Arbitrary Tensor Network Algorithm: Theory, Methods and Applications

Feng Pan

Centre for Quantum Technologies, National University of Singapore

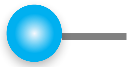
IPAM Tensor Network 2024 workshop

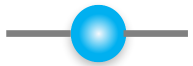
Contents

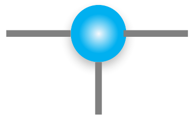
Background & Motivation

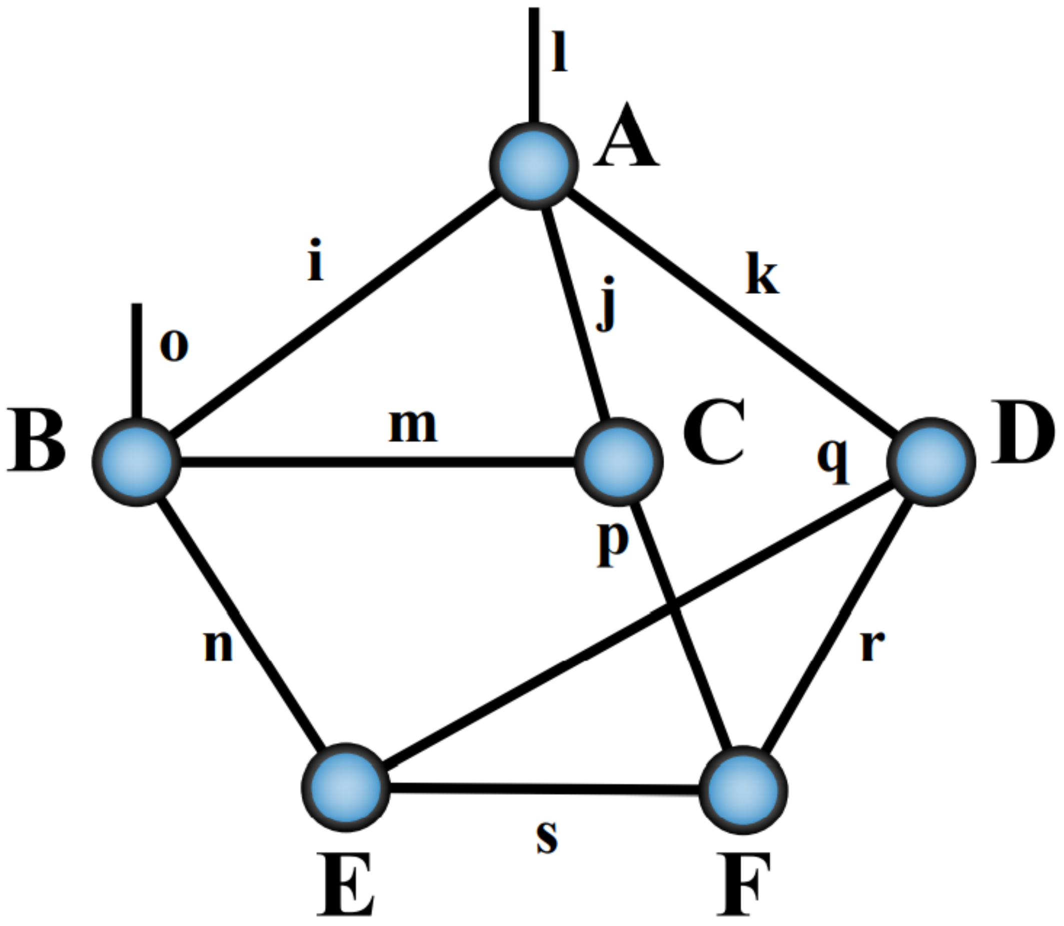
Background – Tensor Network Notations


Scalar

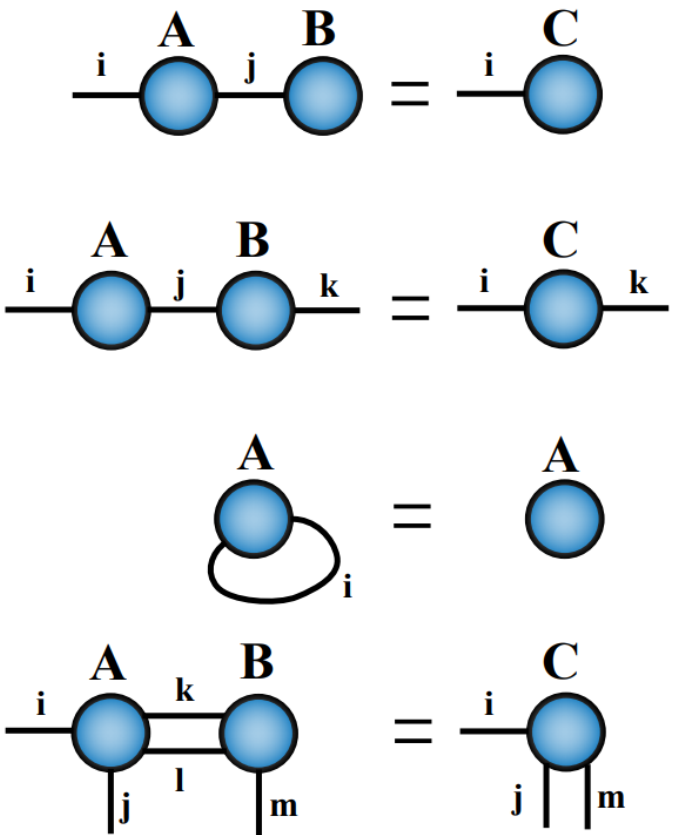

Vector


Matrix


Tensor



Background – Tensor Network Operations



(1) $\sum_j A_{ij} B_j = C_i$

(2) $\sum_j A_{ij} B_{jk} = C_{ik}$

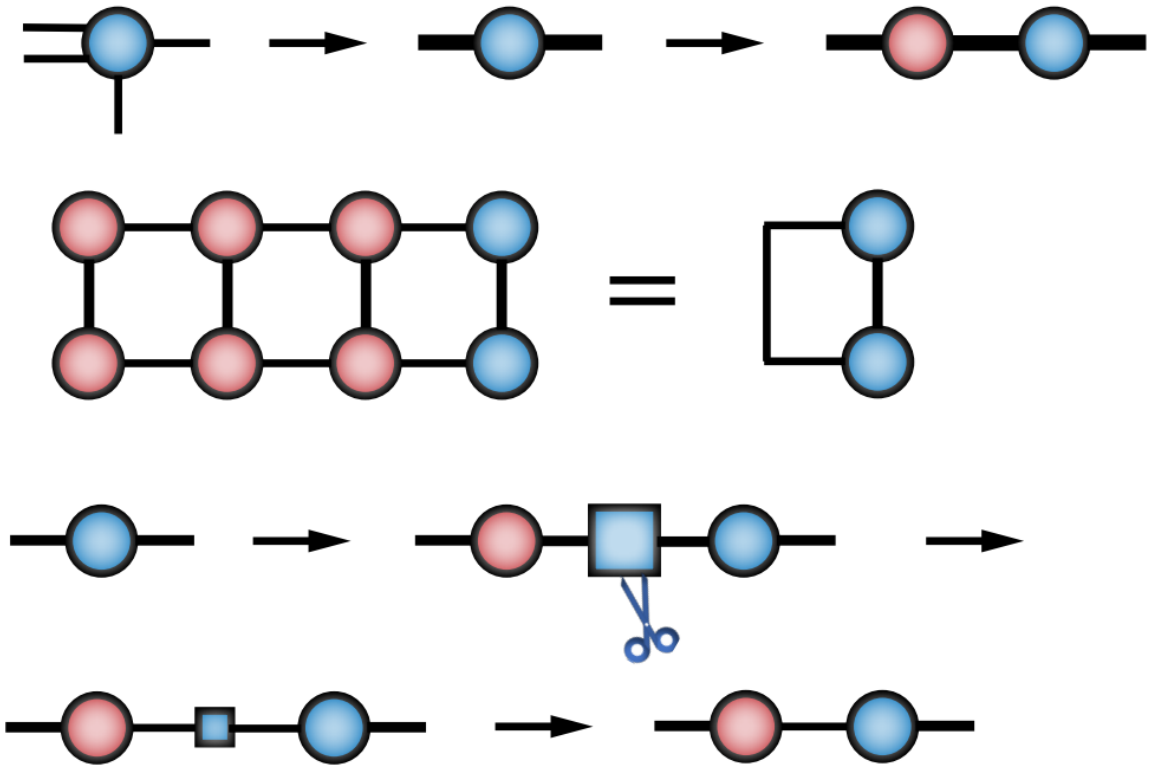
(3) $\sum_i A_{ii} = Tr(A)$

(4) $\sum_{kl} A_{ijkl} B_{klm} = C_{ijm}$

(1)

(2)

(3)



