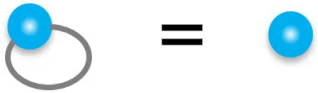
Matrix-vector Multiplication



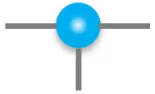
Vector

Matrix Multiplication

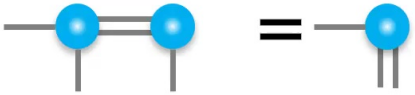
Matrix



Trace



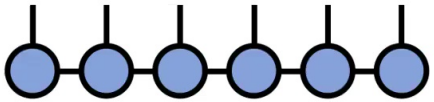
Tensor



Tensor Contraction

Motivation

Matrix Product State /
Tensor Train



PEPS

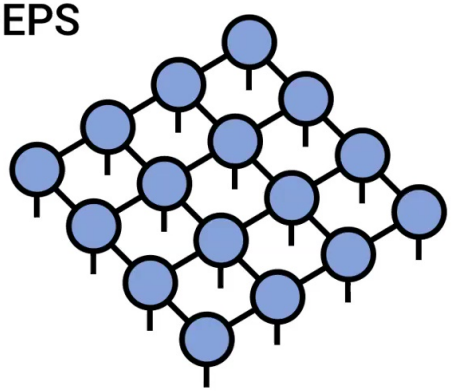
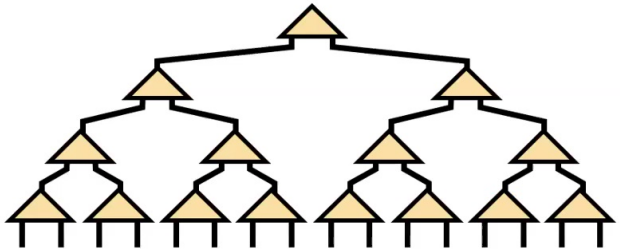
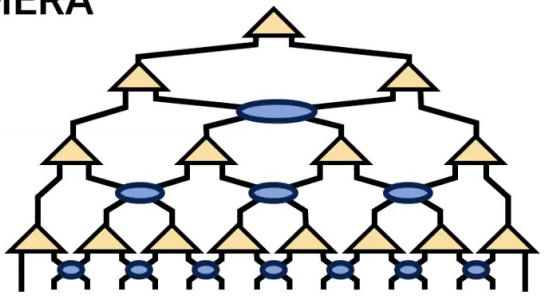


Figure credit:
http://tensornetwork.org/

Tree Tensor Network /
Hierarchical Tucker



MERA



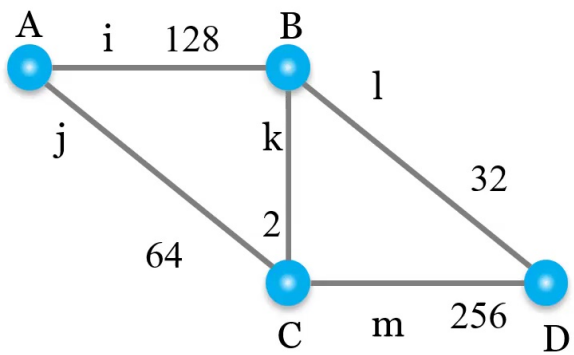
Warmup: Einstein summation convention

Problems involving TN with Arbitrary topology

- Physical problems defined on random graphs (graphical models)

