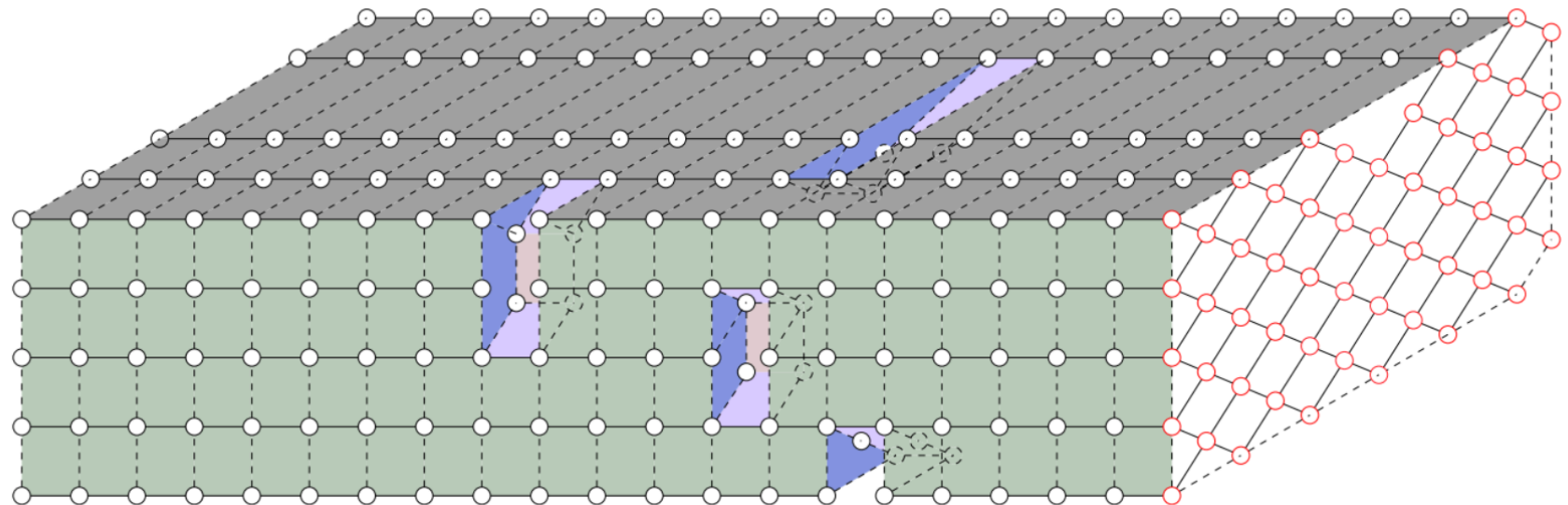
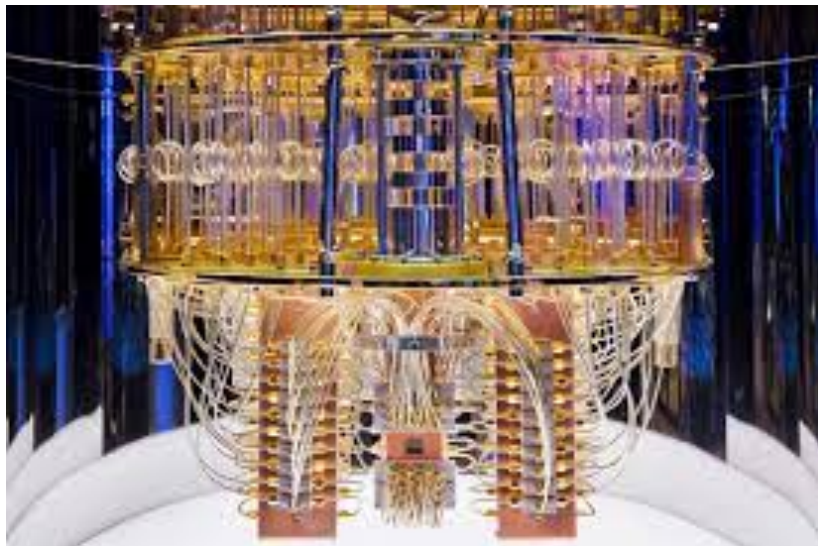


# quantum circuit simulations with Tensor Networks



Pan Zhang  
ITP,CAS

华南师大讲习班  
2022.11.21  
2022.11.23



# Outlines

## Simulation methods of general quantum circuits

- Full amplitude
- Single amplitude
- Approximate simulations: MPS, neural networks, path integrals
- Simulation of stabilizer circuits

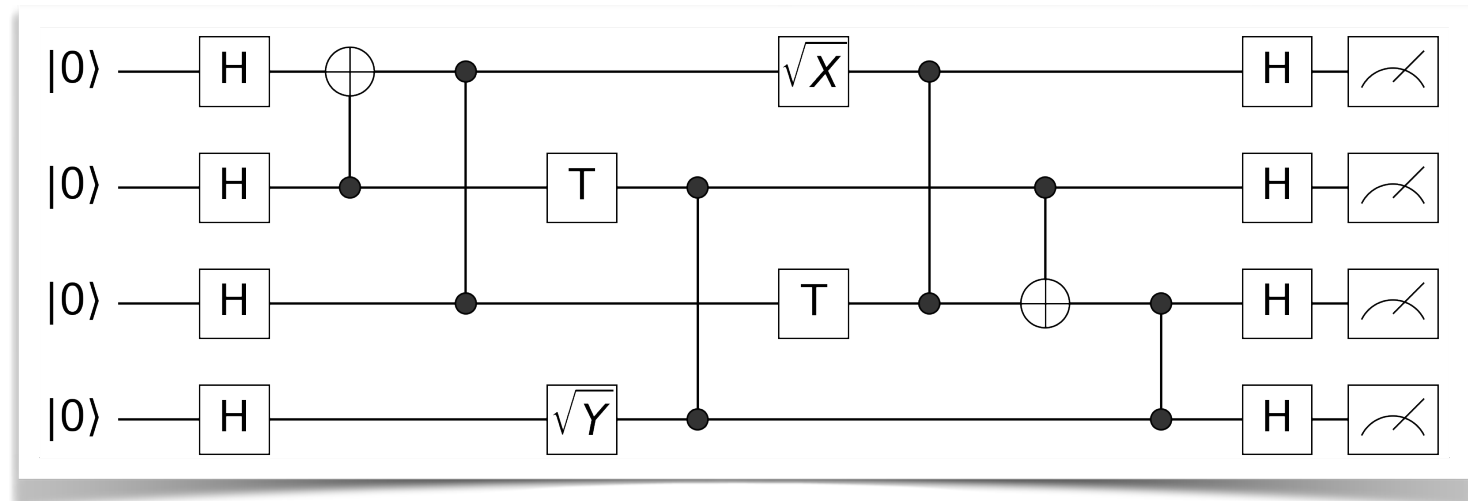
## Simulation of Google's Sycamore quantum circuits

- The big-batch method
- The sparse-state method

# Simulation of quantum circuits

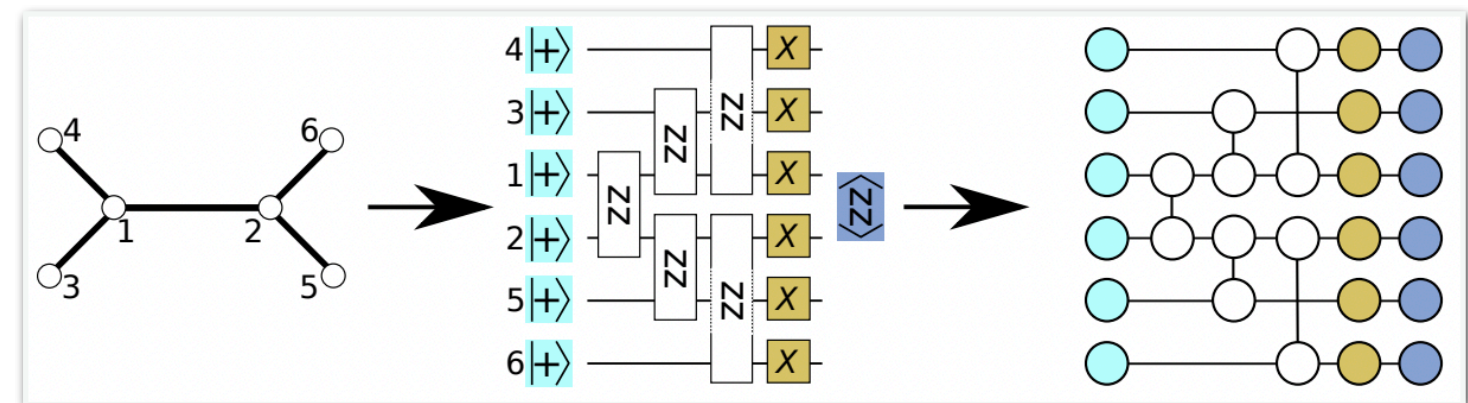
## Computing amplitudes

- Full amplitudes
- Single amplitude



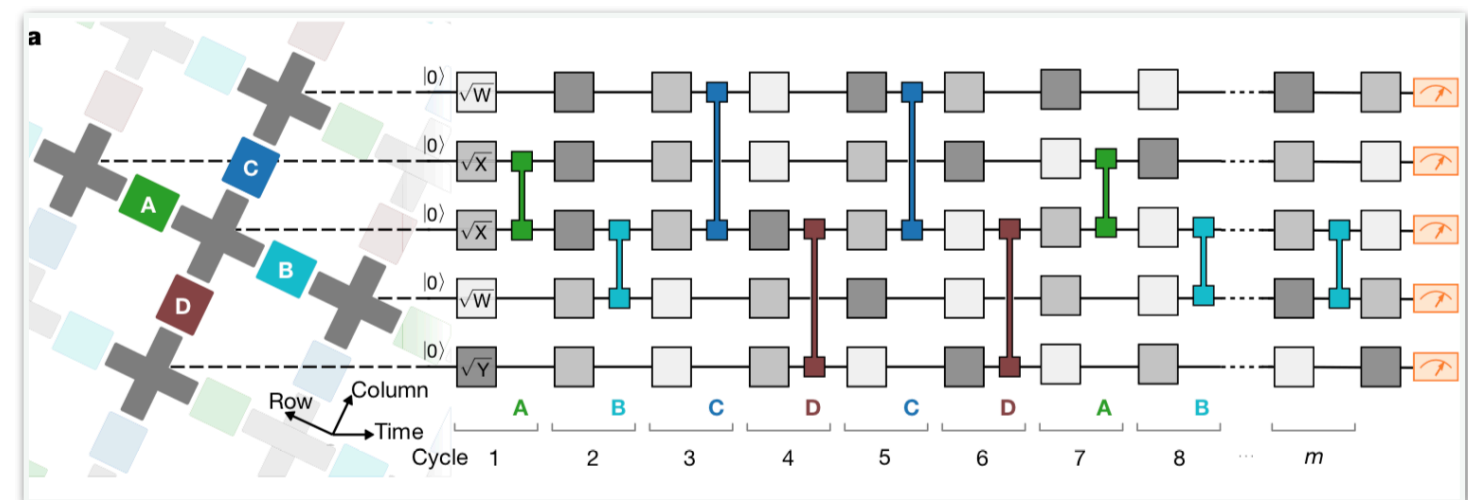
## Computing expectations

- Energy expectations (VQE, QAOA)

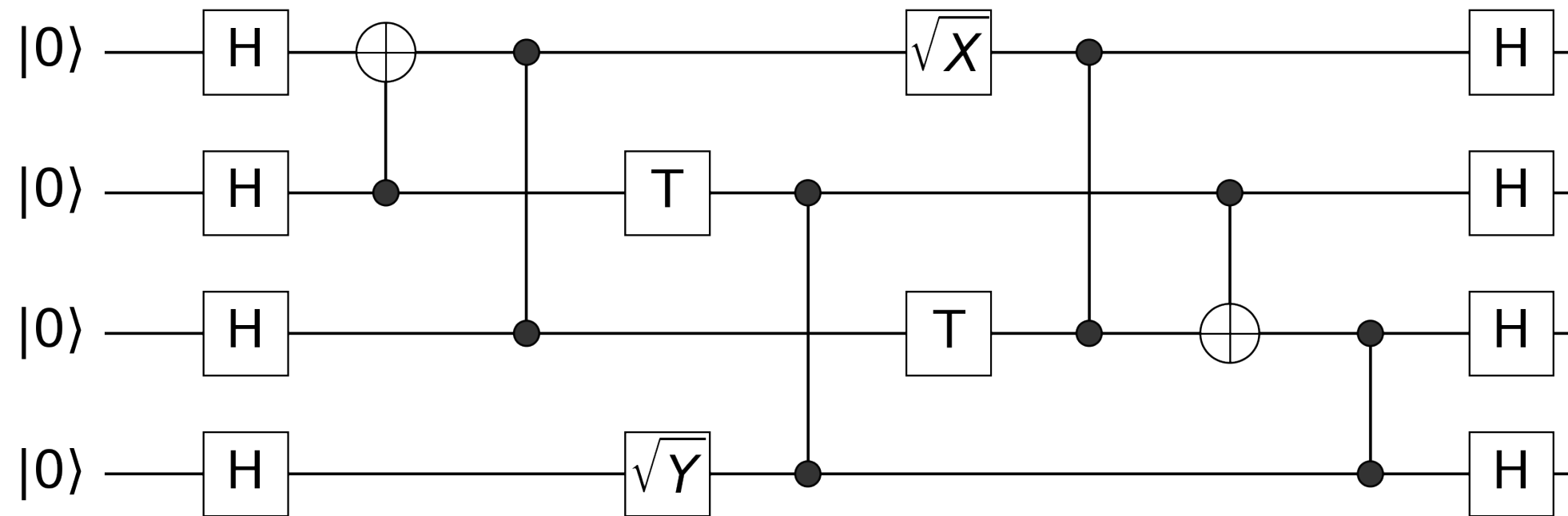


## Sampling problem

- Sycamore problem



# Quantum circuits

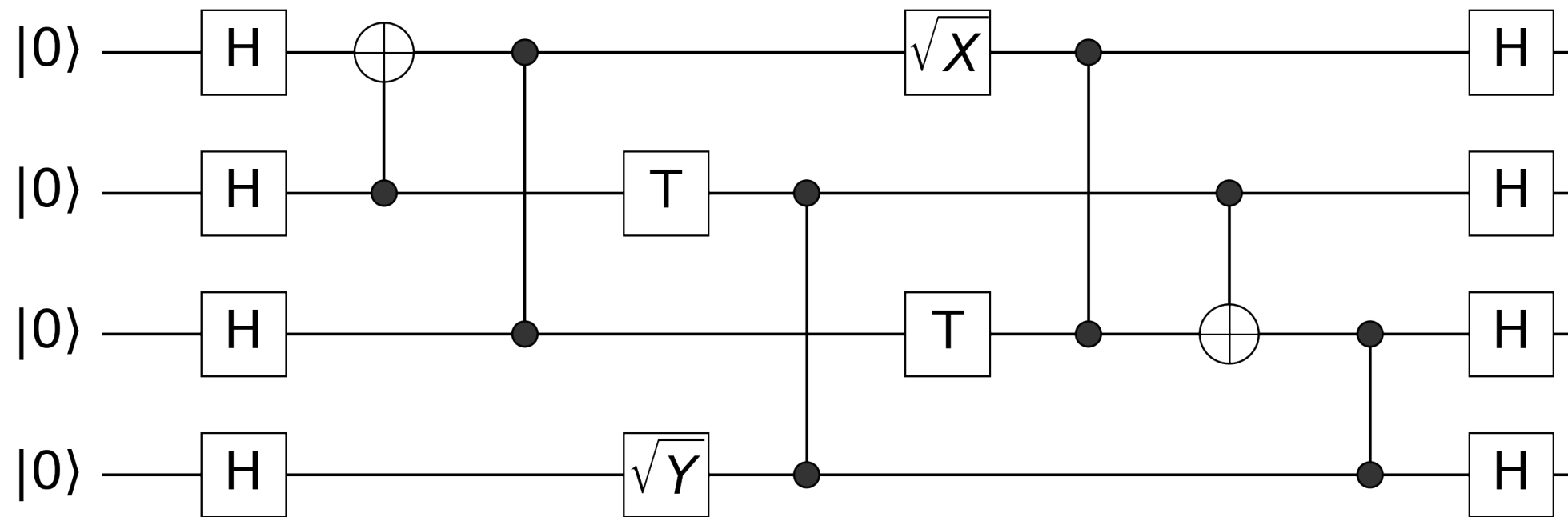


- Start from the initial state
- Applying unitary operators

**Full-amplitude simulation: store and update the state vector**



# States and Gates are tensors



$$|\psi_{\text{init}}\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$$

**Rank one**

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{---} \text{blue circle---}$$

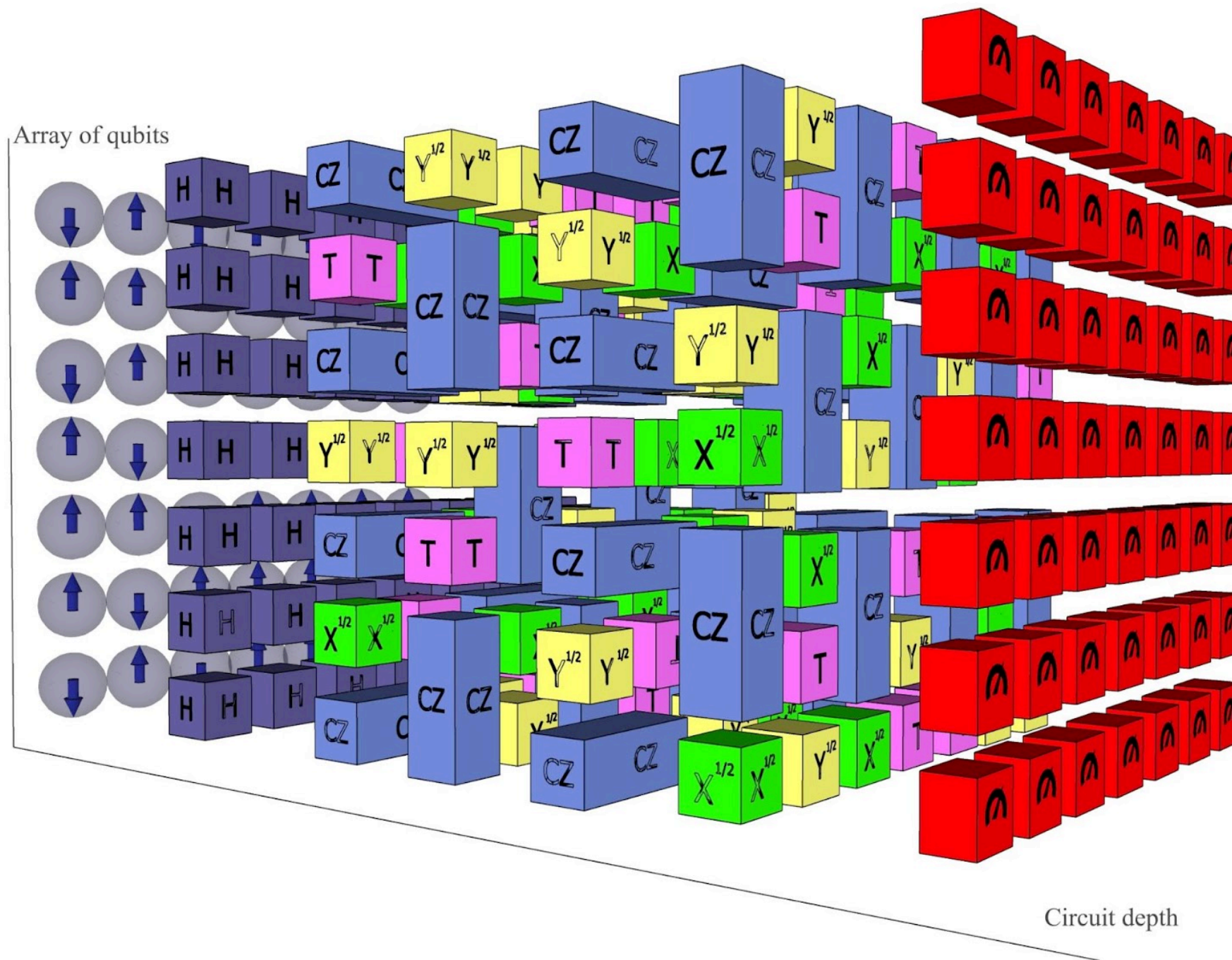
$$\text{---} \boxed{H} \text{---} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{---} \text{blue circle---}$$

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\text{---} \text{blue circle---} \text{---} \text{blue circle---} = \text{---} \text{blue circle---}$$

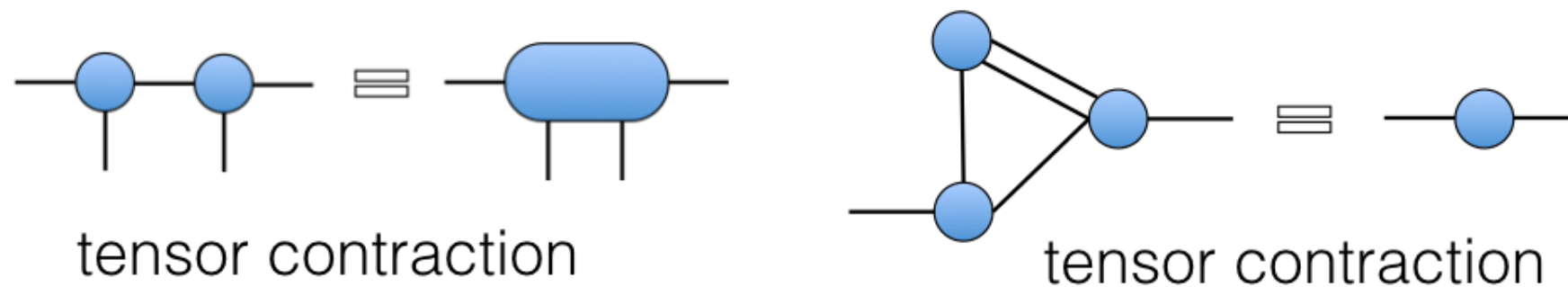
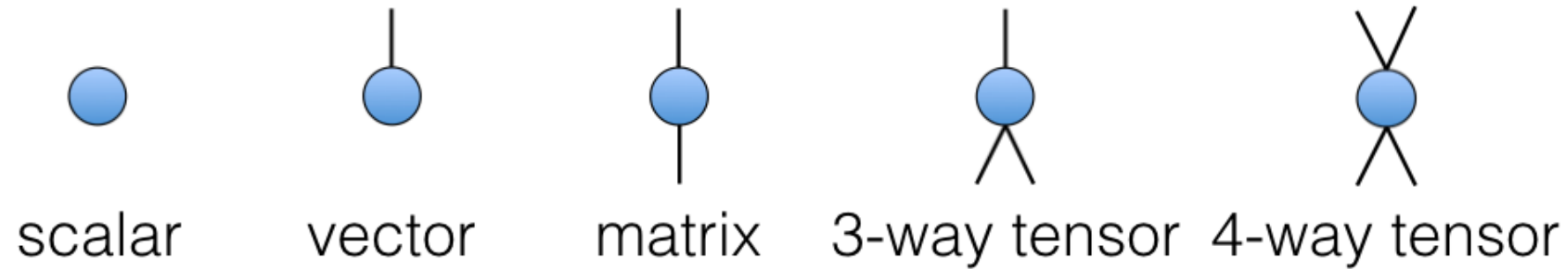
**Each dimension of the state vector is independently operated**

# States and Gates are tensors



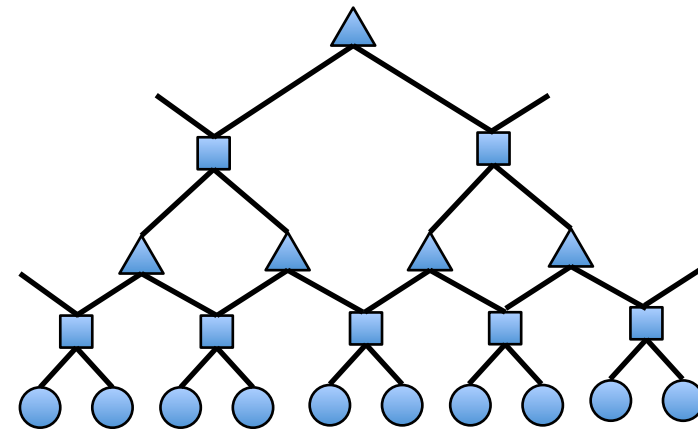
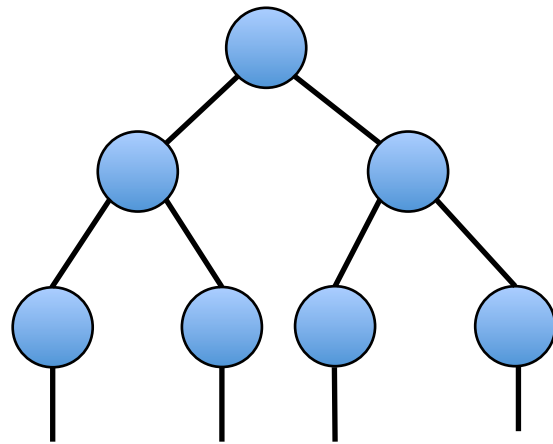
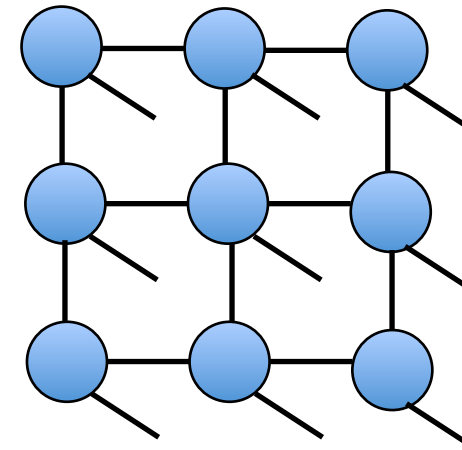
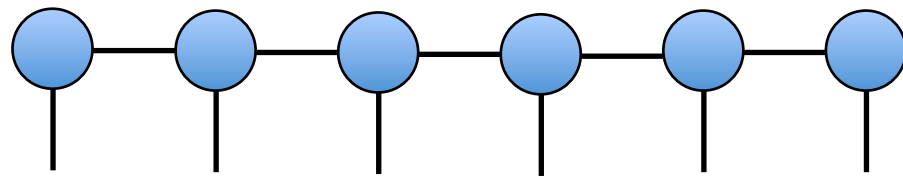
<https://ai.googleblog.com/2018/05/the-question-of-quantum-supremacy.html>