Motivation

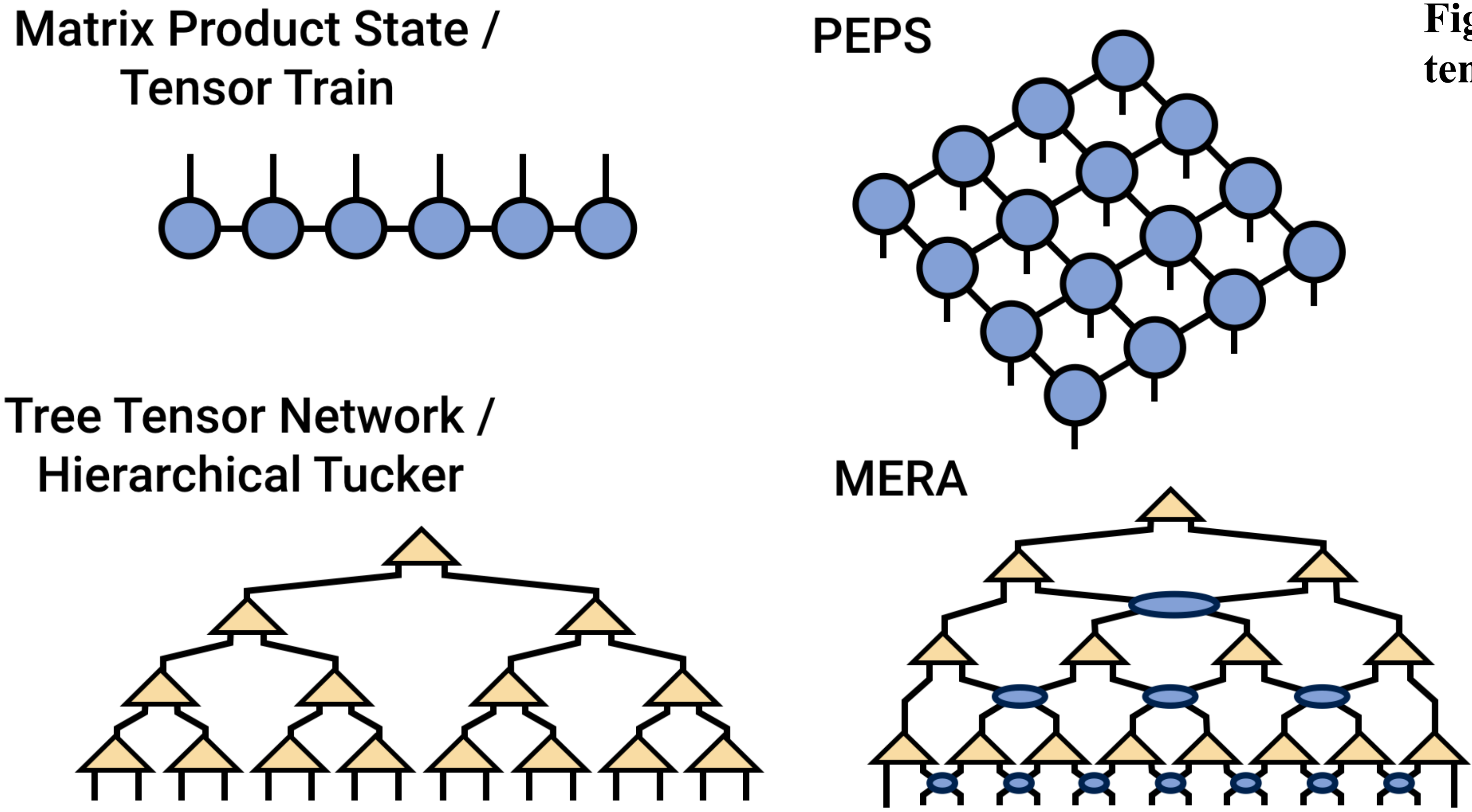


Figure credit: http://tensornetwork.org/

Motivation

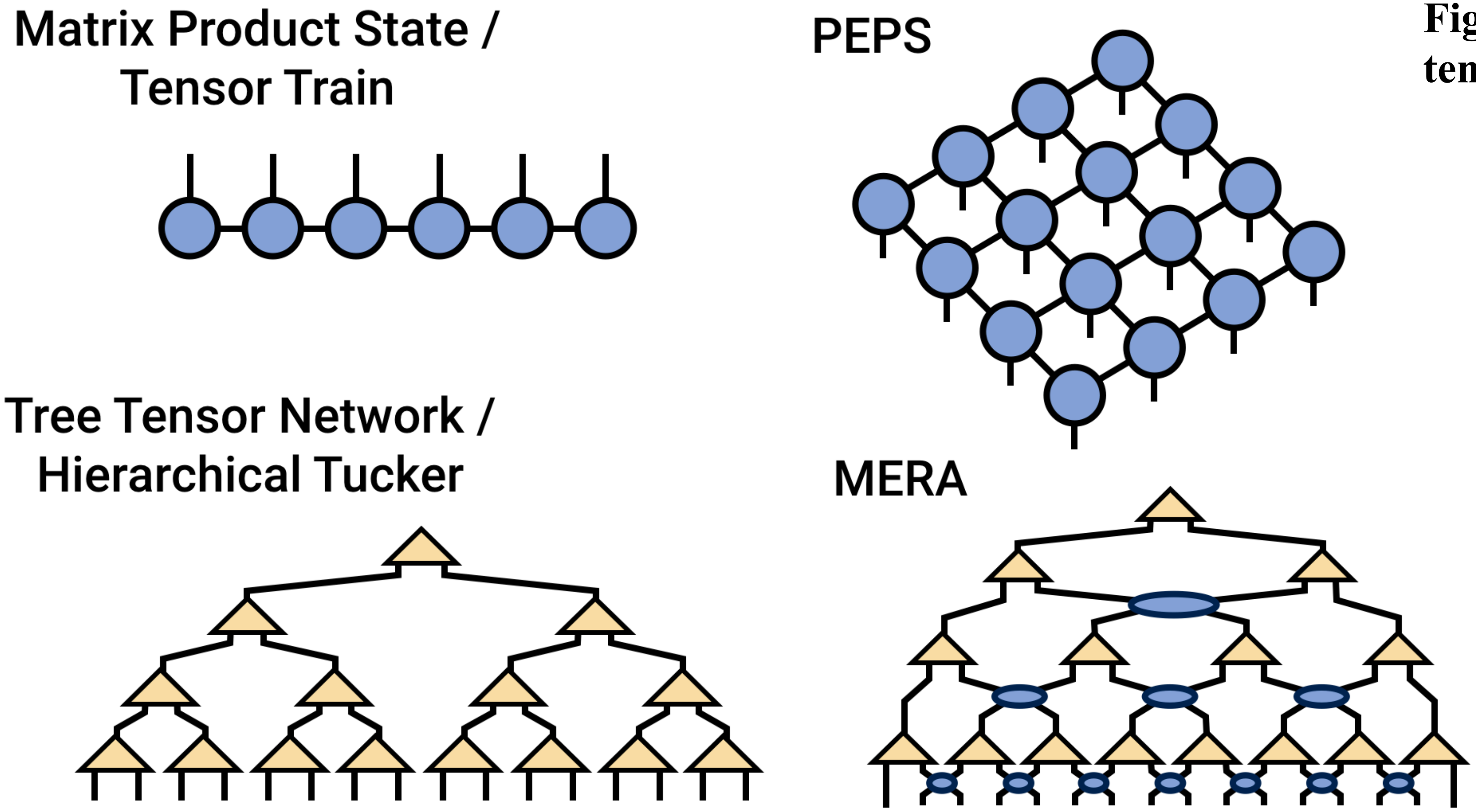


Figure credit: http://tensornetwork.org/

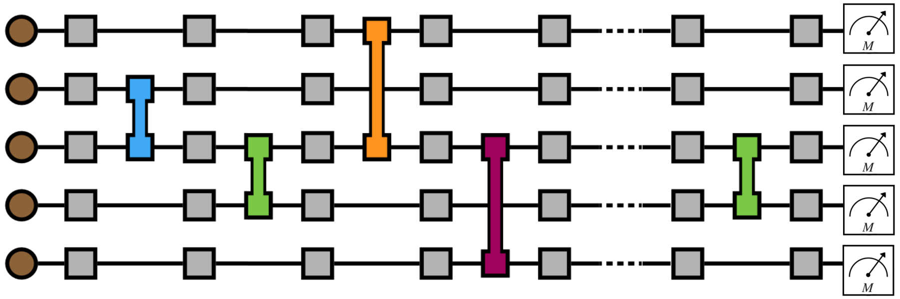
system without any structure?

Motivation

$$\sum_s e^{-\beta \sum_i H_i(\hat{s}_i)}$$

#P problem generally

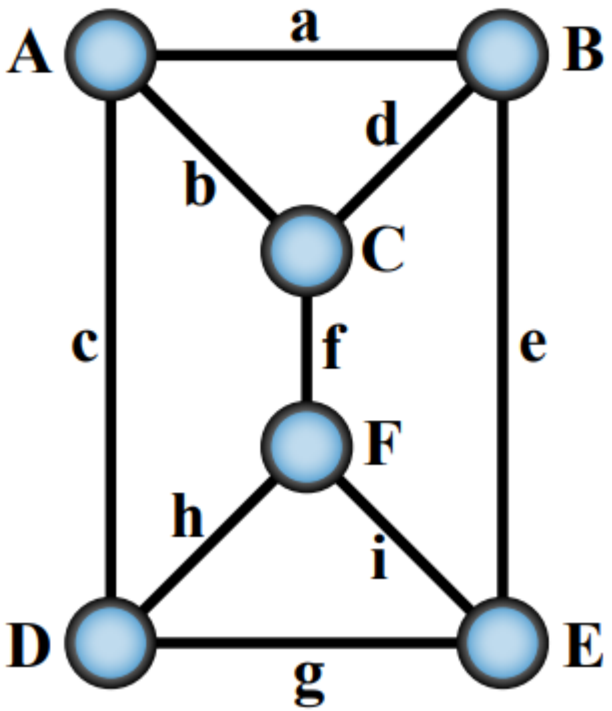
- ➢ Mean-field approximation
- ➢ Tree approximation with Belief Propagation
- ➢ $\sum_s \prod_i T^i_{\hat{s}_i}$ Tensor network contraction

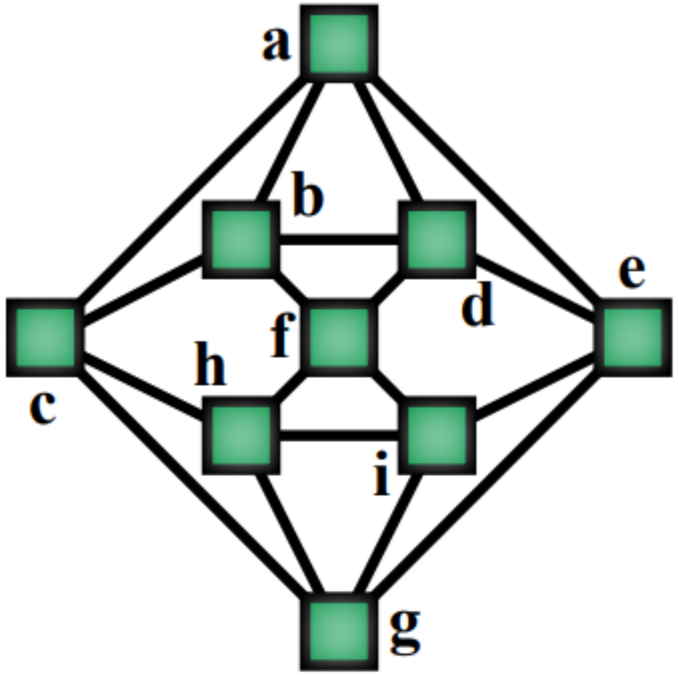


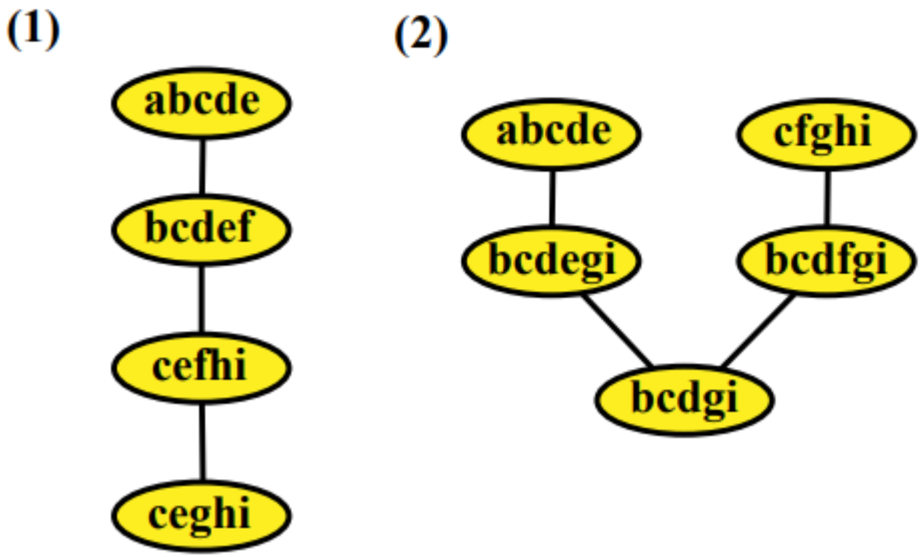
Theoretical Foundation and Methods of arbitrary TN algorithm

Complexity lower bound

Theorem: The space complexity lower bound of an arbitrary tensor network contraction is exponential to the tree width of its line graph.

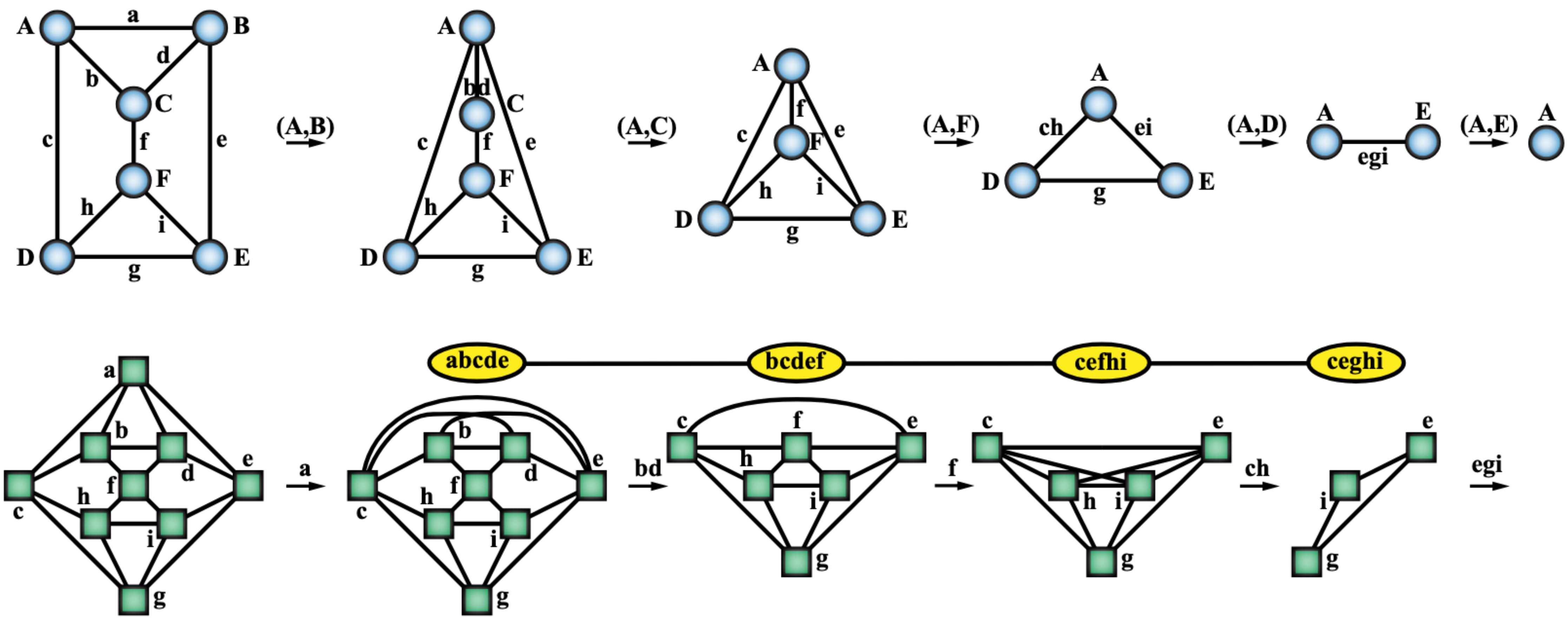






Markov and Shi. *SIAM Journal on Computing 38 (3), 963-981*.