- Random Quantum Circuit (RQC) simulations

Warmup: Einstein summation convention



$$\sum_{ijklm} A_{ij} B_{ikl} C_{jkm} D_{lm}$$

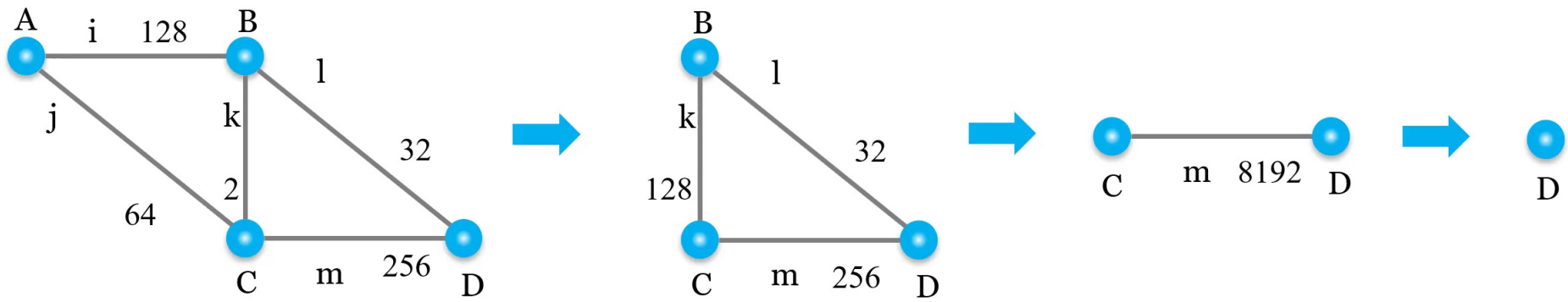
$$A_{ij} B_{ikl} C_{jkm} D_{lm}$$

```
In [1]: import torch

In [2]: A = torch.randn(128, 64)

In [3]: B = torch.randn(128, 2, 32)

In [4]: C = torch.randn(256, 2, 64)

In [5]: D = torch.randn(32, 256)

In [6]: %timeit -n 100 torch.einsum('ab, acd, ecb, de', A, B, C, D)
411 µs ± 130 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

In [7]: %timeit -n 100 torch.einsum('acd, ecb, ab, de', B, C, A, D)
95.5 ms ± 1.92 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

https://github.com/dgasmith/opt_einsum

Warmup: Einstein summation convention

- Time complexity: measure the number of floating-point operations (TC)
- Space complexity: measure the size of largest intermediate tensor (SC)



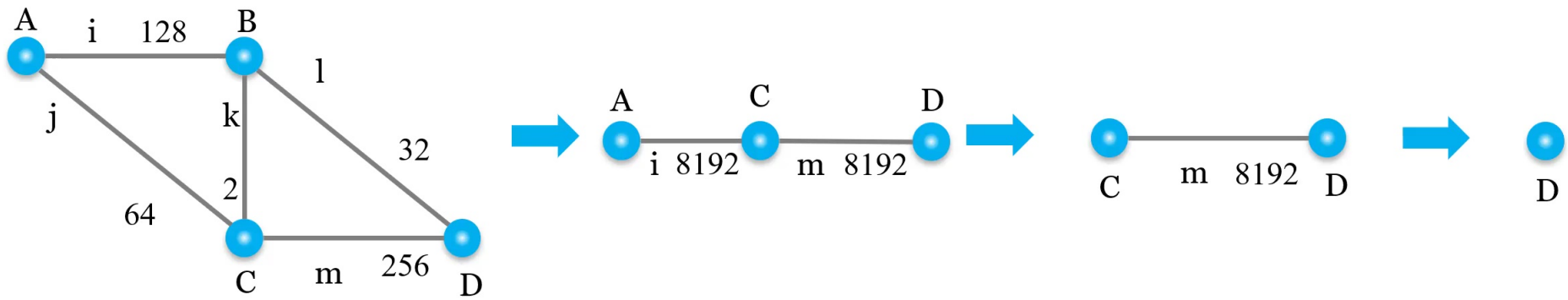
SC: 2^{15} TC: 2^{19} SC: 2^{15} TC: 2^{20} SC: 2^{13} TC: 2^{13}

$$\mathrm{TC}(\mathrm{ABCD}) \approx 2^{20}$$

$$\mathrm{SC}(\mathrm{ABCD}) = 2^{15}$$

Warmup: Einstein summation convention

- Time complexity: measure the number of floating-point operations (TC)
- Space complexity: measure the size of largest intermediate tensor (SC)



SC: 2^{15} TC: 2^{27} SC: 2^{26} TC: 2^{26} SC: 2^{13} TC: 2^{13}

$$\mathrm{TC}(\mathrm{ABCD}) \approx 2^{20}$$

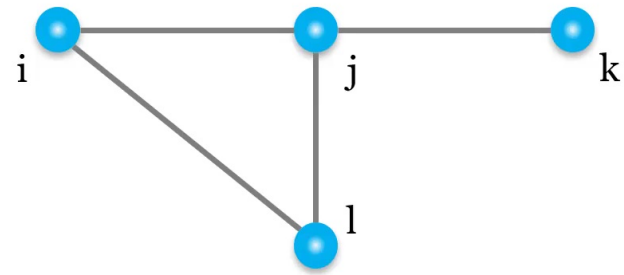
$$\mathrm{SC}(\mathrm{ABCD}) = 2^{15}$$

$$\mathrm{TC}(\mathrm{BCAD}) \approx 2^{27}$$

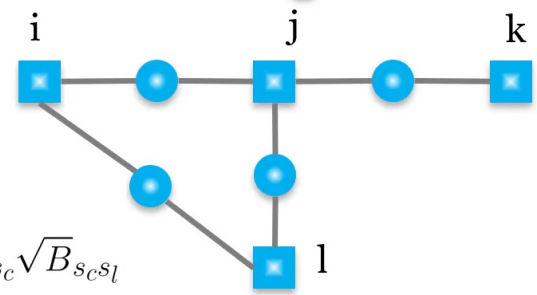
$$\mathrm{SC}(\mathrm{BCAD}) = 2^{26}$$

Physical problems defined on random graphs

$$Z = \sum_{s_i s_j s_k s_l} e^{\beta(J_{ij} s_i s_j + J_{jk} s_j s_k + J_{il} s_i s_l + J_{jl} s_j s_l)}$$

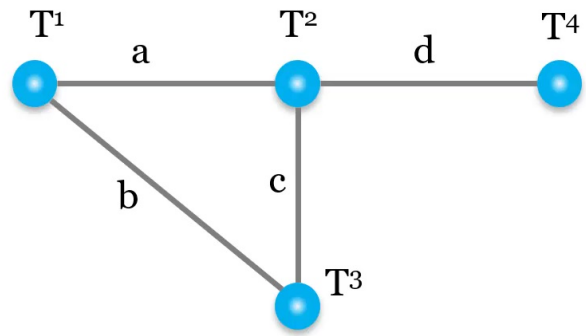


$$= \sum_{s_i s_j s_k s_l} B_{s_i s_j} B_{s_j s_k} B_{s_i s_l} B_{s_j s_l} \text{ where } B_{s_i s_j} = \begin{pmatrix} e^{\beta J_{ij}} & e^{-\beta J_{ij}} \\ e^{-\beta J_{ij}} & e^{\beta J_{ij}} \end{pmatrix}_{s_i s_j}$$



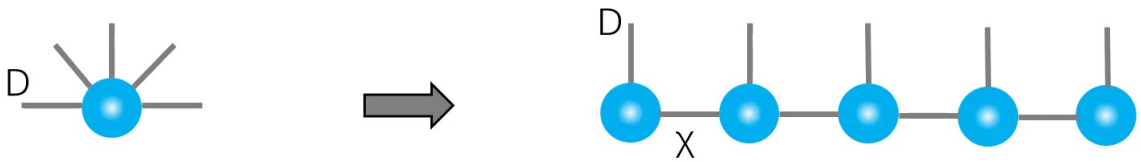
$$= \sum_{s_i s_j s_k s_l s_a s_b s_c s_d} \sqrt{B}_{s_i s_a} \sqrt{B}_{s_a s_j} \sqrt{B}_{s_j s_d} \sqrt{B}_{s_d s_k} \sqrt{B}_{s_i s_b} \sqrt{B}_{s_b s_l} \sqrt{B}_{s_j s_c} \sqrt{B}_{s_c s_l}$$