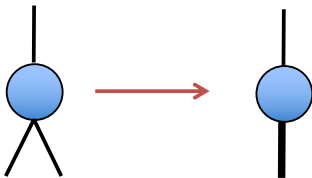
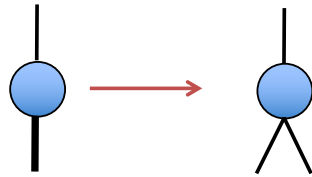
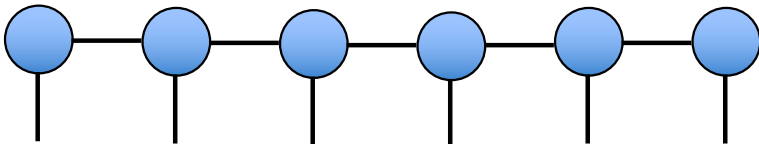
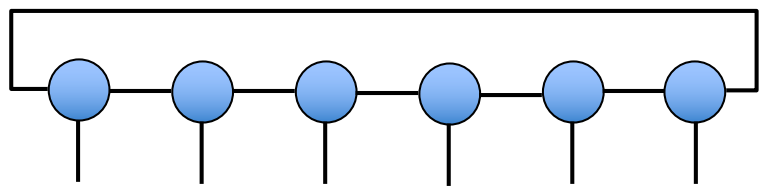
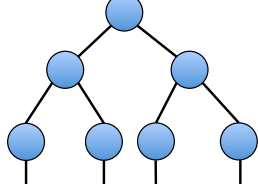
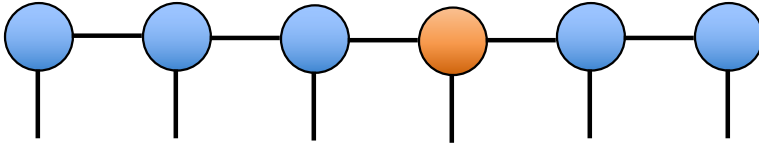
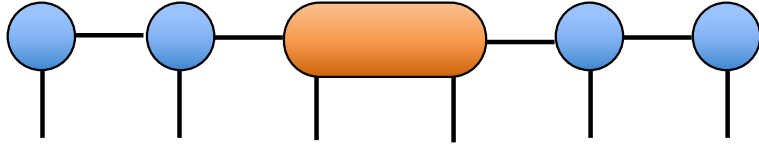
# Diagram notation of tensor networks



**Only Linear operations !!!**

# Tensor networks in physics: imposing prior of physical wave functions

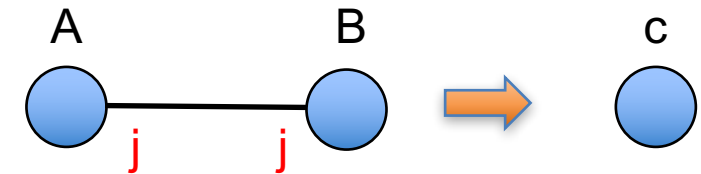


In Physics	Out of Physics	Diagram
grouping of indices	unfolding, matricization	
splitting of indices	tensorizing	
matrix product states	tensor train decomposition	
periodic boundary MPS	tensor chain decomposition	
tree tensor networks	hierarchical Tucker decomposition	
single-site DMRG	alternating least square	
two-site DMRG	modified alternating least square	

# Einsum notations of tensor network contractions

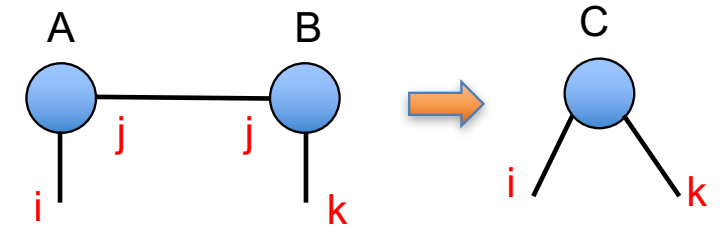
$$c = \text{einsum}(A, B, "j, j")$$

$$c = A \cdot B$$



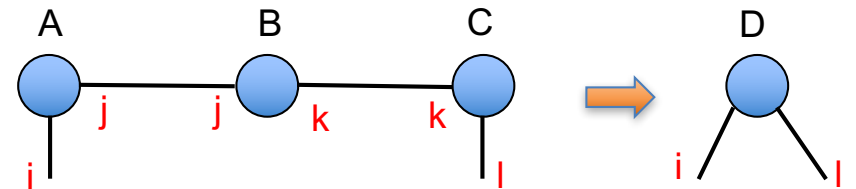
$$C = \text{einsum}(A, B, "ij, jk \rightarrow ik")$$

$$C = AB$$



$$D = \text{einsum}(A, B, C, "ij, jk, kl")$$

$$D = ABC$$



**Space complexity:** the dimension of the largest tensor

**Time complexity:** product of dimensions of all unique indices

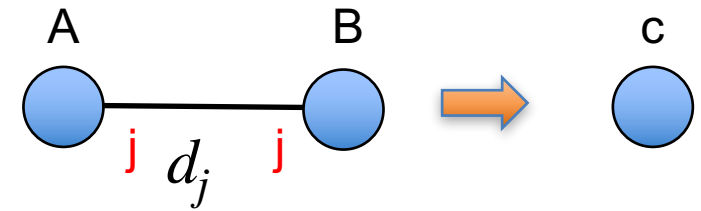
# Computational complexity

## Time complexity

$$c = \sum_{j=1}^{d_j} A_j B_j$$

$d_j$

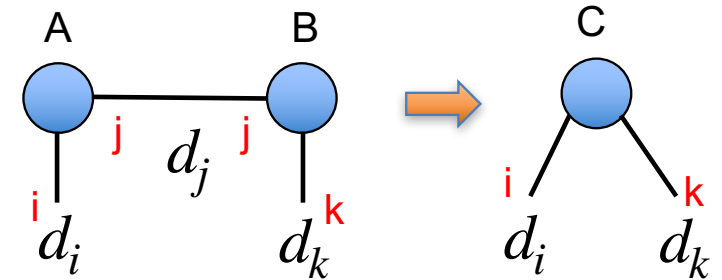
$$c = A \cdot B$$



$$C_{ik} = \sum_{j=1}^{d_j} A_{ij} B_{jk}$$

$d_i d_j d_k$

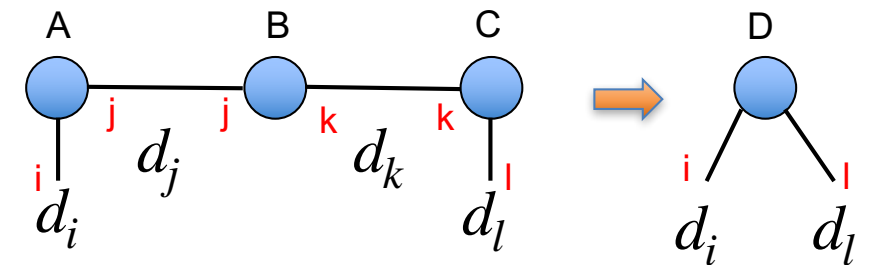
$$C = AB$$



$$D_{il} = \sum_{j=1}^{d_j} \sum_{k=1}^{d_k} A_{ij} B_{jk} C_{kl}$$

$d_i d_j d_k d_l$

$$D = ABC$$

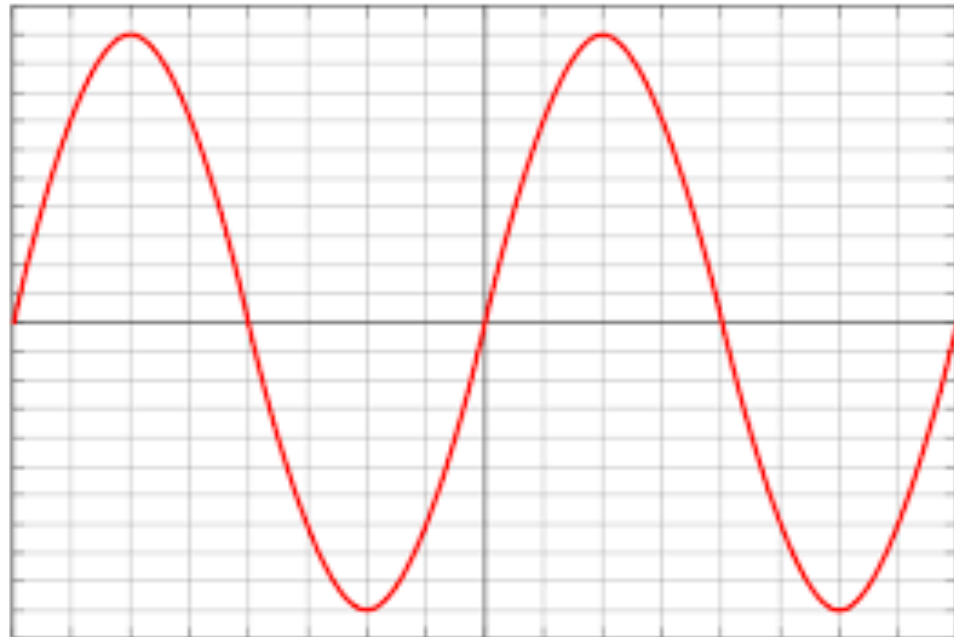


**Space complexity: the dimension of the largest tensor**

**Time complexity: product of dimensions of all unique indices**

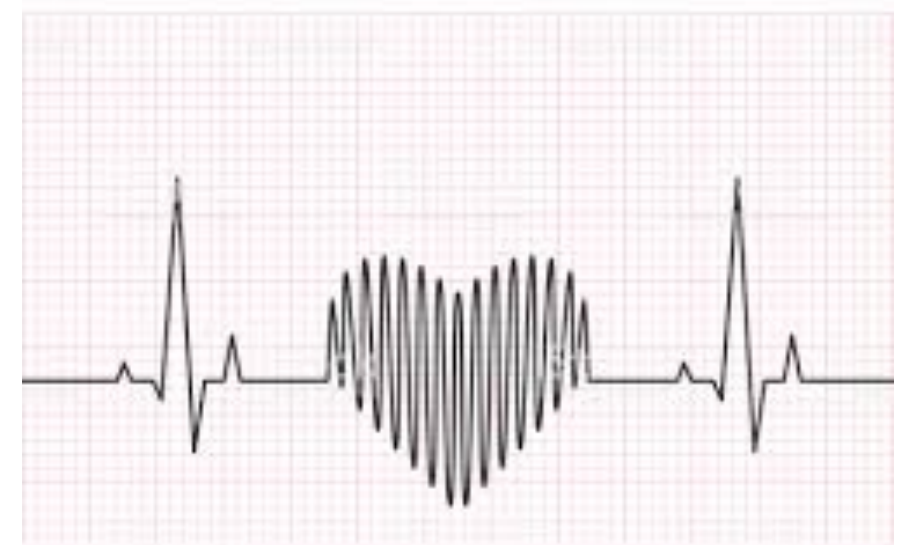
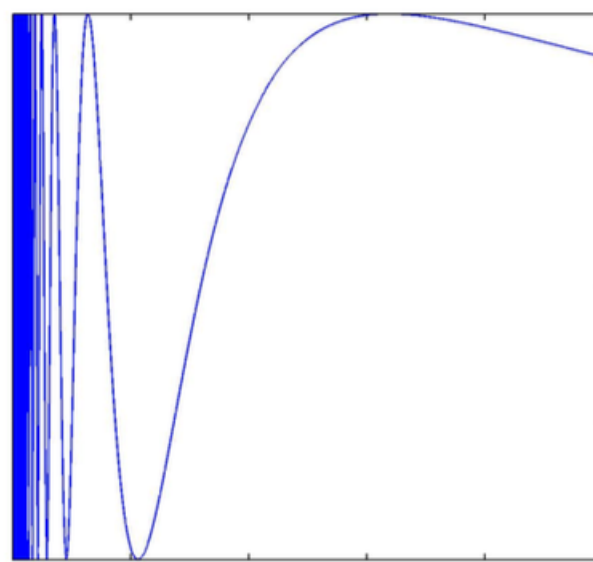
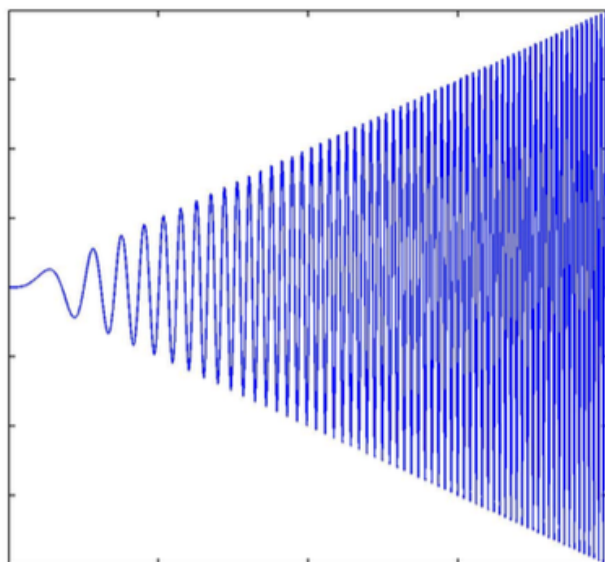


# Automatically detecting structures in the data

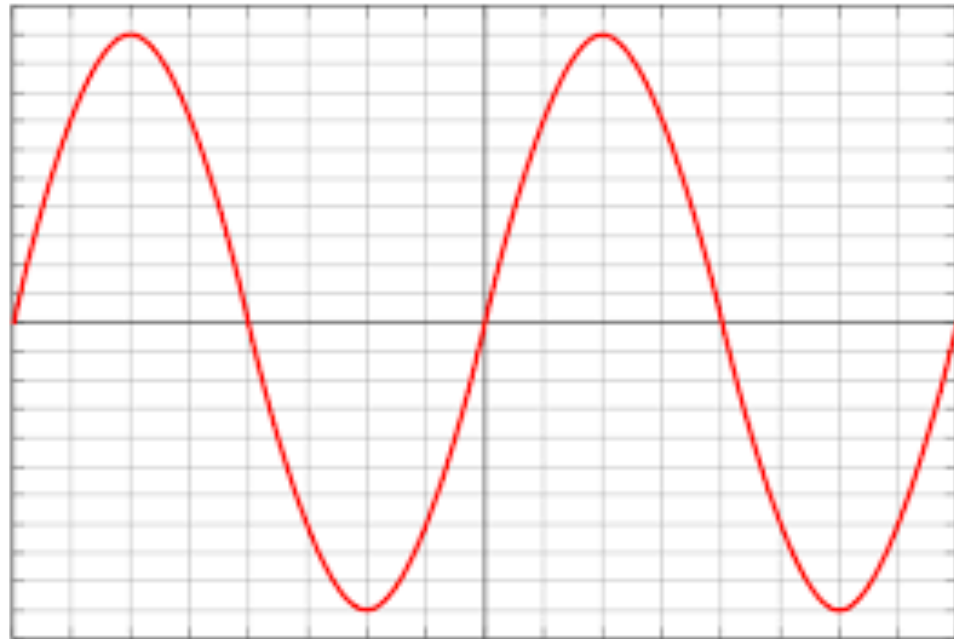


[ 0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,  
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,  
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,  
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,  
.....  
.....  
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,  
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,  
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,  
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938 ]

$10^6$  data points in a vector

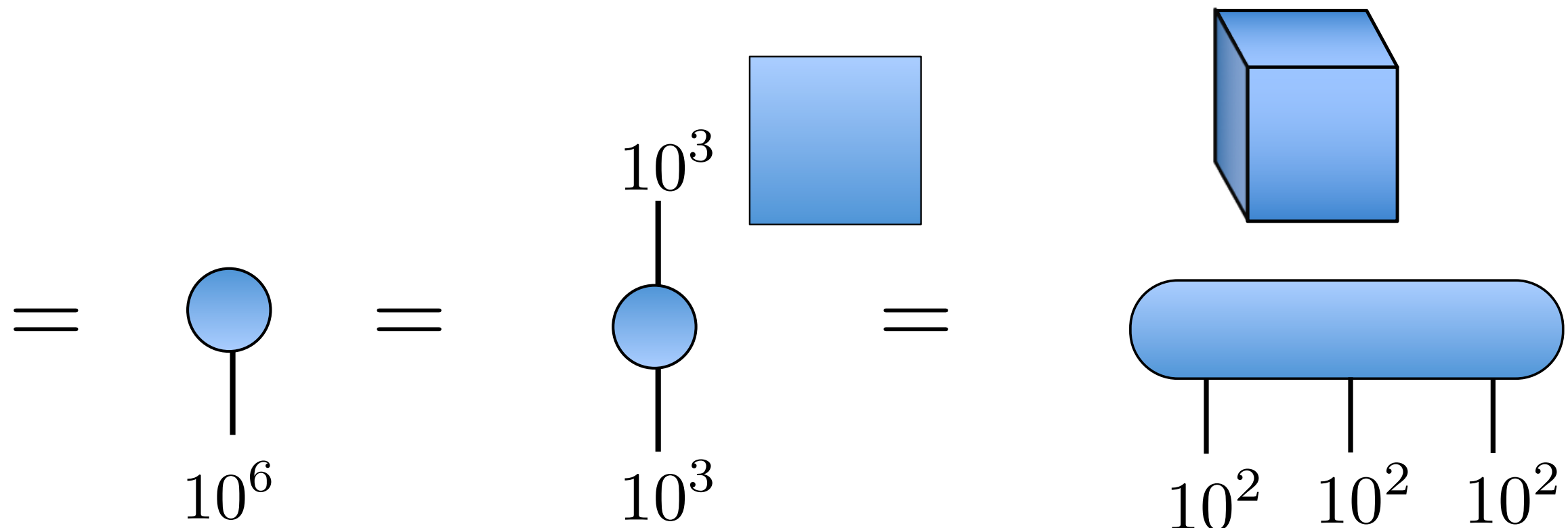


# Automatically detecting structures in the data

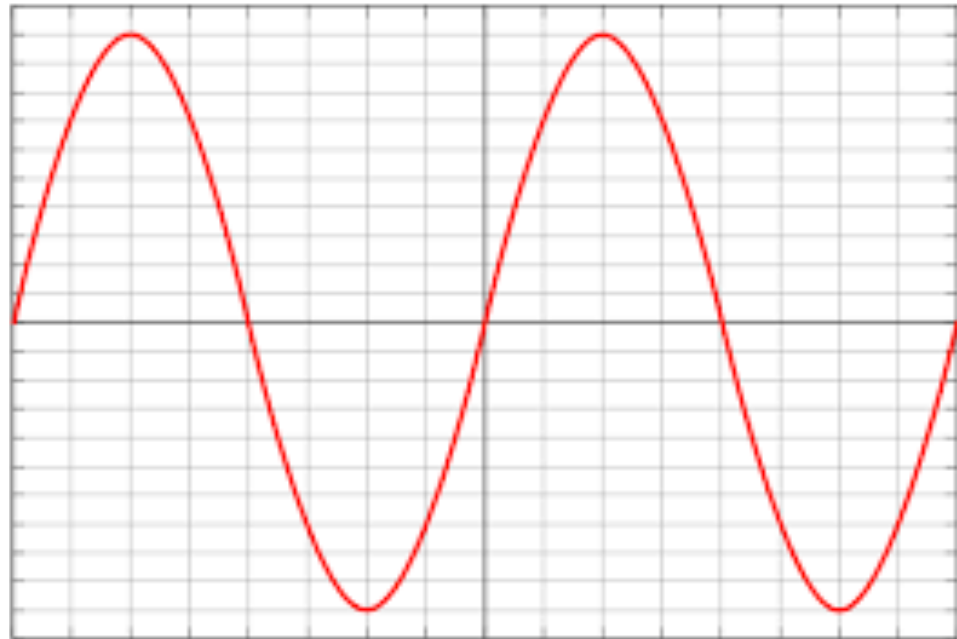


[ 0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,  
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,  
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,  
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,  
.....  
.....  
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,  
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,  
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,  
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938 ]

$10^6$  data points in a vector

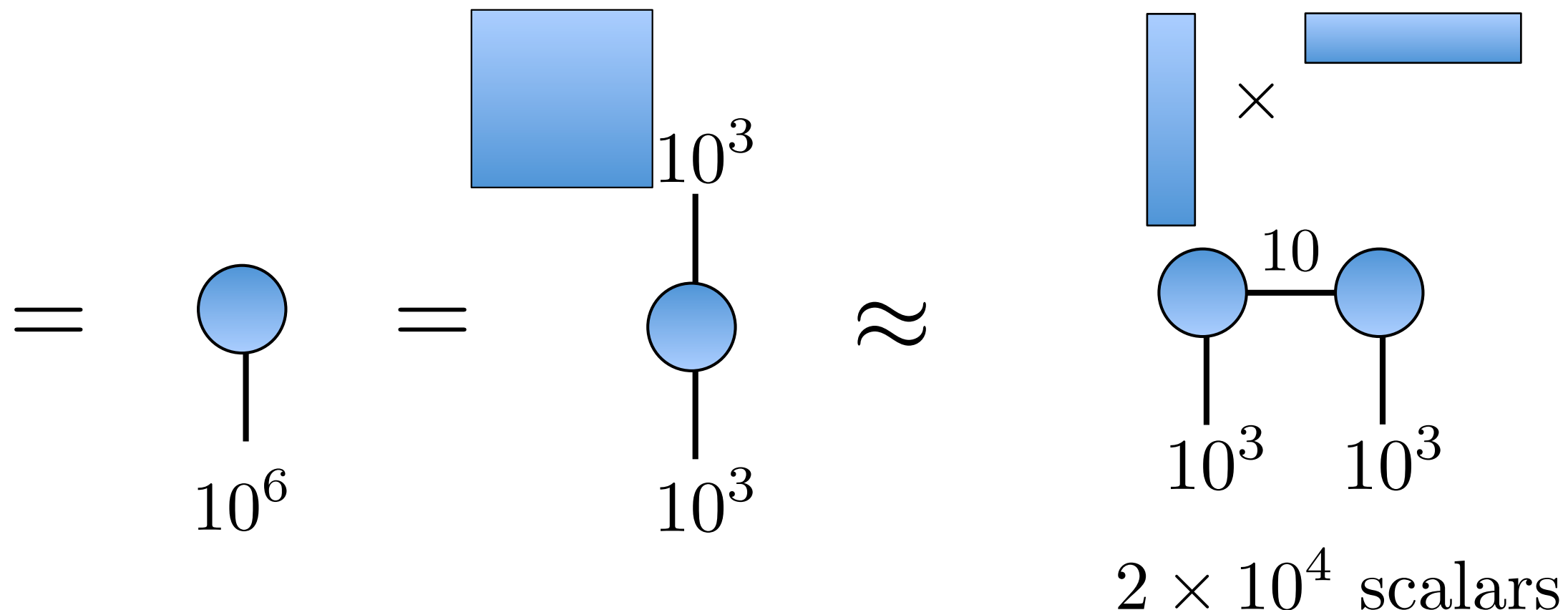


# Automatically detecting structures in the data

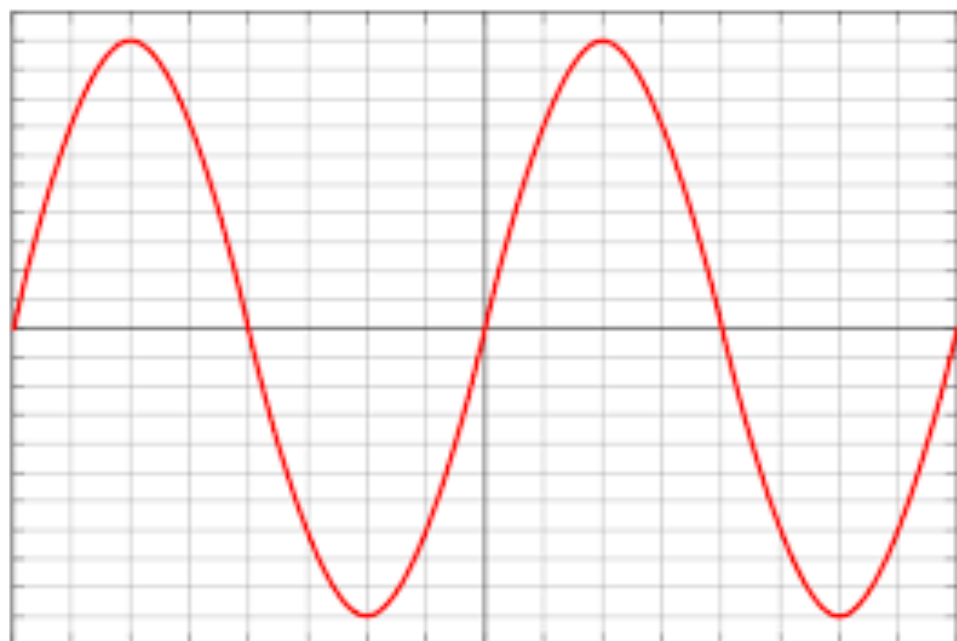


[ 0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,  
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,  
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,  
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,  
.....  
.....  
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,  
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,  
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,  
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938 ]