Tree decomposition and contraction order



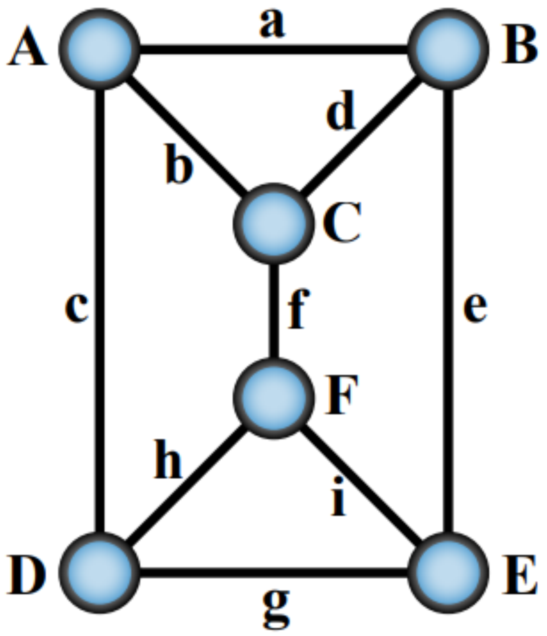
Methods for finding good contraction orders

Finding a good contraction order is a hard combinatorial optimization problem, due to

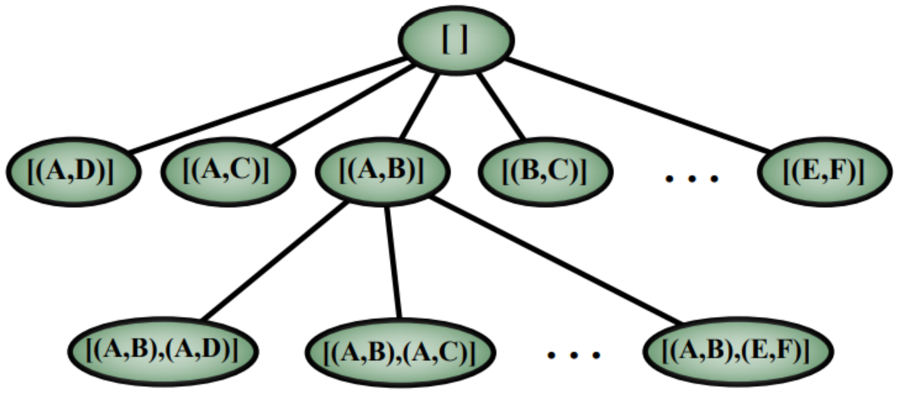
complex target function + large solution space + sequence dependence

Here we introduce four ways to find contraction orders

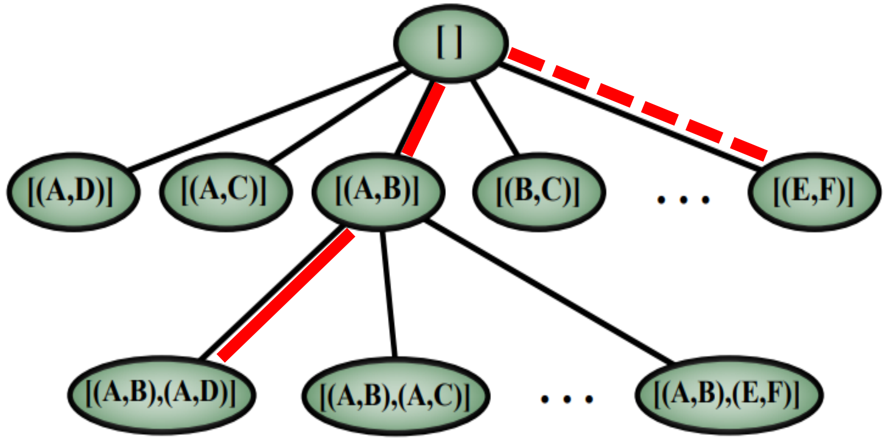
- Greedy method
- Branch and Bound method
- Graph partition method
- Local update method



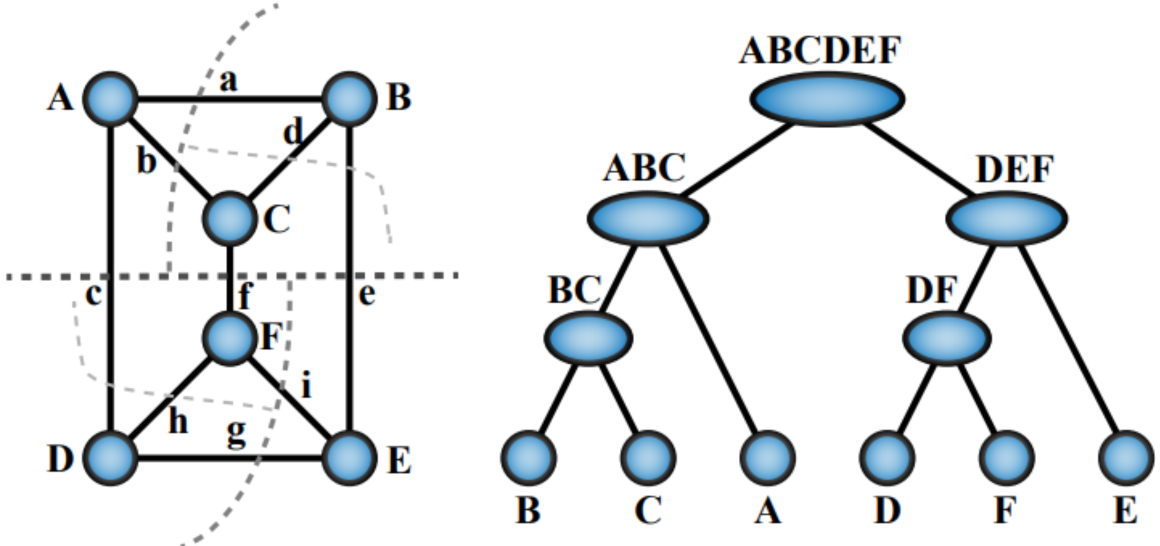
Greedy method:



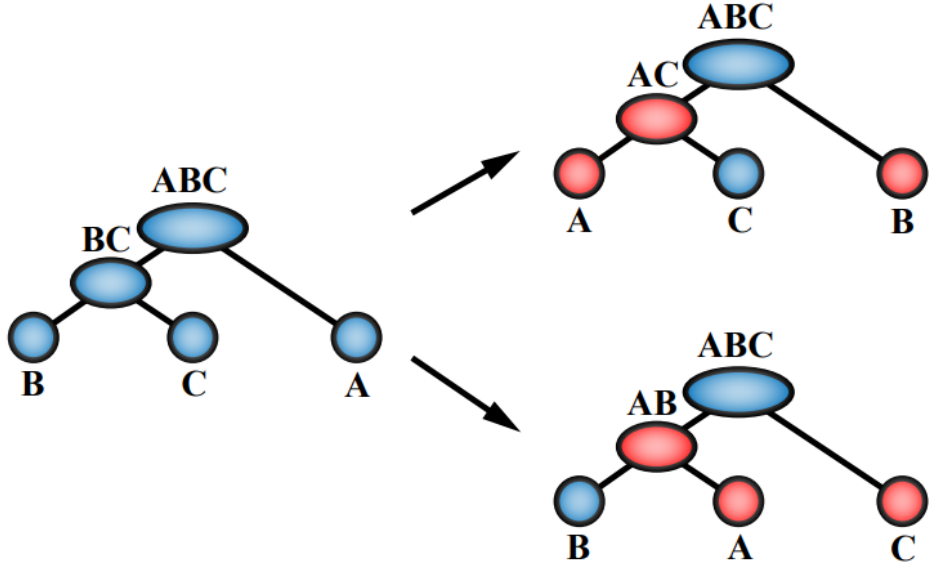
Branch and Bound method:



Graph partition method:



Local update method:

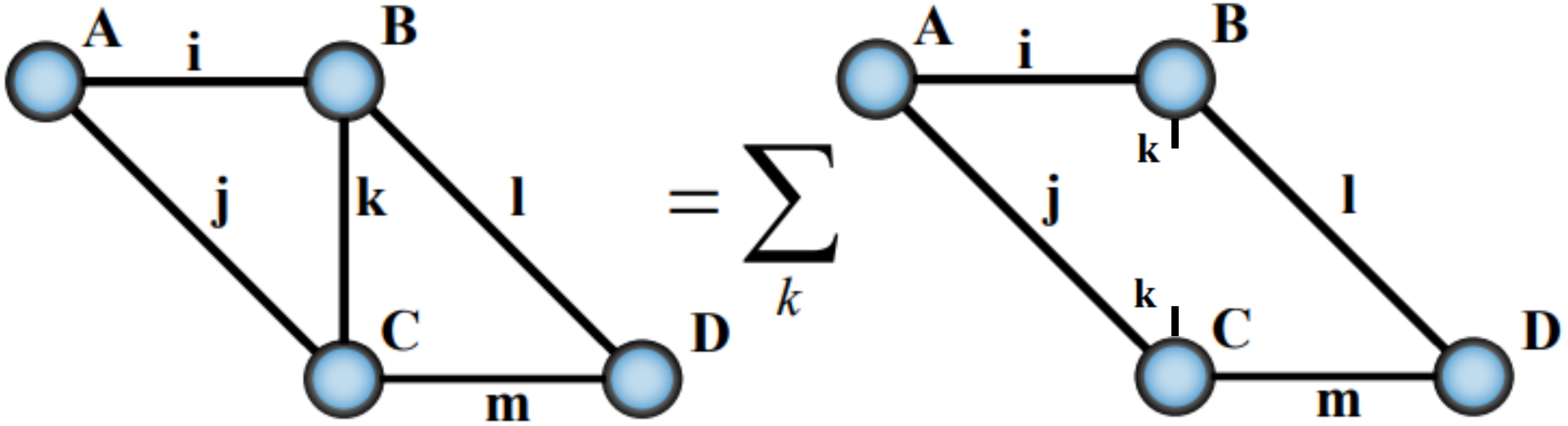


Gray, Kourtis. *Quantum* 5 (2021): 410.

Kalachev, Pavel and Yung. *arXiv:2108.5665* (2021).