$$= \sum_{s_a s_b s_c s_d} T^1_{s_a s_b} T^2_{s_a s_c s_d} T^3_{s_b s_c} T^4_{s_d}$$



MPS calculus - Before contracting

Large intermediate tensor (space complexity):



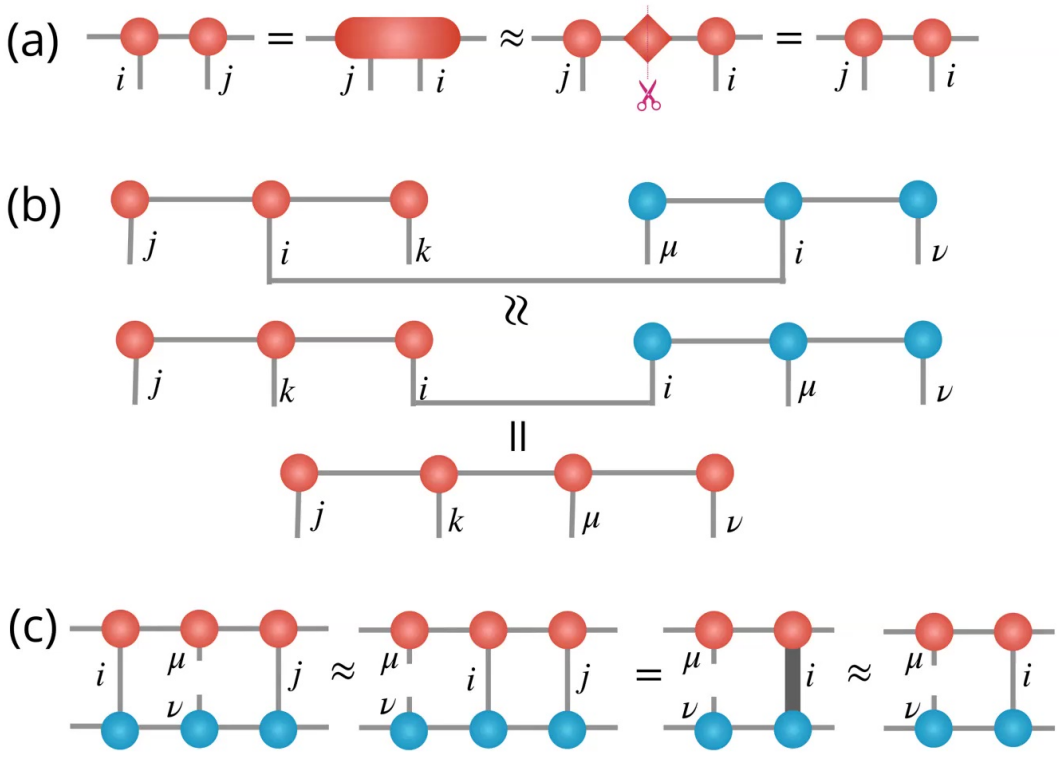
Time complexity:

Control D and χ by adopting singular value decompositions (SVD)

Contraction order:

Heuristically choose from current TN

MPS calculus - Contraction operations

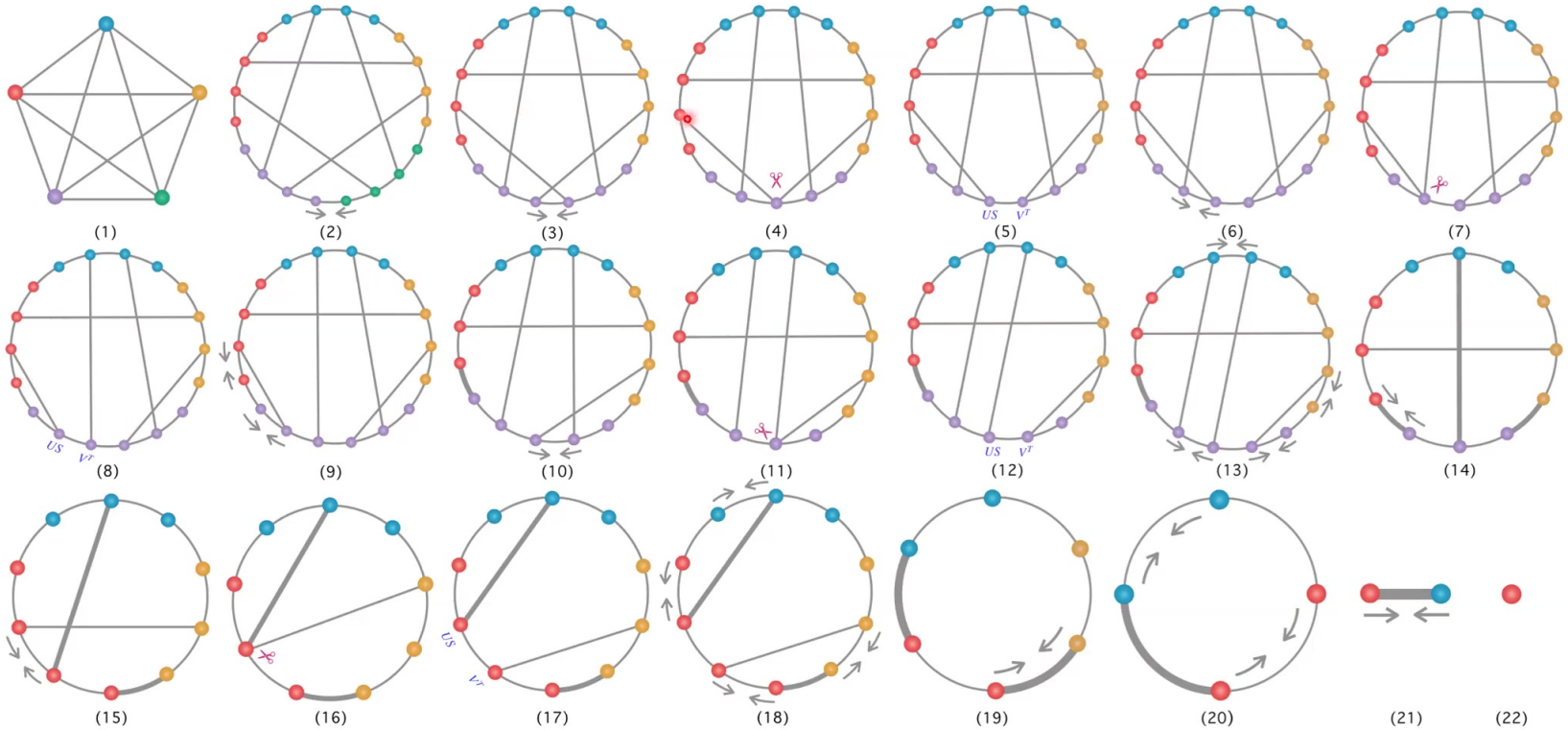


Swap

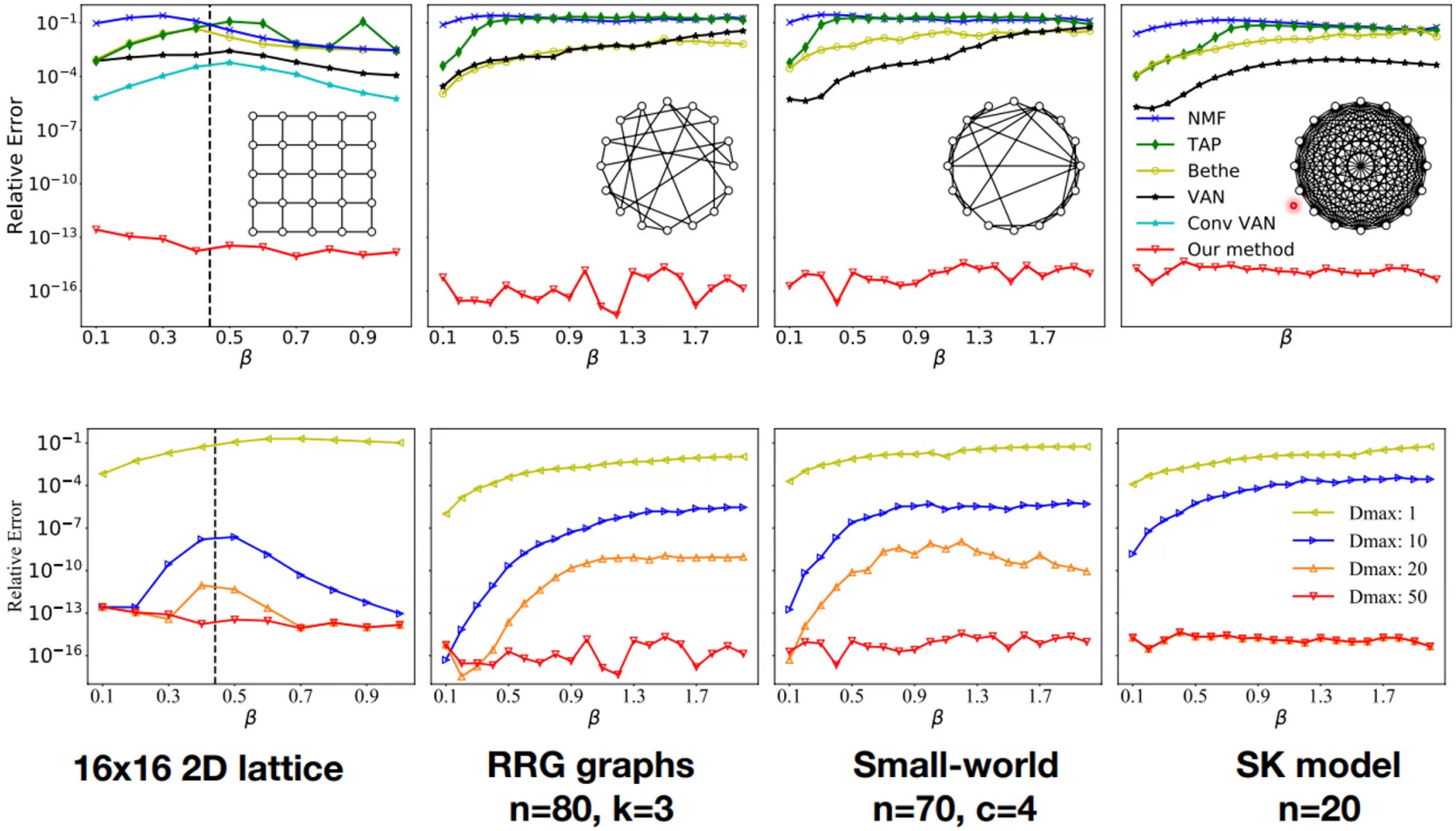
Contract

Merge

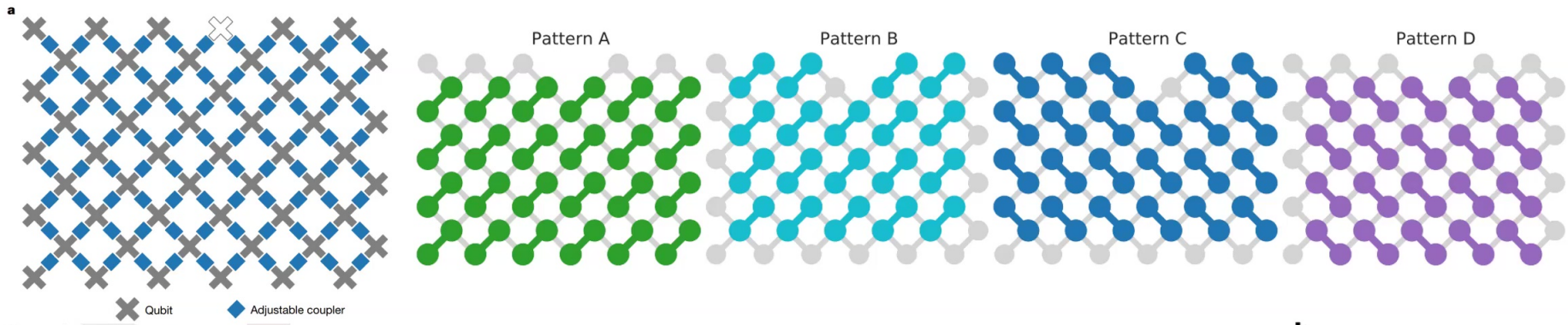
Contract an Arbitrary TN with an example

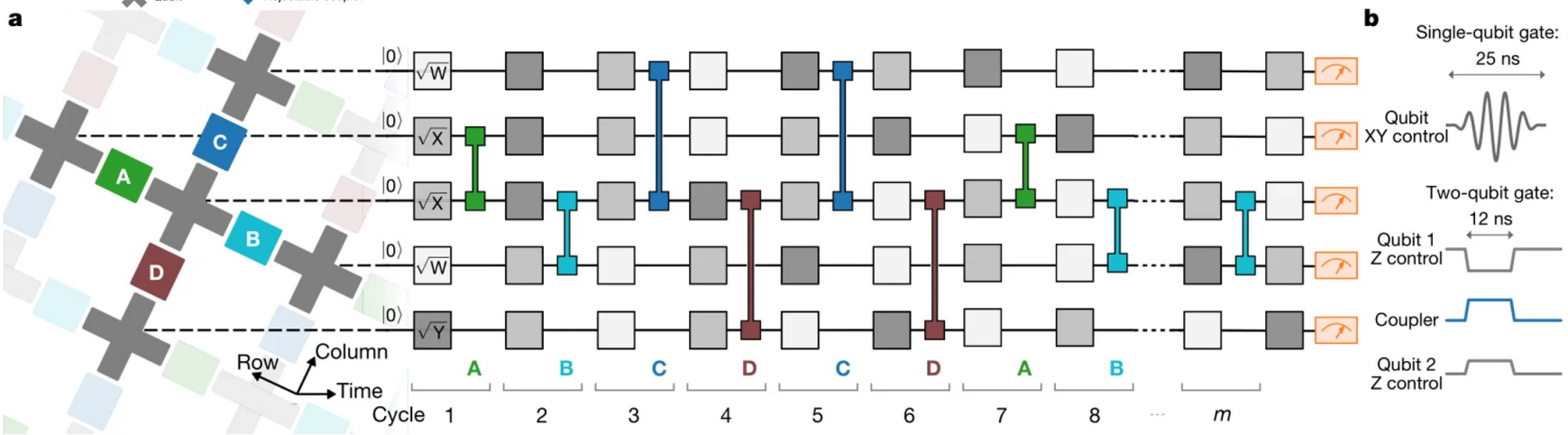


Some results on free energy calculation



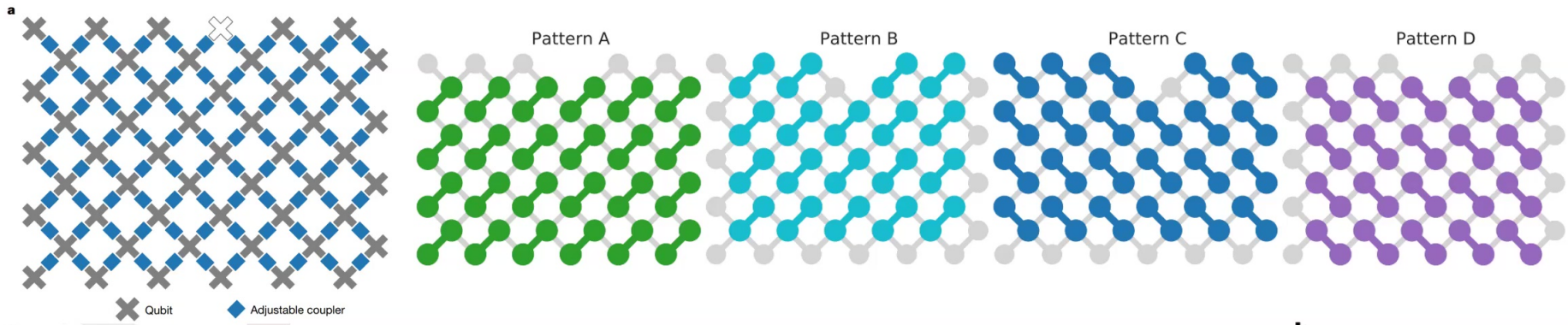
Sycamore quantum circuits and RQC sampling

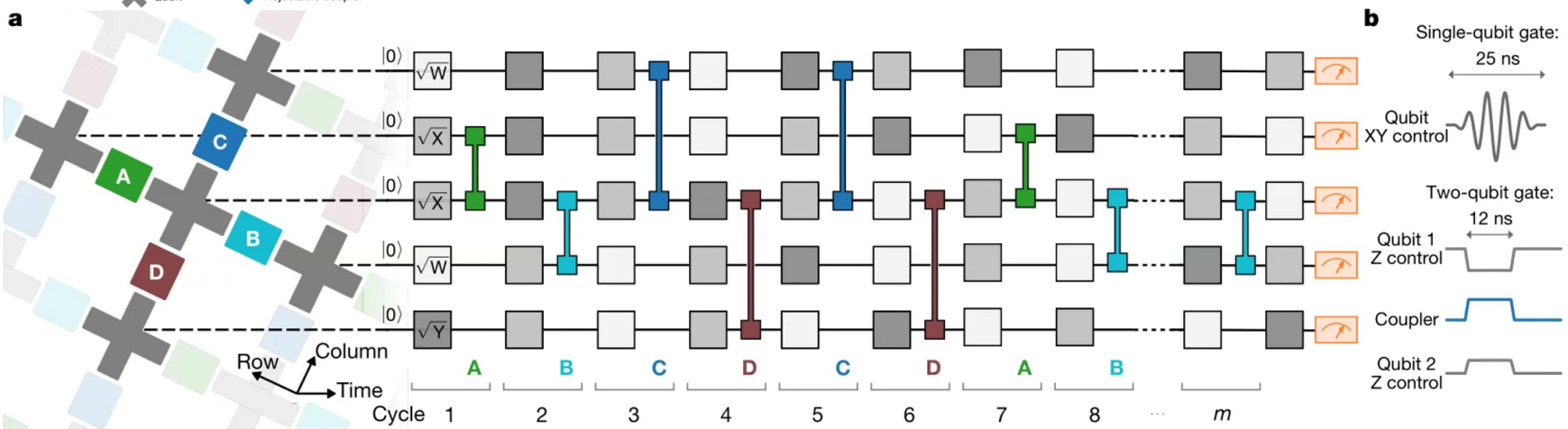




Arute, et al. *Nature* 574.7779 (2019): 505-510.

Sycamore quantum circuits and RQC sampling





200 seconds 1,000,000 bistrings 0.2% fidelity

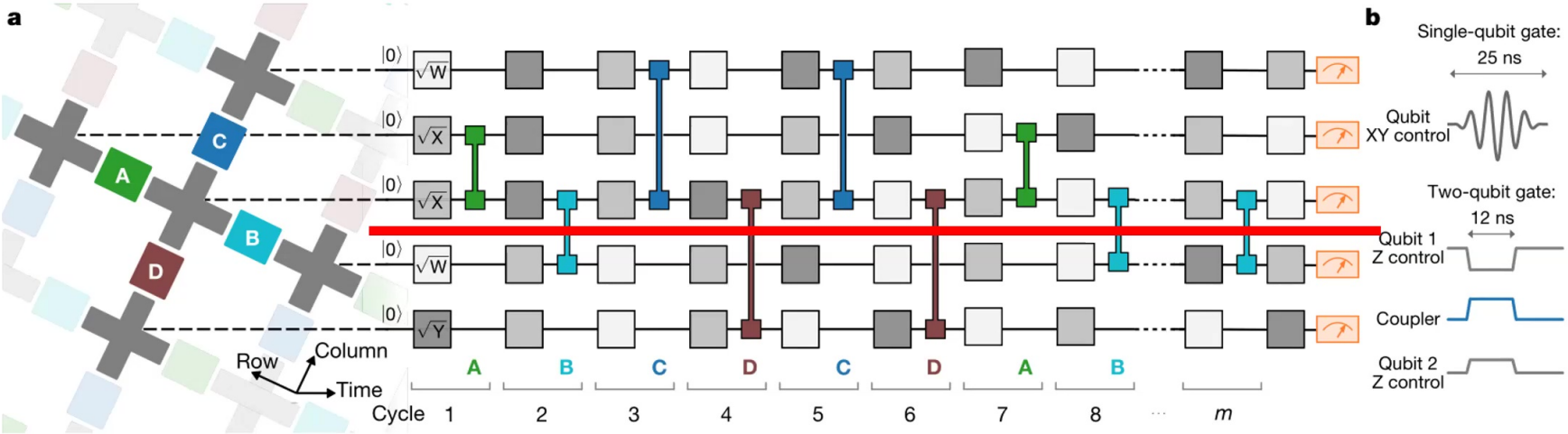
Arute, et al. *Nature* 574.7779 (2019): 505-510.

Quantum Supremacy demonstration

Classical simulation candidates:

1. Schrodinger-Feynman algorithm (Google's choice)

2. Tensor network based algorithm



Quantum Supremacy demonstration

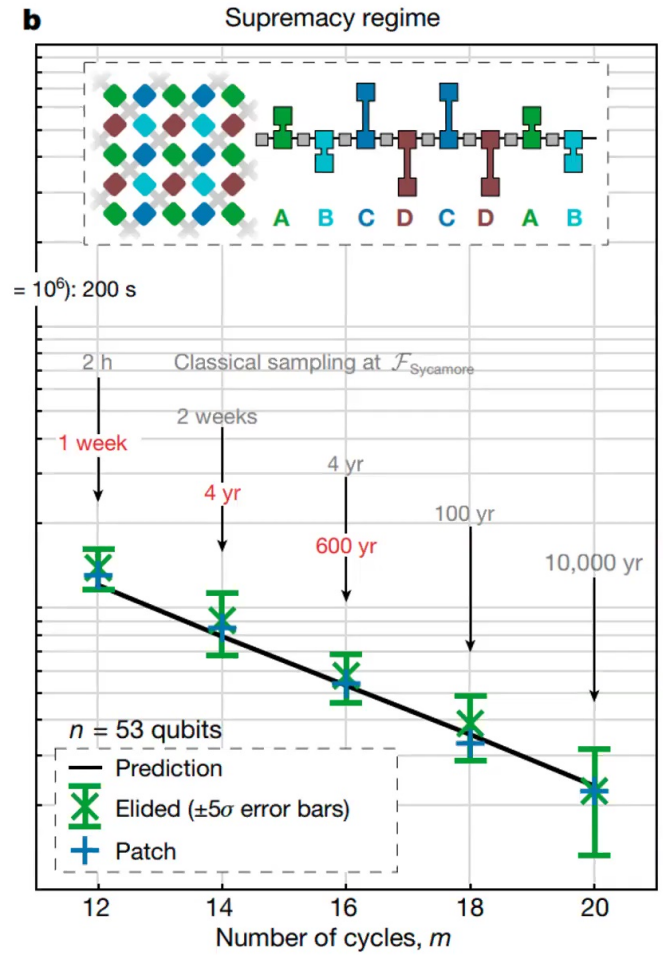
Classical simulation candidates:

1. Schrodinger-Feynman algorithm (Google's choice)

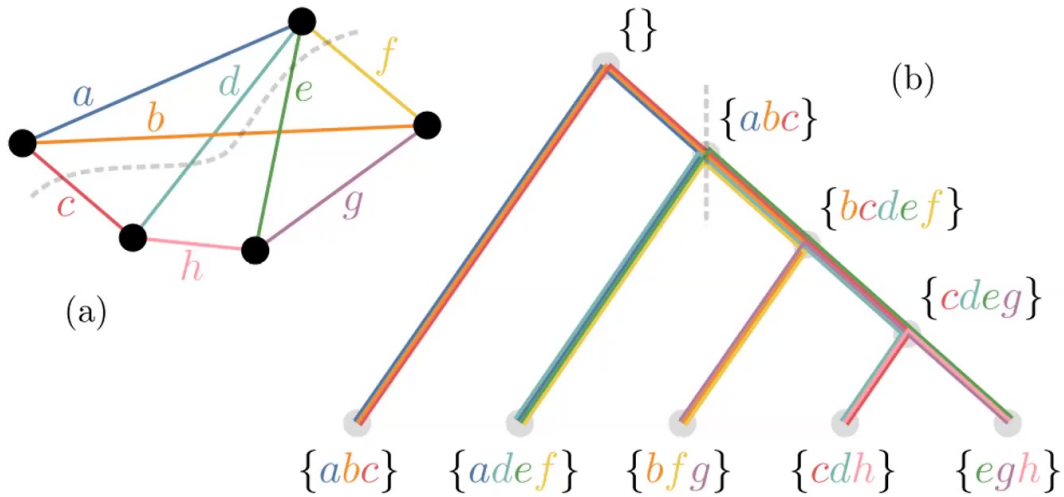
2. Tensor network based algorithm

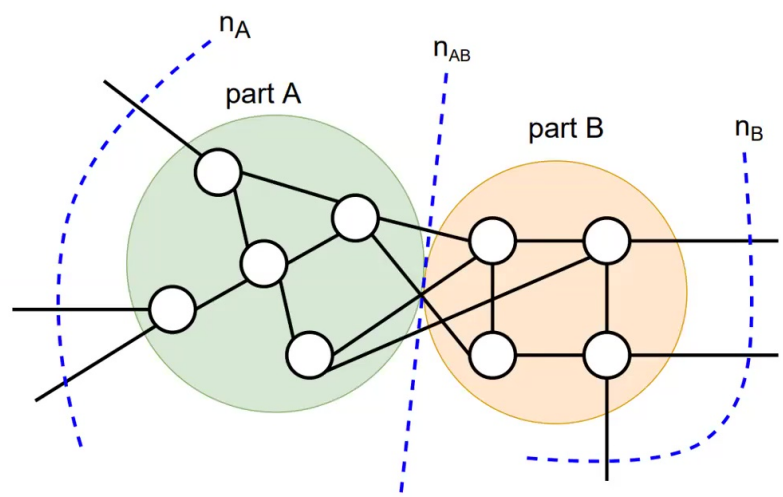
qubits, n	cycles, m	total #paths	fidelity	run time
53	12	$4^{17}2^4$	1.4%	2 hours
53	14	$4^{21}2^4$	0.9%	2 weeks
53	16	$4^{25}2^3$	0.6%	4 years
53	18	$4^{28}2^3$	0.4%	175 years
53	20	$4^{31}2^4$	0.2%	10000 years

TABLE XI. Approximate qsimh run times using one million CPU cores extrapolated from the average simulation run time for 1000 simulation paths on one CPU core.



Contraction Tree and Bipartition





Time complexity: $2^{n_A} \cdot 2^{n_{AB}} \cdot 2^{n_B}$

Space complexity: $\max(2^{n_A+n_{AB}}, 2^{n_{AB}+n_B}, 2^{n_A+n_B})$

Gray, Kourtis. *Quantum* 5 (2021): 410.

Fidelity and Linear cross entropy benchmark (XEB)