$10^6$  data points in a vector

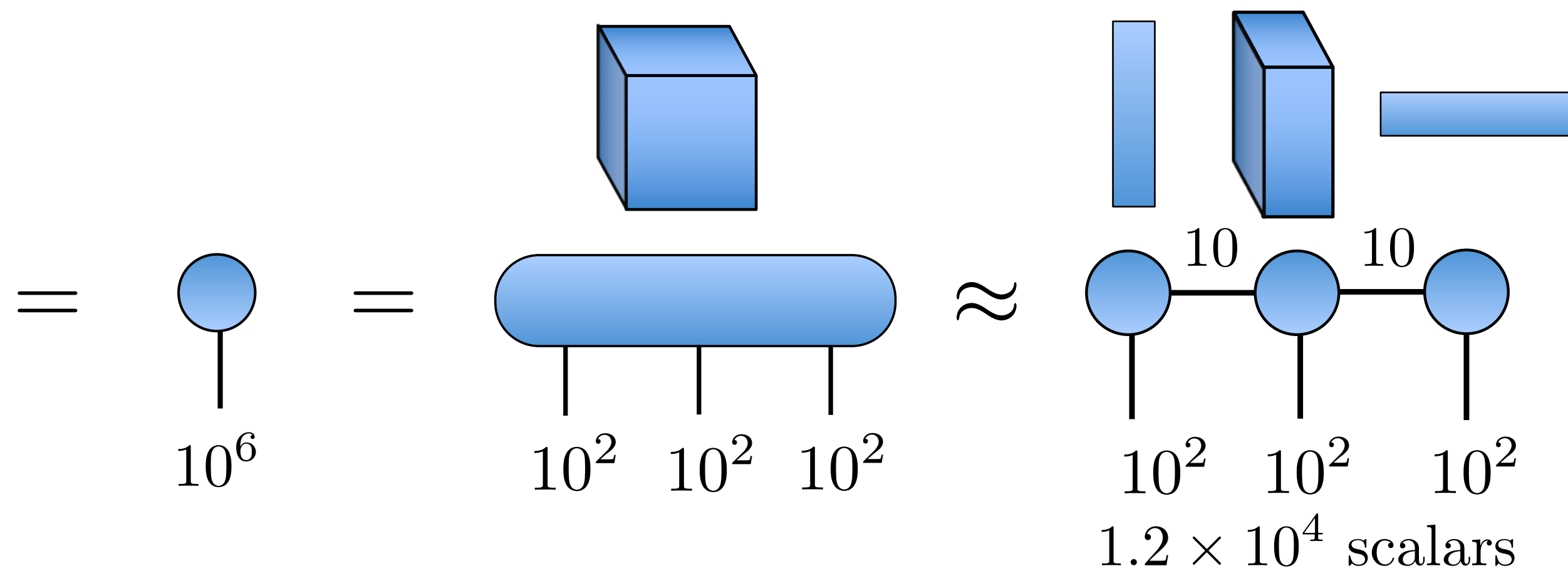


# Automatically detecting structures in the data



[ 0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,  
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,  
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,  
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,  
.....  
.....  
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,  
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,  
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,  
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938 ]

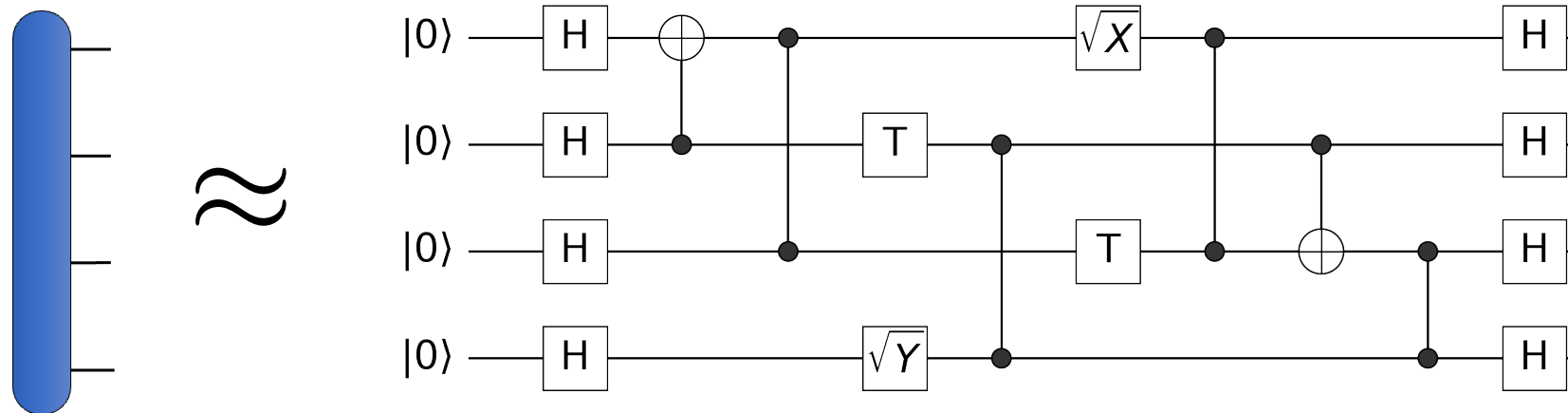
$10^6$  data points in a vector



# Two problems

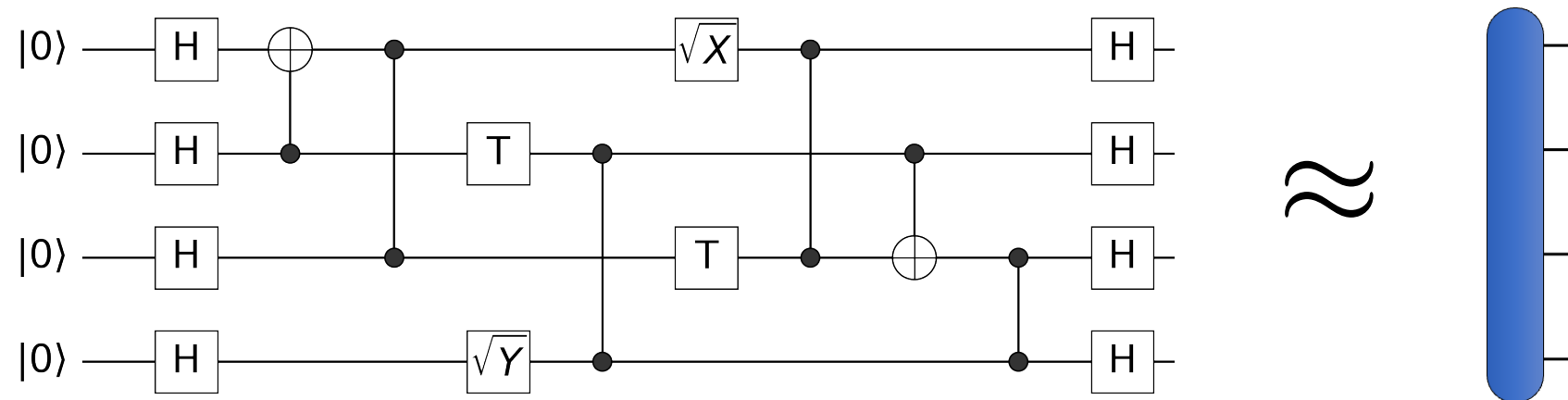
## Tensor decomposition









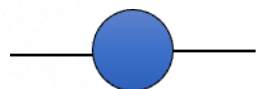




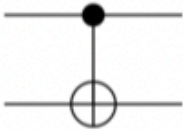
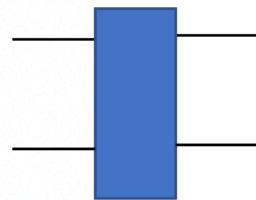
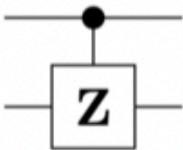
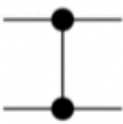
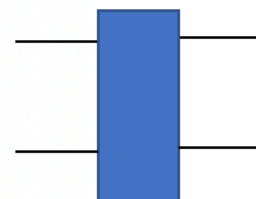

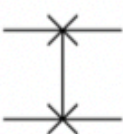
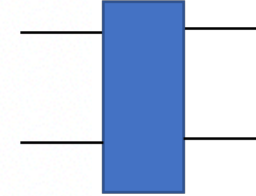
(Design / Learn a circuit)



## Tensor contraction

(Simulate a circuit)




Operator	Gate(s)		Matrix	
Pauli-X (X)			$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	
Pauli-Y (Y)			$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	
Pauli-Z (Z)			$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	
Hadamard (H)			$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	
Phase (S, P)			$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	
$\pi/8$ (T)			$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	
Controlled Not (CNOT, CX)			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	
Controlled Z (CZ)			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	
SWAP			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	

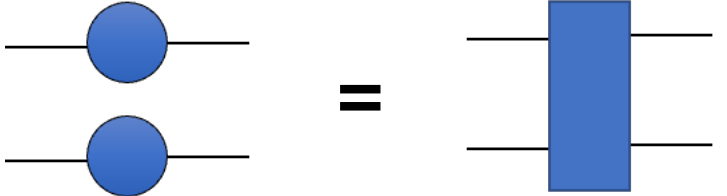


# Gate composition

$$|\psi\rangle \text{---} \boxed{Y} \text{---} \boxed{X} \text{---} = \text{---} \boxed{X \cdot Y} \text{---} \quad XY |\psi\rangle$$

$$C = X \cdot Y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} = iZ$$


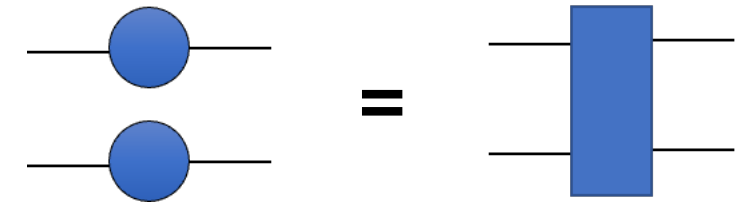
$$\begin{array}{l} |\psi\rangle \text{---} \boxed{Y} \text{---} Y|\psi\rangle \\ |\phi\rangle \text{---} \boxed{X} \text{---} X|\phi\rangle \end{array} \Leftrightarrow \left. \begin{array}{l} |\psi\rangle \text{---} \\ |\phi\rangle \text{---} \end{array} \right\} \boxed{Y \otimes X} \left. \begin{array}{l} \text{---} \\ \text{---} \end{array} \right\} (Y \otimes X) |\psi \otimes \phi\rangle$$

$$Y \otimes X = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & -i \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ i \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & 0 \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{bmatrix}$$


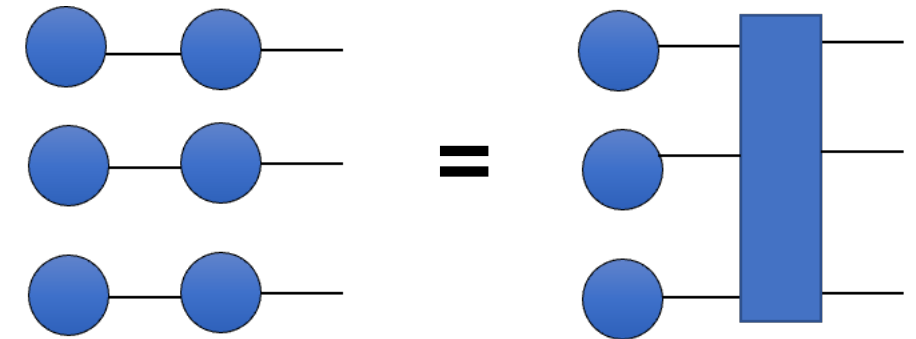


# Parallel operations

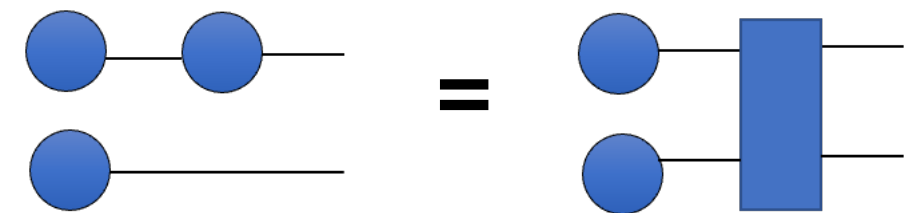
$$H \otimes H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$



$$\left. \begin{array}{l} |\psi_2\rangle \text{---} [H] \text{---} \\ |\psi_1\rangle \text{---} [H] \text{---} \\ |\psi_0\rangle \text{---} [H] \text{---} \end{array} \right\} H^{\otimes 3} |\psi\rangle = H|\psi_2\rangle \otimes H|\psi_1\rangle \otimes H|\psi_0\rangle$$



$$|\psi\rangle \left\{ \begin{array}{l} \text{---} [H] \text{---} \\ \text{---} \end{array} \right\} = \begin{array}{l} \text{---} [H] \text{---} \\ \text{---} [I] \text{---} \end{array} = \begin{array}{l} \text{---} [H \otimes I] \text{---} \end{array} \left\} (H \otimes I) |\psi\rangle$$



# Full-amplitude simulation: the Schrödinger algorithm

HiQ

Cirq

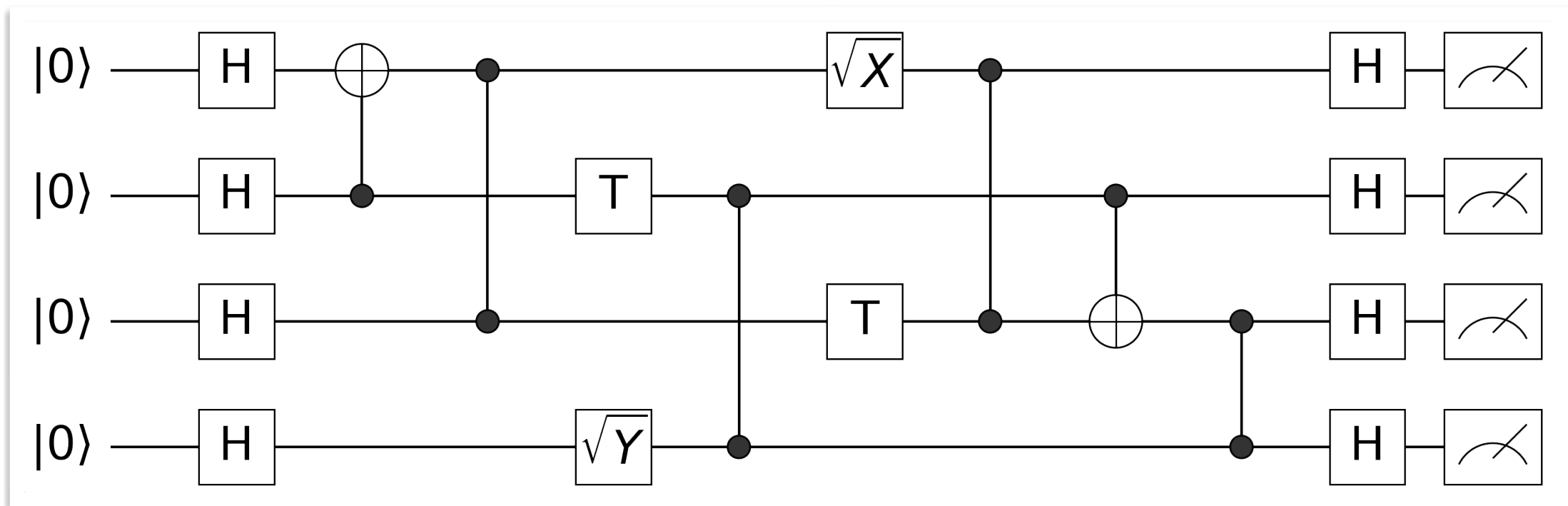
**Yao.jl**

Qiskit

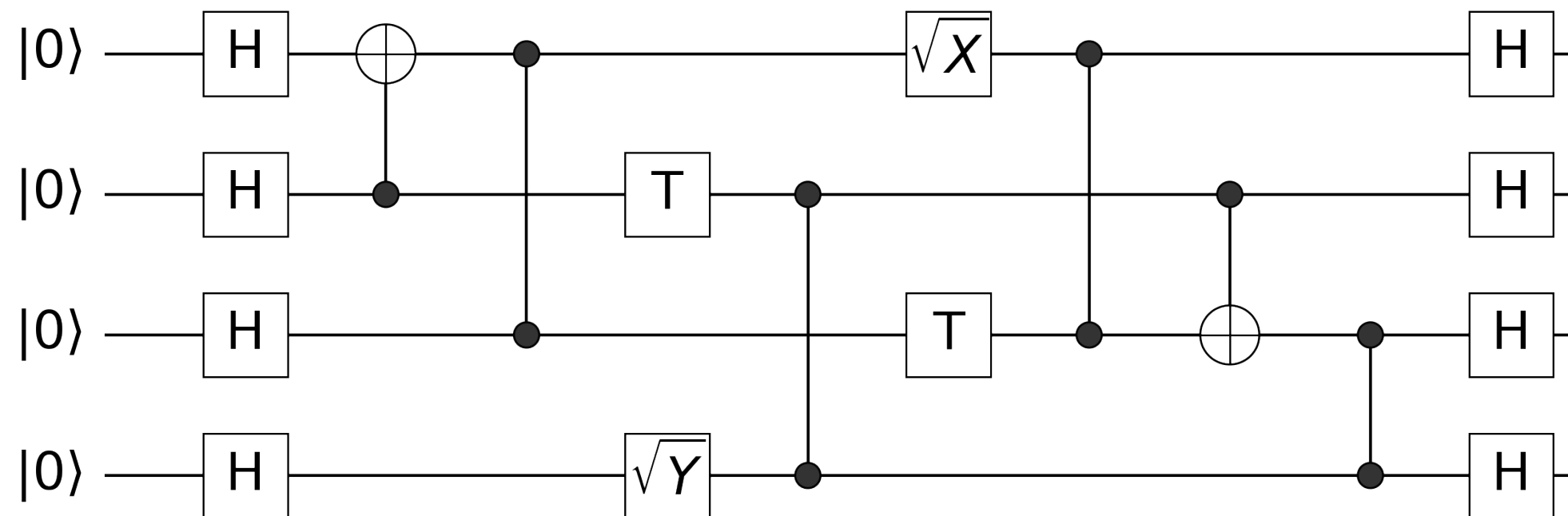
Qulacs

Azure Quantum

Quantum Paddle



# The Schrödinger algorithm



$$|\psi\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$$

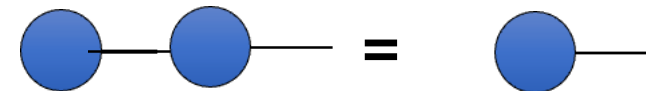
**Rank one**

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



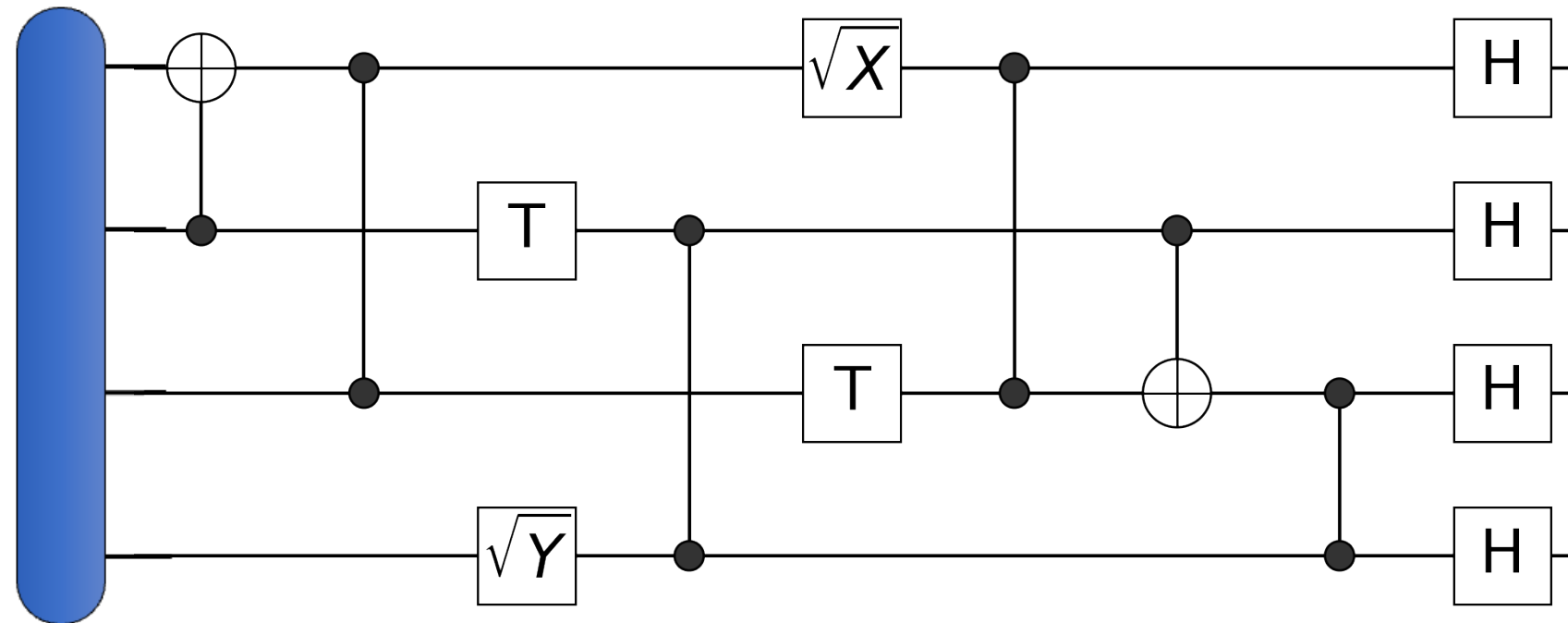
$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\text{---} \boxed{\text{H}} \text{---} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$



**Each dimension of the state vector is independently operated**

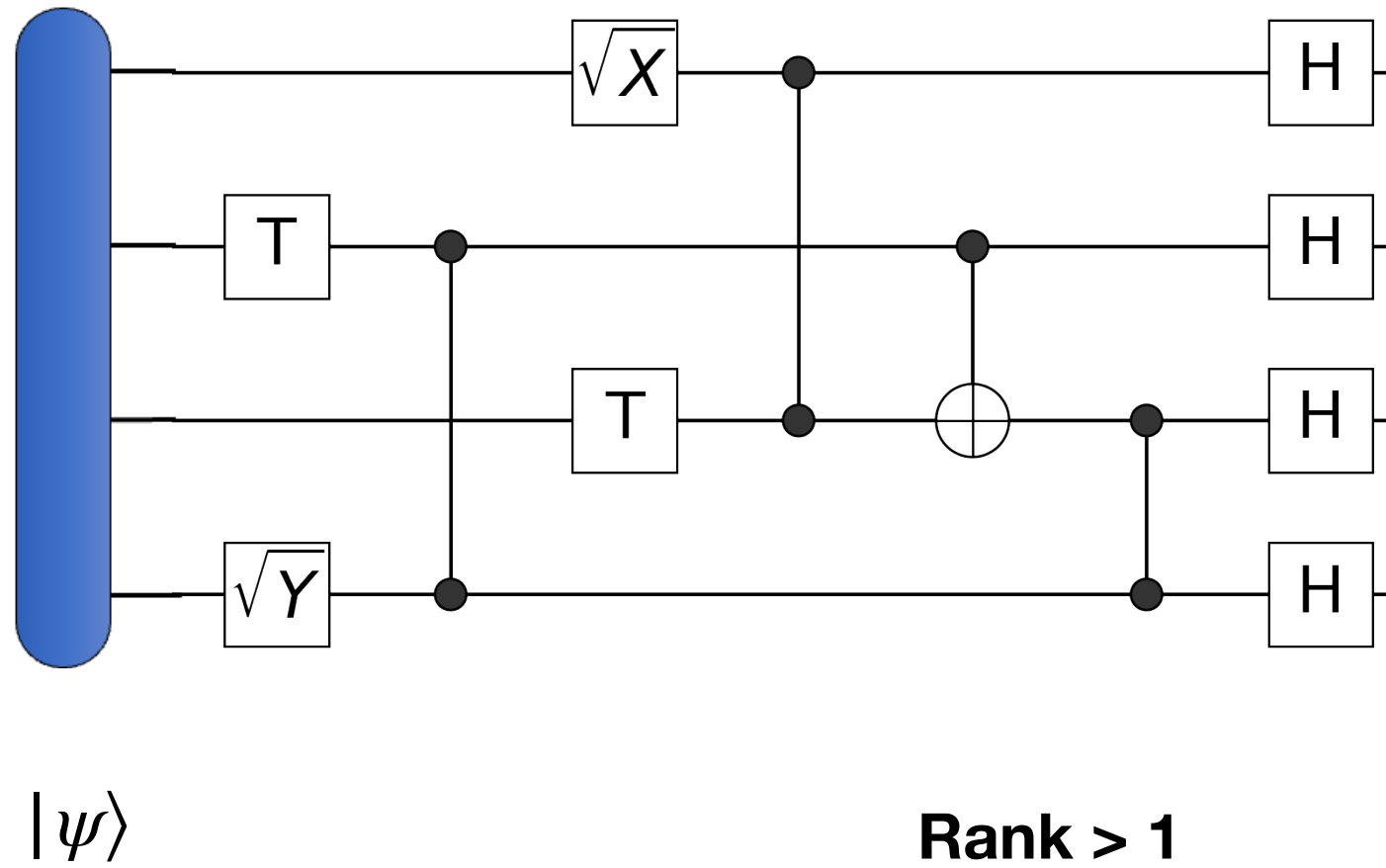
# The Schrödinger algorithm



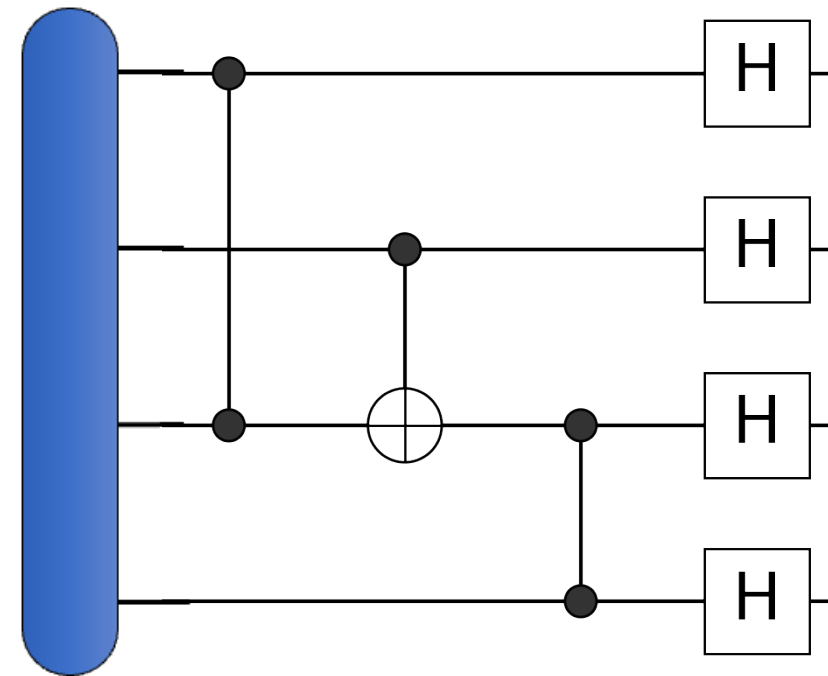
$$|\psi\rangle = |+\rangle \otimes |+\rangle \otimes |+\rangle \otimes |+\rangle \quad \text{Rank one}$$

- **Permute+ reshape** the tensor to a matrix; reshape the gate to a matrix; apply matrix multiplications
- Use einsum, tensordot ...