Other techniques in the real numerical contraction

- Tensor slicing



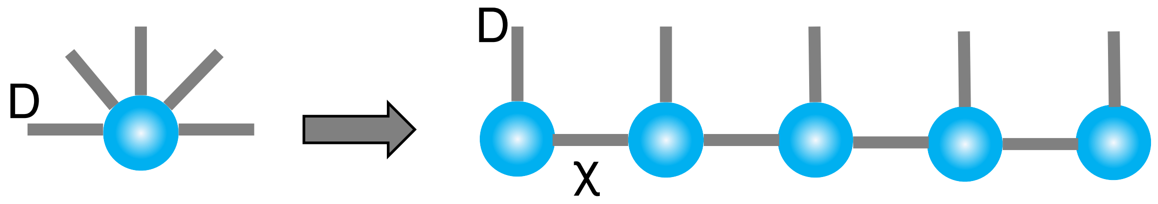
- Computational efficiency

  Balance the memory read/write operations and floating-point operations

Approximate arbitrary TN algorithm

Approximate arbitrary TN algorithm

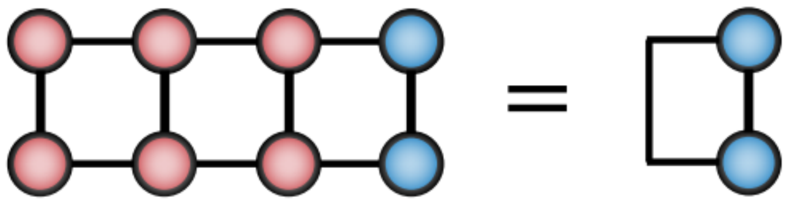
1. Exponentially large complexity — Using matrix product state (MPS)



2. How to do approximation? — DMRG like low-rank approximation scheme

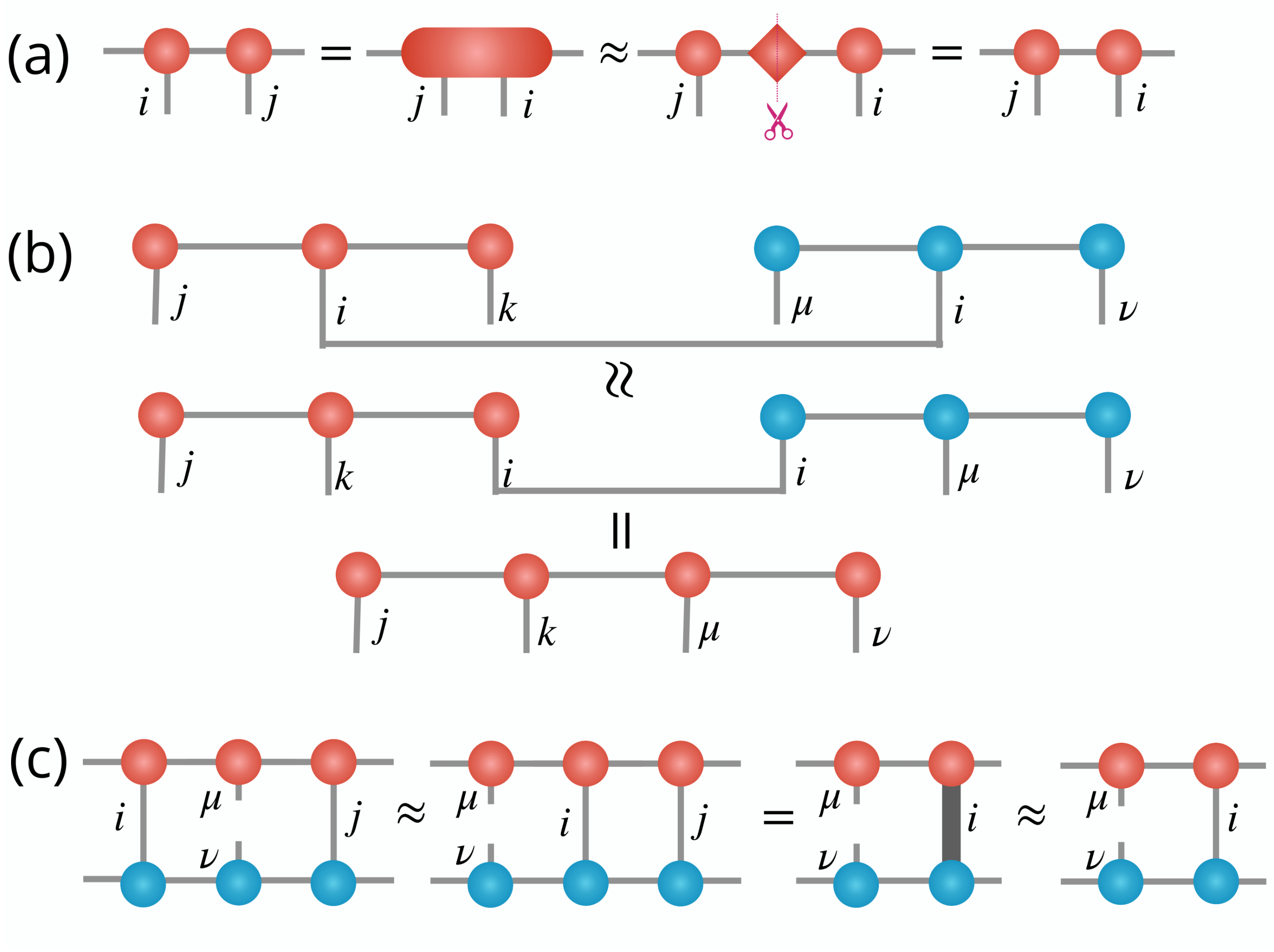
$$D_{\max} \qquad \chi_{\max}$$

3. How to control the error? — Using the canonical form of MPS



4. Contraction order? — Greedily chosen from the current TN

MPS calculus operations

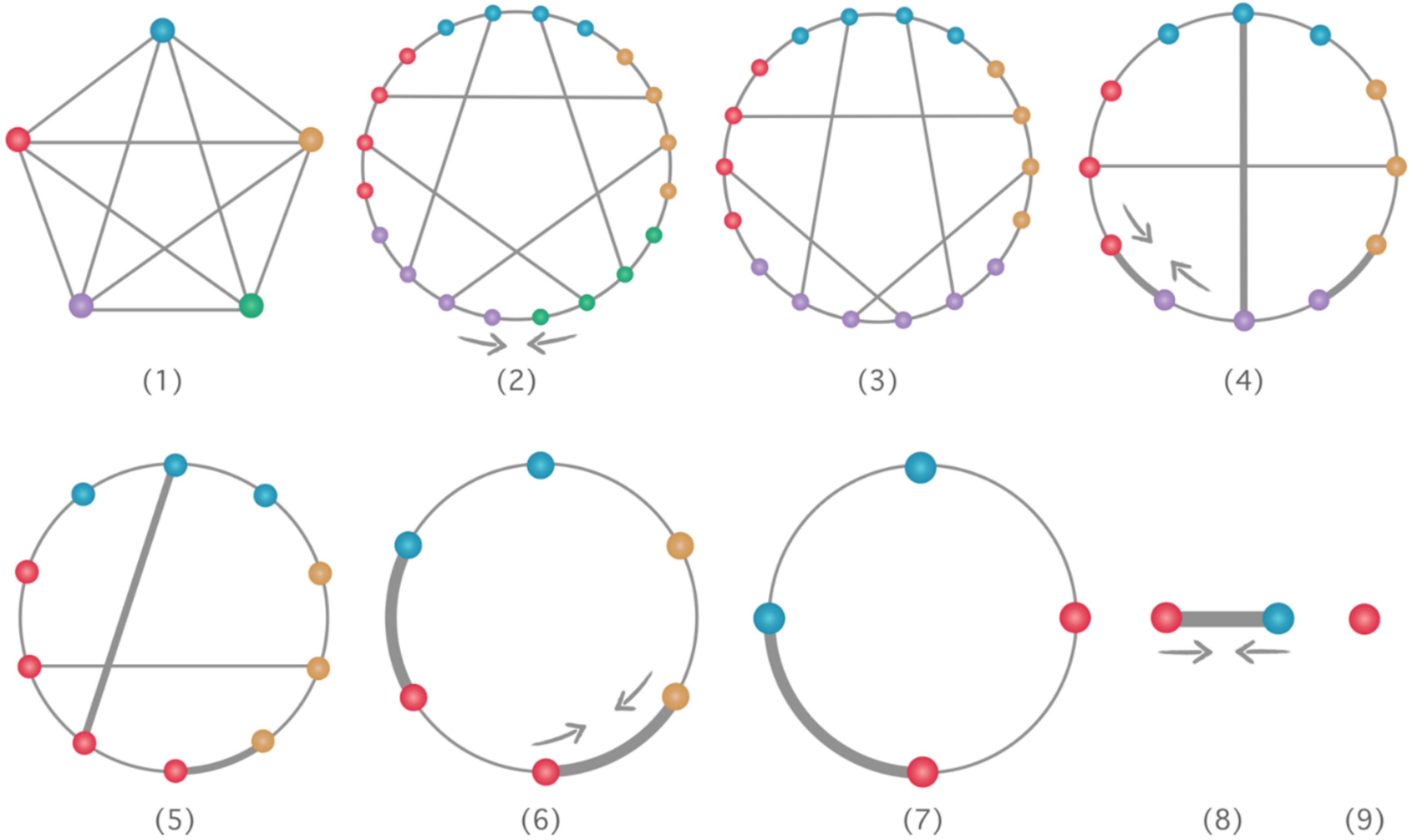


Swap

Contract

Merge

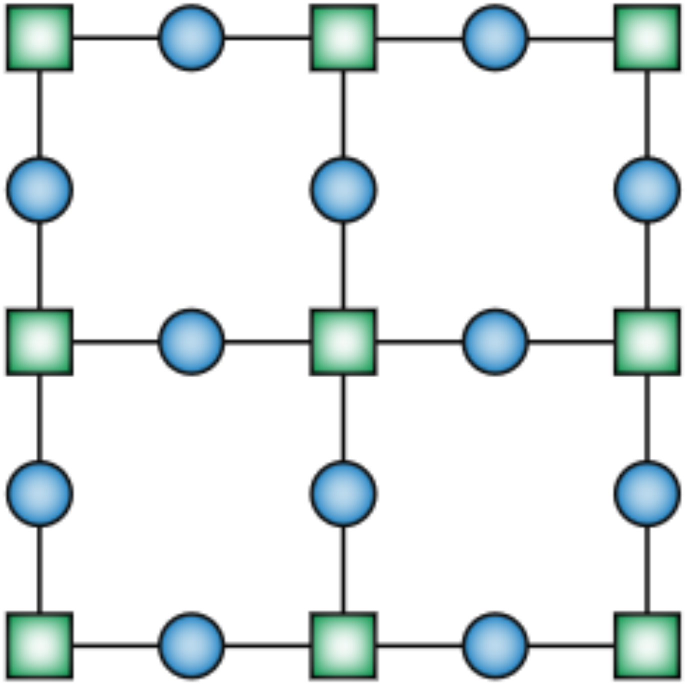
Contract an Arbitrary TN with an example



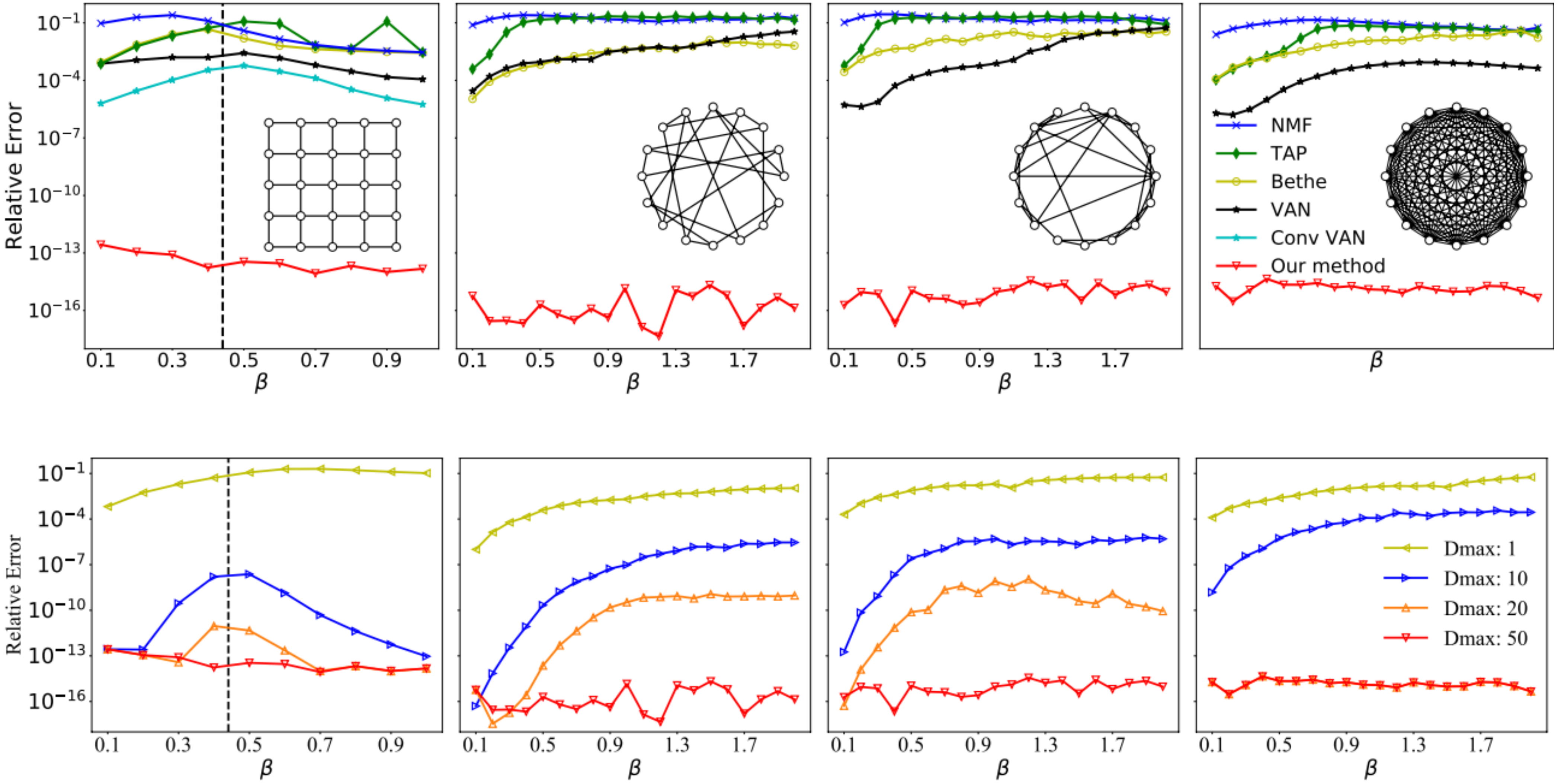
Pan, et al. *Phys. Rev. Lett.* 125.060503 (2020)

Map calculation of physical properties to TN

$$F(\beta) = -\frac{1}{\beta N} \ln\left(\sum_{\vec{s}} \prod_{(i,j)} e^{\beta J_{ij} s_i s_j} \right)$$
$$= -\frac{1}{\beta N} \ln\left(\sum_{\vec{b}} \bigotimes_{i} \mathcal{I}_{b_i^{d_i}} \bigotimes_{(i,j)} \mathcal{B}_{b_i b_j} \right)$$



Some results on partition function calculation



16x16 2D lattice

RRG graphs n=80, k=3

Small-world n=70, c=4

SK model n=20

Exact arbitrary TN algorithm

Exact arbitrary TN algorithm

There are some circumstances when problems require exact results or there is no intrinsic low-rank structure:

For example, Simulation of quantum circuits with quantum gate (fSim) whose decomposition spectrum is flat.