- Traditional definition of fidelity $$\mathcal{F} = \frac{|\langle\psi_n|\psi\rangle|^2}{|\langle\psi_n|\psi_n\rangle|^2|\langle\psi|\psi\rangle|^2}$$

- Since RQC sampling can not acquire all samples, use XEB fidelity instead

$$\mathcal{F}_{\mathrm{XEB}} = \frac{2^n}{L}\sum_{i=1}^{L} p(x_i) - 1$$

Fidelity and Linear cross entropy benchmark (XEB)

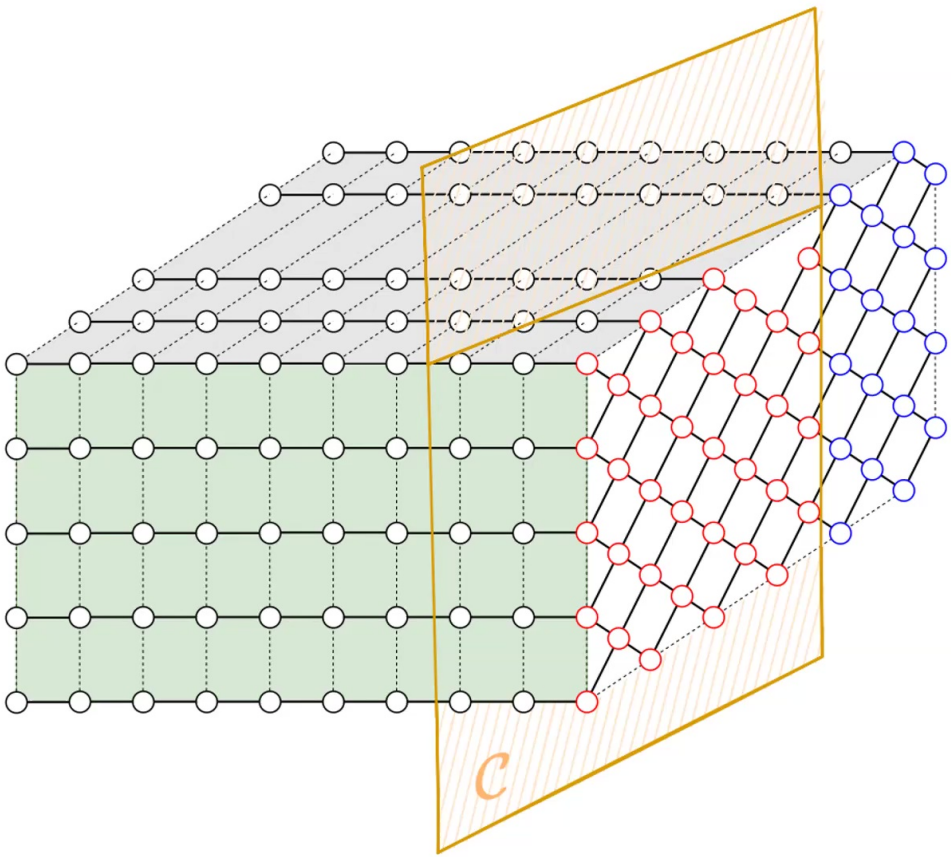
- Traditional definition of fidelity $$\mathcal{F} = \frac{|\langle\psi_n|\psi\rangle|^2}{|\langle\psi_n|\psi_n\rangle|^2|\langle\psi|\psi\rangle|^2}$$

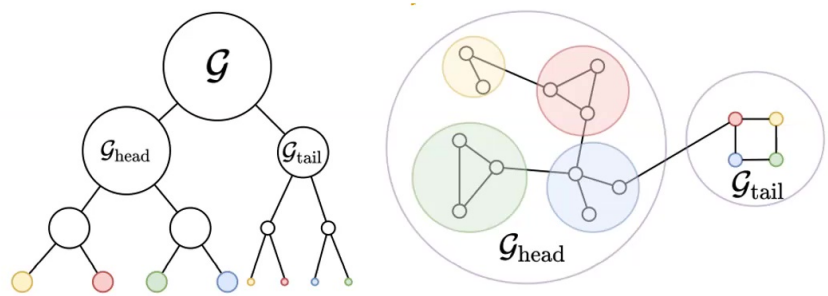
- Since RQC sampling can not acquire all samples, use XEB fidelity instead

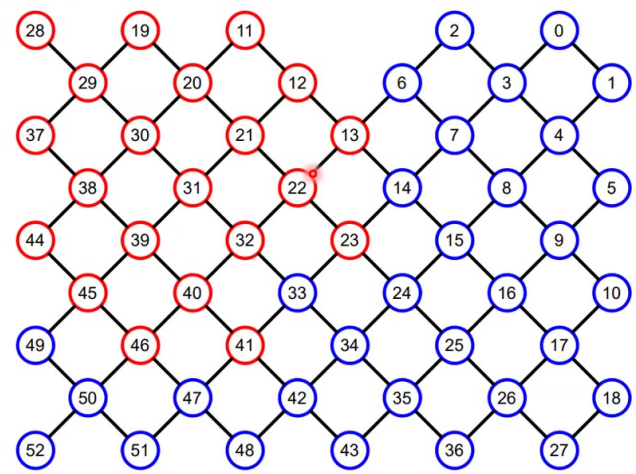
$$\mathcal{F}_{\mathrm{XEB}} = \frac{2^n}{L}\sum_{i=1}^{L} p(x_i) - 1$$

- XEB can be spoofed: samples with large probabilities

Bighead method

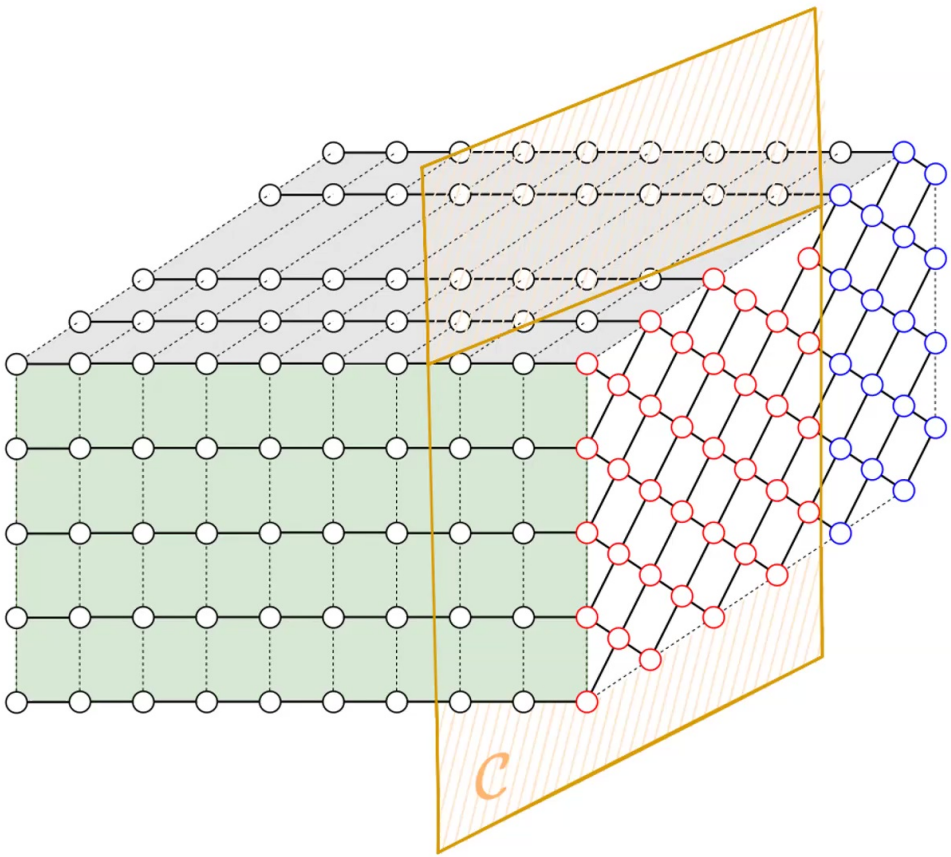


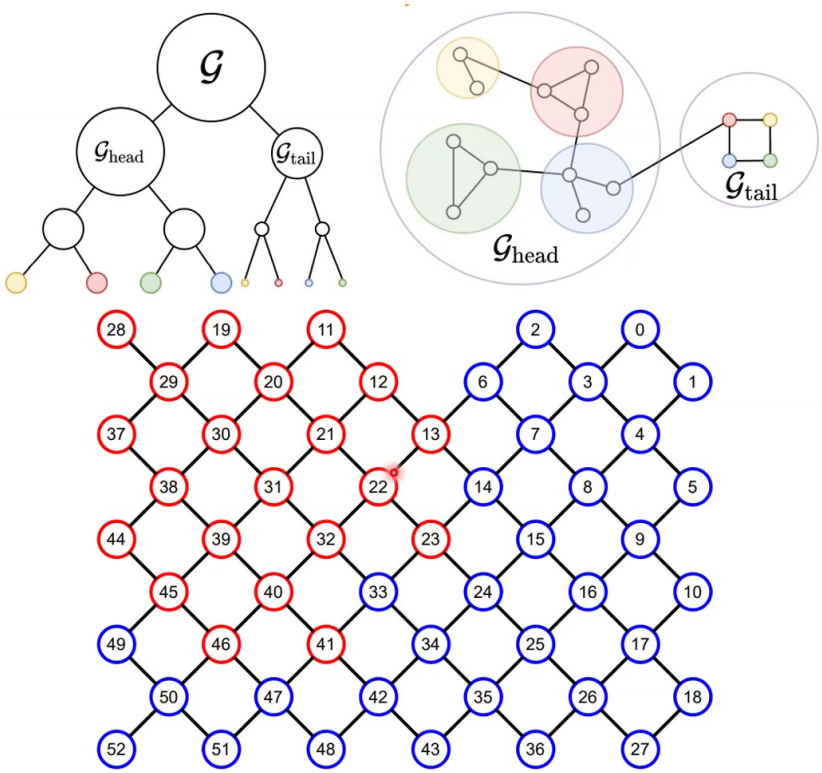




20 cycles

Bighead method





20 cycles

https://github.com/kahypar/kahypar

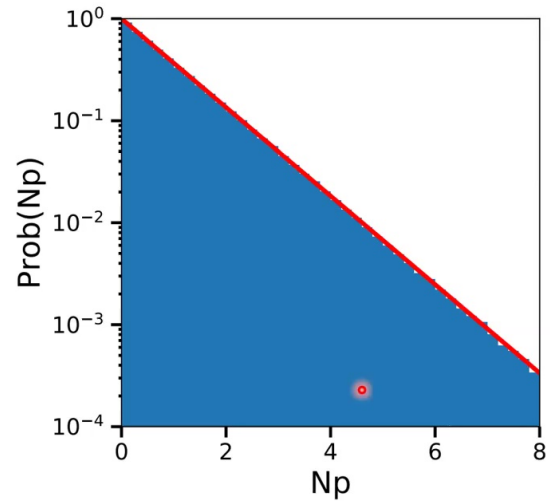
Results and the spoofing of XEB

	# bitstrings	Time complexity	Space complexity	Computational time	Computational hardware
Google [1]	10^6	—	—	10,000 years	Summit supercomputer
Cotengra [12]	1	3.10×10^{22}	2^{27}	3,088 years	One NVIDIA Quadro P2000
Alibaba [18]	64	6.66×10^{18}	2^{29}	267 days	One V100 GPU
Ours	**2097152**	4.51×10^{18}	2^{30}	149 Days	One A100 GPU

TABLE II. Comparison of computational cost among different methods on Sycamore circuit with 53 qubits and 20 cycles.

5 days in 60 GPUs

bitstring	amplitude	probability
0000000000000000000000010000011010000001000010000000	$-2.97 \times 10^{-8} + 2.06 \times 10^{-8}i$	1.31×10^{-15}
0000000000000000000000000000000000000000100001000000	$1.50 \times 10^{-8} + 3.85 \times 10^{-9}i$	2.39×10^{-16}
0000000000011100000111110000111110000111110011000000	$-3.17 \times 10^{-9} - 5.45 \times 10^{-9}i$	3.97×10^{-17}
0000000000011100000111110000111110000111100010100000	$-1.89 \times 10^{-10} + 3.13 \times 10^{-9}i$	9.86×10^{-18}
0000000000000000000000000000010000000001110001000000	$8.07 \times 10^{-10} + 4.35 \times 10^{-10}i$	8.41×10^{-19}



Results and the spoofing of XEB