# The Schrödinger algorithm



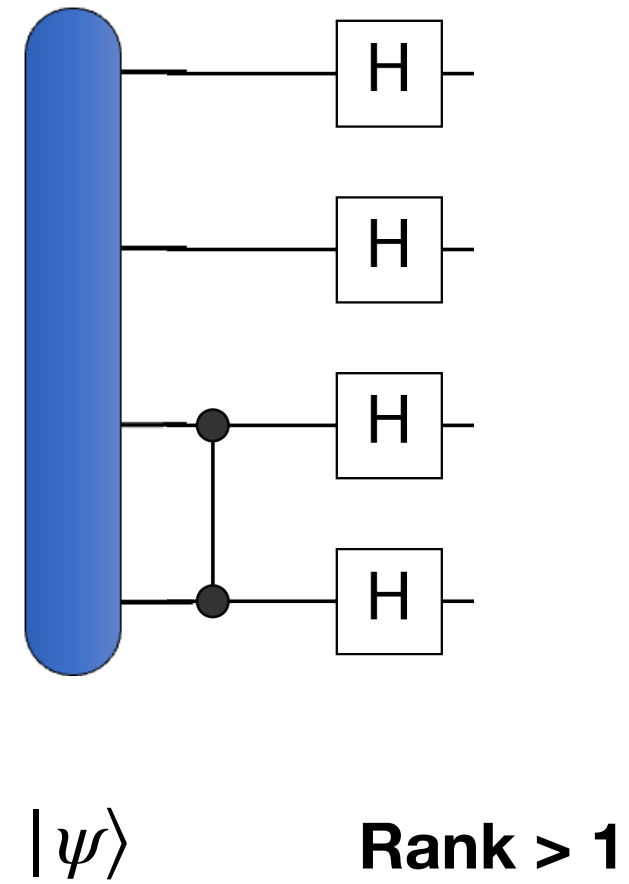
# The Schrödinger algorithm



$|\psi\rangle$

**Rank > 1**

# The Schrödinger algorithm





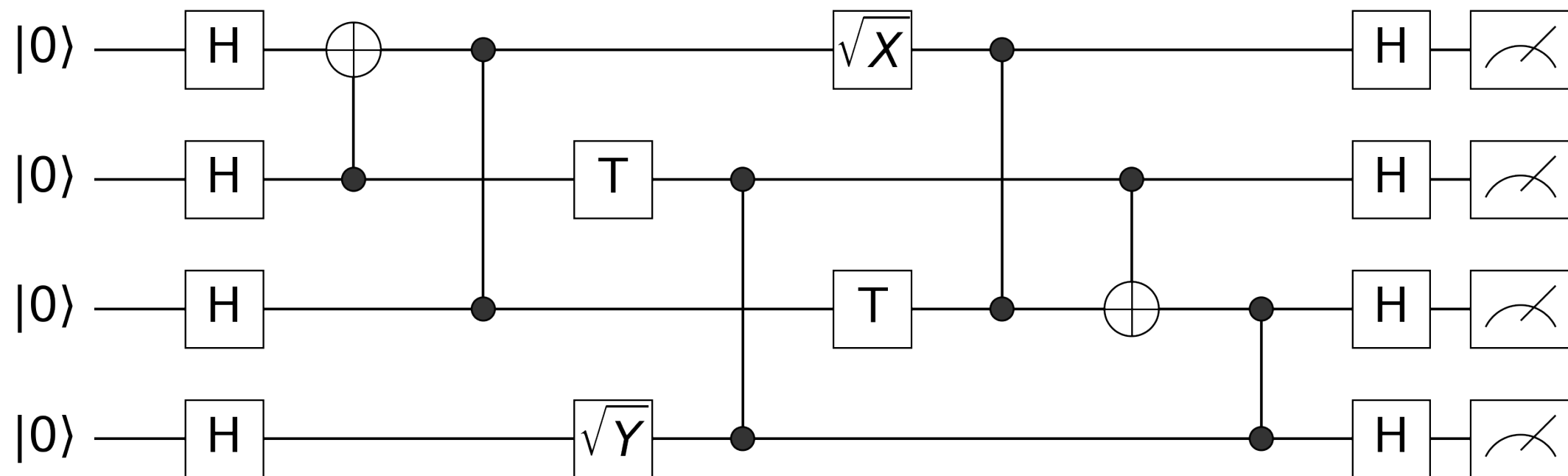
# The Schrödinger algorithm



$|\psi\rangle$

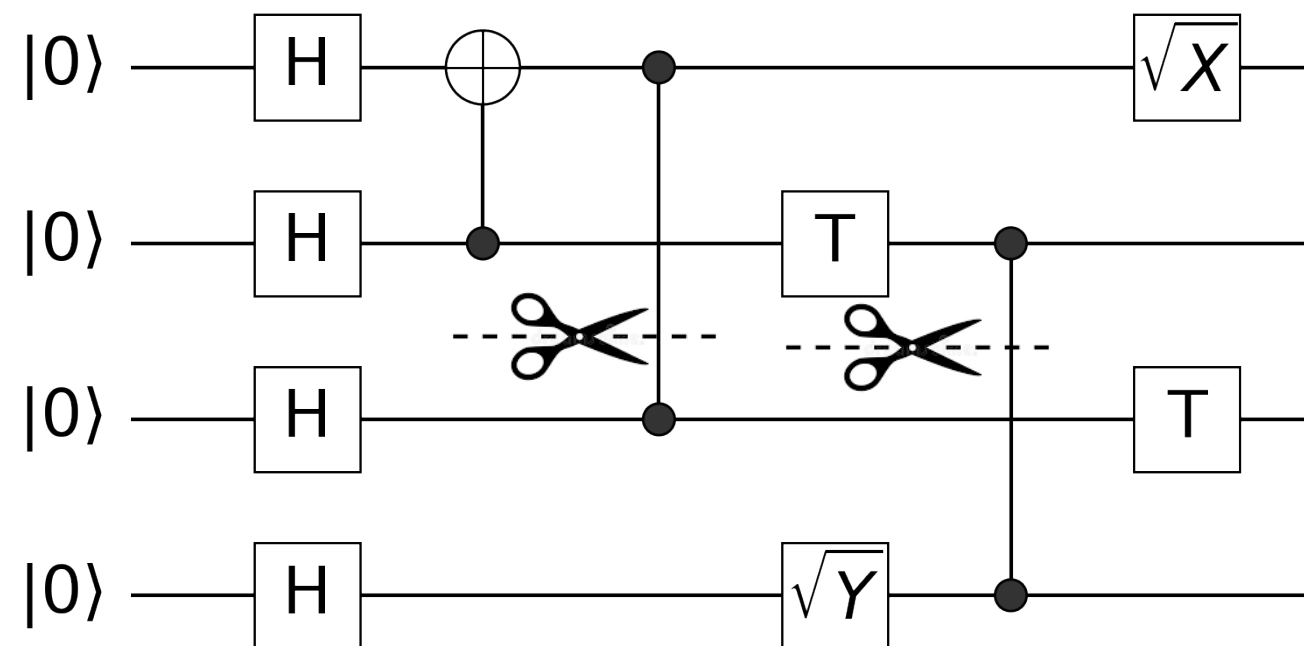
**Rank = 4**

# The Schrödinger algorithm

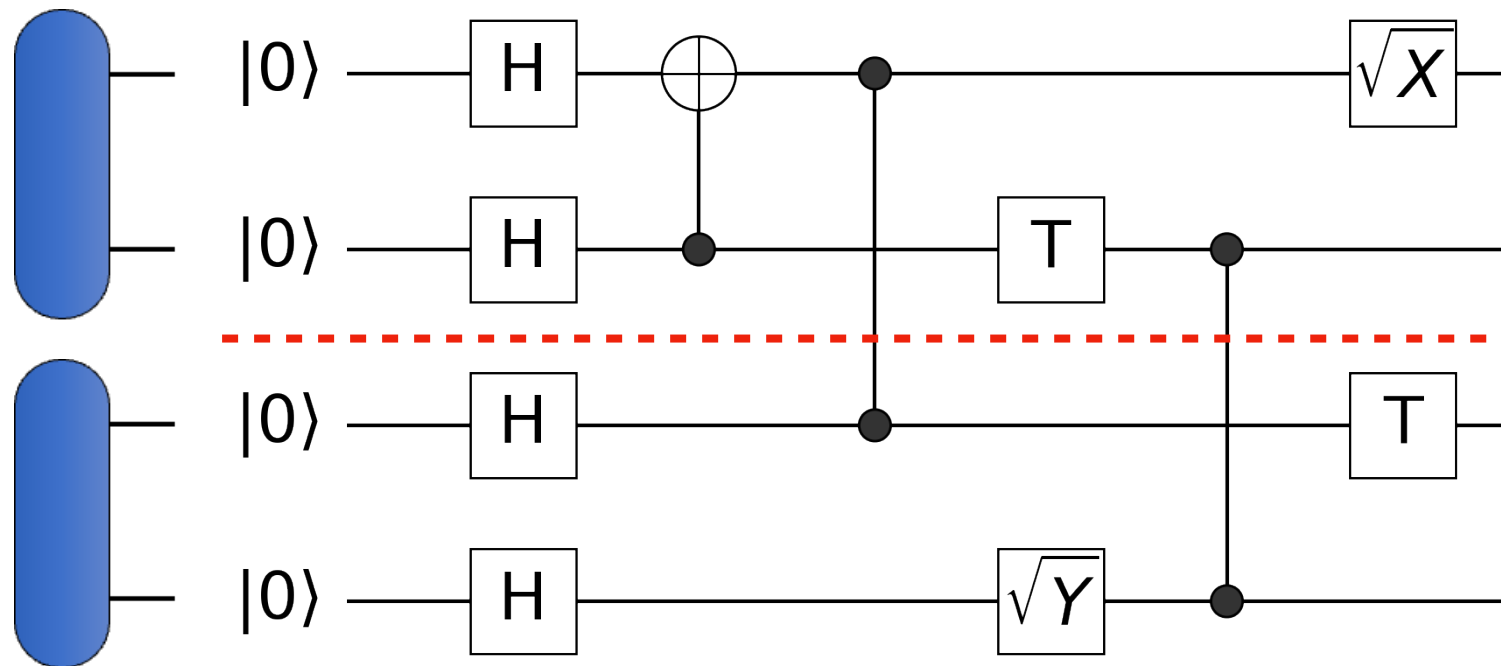


#qubits	Space dimension	Space complexity	Storage Device
10	$2^{10} = 1024$	16 K bytes	
20	$2^{20} = 60536$	16 M bytes	
30	$2^{30} = 1073741824$	16 G bytes	Laptop
40	$2^{40} = 1099511627776$	16 T bytes	Cluster
50	$2^{50} = 1125899906842624$	16 P bytes	Supercomputer
53	$2^{53} = 9007199254740992$	128 P bytes	All hard disks of supercomputer