Hence, we will also need exact arbitrary TN algorithms.

Exact arbitrary TN algorithm - Challenges

- Exponentially large complexity

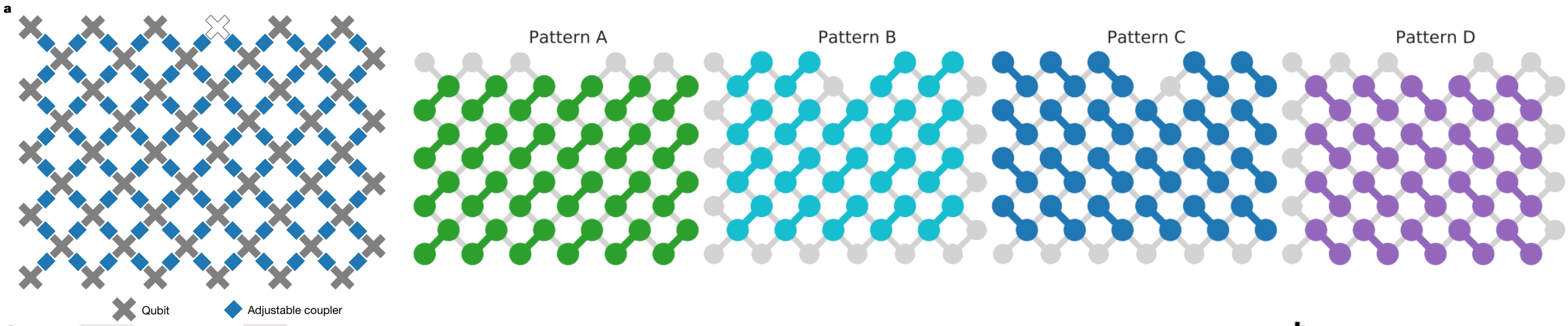
- ~~How to do approximation?~~

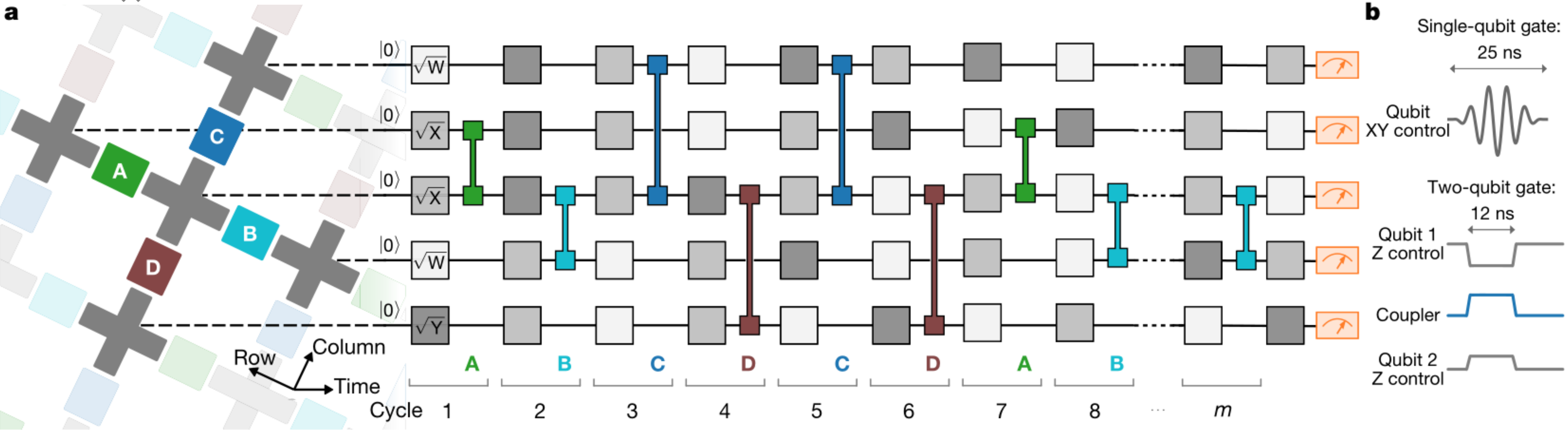
- ~~How to control the error?~~

- Contraction order

- For quantum circuit simulations: how to sample bitstrings

Sycamore chip and Random Circuit Sampling

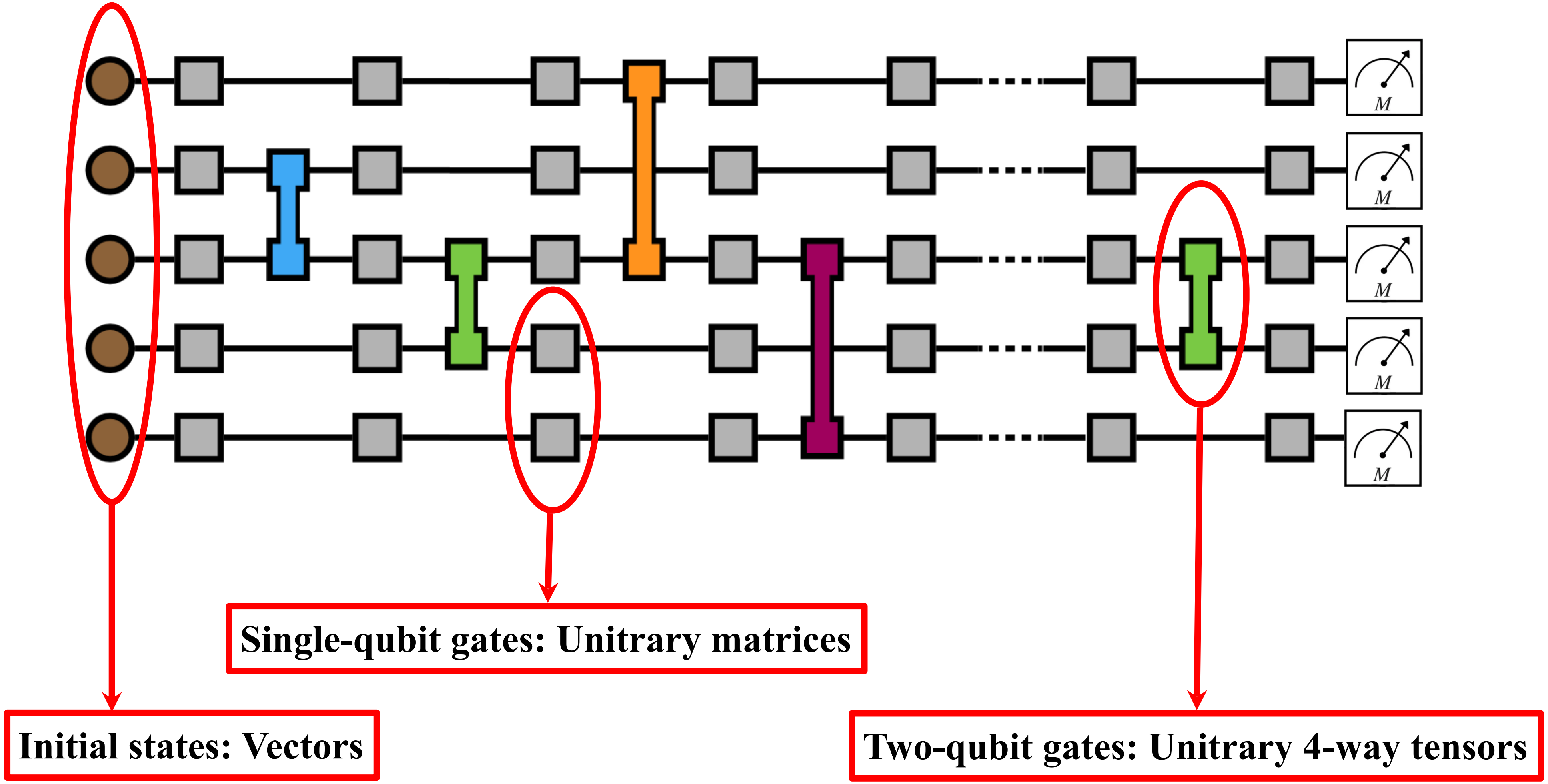




200 seconds 1,000,000 bit-strings 0.2% XEB fidelity

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

Tensor Network and QCircuits



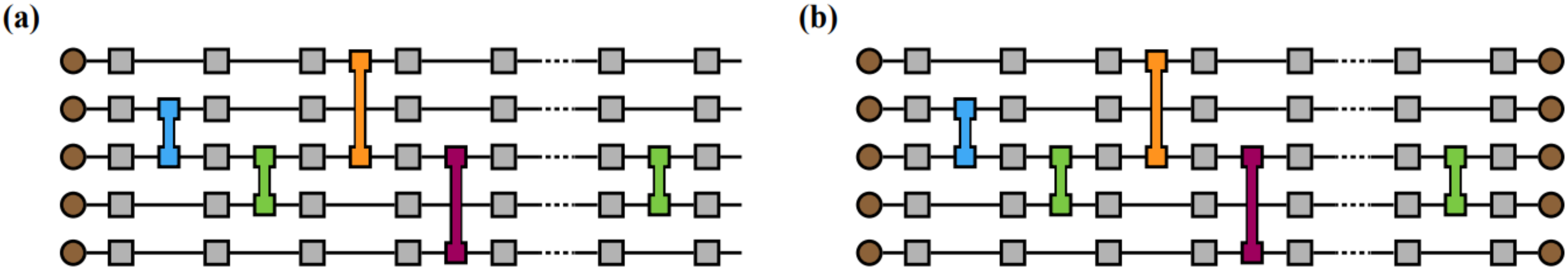
$$\langle 0|U_c|x\rangle = \sum_{s}\prod_{i} U^{i}_{\hat{s}_i}$$

Another #P problems without any low-rank structure to utilize

Quantum Supremacy demonstration

Classical simulation candidates:
1. Schrodinger-Feynman algorithm (Google's choice)

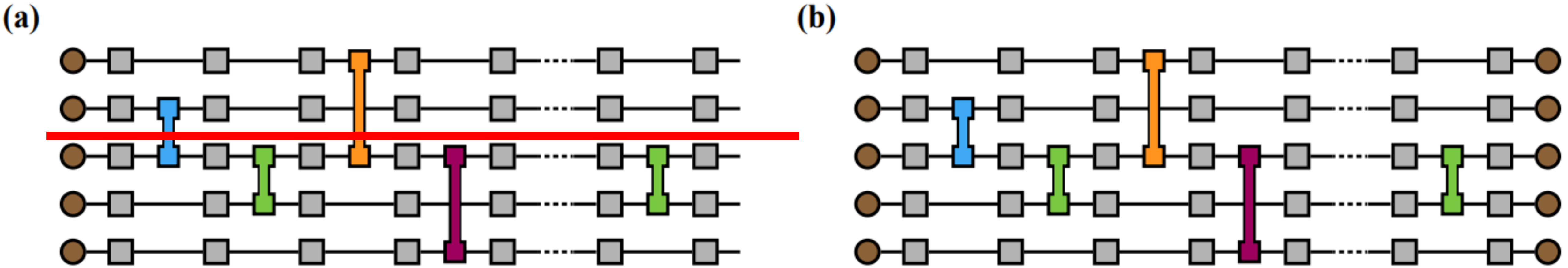
2. Tensor network contraction algorithm



Quantum Supremacy demonstration

Classical simulation candidates:
1. Schrodinger-Feynman algorithm (Google's choice)