	# bitstrings	Time complexity	Space complexity	Computational time	Computational hardware
Google [1]	10^6	—	—	10,000 years	Summit supercomputer
Cotengra [12]	1	3.10×10^{22}	2^{27}	3,088 years	One NVIDIA Quadro P2000
Alibaba [18]	64	6.66×10^{18}	2^{29}	267 days	One V100 GPU
Ours	**2097152**	4.51×10^{18}	2^{30}	149 Days	One A100 GPU

TABLE II. Comparison of computational cost among different methods on Sycamore circuit with 53 qubits and 20 cycles.

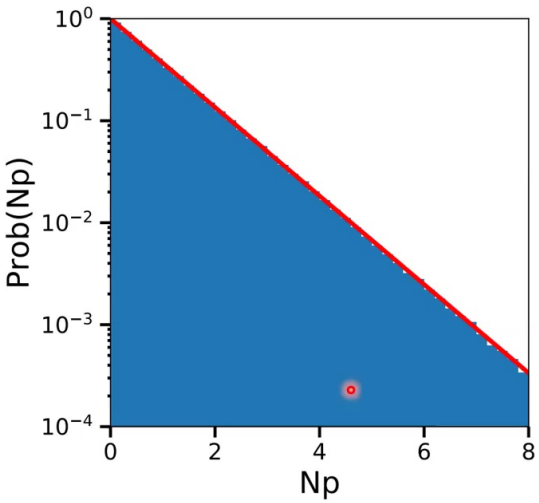
5 days in 60 GPUs

bitstring	amplitude	probability
0000000000000000000000010000011010000001000010000000	$-2.97 \times 10^{-8} + 2.06 \times 10^{-8}i$	1.31×10^{-15}
0000000000000000000000000000000000000000100001000000	$1.50 \times 10^{-8} + 3.85 \times 10^{-9}i$	2.39×10^{-16}
0000000000011100000111110000111110000111110011000000	$-3.17 \times 10^{-9} - 5.45 \times 10^{-9}i$	3.97×10^{-17}
0000000000011100000111110000111110000111100010100000	$-1.89 \times 10^{-10} + 3.13 \times 10^{-9}i$	9.86×10^{-18}
0000000000000000000000000000010000000000111000100000	$8.07 \times 10^{-10} + 4.35 \times 10^{-10}i$	8.41×10^{-19}

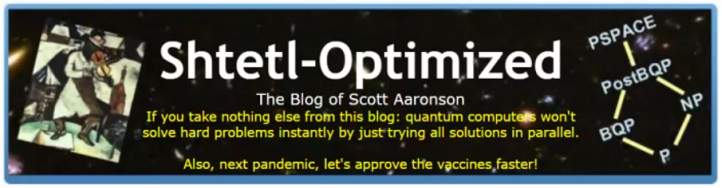
2,097,152 bitstring samples with 0% fidelity

1,000,000 bitstring samples with 73.9% fidelity

>> 0.2%



Feedbacks of this work



« The Zen Anti-Interpretation of Quantum Mechanics Sayonara Majorana? »

Another axe swung at the Sycamore

So there's an interesting new paper on the arXiv by Feng Pan and Pan Zhang, entitled "Simulating the Sycamore supremacy circuits." It's about a new tensor contraction strategy for classically simulating Google's 53-qubit quantum supremacy experiment from Fall 2019. Using their approach, and using just 60 GPUs running for a few days, the authors say they managed to generate a million *correlated* 53-bit strings—meaning, strings that all agree on a specific subset of 20 or so bits—that achieve a high linear cross-entropy score.

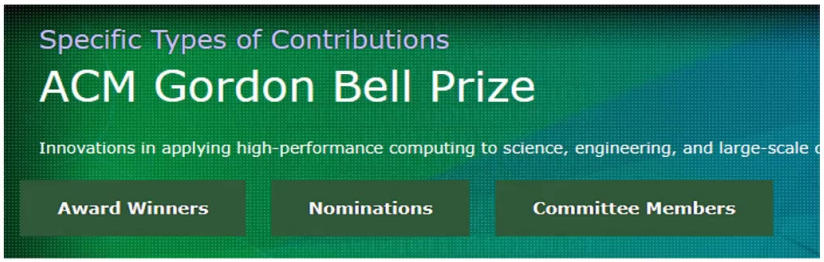
Alas, I haven't had time this weekend to write a "proper" blog post about this, but several people have by now emailed to ask my opinion, so I thought I'd share the brief response I sent to a journalist.