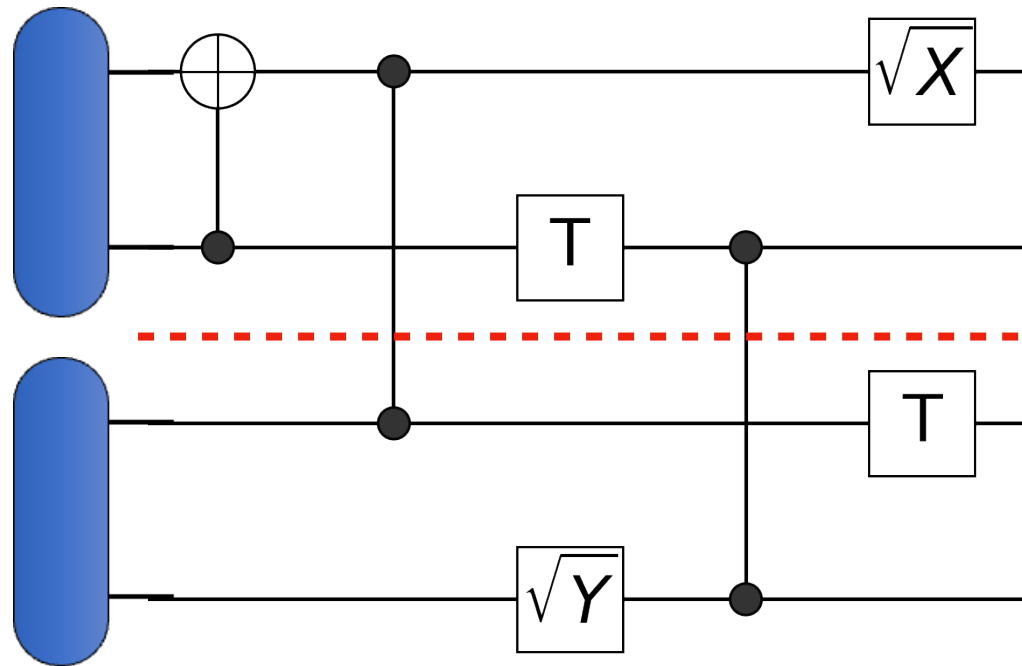
# The Schrödinger-Feynmann algorithm



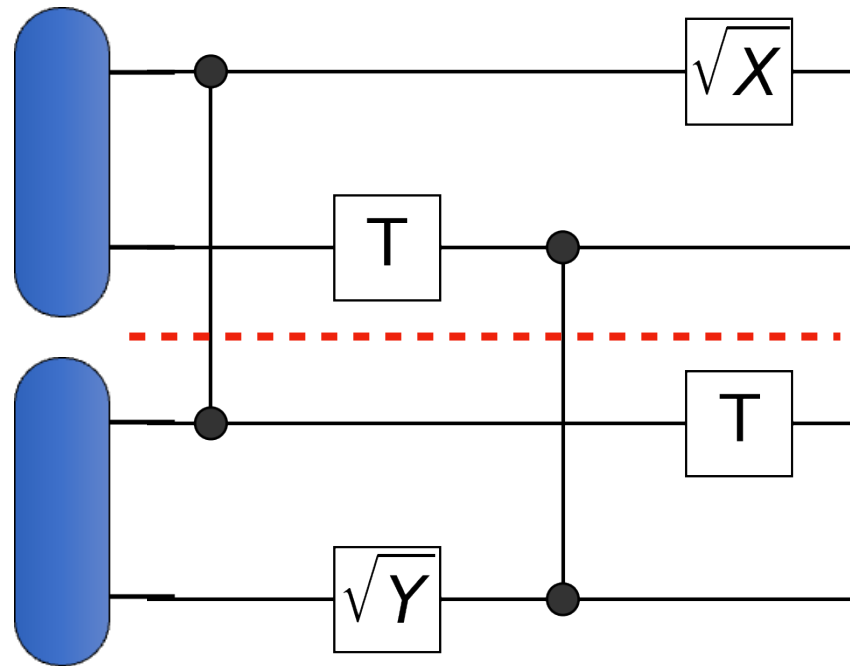
# The Schrödinger-Feynmann algorithm



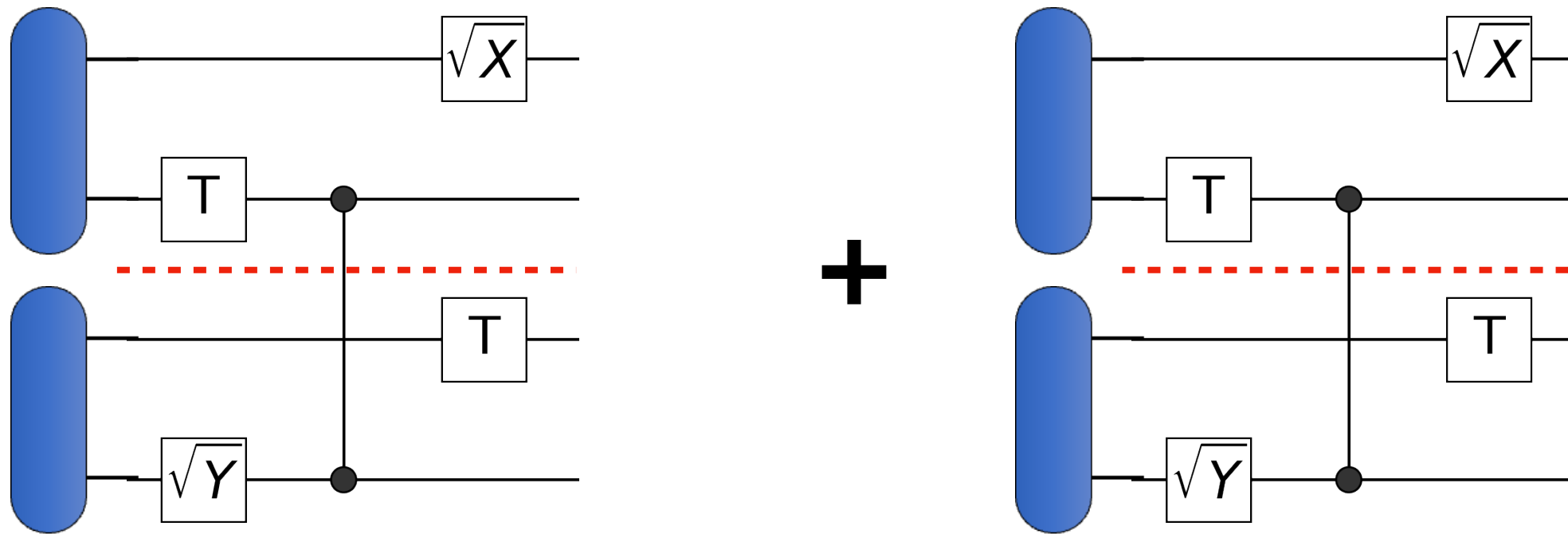
# The Schrödinger-Feynmann algorithm



# The Schrödinger-Feynmann algorithm

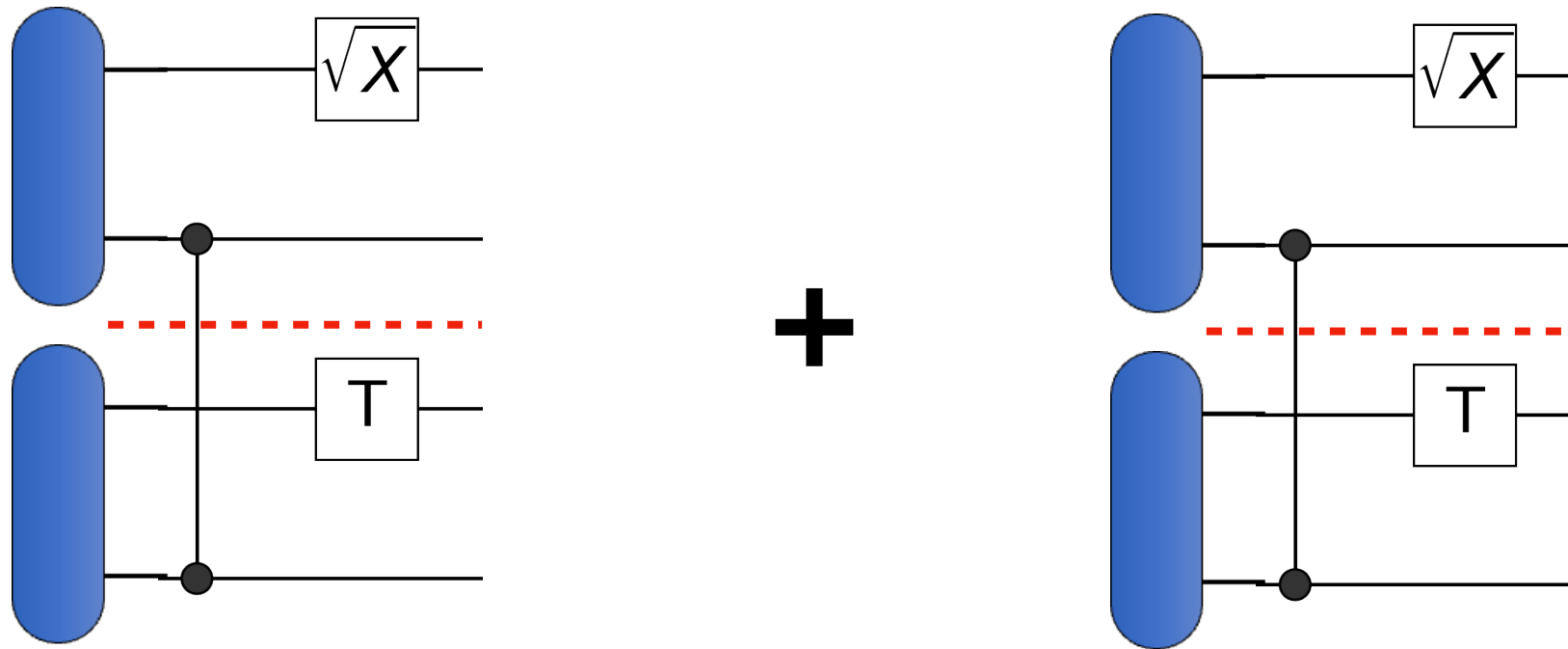


# The Schrödinger-Feynmann algorithm

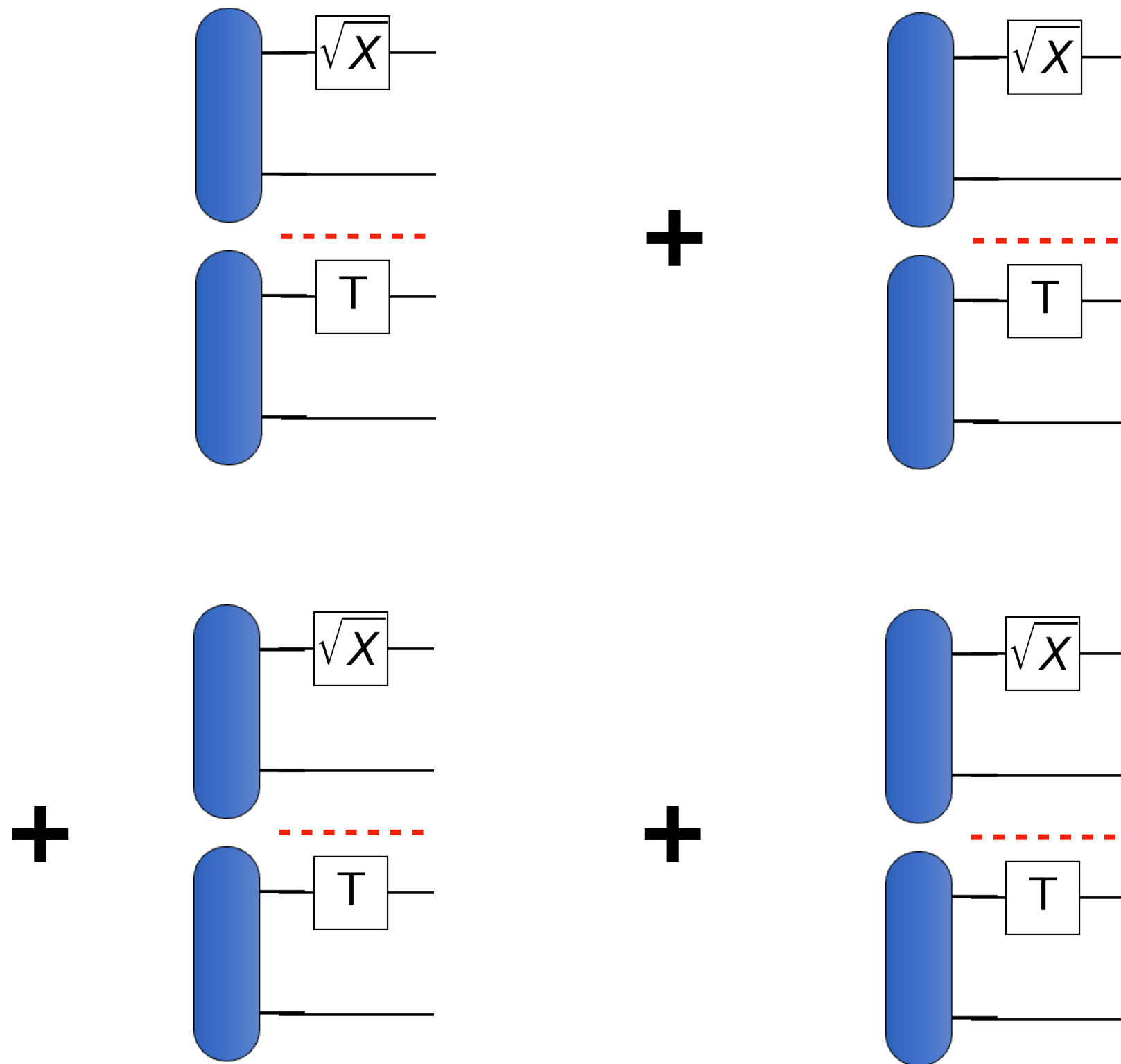




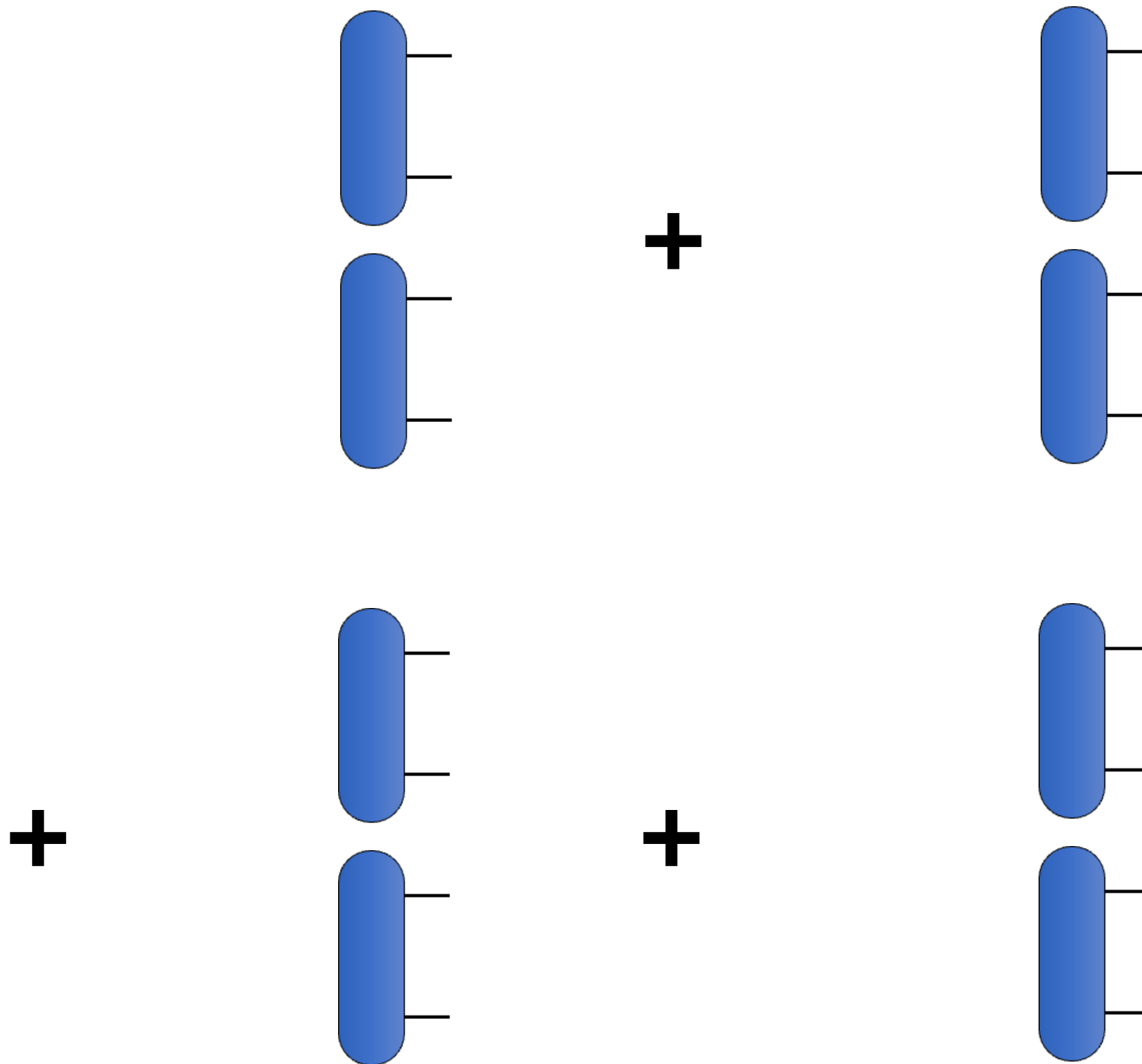
# The Schrödinger-Feynmann algorithm



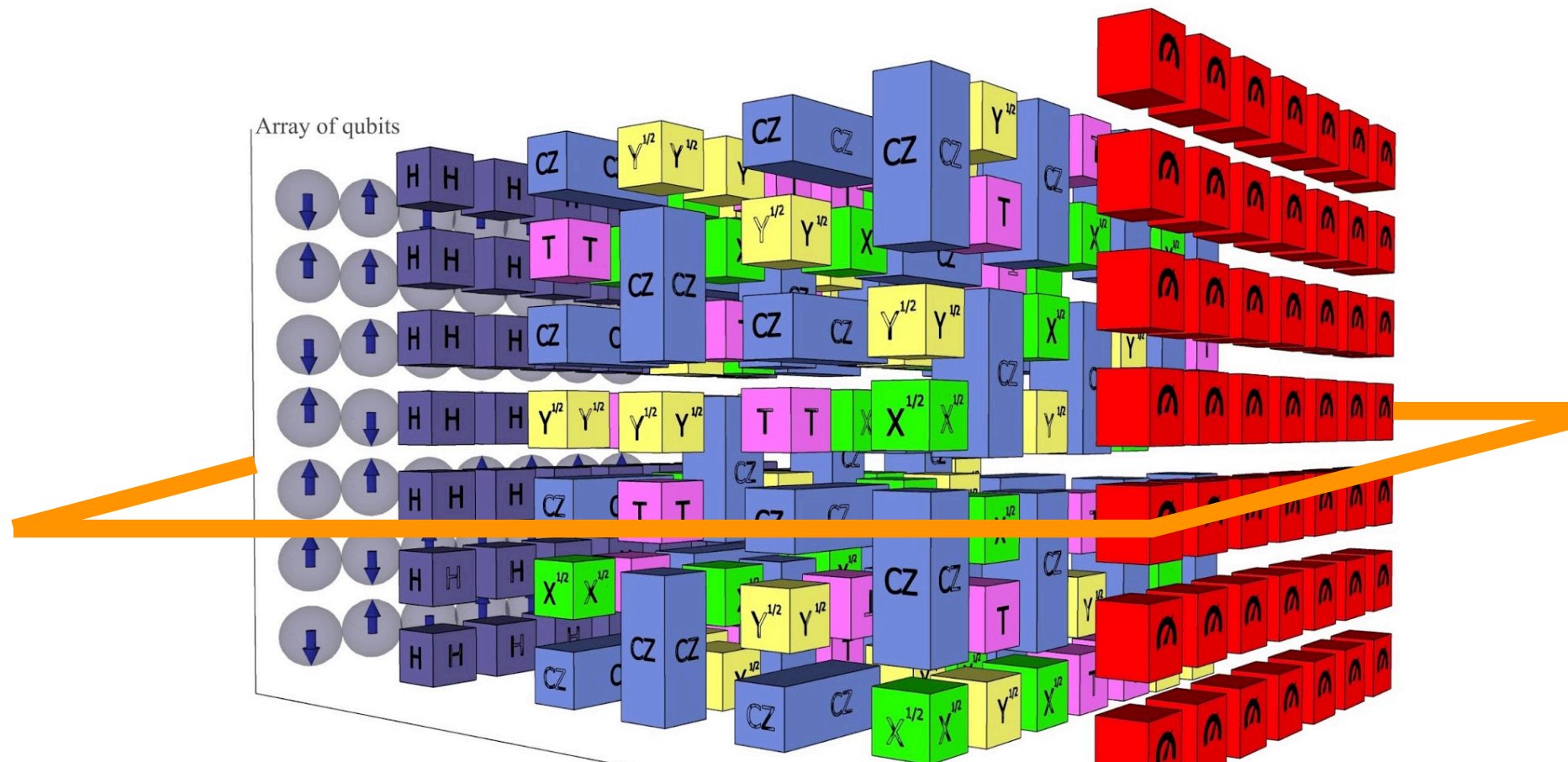
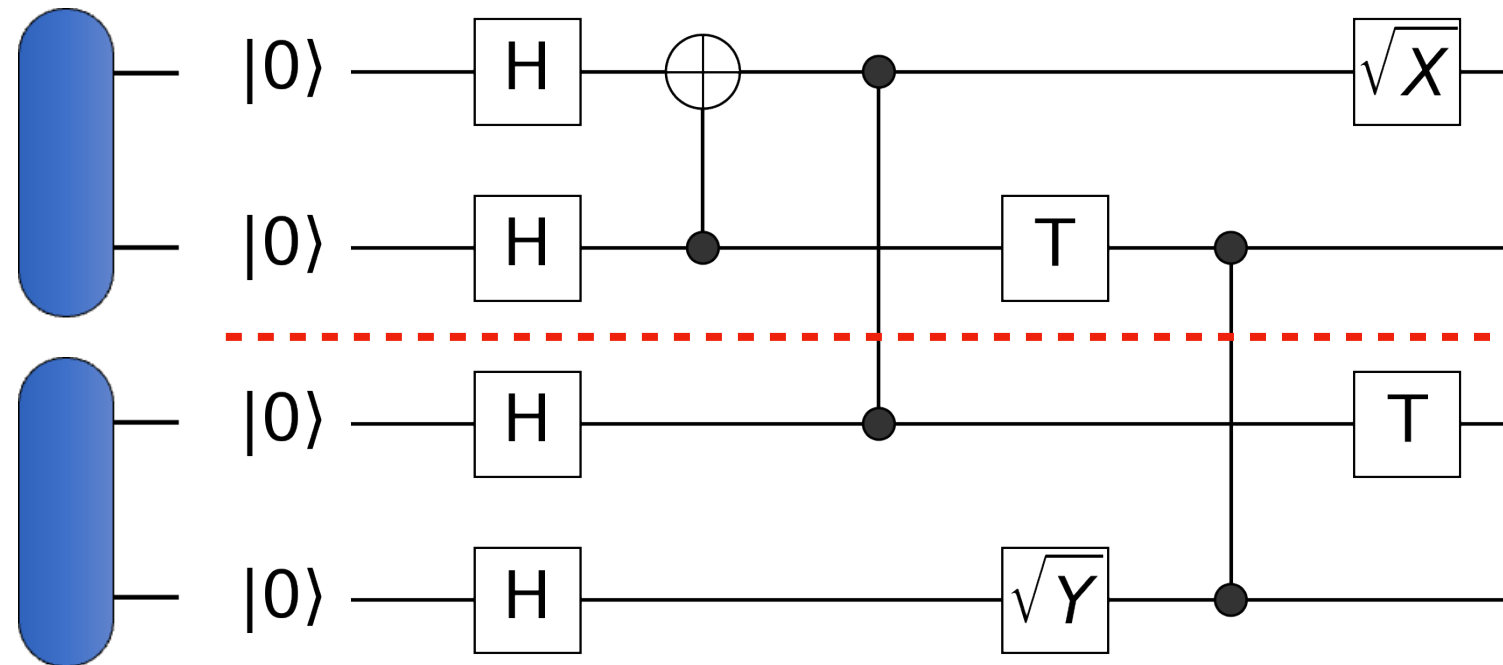
# The Schrödinger-Feynmann algorithm



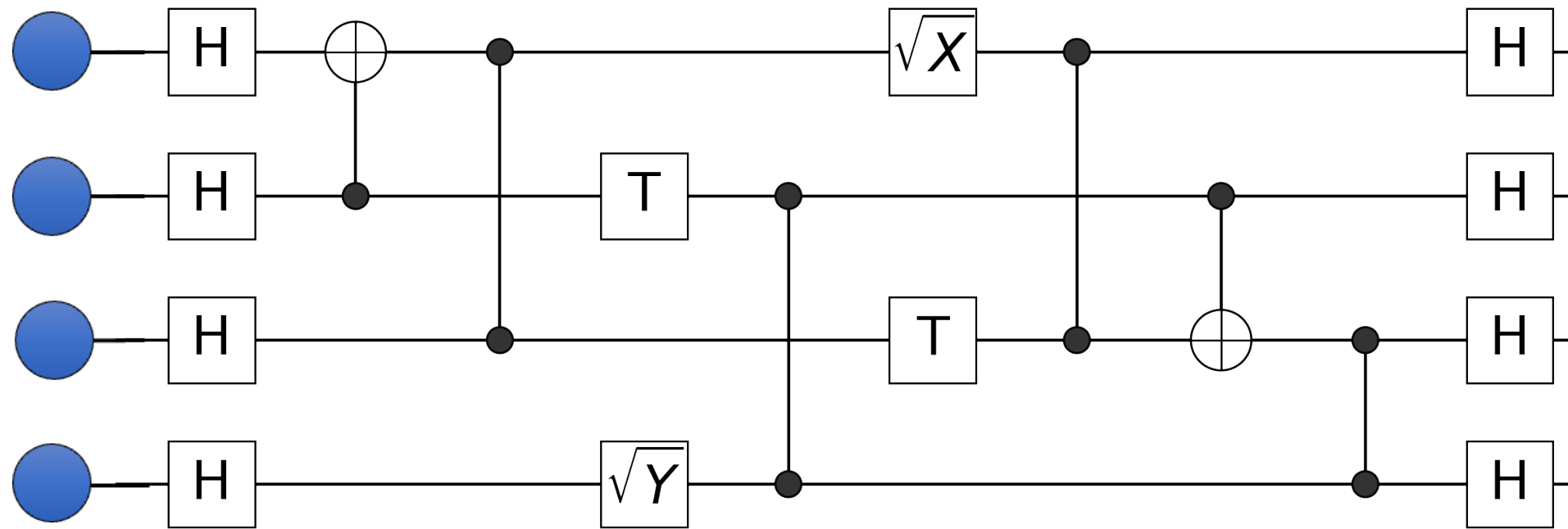
# The Schrödinger-Feynmann algorithm



Problem of the Schrödinger-Feynmann algorithm:  
Complexity grows exponential with number of cuts



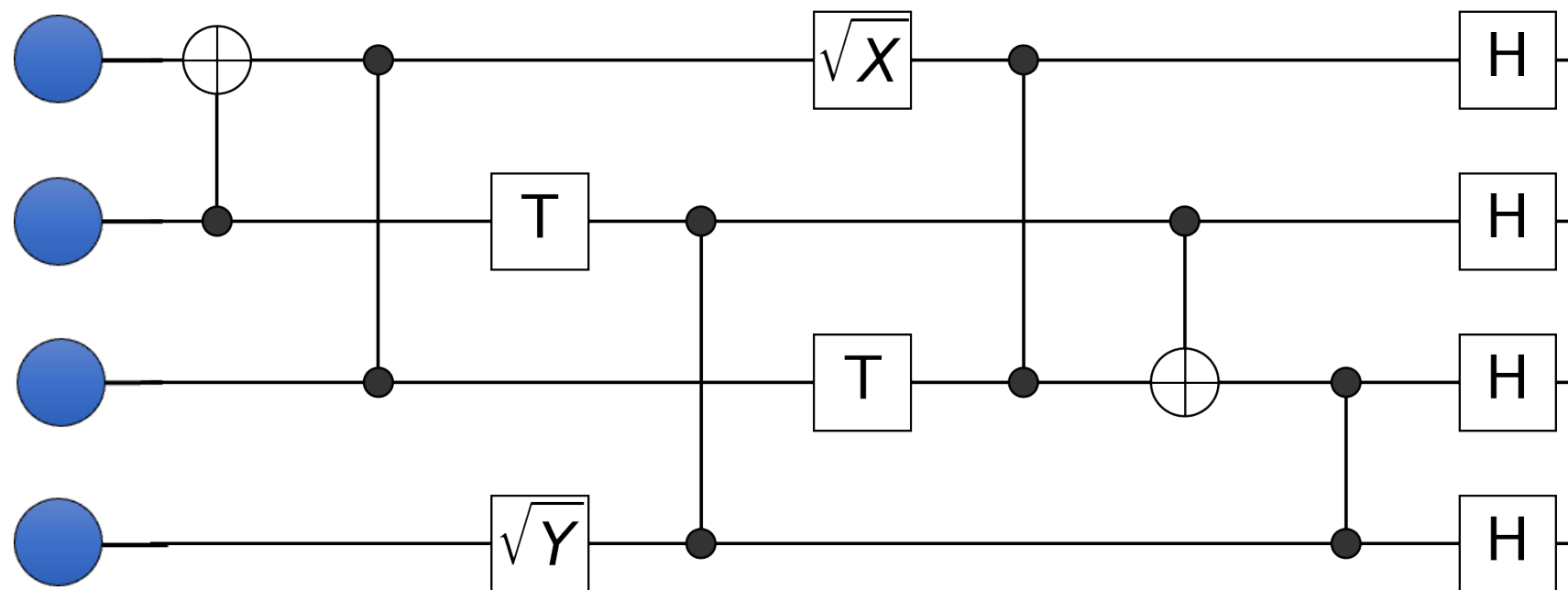
# MPS algorithm



$$|\psi\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$$

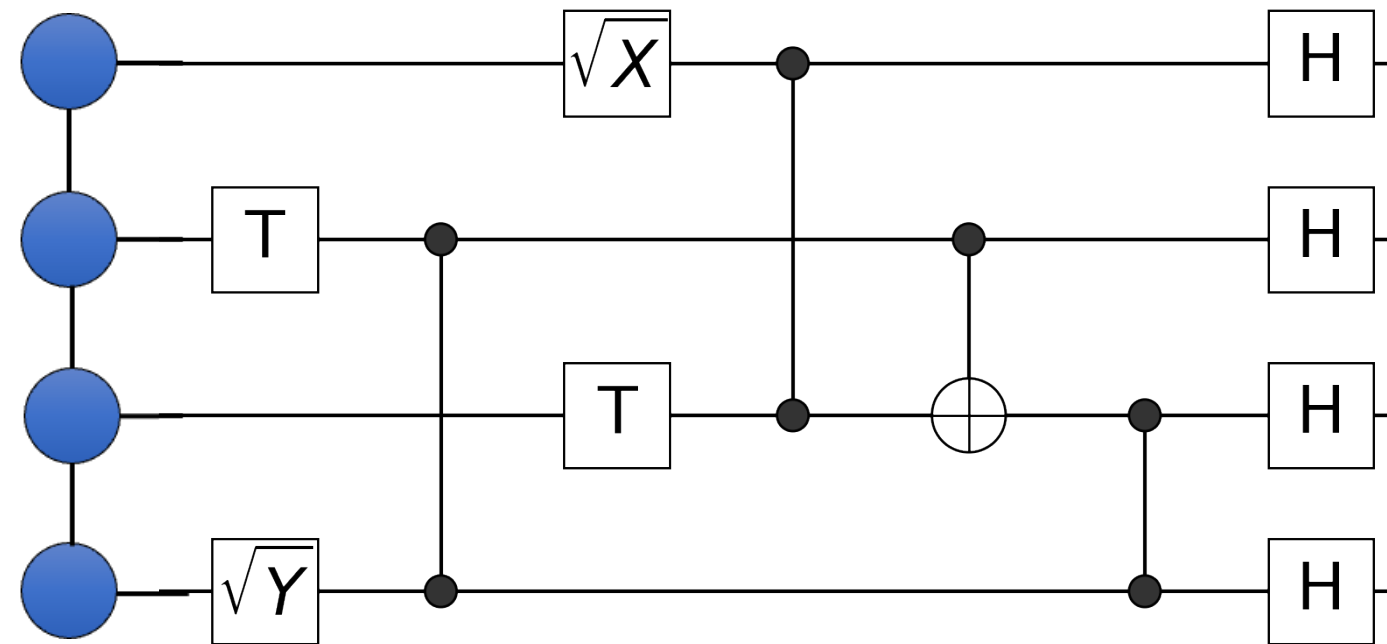
## Rank one

# MPS algorithm



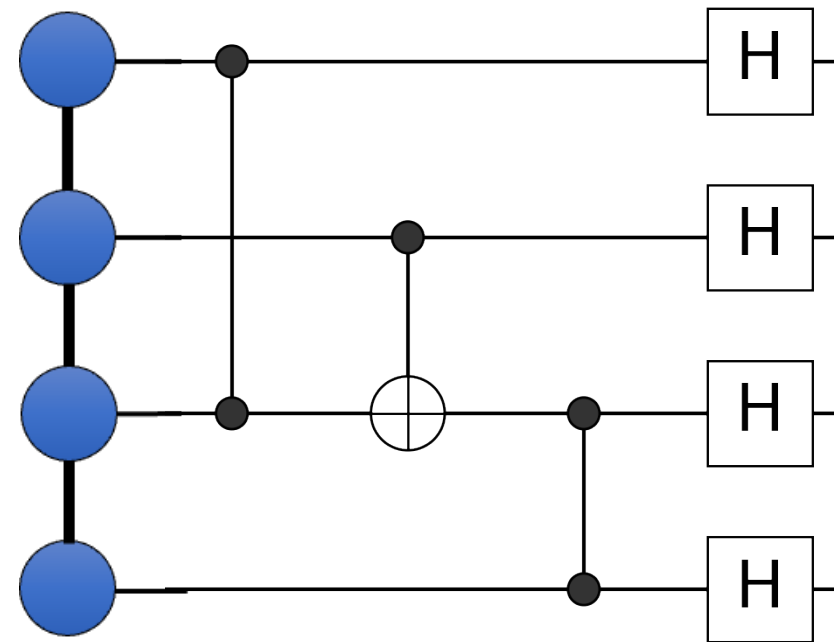
$$|\psi\rangle = |+\rangle \otimes |+\rangle \otimes |+\rangle \otimes |+\rangle \quad \text{Rank one}$$

# MPS algorithm


$$|\psi\rangle$$

## Rank > 1

# MPS algorithm

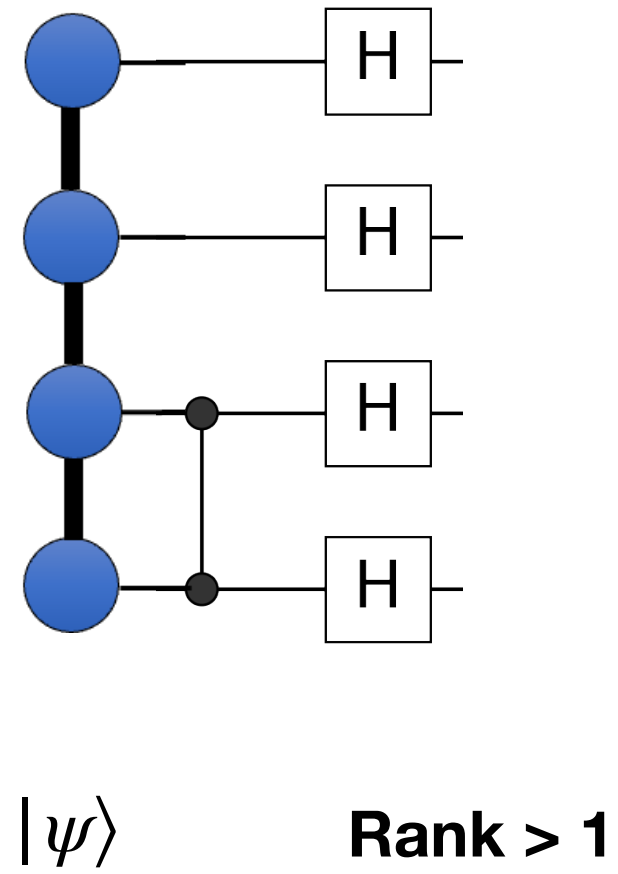


$|\psi\rangle$

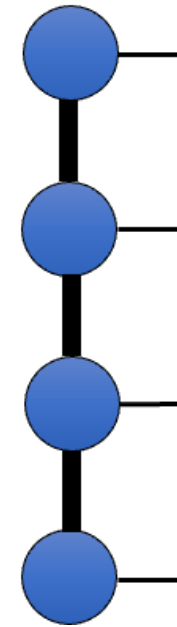
**Rank > 1**



# MPS algorithm



# MPS algorithm

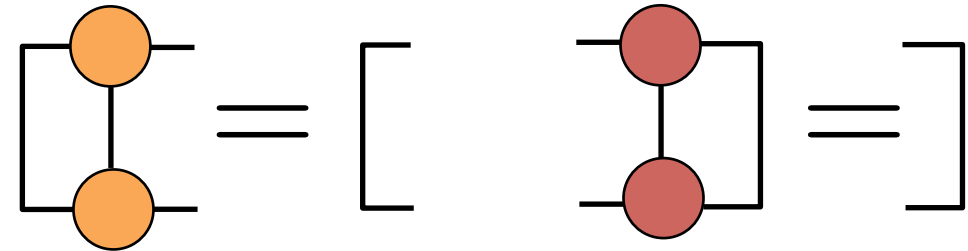
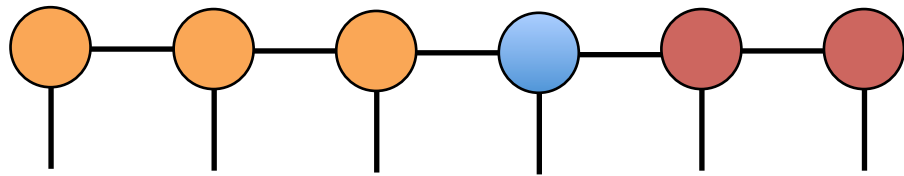


$|\psi\rangle$

**Rank = 4**

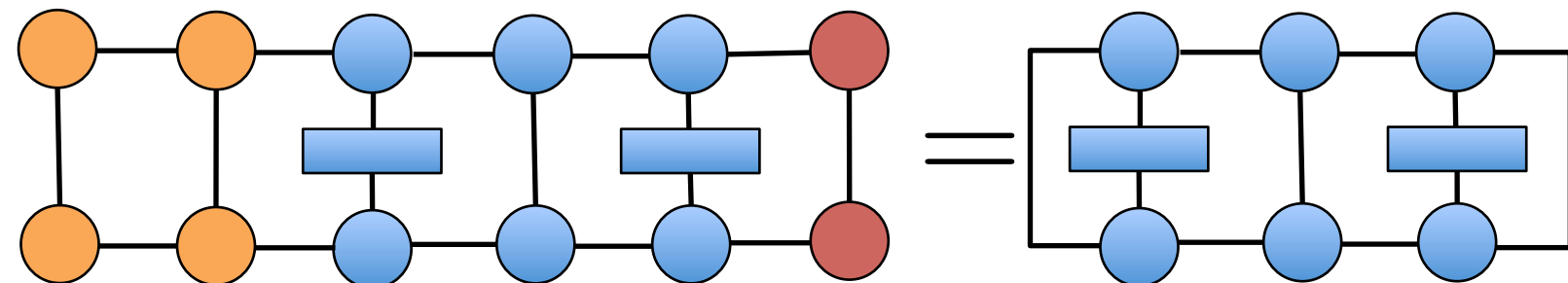
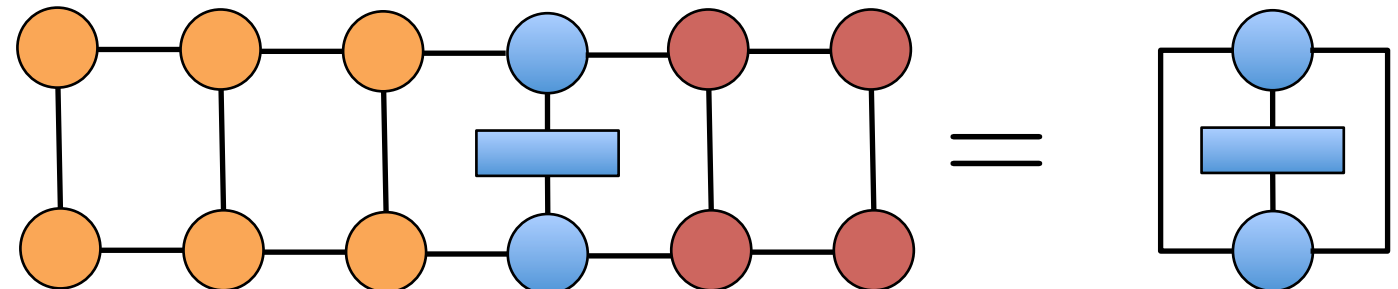
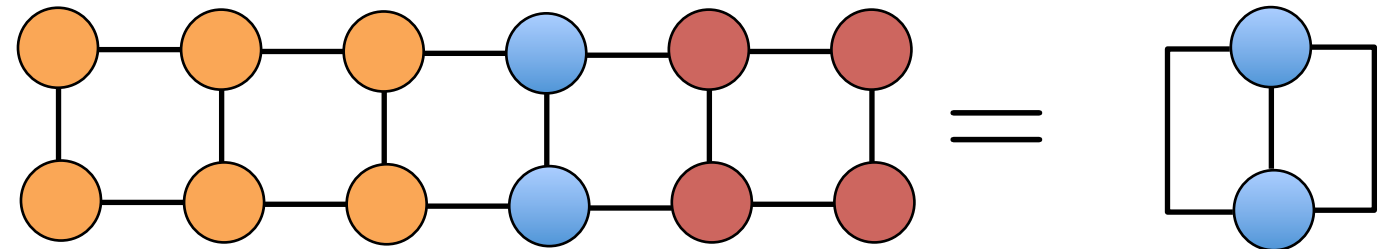
# Canonical forms of MPS

Analogous to the Tucker decomposition and HOSVDs, MPS has the benefits of orthogonality.