2. Tensor network contraction algorithm



Quantum Supremacy demonstration

Classical simulation candidates:
1. Schrodinger-Feynman algorithm (Google’s choice)

2. Tensor network contraction algorithm

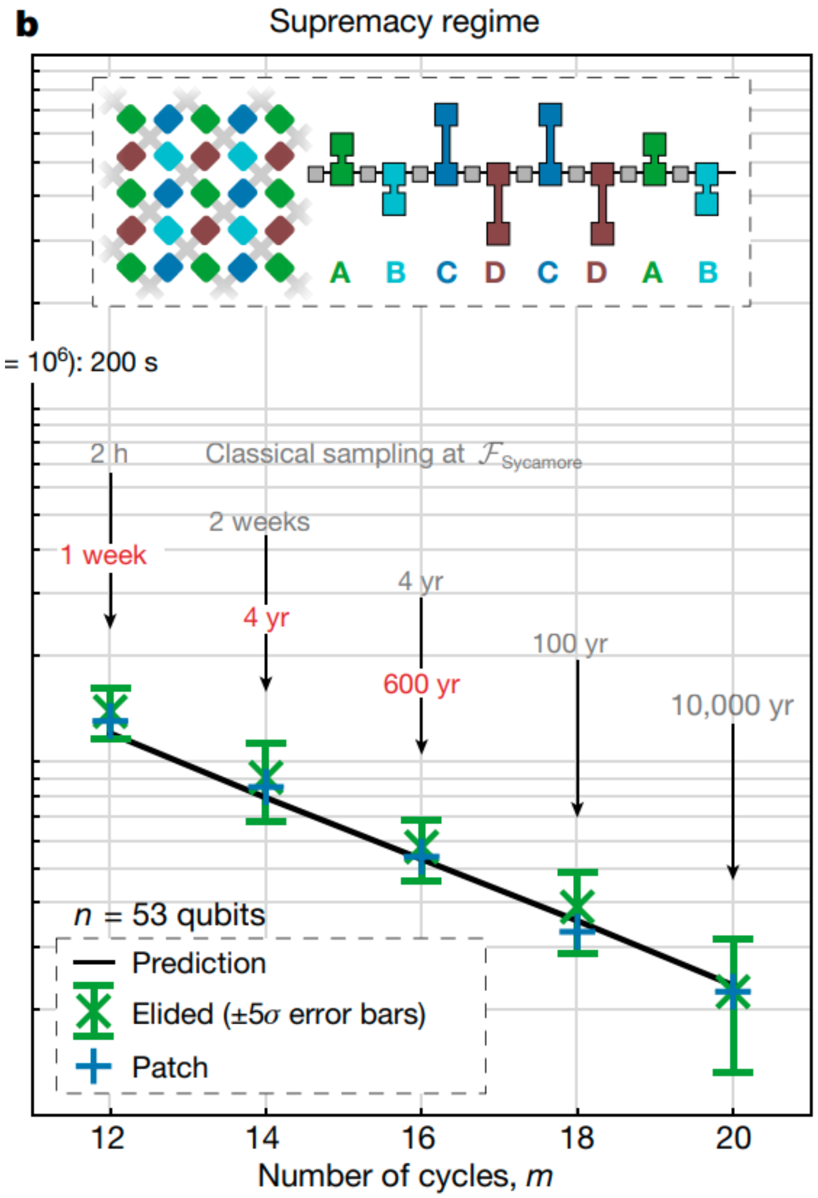
Quantum Supremacy demonstration

Classical simulation candidates:
1. Schrodinger-Feynman algorithm (Google's choice)

2. Tensor network contraction 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.



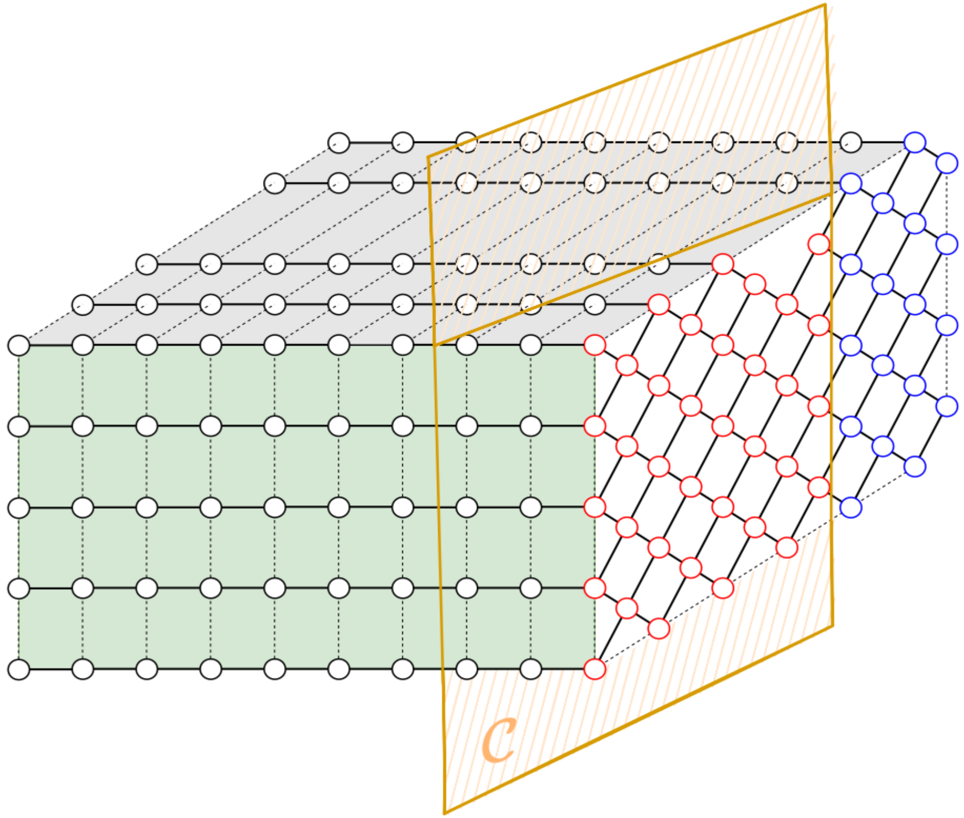
Fidelity and Linear cross entropy benchmark (XEB)

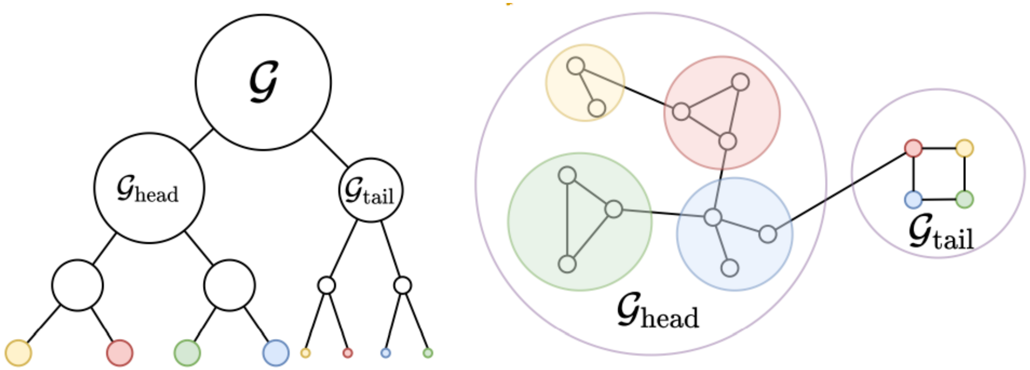
- DM of the output state $\rho = \mathcal{F}|\psi_U\rangle\langle\psi_U| + (1-\mathcal{F})I/2^n$
- Traditional definition of fidelity $\mathcal{F} = \langle\psi_U|\rho|\psi_U\rangle$
- It is impossible to calculate the fidelity for such experiment, use XEB fidelity instead

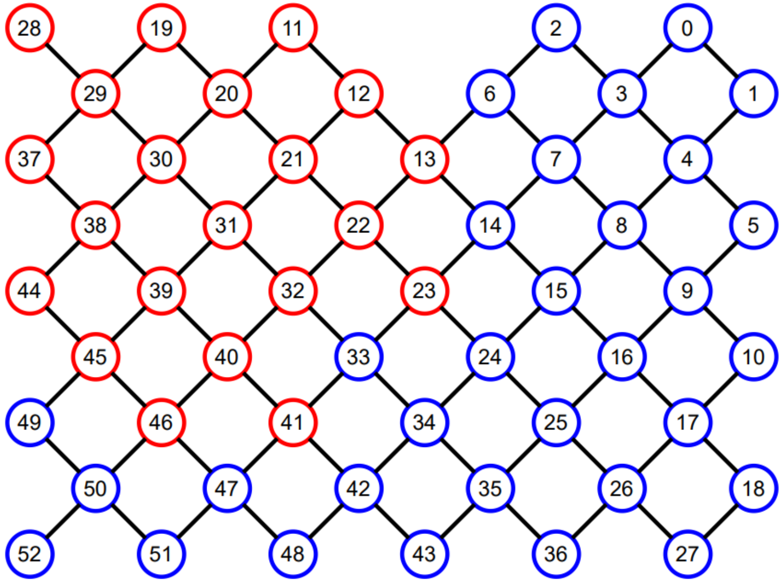
$$\mathcal{F}_{\mathrm{XEB}} = \frac{2^n}{L}\sum_{i=1}^{L} p_U(x_i) - 1$$

- XEB can be spoofed: samples with large probabilities

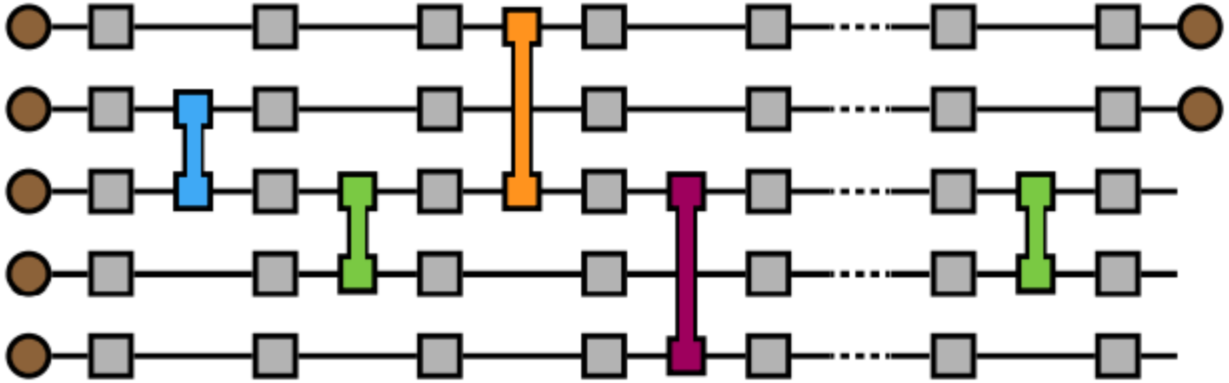
Big-batch method







20 cycles



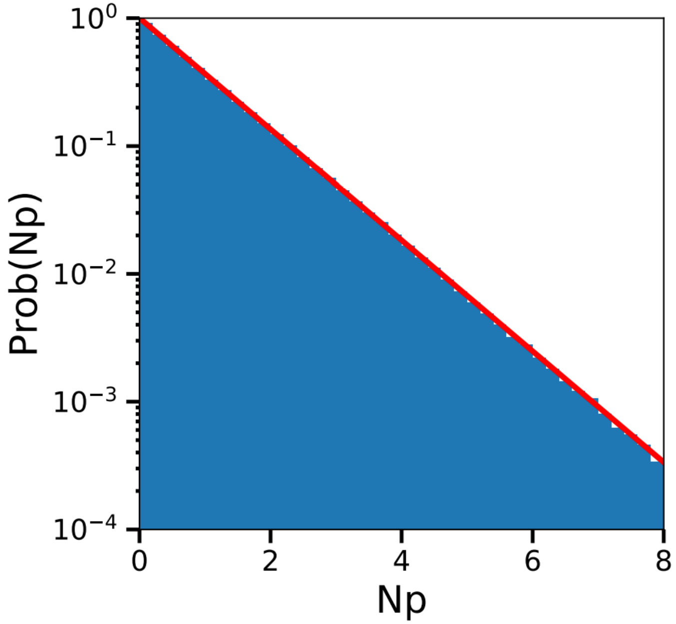
Results and the spoofing of XEB

	# 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
00000000000000000000000100000110100000010000100000000	$-2.97 \times 10^{-8} + 2.06 \times 10^{-8}i$	1.31×10^{-15}
00000000000000000000000000000000000000000100001000000	$1.50 \times 10^{-8} + 3.85 \times 10^{-9}i$	2.39×10^{-16}
00000000000111000001111100001111100001111100110000000	$-3.17 \times 10^{-9} - 5.45 \times 10^{-9}i$	3.97×10^{-17}
00000000000111000001111100001111100001111000101000000	$-1.89 \times 10^{-10} + 3.13 \times 10^{-9}i$	9.86×10^{-18}
00000000000000000000000000000100000000011100010000000	$8.07 \times 10^{-10} + 4.35 \times 10^{-10}i$	8.41×10^{-19}