This does look like a significant advance on simulating Sycamore-like random quantum circuits! Since it's based on tensor networks, you don't need the literally largest supercomputer on the planet filling up tens of petabytes of hard disk space with amplitudes, as in the brute-force strategy proposed by IBM. Pan and Zhang's strategy seems most similar to the strategy previously proposed by Alibaba, with the key difference being that the new approach generates millions of correlated samples rather than just one.

I guess my main thoughts for now are:

1. Once you knew about this particular attack, you could evade it and get back to where we were before by switching to a more sophisticated verification test — namely, one where you not only computed a Linear XEB score for the observed samples, you *also* made sure that the samples didn't share too many bits in common. (Strangely, though, the paper never mentions this point.)
2. The other response, of course, would just be to redo random circuit sampling with a slightly bigger quantum computer, like the ~70-qubit devices that Google, IBM, and others are now building!

Anyway, very happy for thoughts from anyone who knows more.



Home > ACM Gordon Bell Prize

About ACM Gordon Bell Prize

The Gordon Bell Prize is awarded each year to recognize outstanding achievement in high-performance computing. The purpose of the award is to track the progress over time of parallel computing, with particular emphasis on rewarding innovation in applying high-performance computing to applications in science, engineering, and large-scale data analytics. Prizes may be awarded for peak performance or special achievements in scalability and time-to-solution on important science and engineering problems. Financial support of the $10,000 award is provided by Gordon Bell, a pioneer in high-performance and parallel computing.

Recent Gordon Bell Prize News

2021 ACM Gordon Bell Prize Awarded to Team for Achieving Real-Time Simulation of Random Quantum Circuit

ACM, the Association for Computing Machinery, named a 14-member team, drawn from Chinese institutions, recipients of the 2021 ACM Gordon Bell Prize for their project, Closing the "Quantum Supremacy" Gap: Achieving Real-Time Simulation of a Random Quantum Circuit Using a New Sunway Supercomputer.