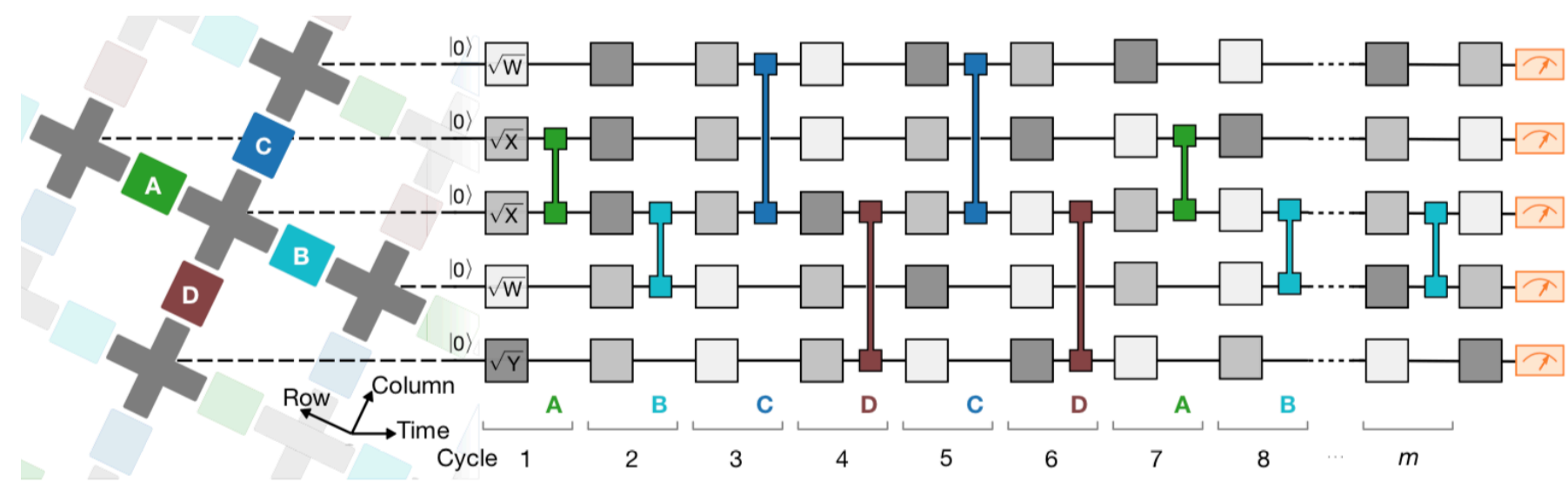


## Benefits

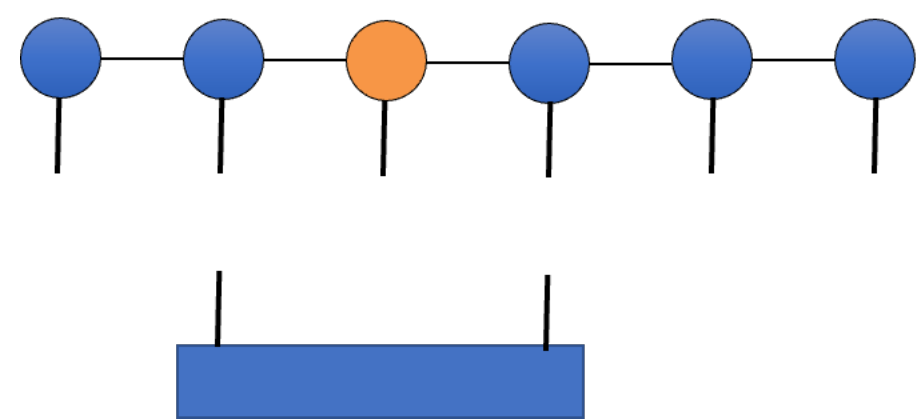
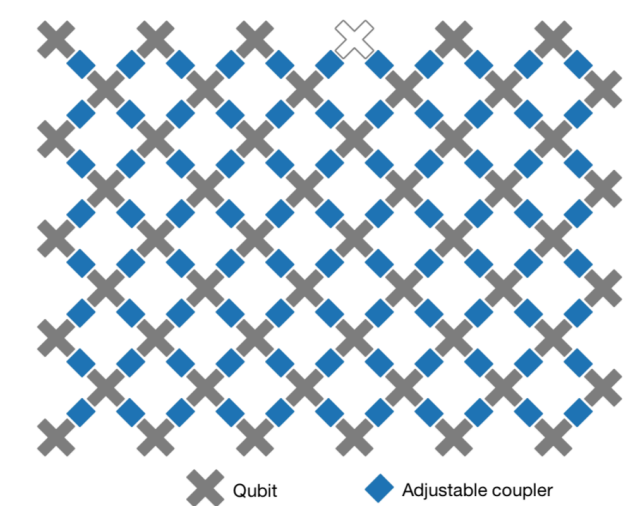
- Fixed gauge, no ambiguity
- Easy norm computation
- Easy expectation/  
correlation computation
- Always good conditioned



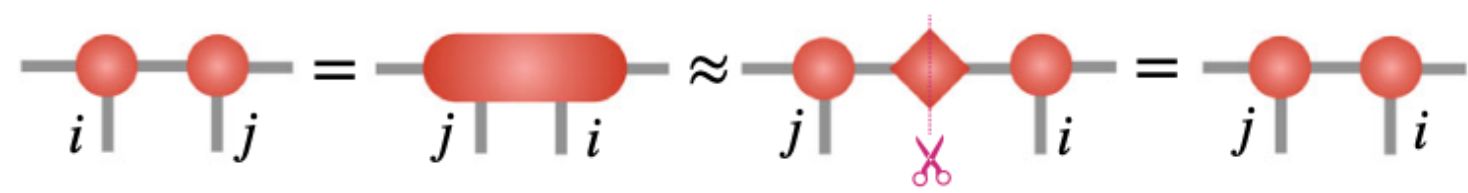
# Problem of MPS simulation



2D Layout



Non-local operators



SWAP

# Results of MPS simulations of 2D circuits with CZ gates

PHYSICAL REVIEW X **10**, 041038 (2020)

Featured in Physics

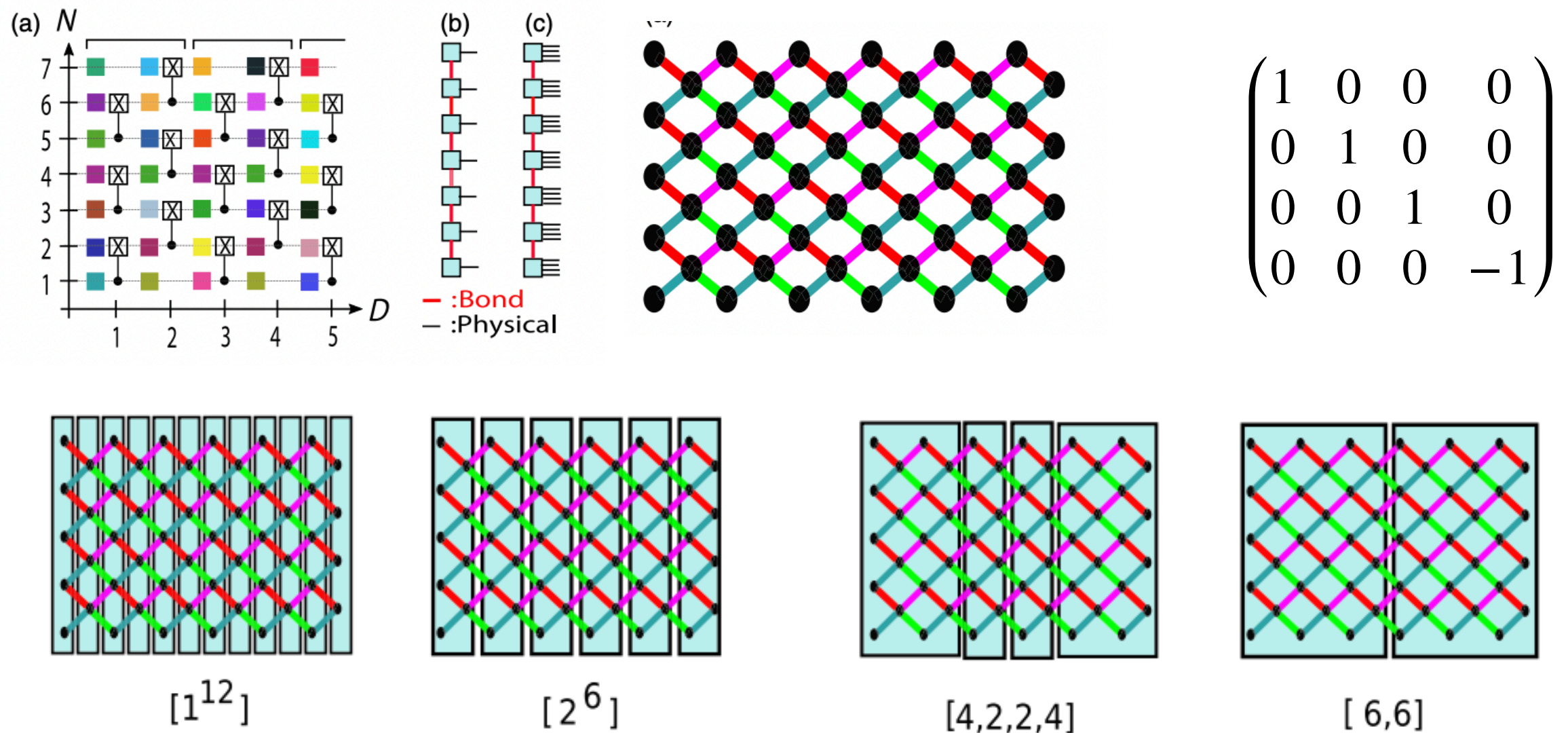
## What Limits the Simulation of Quantum Computers?

Yiqing Zhou<sup>1,2</sup>, E. Miles Stoudenmire<sup>2</sup> and Xavier Waintal<sup>3</sup>

<sup>1</sup>Department of Physics, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA

<sup>2</sup>Center for Computational Quantum Physics, Flatiron Institute, New York, New York 10010, USA

<sup>3</sup>Univ. Grenoble Alpes, CEA, IRIG-Pheliqs, 38054 Grenoble, France



# Results of MPS simulations of 2D circuits with CZ gates

PHYSICAL REVIEW X **10**, 041038 (2020)

Featured in Physics

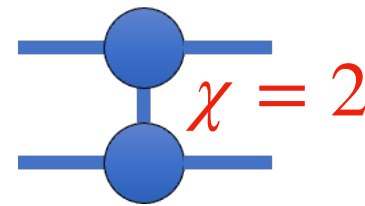
## What Limits the Simulation of Quantum Computers?

Yiqing Zhou<sup>1,2</sup>, E. Miles Stoudenmire<sup>2</sup>, and Xavier Waintal<sup>3</sup>

<sup>1</sup>Department of Physics, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA

<sup>2</sup>Center for Computational Quantum Physics, Flatiron Institute, New York, New York 10010, USA

<sup>3</sup>Univ. Grenoble Alpes, CEA, IRIG-Pheliqs, 38054 Grenoble, France



$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$
$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$
$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix}$$

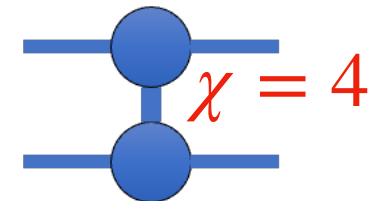
**Singular values:**  
 $[\sqrt{2}, \sqrt{2}, 0, 0]$

```
import numpy as np
a = np.array([[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,-1]])
a = a.reshape([2,2,2,2])
b = a.transpose([0,2,1,3])
b = b.reshape(4,4)
u,d,v = np.linalg.svd(b)
```

# Results of MPS simulations of 2D circuits with fSim gates

**fSim gate has a flat prior**

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



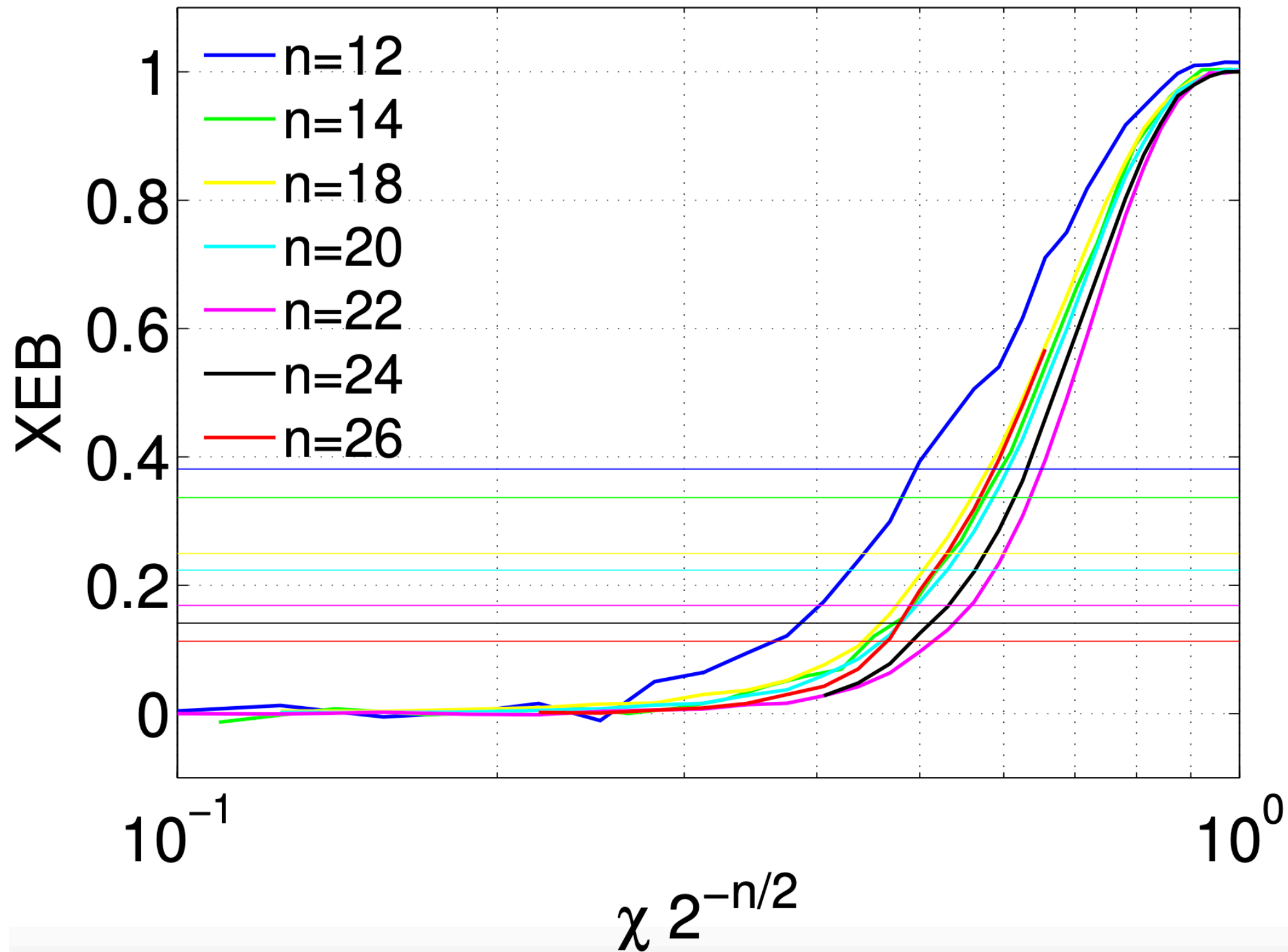
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -i \sin \theta & 0 \\ 0 & -i \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & \cos \theta \\ 0 & 0 & -i \sin \theta & 0 \\ 0 & -i \sin \theta & 0 & 0 \\ \cos \theta & 0 & 0 & e^{-i\phi} \end{pmatrix}$$

**|Singular values|<sup>2</sup>**  
 $[1, 1, \sin^2(\theta), \sin^2(\theta)]$

```
from sympy import *
theta = symbols('theta')
phi = symbols('phi')
a=Matrix([[1,0,0,cos(theta)],[0,0,I*sin(theta),0],[0,I*sin(theta),0,0],[0,0,0,exp(-I*phi)]])
a.eigenvals()
```

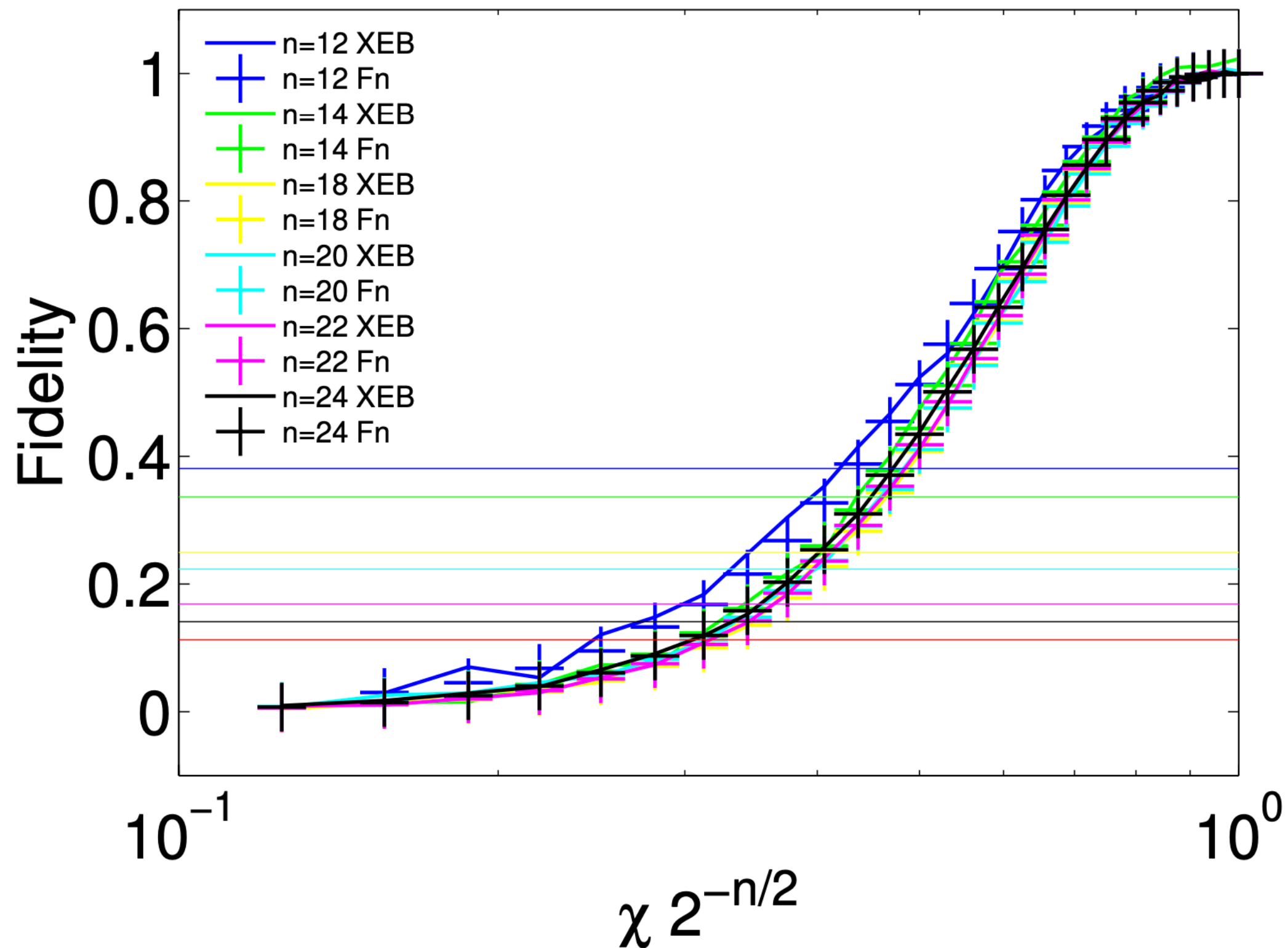
# Results of MPS simulations of Sycamore circuits



Our results

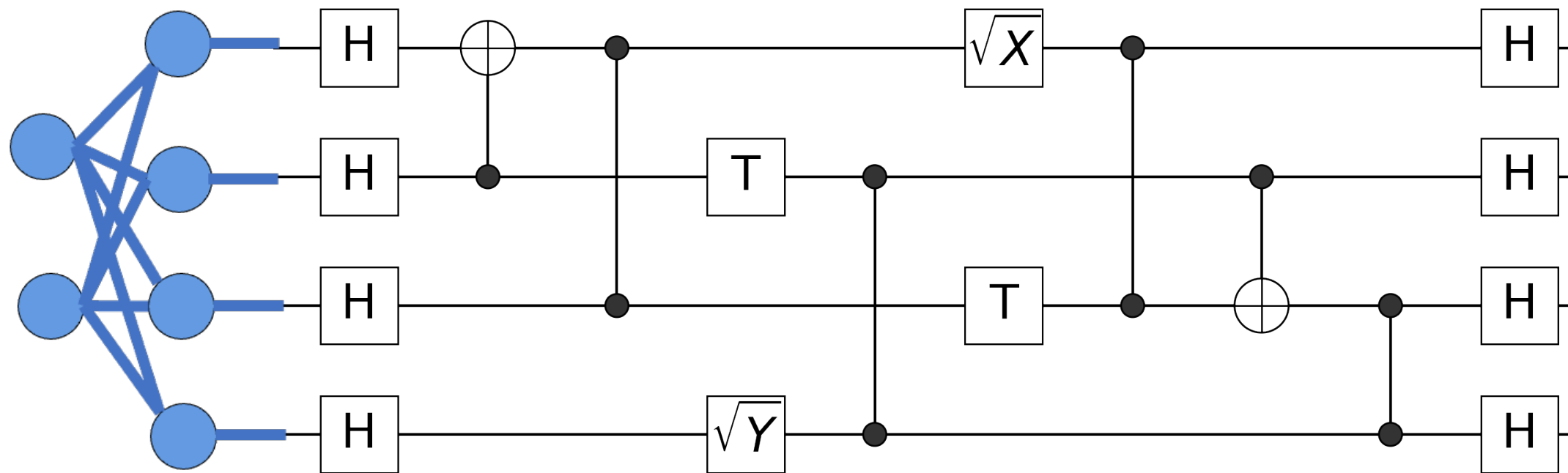


# Results of Group MPS simulations of Sycamore circuits



**Our results**

# Neural network simulation of quantum circuits



Tensor network (e.g. MPS) states:

low-rank, weak entanglements

efficient and accurate to compute inner product

Neural network states

high-rank

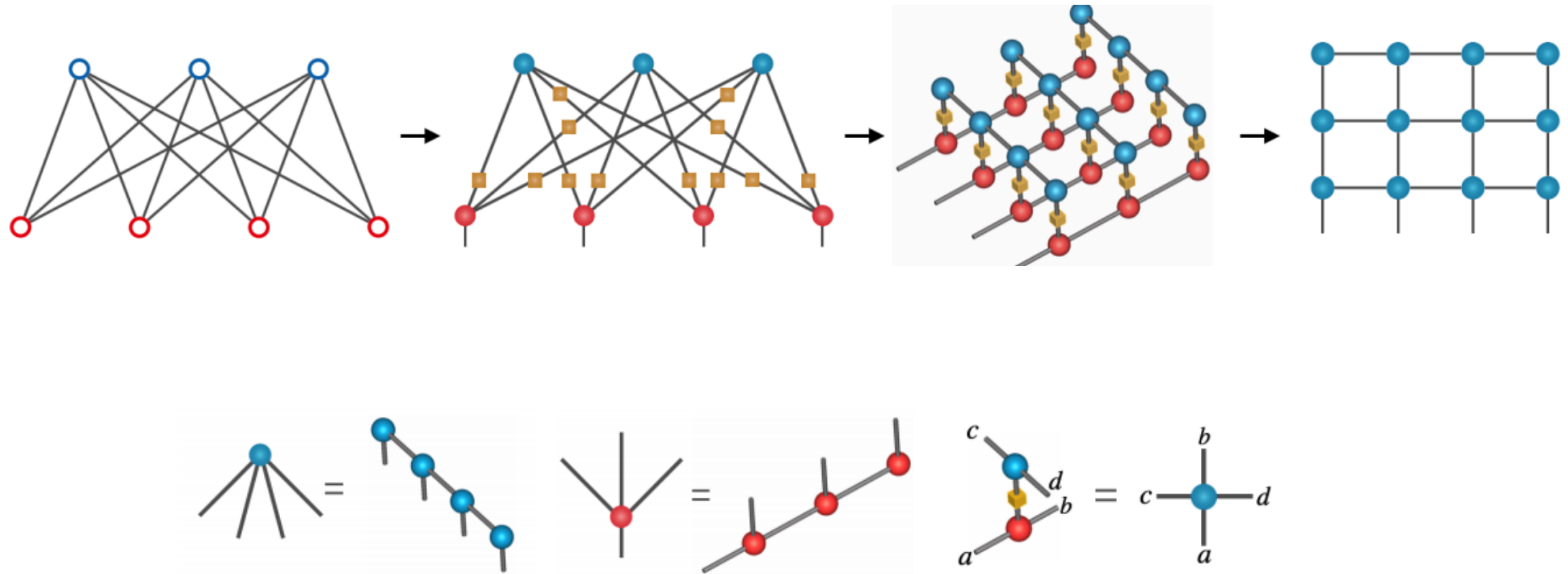
difficult to compute inner products, need sampling