2,097,152 bitstring samples with 0% XEB fidelity

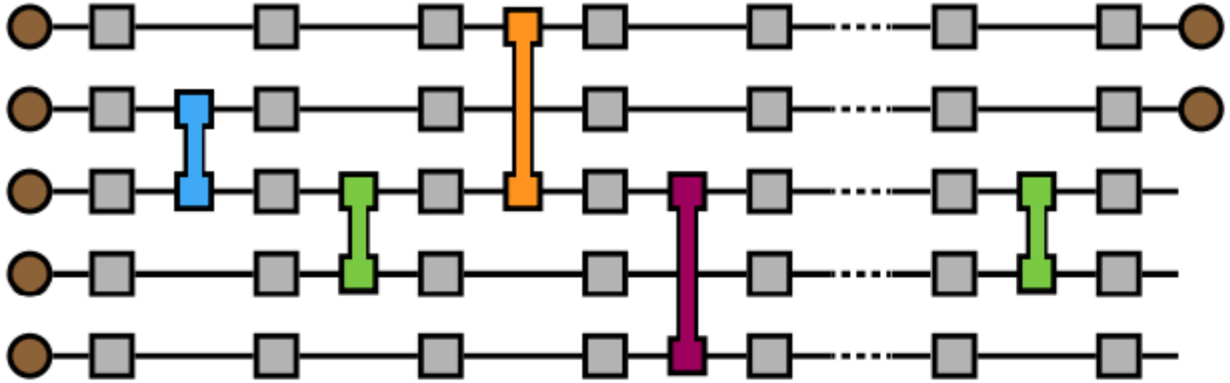
1,000,000 bitstring samples with 73.9% XEB fidelity

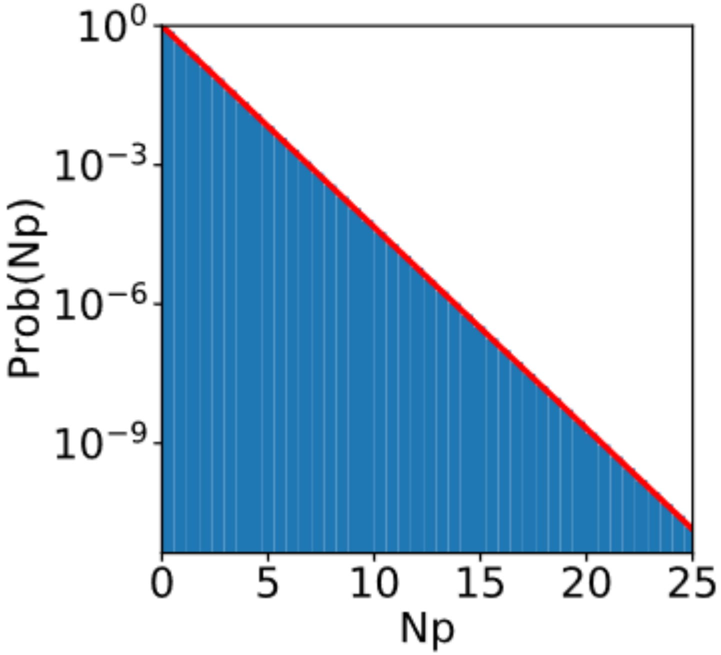
>> 0.2%

Pan, and Zhang. *Phys. Rev. Lett.* 128.030501 (2022).

Full amplitude simulation

We can also use big-batch method to do full amplitude simulation, by enumerating configurations of closed qubits.





50 qubits, 14 cycles, EFGH sequence using 100 GPUs in 10 days

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.

Can be defended using a slightly more complicated benchmark such as adding a preprocessing step to detect correlated samples.

Sparse-state tensor network simulation

Observation of quantum supremacy experiments:

- ➢ The number of bitstrings obtained by experiments will not be exponentially large.
- ➢These bitstrings will compose a sparse-state of the full Hilbert space.

Thus, calculating multiple bitstring amplitudes becomes the tensor network contraction below

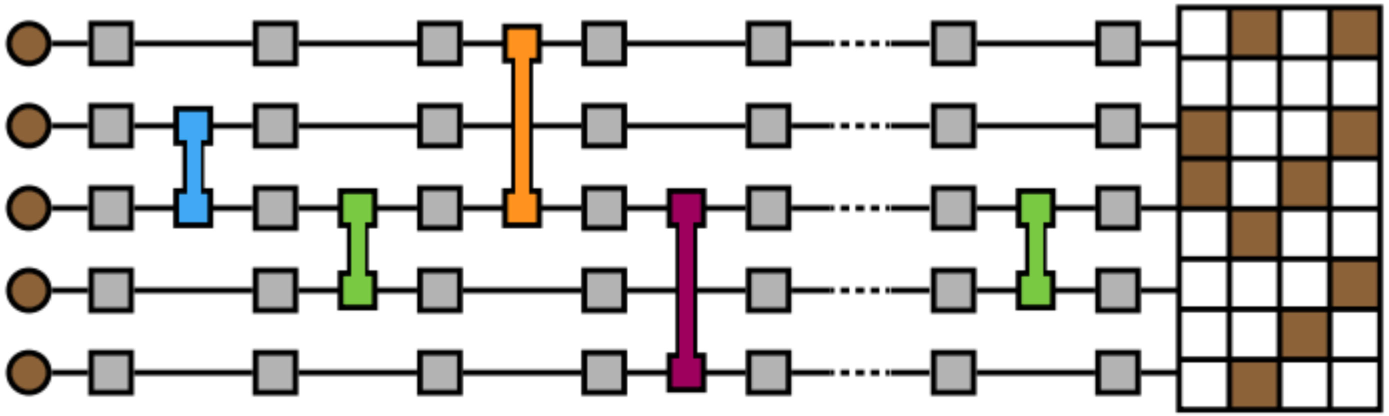
Sparse-state tensor network simulation

Observation of quantum supremacy experiments:

➢ The number of bitstrings obtained by experiments will not be exponentially large.

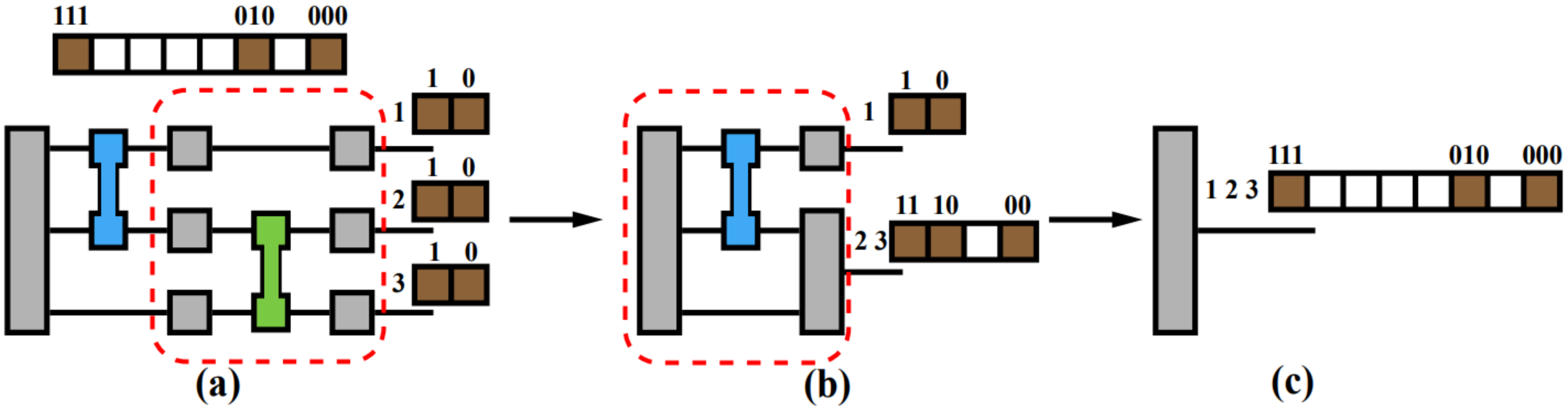
➢These bitstrings will compose a sparse-state of the full Hilbert space.

Thus, calculating multiple bitstring amplitudes becomes the tensor network contraction below



Sparse-state contraction

The core of sparse-state contraction is explicitly listing the dimension of open qubits and determine which entries should be calculated



Arbitrary bit-string means we can circumvent the correlated bitstring problems, and there is no way to tell from our samples from the quantum samples

Results

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 → dozens of seconds in exaflops supercomputer
Quantum supremacy on Sycamore53 does not hold!

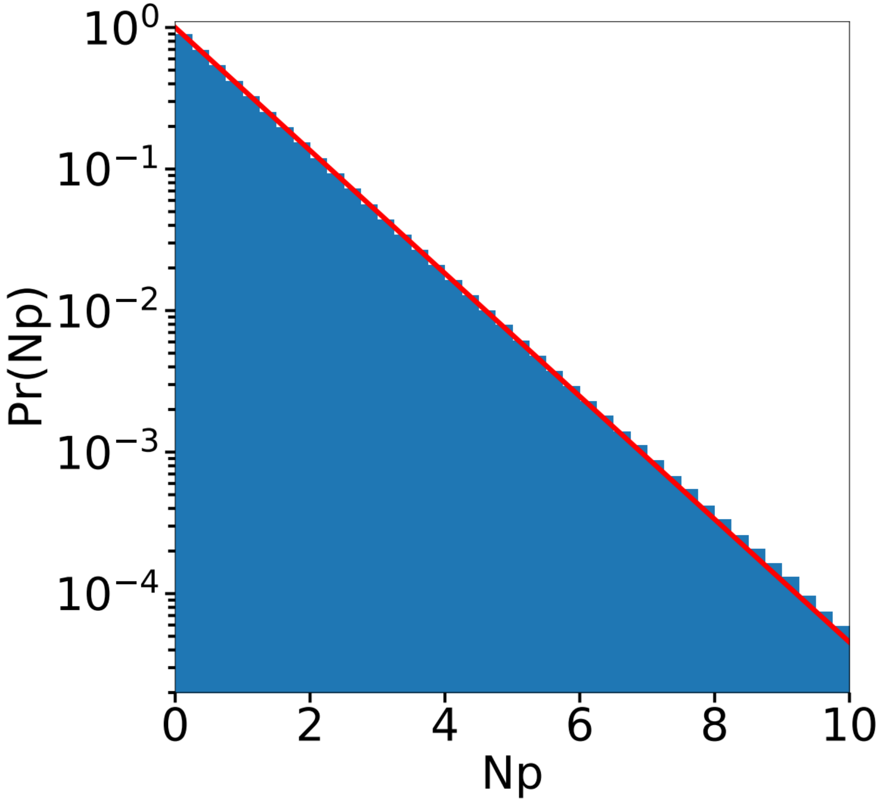
Results

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}	

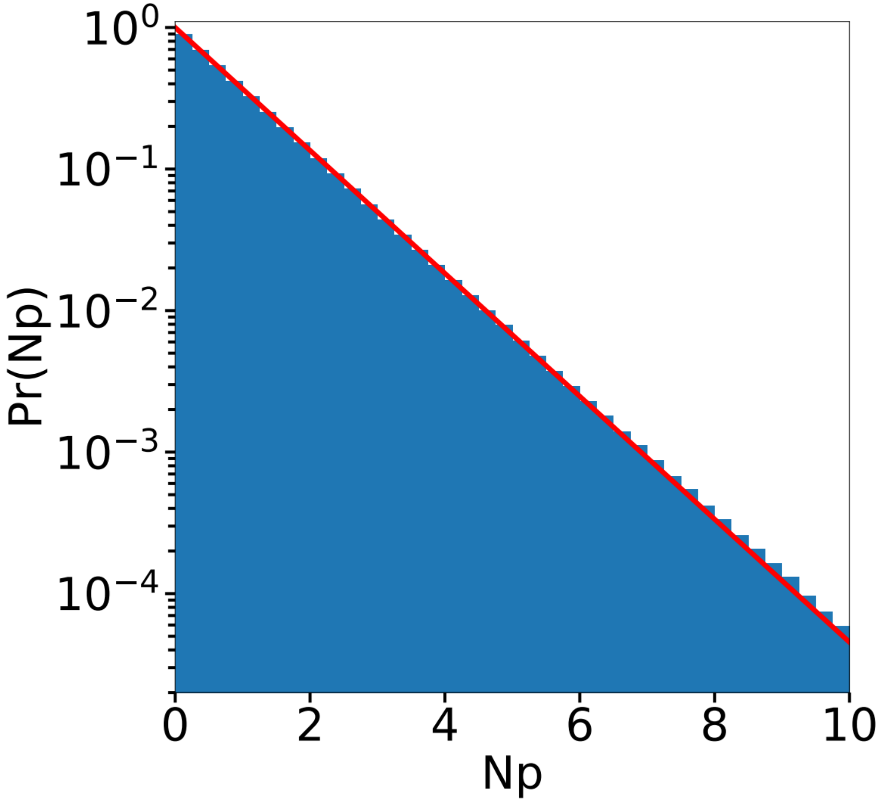
	# bitstrings	Time complexity	Space complexity
Google [1]	10^6	—	—
Cotengra [12]	1	3.10×10^{22}	2^{27}
Alibaba [18]	64	6.66×10^{18}	2^{29}
Ours	**2097152**	4.51×10^{18}	2^{30}