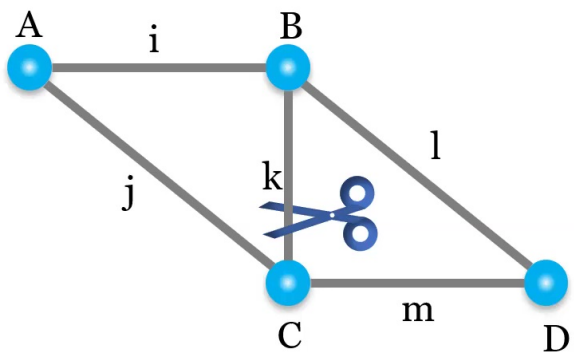
The members of the winning team are: Yong (Alexander) Liu, Xin (Lucy) Liu, Fang (Nancy) Li, Yuling Yang, Jiawei Song, Pengpeng Zhao, Zhen Wang, Dajia Peng, and Huarong Chen of Zhejiang Lab, Hangzhou and the National Supercomputing Center in Wuxi; Haohuan Fu and Dexun Chen of Tsinghua University, Beijing, and the National Supercomputing Center in Wuxi; Wenzhao Wu of the National Supercomputing Center in Wuxi; and Heliang Huang and Chu Guo of the Shanghai Research Center for Quantum Sciences.

Two improvements

- Use the low-rank structures and slicing to decrease the contraction complexity.

- Move the cutting plane forward, make the tail part includes all qubits.

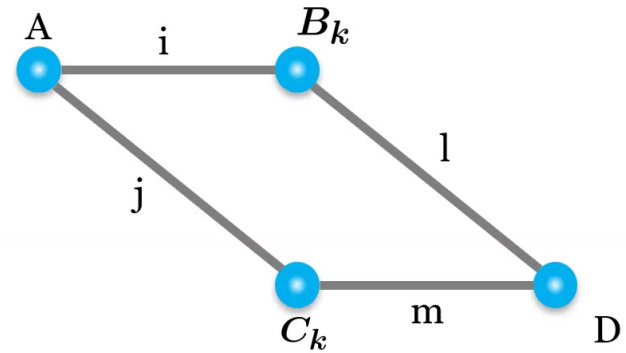
Slicing



$$\sum_{ijklm} A_{ij} B_{ikl} C_{jkm} D_{lm}$$

$$=$$

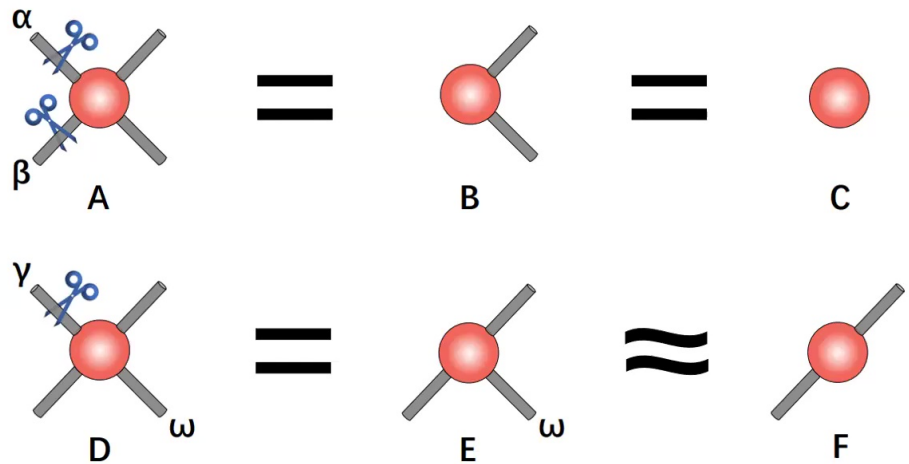
$$\sum_{k}$$



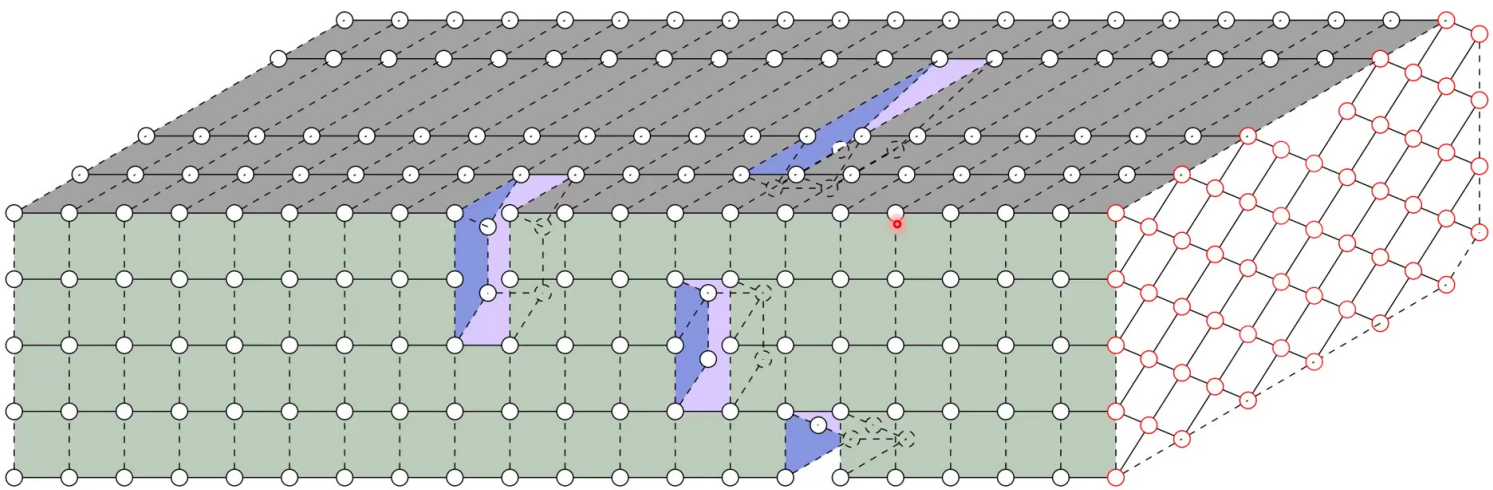
$$\sum_{k} \sum_{ijlm} A_{ij} B_{ikl} C_{jkm} D_{lm}$$

Low-rank structures and slicing

$$\mathrm{fSim}(\theta,\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -i\sin\theta & 0 \\ 0 & -i\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{bmatrix}$$

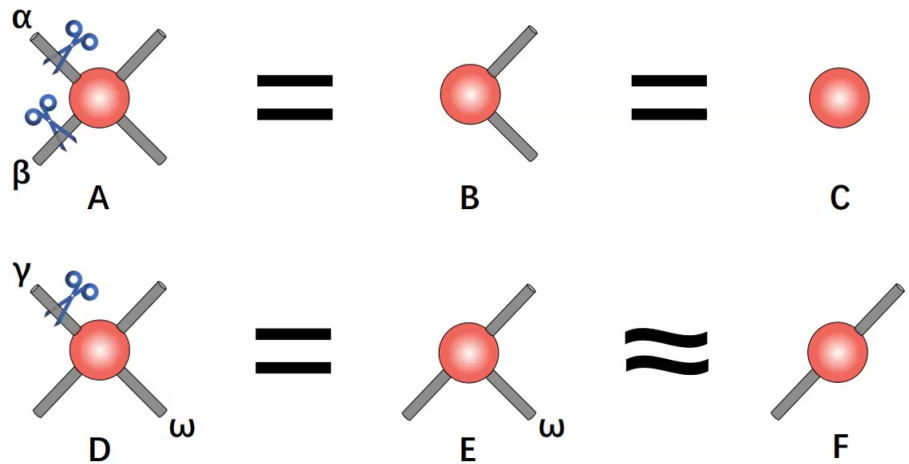


$$F_{\text{estimate}} \approx 2^{-K} \prod_{i=1}^{k} (\sin^2(\theta_i) + 1)/2$$



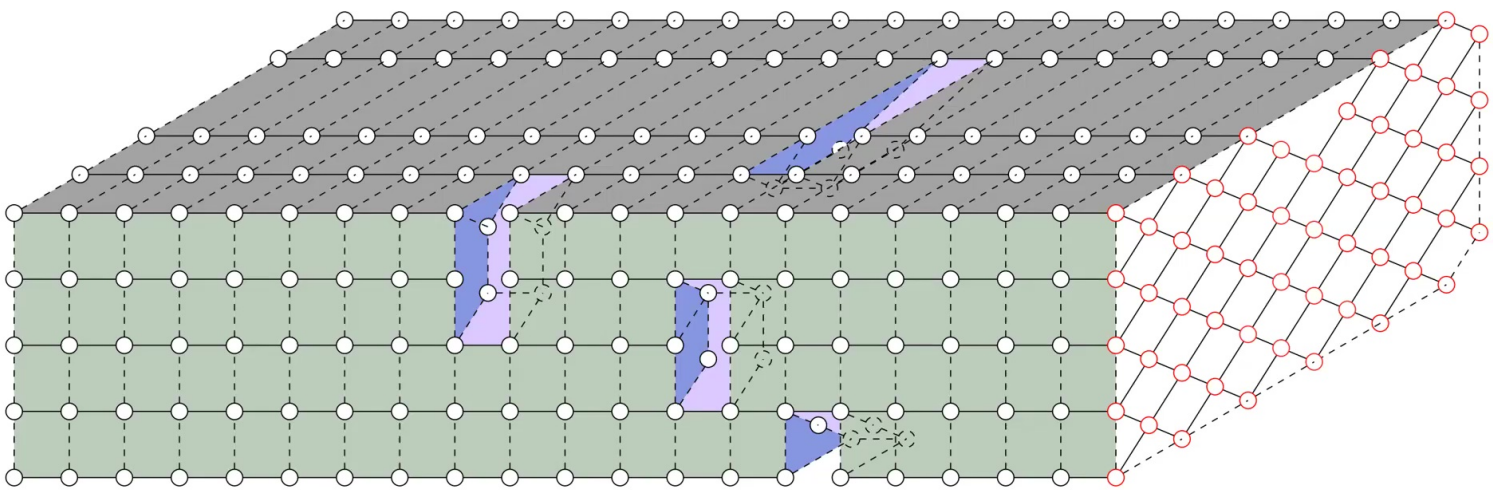
Low-rank structures and slicing

$$\mathrm{fSim}(\theta,\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -i\sin\theta & 0 \\ 0 & -i\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{bmatrix}$$

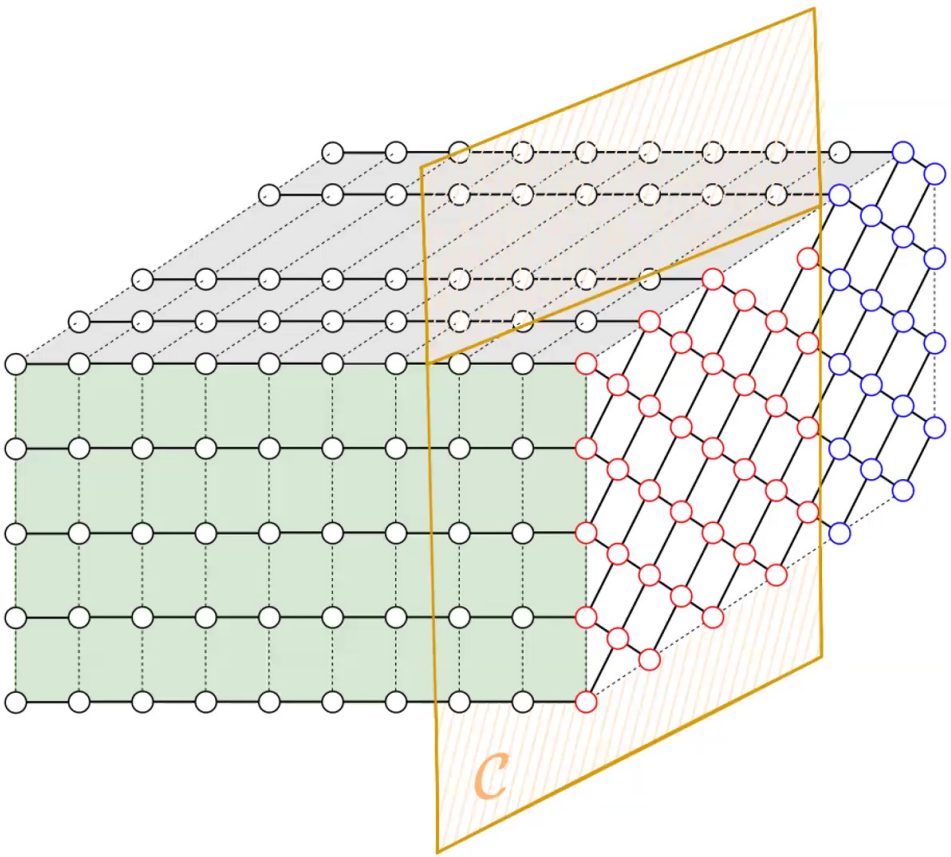


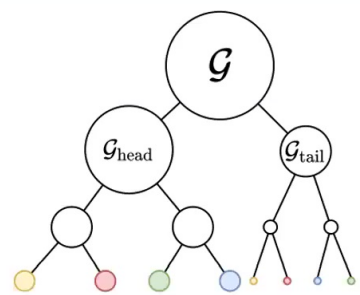
$$F_{\text{estimate}} \approx 2^{-K} \prod_{i=1}^{k} (\sin^2(\theta_i) + 1)/2$$

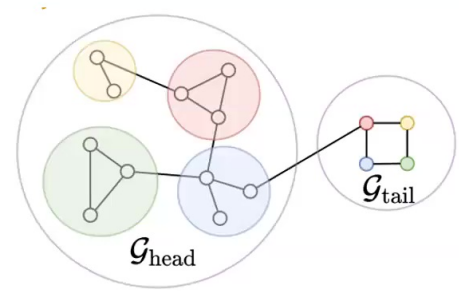
$$= 2^{-8} \times 0.9565 \approx 0.0037$$



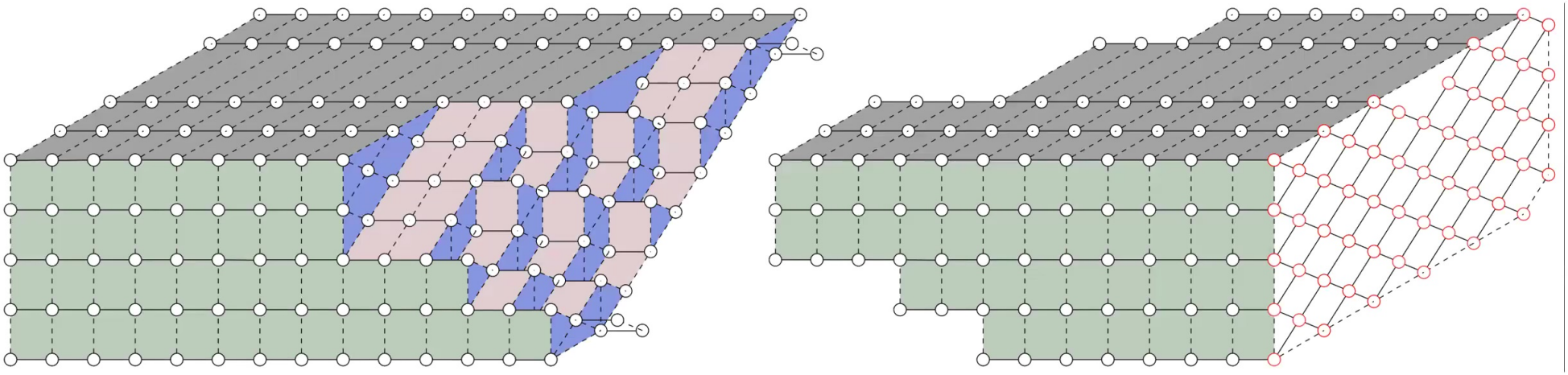
Sparse-state contraction







Sparse-state contraction



Results

Data	Original	Branch merge
T_c head one sub-task	2.3816×10^{13}	6.967×10^{13}
T_c tail one sub-task	2.9425×10^{13}	8.796×10^{13}