variational optimization with a loss function  $D(|\psi\rangle, A|\psi\rangle)$

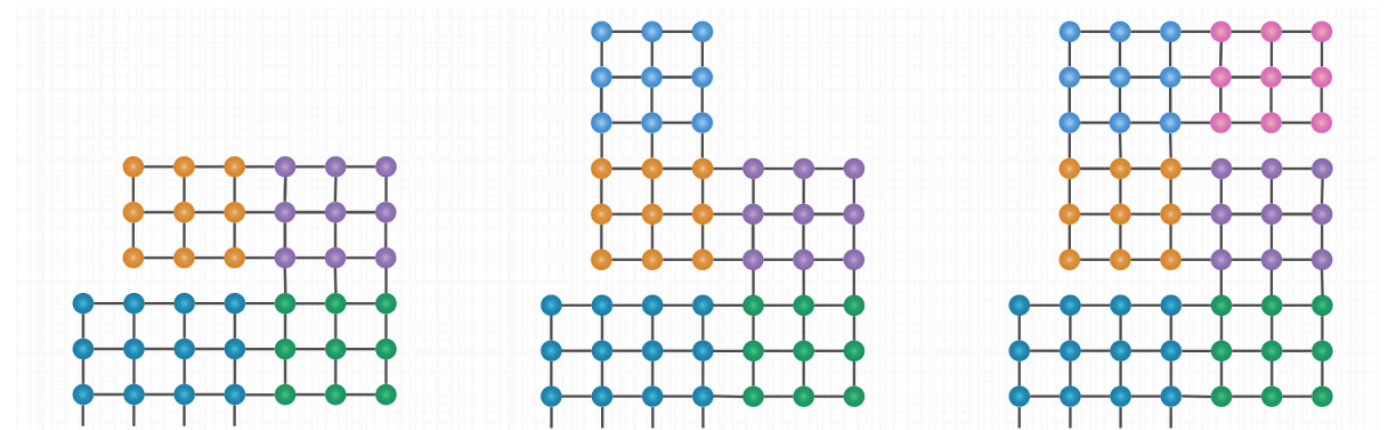
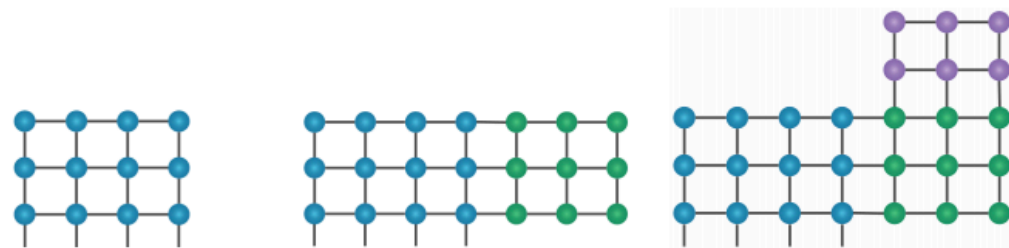
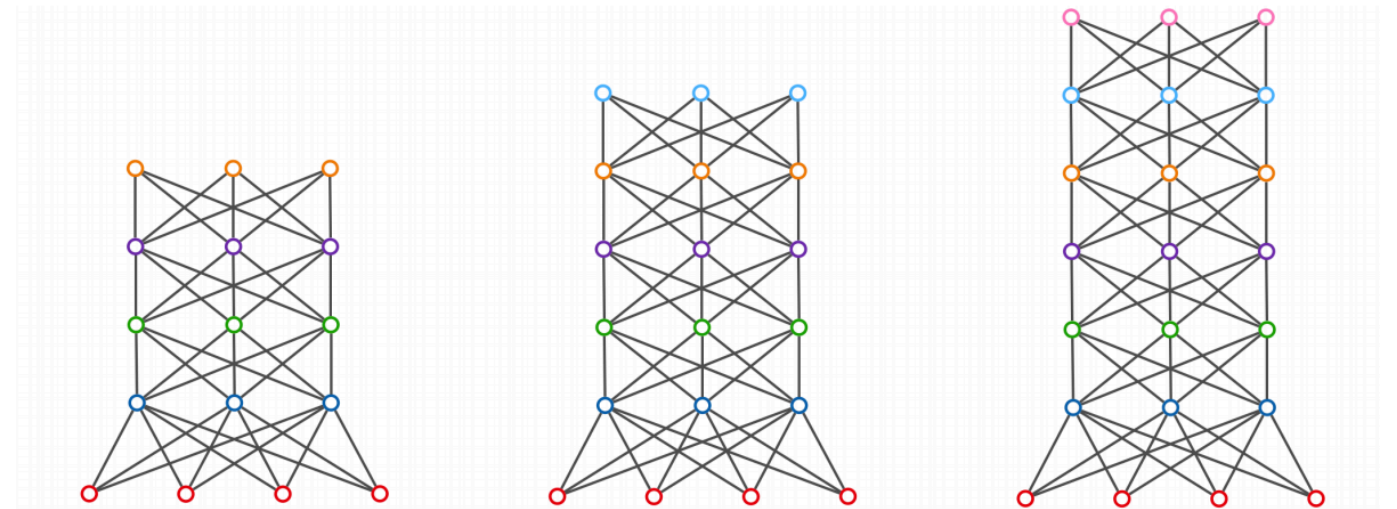
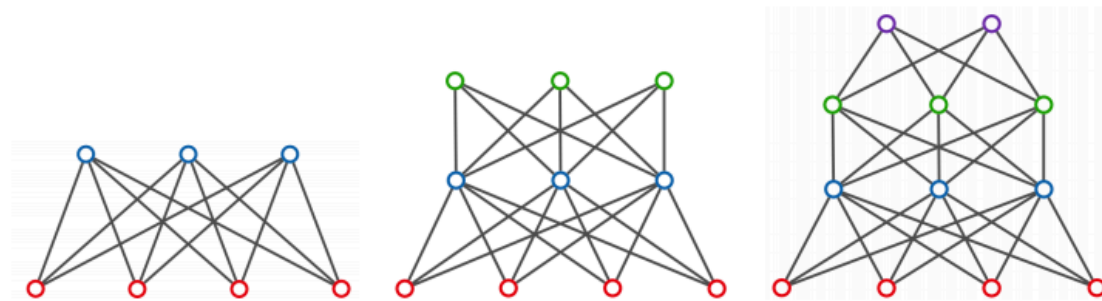
M. Medvidovic and G. Carleo, npj Quantum information **7**, 1 (2021)

B. Jonsson, B. Bauer, G. Carleo, arXiv:1808.05232 (2018)

# Neural network simulation of quantum circuits

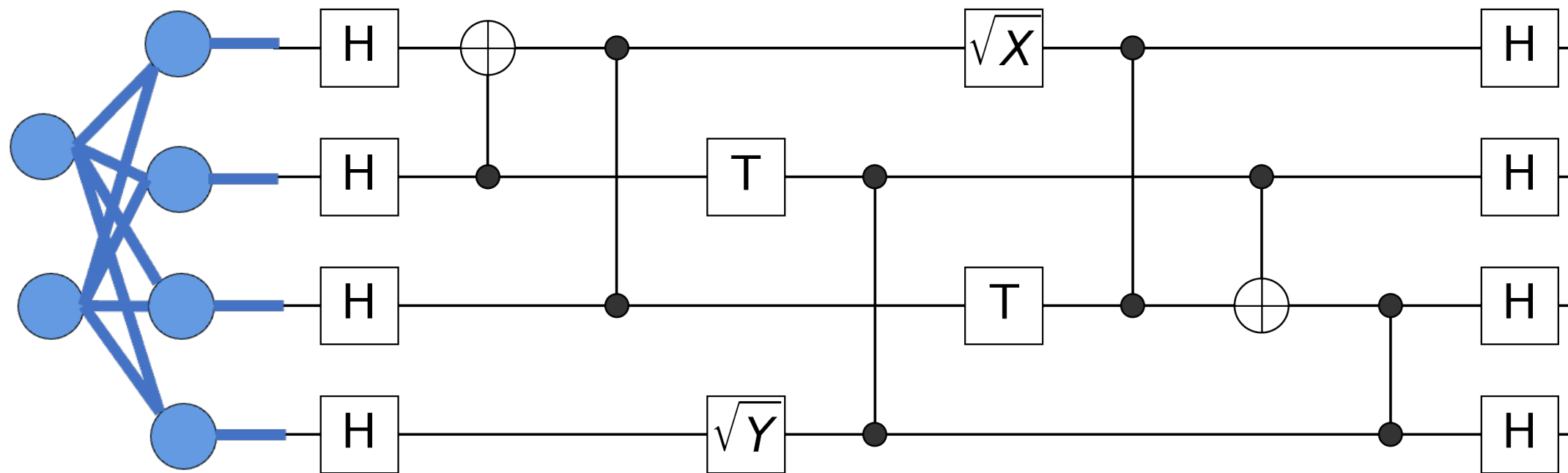


# Deep Boltzmann Machines



## 2D Tensor Network

# Neural network simulation of quantum circuits



Tensor network (e.g. MPS) states:

low-rank, weak entanglements

efficient and accurate to compute inner product

Neural network states

high-rank

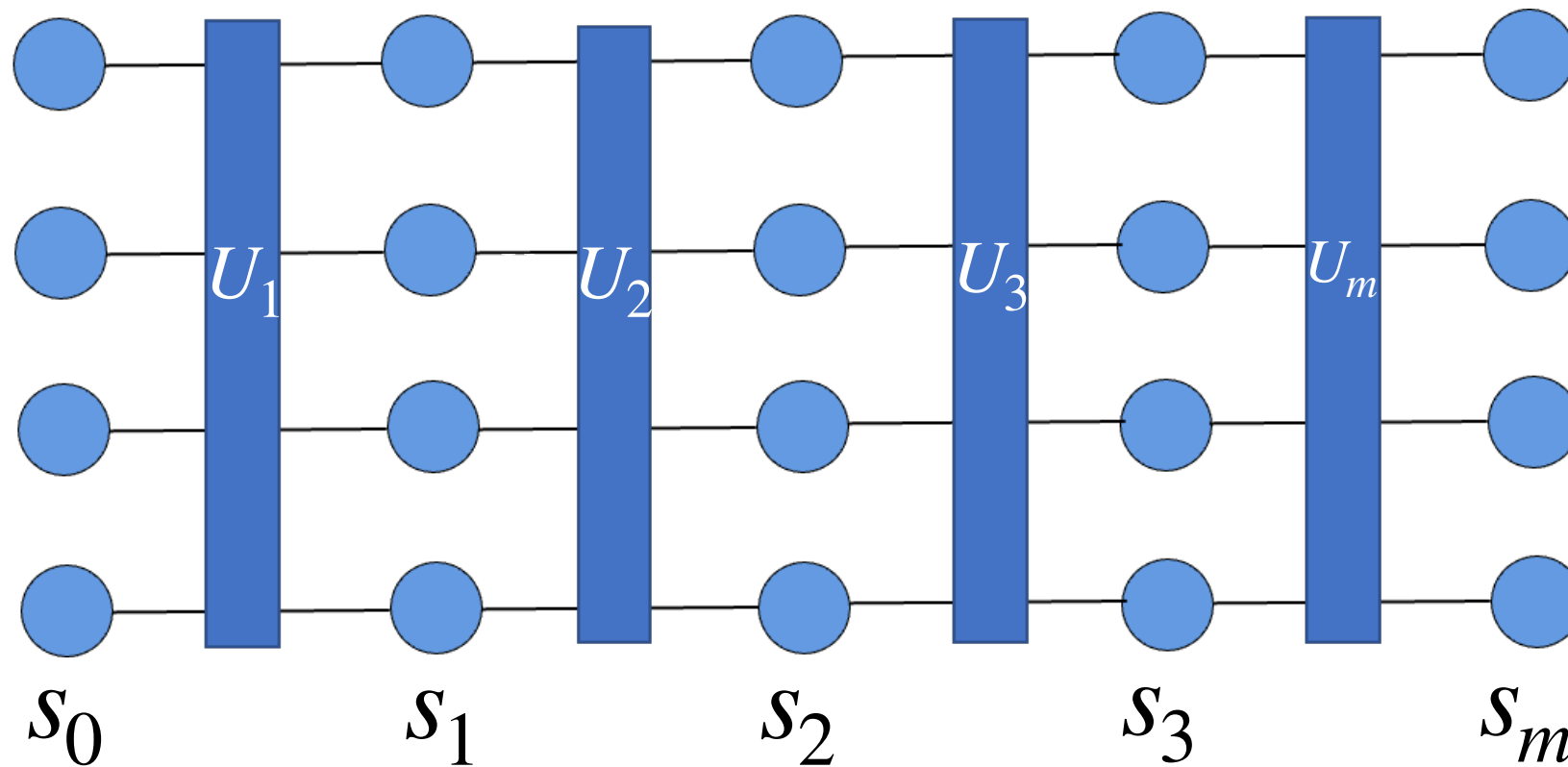
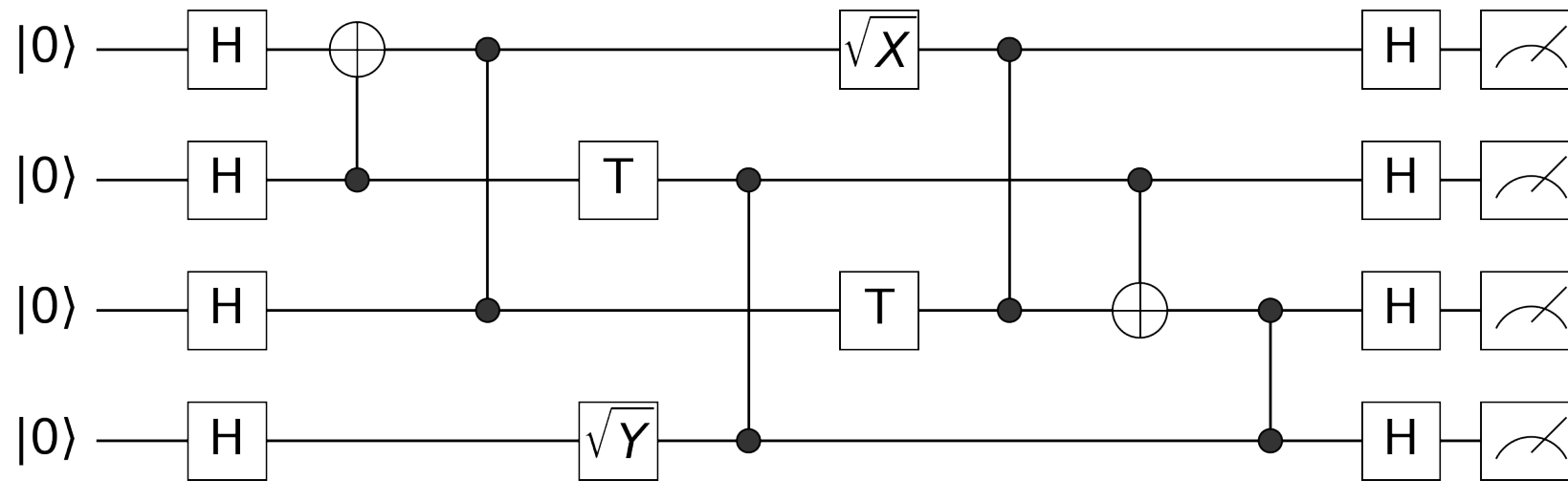
difficult to compute inner products, need sampling

variational optimization with a loss function  $D(|\psi\rangle, A|\psi\rangle)$

M. Medvidovic and G. Carleo, npj Quantum information **7**, 1 (2021)

B. Jonsson, B. Bauer, G. Carleo, arXiv:1808.05232 (2018)

# Tensor network contraction and Path Integral

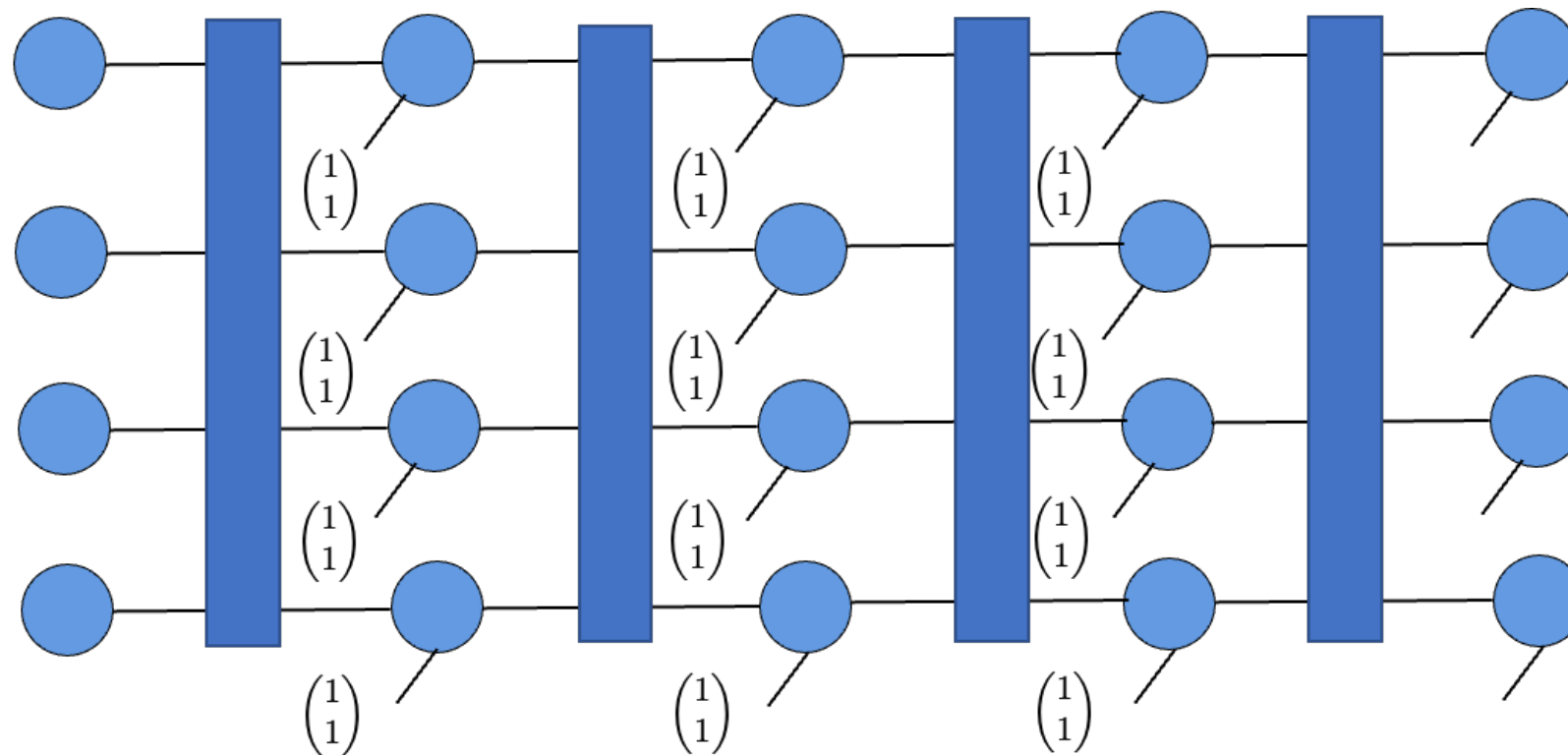
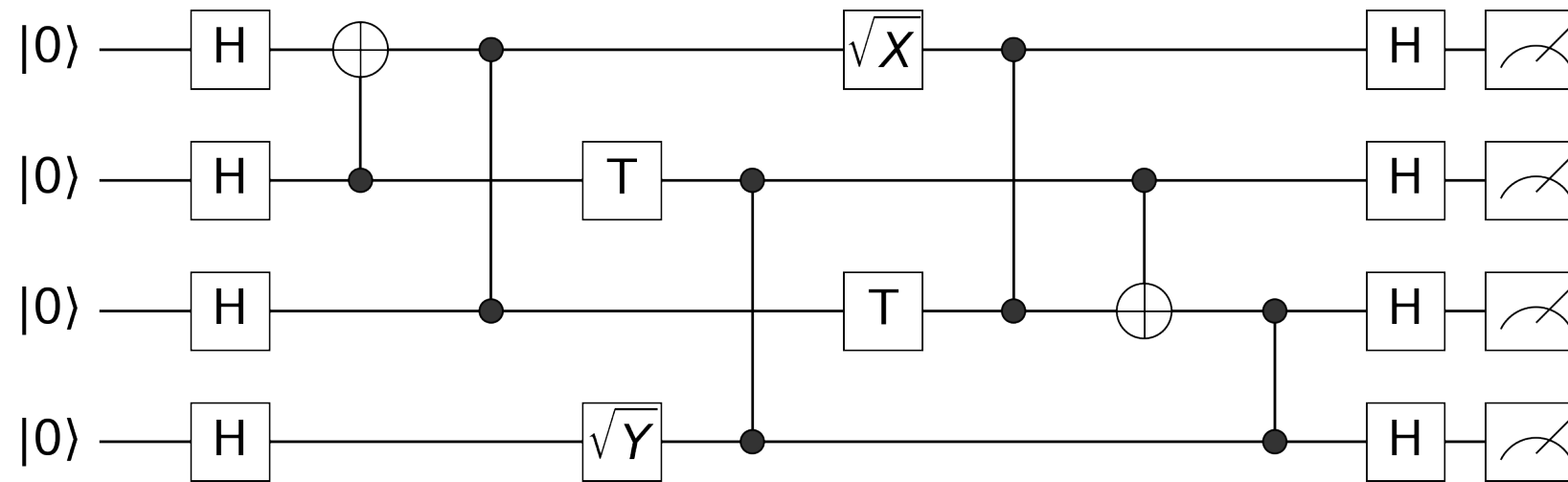


$$s_t = \prod_{i=1}^n s_i^t$$

$$\langle \psi | s_m \rangle = \sum_{s_1} \sum_{s_2} \cdots \sum_{s_{m-1}} \prod_{t=1}^m \langle s_{t-1} | U_t | s_t \rangle$$

**Totally  $2^{n(m-2)}$  paths**

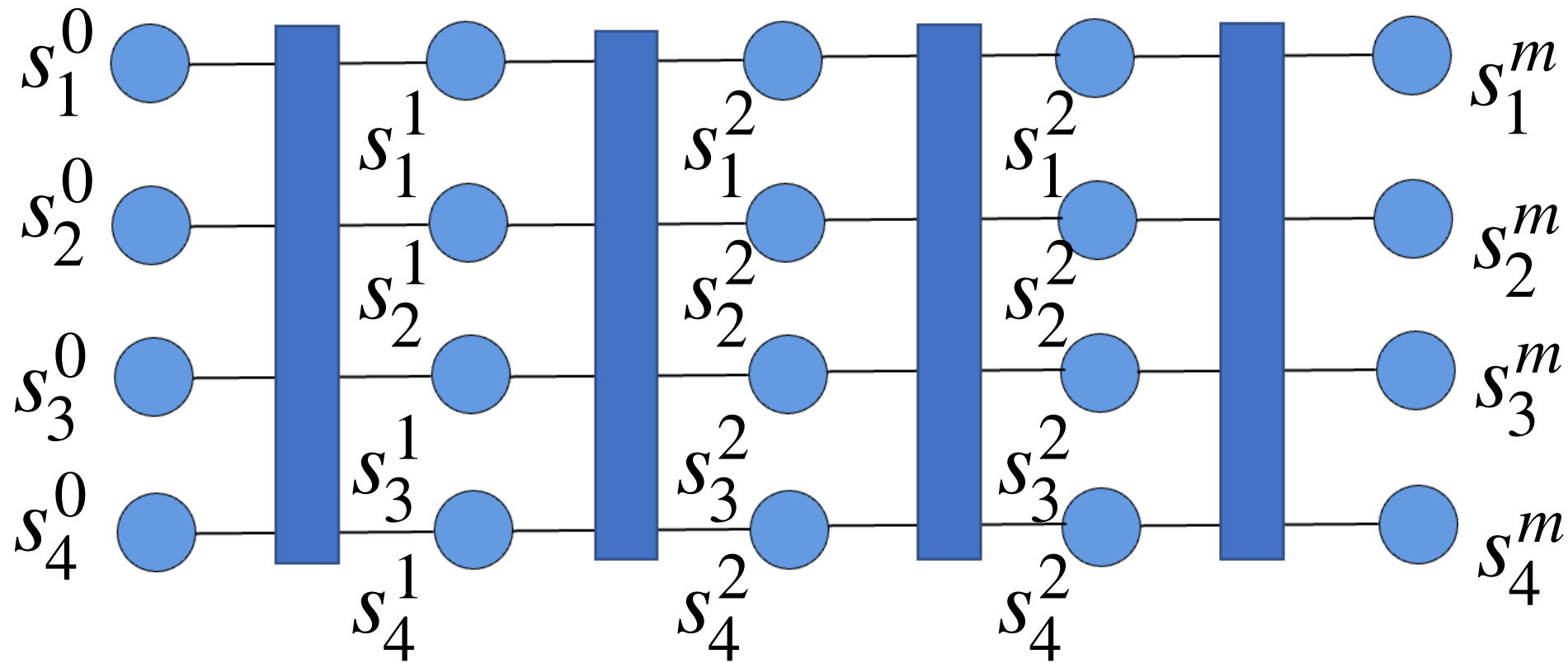
# Tensor network contraction and **complex Ising model**



**Final state = Marginals of complex Ising model**

**Single amplitude  $\langle \psi | s_m \rangle =$  **Partition function** of complex Ising model**

# Final state and statistical mechanics model



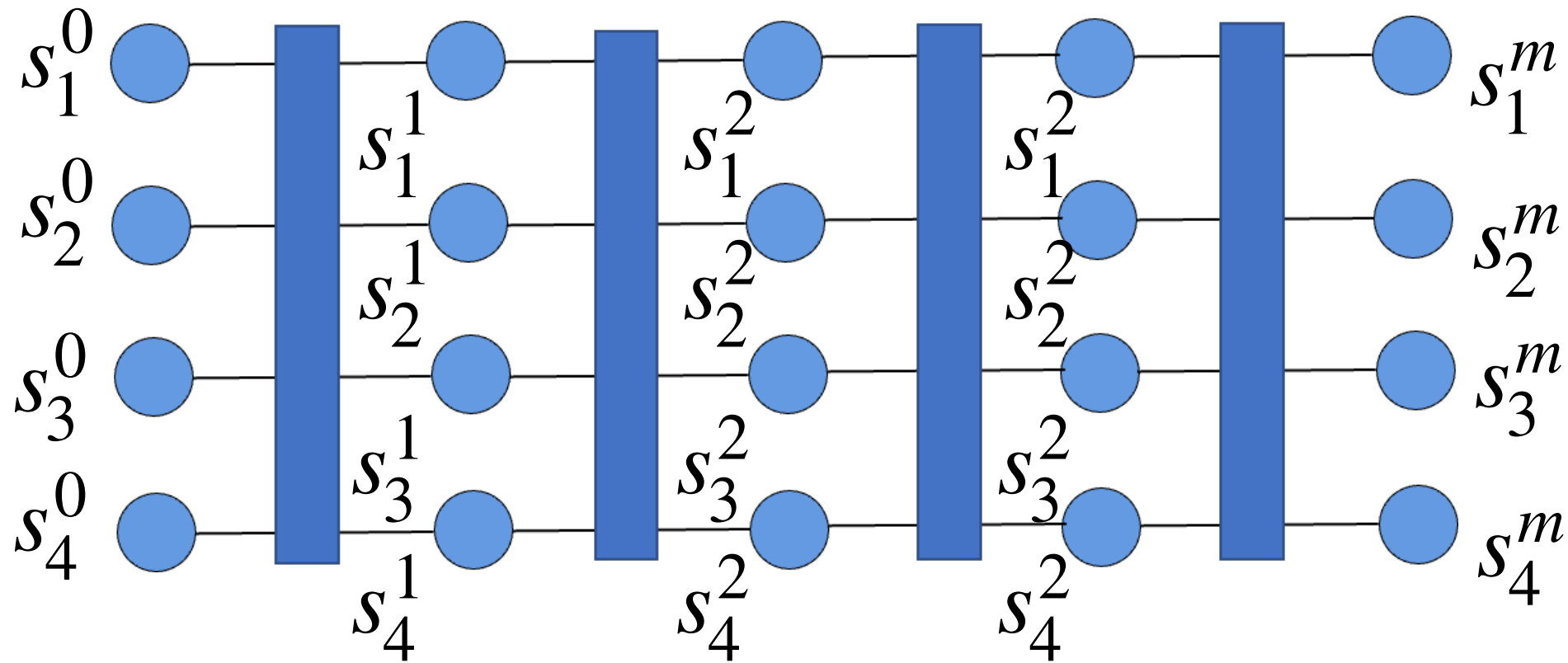
$$P(s_1^m, s_2^m, s_3^m, \dots, s_n^m) = \frac{1}{Z} \left| \sum_{s_1} \sum_{s_2} \dots \sum_{s_{m-1}} \prod_{t=1}^m \langle s_{t-1} | U_t | s_t \rangle \right|^2$$

$$= \frac{1}{Z} \left| \sum_{s_1^1} \sum_{s_2^1} \dots \sum_{s_n^{m-1}} e^{-E(s_1^1, s_2^1, \dots, s_n^{m-1})} \right|^2$$

Single amplitude  $\langle \psi | s_m \rangle = \text{Energy of complex Stat. Mech.}$



# Classical variational / sampling methods ?



Energy functions are difficult to compute.