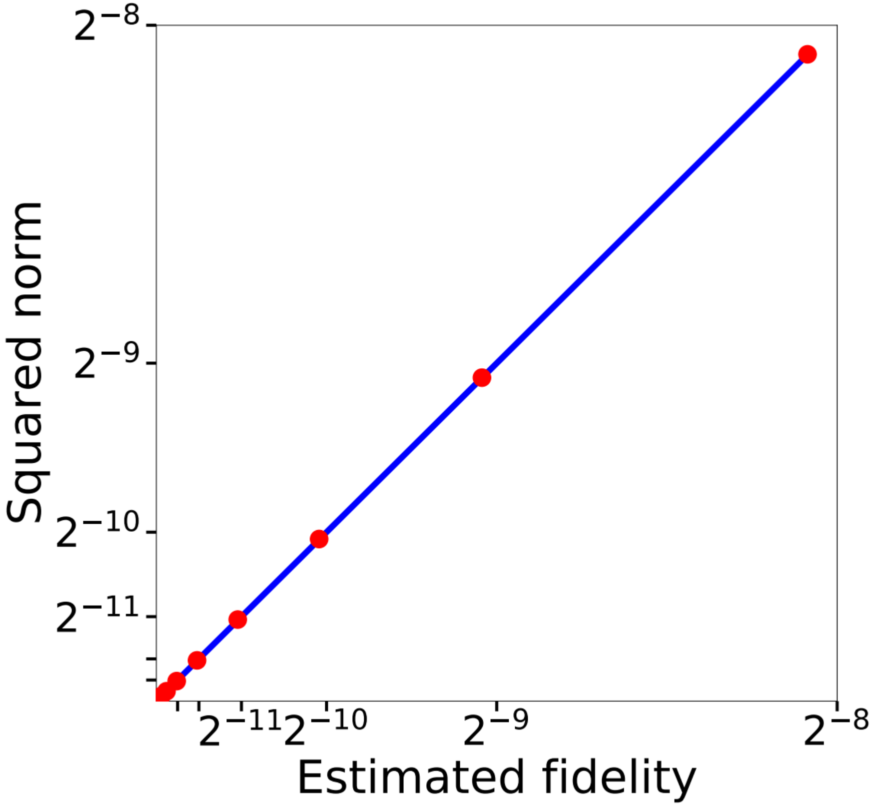
Results

Results

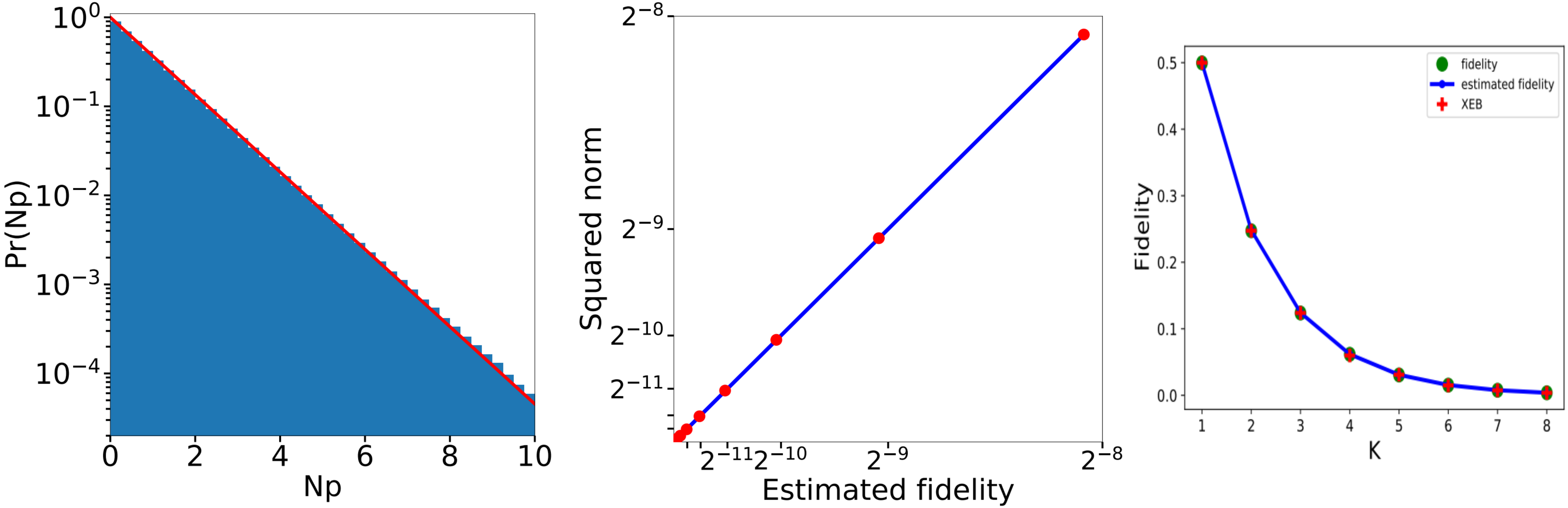


Results


Pr(Np)
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0 2 4 6 8 10
Np

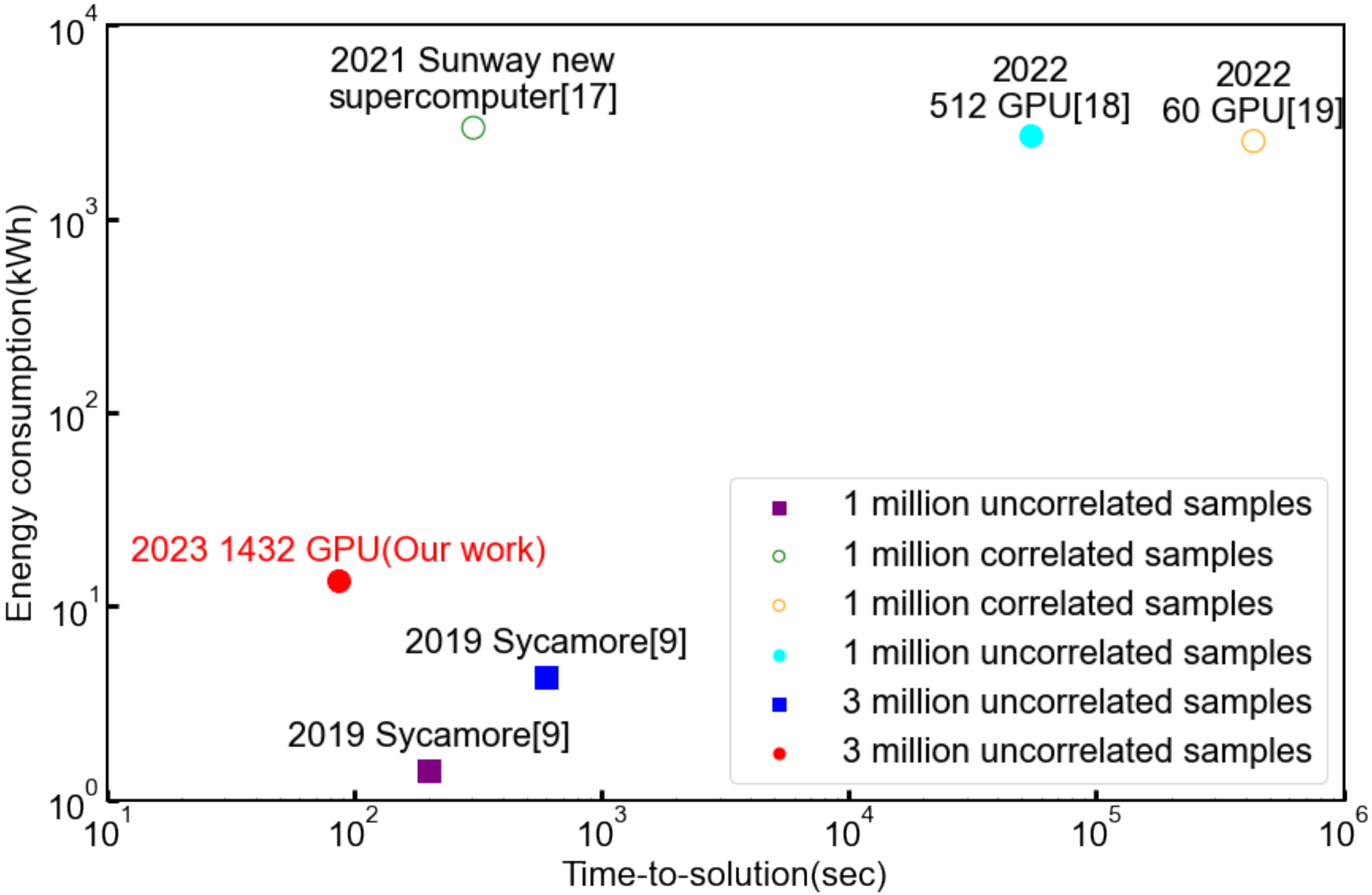

Squared norm
2^{-8}
2^{-9}
2^{-10}
2^{-11}
2^{-11} 2^{-10} 2^{-9} 2^{-8}
Estimated fidelity

Results



Pan, Chen and Zhang. *Phys. Rev. Lett.* 129.090502 (2022).

Advantage on Power Consumption?



1. improved slicing scheme
2. post-selection

Zhao, Zhong, Pan, et al ***in preparation***

Conclusion

Conclusion

- Arbitrary tensor network algorithms are very efficient tools for solving specific #P problems, either with low-rank structure or not.

- Their improvements are highly related to algorithmic and hardware developments (the hardware requirements are highly similar to large language models).

- There are many applications:
  - Classical simulation/validation of quantum computational tasks
  - Calculation of physical properties defined on complex systems
  - Exploring solution space of combinatorial optimization problems (tropical algebra)

Thanks for your attention!

$$
\begin{aligned}
E^* &= -\lim_{\beta\to\infty}\frac{1}{\beta}\sum_s e^{-\beta E(s)}\\
&= -\lim_{\beta\to\infty}\frac{1}{\beta}\sum_s \prod_{i<j} e^{\beta J_{ij}s_i s_j}\prod_i e^{h_i s_i}
\end{aligned}
$$

$$
\lim_{\beta\to\infty}\frac{1}{\beta}\ln(e^{\beta x}+e^{\beta y}) = x\oplus y, \qquad \frac{1}{\beta}\ln(e^{\beta x}\cdot e^{\beta y}) = x\odot y
$$

Jin-Guo Liu, Lei Wang, and Pan Zhang. "Tropical tensor network for ground states of spin glasses." Physical Review Letters 126.9 (2021): 090506.

Jin-Guo Liu, et al. "Computing solution space properties of combinatorial optimization problems via generic tensor networks." SIAM Journal on Scientific Computing 45.3 (2023): A1239-A1270.