Overall T_c (2^{16} sub-tasks)	3.489×10^{18}	1.033×10^{19}
Space complexity	2^{30}	

Results

Data	Original	Branch merge
T_c head one sub-task	2.3816×10^{13}	6.967×10^{13}
T_c tail one sub-task	2.9425×10^{13}	8.796×10^{13}
Overall T_c (2^{16} sub-tasks)	3.489×10^{18}	1.033×10^{19}
Space complexity	2^{30}	

	# bitstrings	Time complexity	Space complexity
Google [1]	10^6	—	—
Cotengra [12]	1	3.10×10^{22}	2^{27}
Alibaba [18]	64	6.66×10^{18}	2^{29}
Ours	**2097152**	4.51×10^{18}	2^{30}

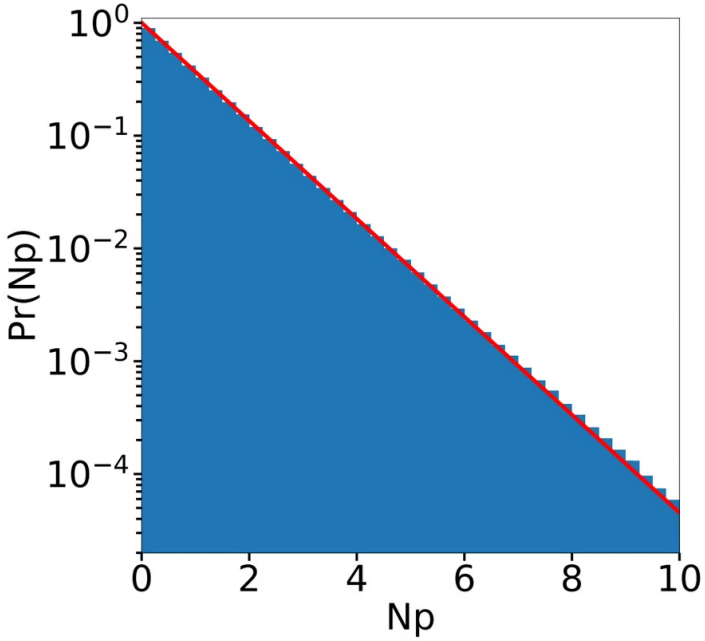
Results

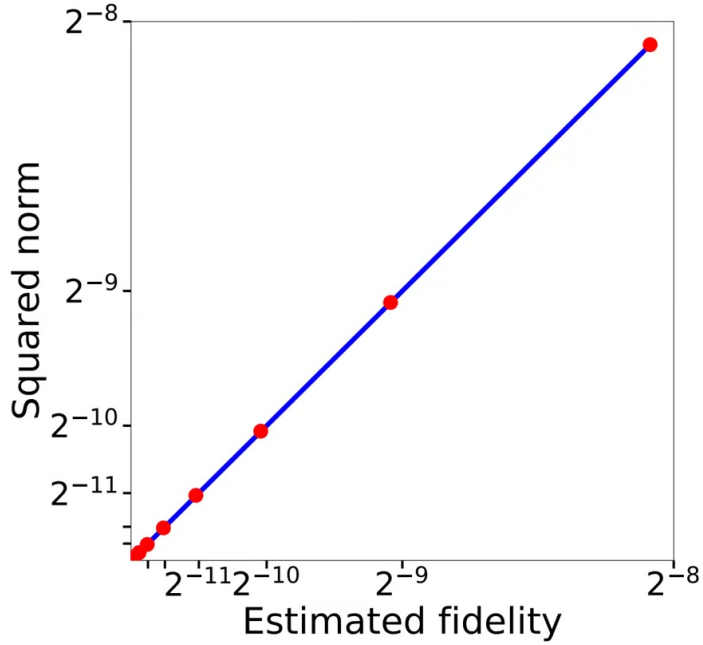
Data	Original	Branch merge
T_c head one sub-task	2.3816×10^{13}	6.967×10^{13}
T_c tail one sub-task	2.9425×10^{13}	8.796×10^{13}
Overall T_c (2^{16} sub-tasks)	3.489×10^{18}	1.033×10^{19}
Space complexity	2^{30}	

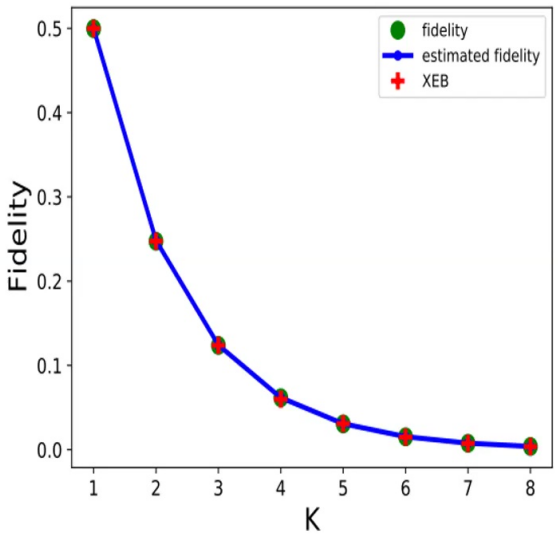
	# bitstrings	Time complexity	Space complexity
Google [1]	10^6	—	—
Cotengra [12]	1	3.10×10^{22}	2^{27}
Alibaba [18]	64	6.66×10^{18}	2^{29}
Ours	**2097152**	4.51×10^{18}	2^{30}

15 hours in 512 GPUs → several seconds in supercomputer
Quantum supremacy rejected!

Results


Pr(Np)
Np


Squared norm
Estimated fidelity


fidelity
estimated fidelity
XEB
Fidelity
K

Some thoughts

- This work has equivocally rejected the quantum supremacy claim of Google with their Sycamore quantum circuits, current RQC samplings experiments still need to improve the fidelity.

- Classical simulation of the RQC sampling task just starts, with possible further improvements on order finding techniques and performance of supercomputers.

- Quantum supremacy (quantum computational advantage) will be settled sometime in the future. But now there will be a race between quantum experiments and classical methods.

Conclusion and Insights

- Arbitrary tensor network algorithms are useful for specific physical problems and show their great performance over traditional methods.

- Their improvements are highly related to the algorithmic and hardware developments.

- Their applications can be enormous:
  - Validation and design of quantum algorithms for combinatorial optimization.
  - Effective and accurate simulation of the quantum/classical systems.

Thanks for your attention!