Sign problem for sampling hidden variables.

No variational principles for joint amplitude of all variables.

# Trading fidelity with complexity

Using approximate state, e.g. MPS

Sampling from a mixed distribution ( a small portion from true distribution + a large part from pure noise)

- Noisy state:  $\rho = f|\psi\rangle\langle\psi| + (1-f)\frac{1}{2^n}$
- 2000 bit strings with from  $|\psi\rangle\langle\psi|$  and 998000 from uniform distribution

Summing over a fraction of paths in the path-integral representation.

# Simulation of stabilizer circuits

Stabilizer circuits:

- Gates are

$$\text{CNOT } C_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} : \text{create entanglements}$$

$$\text{Hadamard } H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} : \text{create superpositions}$$

$$\text{Cphase } S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} : \text{add complex phases}$$

- Stabilizer states from  $|000\dots 0\rangle$

Creates complex entanglements, but not universal

*Gottesman-Knill Theorem:* Stabilizer circuits can be simulated in polynomial time:  
**storing stabilizers (generators of the stabilizer sub-group) rather than the state**

## Stabilizers: single qubit

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$X^2 = Y^2 = Z^2 = I^2 = I$$

$$XY = iZ, YX = -iZ, YZ = iX, ZY = -iX, ZX = iY, XZ = -iY$$

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad |+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$Z|0\rangle = |0\rangle: Z \text{ stabilizes } |0\rangle, \quad X|+\rangle = |+\rangle: X \text{ stabilizes } |+\rangle$$

Stabilizer group for  $|0\rangle$  is  $\{I, Z\} = \langle Z \rangle$

Stabilizer group for  $|+\rangle$  is  $\{I, X\} = \langle X \rangle$

## Stabilizers: two and more qubits

GHZ state:  $|\phi_+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

$$Z_1 Z_2 |\phi_+\rangle = |\phi_+\rangle, X_1 X_2 |\phi_+\rangle = |\phi_+\rangle, -Y_1 Y_2 |\phi_+\rangle = |\phi_+\rangle, I_1 I_2 |\phi_+\rangle = |\phi_+\rangle$$

Stabilizer group for  $|\phi_+\rangle$  is  $\{II, XX, ZZ, -YY\} = \langle XX, ZZ \rangle$

Stabilizer group for  $|\phi_-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$  is

$$\{II, -XX, ZZ, YY\} = \langle -XX, ZZ \rangle$$

The  $n$ -qubit stabilizer states can be determined by a stabilizer group of size  $2^n$ , which has  $n$  generators.

Rather than storing the state vector  $|\phi\rangle$  with  $2^n$  parameters, storing the  $n$  stabilizers.

# Stabilizers: check matrix representation

$$P_1 \cong V_4 \cong C_2 \times C_2 \quad I \mapsto 00 \quad X \mapsto 10 \quad Z \mapsto 01 \quad Y \mapsto 11$$

Stabilizers	X checks	Z checks
XIIII	10000	00000
ZIIII	00000	10000
IYIII	01000	01000
IIXZI	00100	00010

$l$  rows, each row indicate a stabilizer

$n$  columns, corresponding to  $n$  qubits

There is also a overall phase (not shown here)

# Stabilizer circuits

Now consider the state after a unitary operator  $U$  applied on an initial state  $|\psi\rangle$  which is stabilized by group  $S$  with  $g \in S$ ,  $g|\psi\rangle = |\psi\rangle$ .

$$U|\psi\rangle = Ug|\psi\rangle = UgU^\dagger Ug|\psi\rangle = (UgU^\dagger)(U|\psi\rangle)$$

- $UgU^\dagger$  stabilizes  $U|\psi\rangle$ .
- If  $UgU^\dagger$  is also a Pauli operator,  $U|\psi\rangle$  is determined by stabilizers specified by  $UgU^\dagger$ .
- **No need to store  $U|\psi\rangle$ , just trace the change of  $S$**

# Stabilizer circuits

For example:

Hadamard gate  $H$ :

$$HXH^\dagger = Z; \quad HYH^\dagger = -Y; \quad HZH^\dagger = X$$

Controlled-Not gate  $U$ :

$$C_x X_1 C_x^\dagger = X_1 X_2; \quad C_x X_2 C_x^\dagger = X_2; \quad C_x Z_1 C_x^\dagger = Z_1; \quad C_x Z_2 C_x^\dagger = Z_1 Z_2$$

Phase gate  $S$ :

$$SXS^\dagger = Y; \quad SZS^\dagger = Z.$$

Actually,  $\{H, C_x, S\}$  generates  $N(G_n)$ , the normalizer of  $G_n$ , Pauli group on  $n$  qubits

$$\text{i.e. for } U \in \langle H, C_x, S \rangle, \quad UG_n U^\dagger = G_n$$

$N(G_n)$  is also known as *Clifford group*



# Simulation of stabilizer circuits

Stabilizers	X checks	Z checks
XIIII	10000	00000
ZIIII	00000	10000
IYIII	01000	01000
IIXZI	00100	00010

To apply  $H$  to the  $i^{\text{th}}$  qubit:

- Swap the  $i^{\text{th}}$  row of the X check to the Z check

To apply  $S$  to the  $i^{\text{th}}$  qubit:

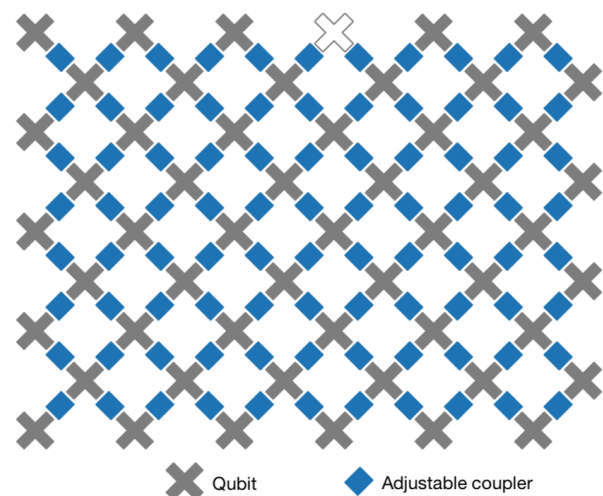
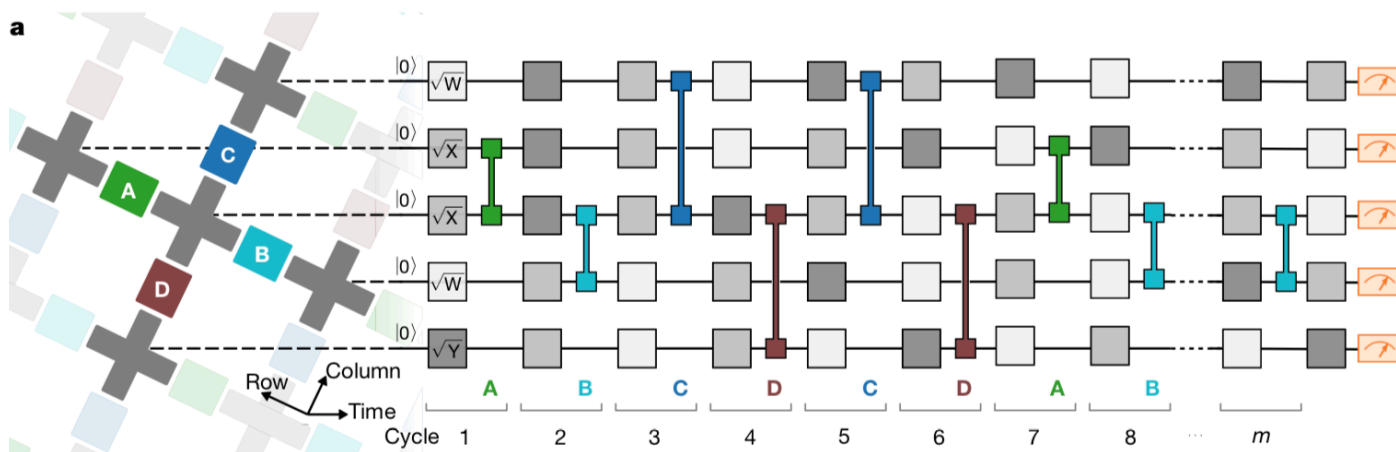
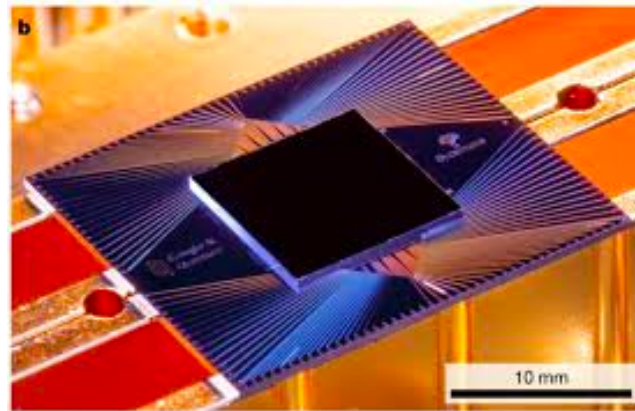
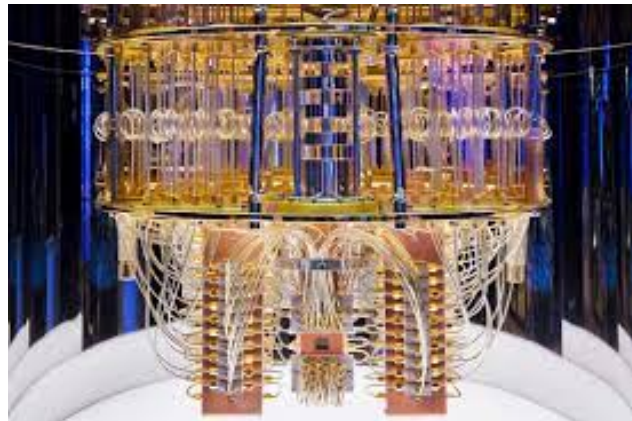
- Bitwise XOR the  $i^{\text{th}}$  row of the X check into the  $i^{\text{th}}$  row of the Z check

To apply  $C_x$  from the  $i^{\text{th}}$  qubit to the  $j^{\text{th}}$  qubit:

- Bitwise XOR the  $i^{\text{th}}$  row of the X check into the  $j^{\text{th}}$  row of the X check
- Bitwise XOR the  $j^{\text{th}}$  row of the Z check into the  $i^{\text{th}}$  row of the Z check

Measurements can also be conveniently (commute or anti-commute with stabilizers).

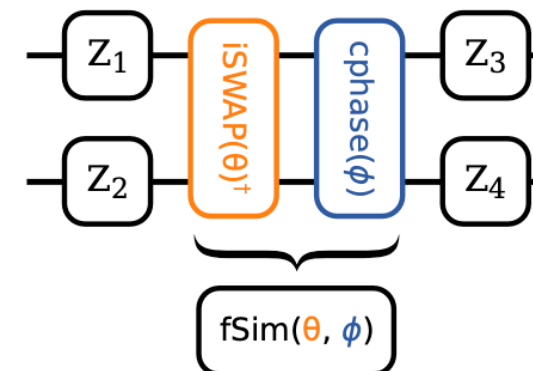
# Google's Sycamore circuits



**53 qubits, 20 cycles**

$$\sqrt{X} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}, \quad \sqrt{Y} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad \sqrt{W} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -\sqrt{i} \\ \sqrt{-i} & 1 \end{bmatrix}.$$

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



**nature**

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [articles](#) > [article](#)

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

## Quantum supremacy using a programmable superconducting processor

[Frank Arute](#), [Kunal Arya](#), ... [John M. Martinis](#)  [+ Show authors](#)

[Nature](#) **574**, 505–510 (2019) | [Cite this article](#)

**878k** Accesses | **1479** Citations | **6167** Altmetric | [Metrics](#)

# Single qubit gates of Sycamore

Each one is a  $\pi/2$ -rotation around an axis lying on the equator of the Bloch sphere.  
Up to a global phase, the gates are

$$\left. \begin{aligned} X^{1/2} &\equiv R_X(\pi/2) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}, \\ Y^{1/2} &\equiv R_Y(\pi/2) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \end{aligned} \right\} \text{single-qubit Clifford gates}$$

$$\begin{aligned} W^{1/2} &\equiv R_{X+Y}(\pi/2) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -\sqrt{i} \\ \sqrt{-i} & 1 \end{bmatrix} && \text{non-Clifford gate.} \\ &= (X + Y)/\sqrt{2} . \end{aligned}$$

# Quantum supremacy

A specific computational task

- No matter whether it is “useful”

Beyond the capabilities of classical super-computers

In the NISQ era:

- Noisy (no error correction)
- Circuits are not so deep
- Fidelity of gates are high

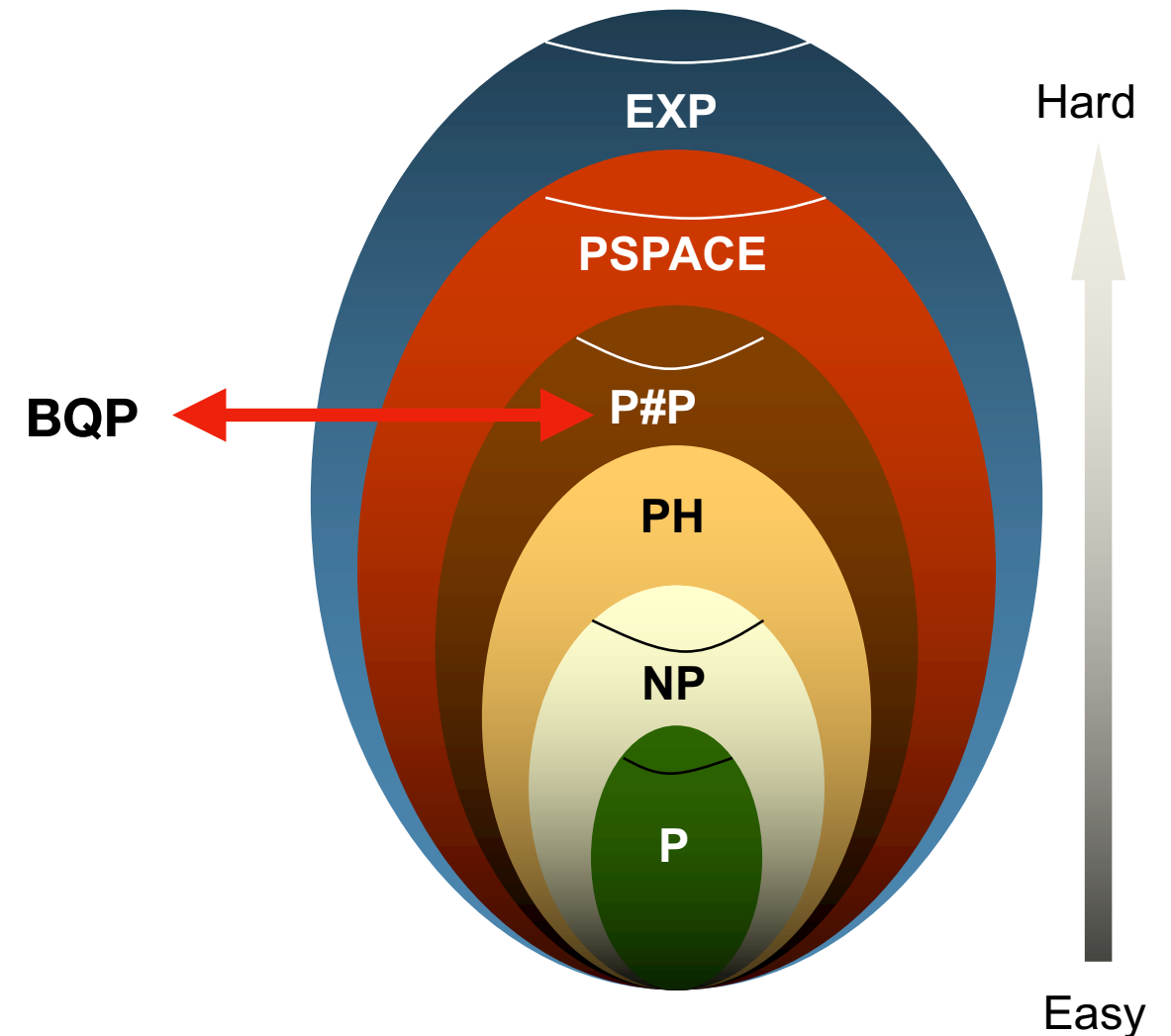
# Quantum supremacy

Aaronson and Chen's conjecture:

A random circuit  $U$  with  $n$  qubits and depth  $\sim \sqrt{n}$ , no classical algorithm can guess if

$$\langle 0^n | U | 0^n \rangle > \text{Median} (\langle 0^n | U | 0^n \rangle)$$

With probability  $\frac{1}{2} + O(2^{-n})$

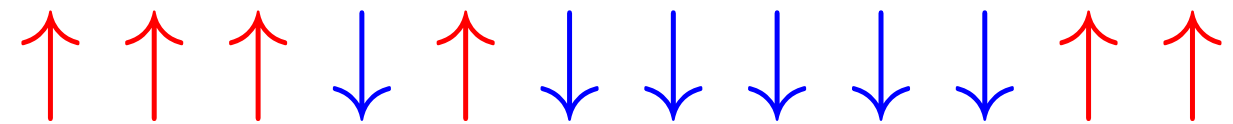




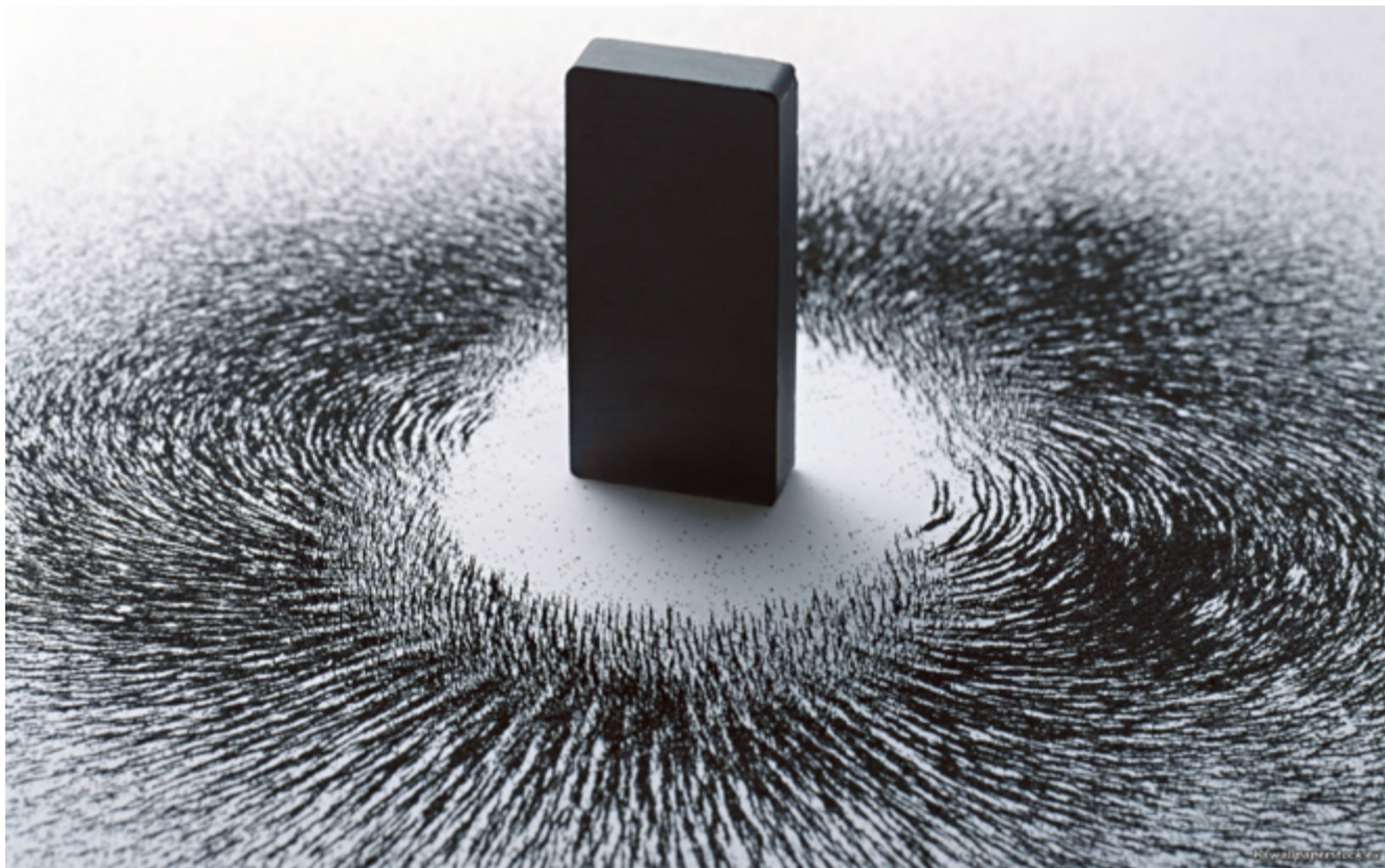
# Statistical Mechanics

$$\mathbf{S} = \{+1, -1\}^n$$

$$P(\mathbf{S}) = \frac{1}{Z} e^{-\beta E(\mathbf{S})}$$



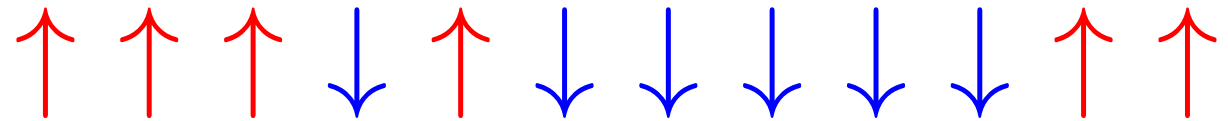
$$Z = \sum_{\mathbf{s}} e^{-\beta E(\mathbf{S})}$$



# Statistical Mechanics

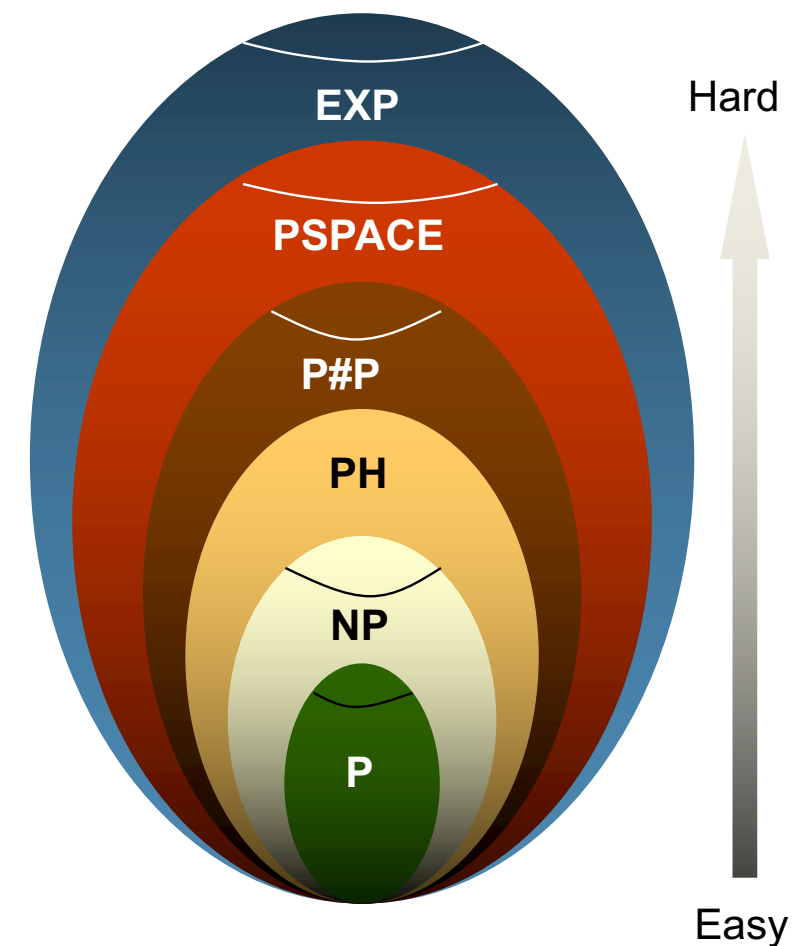
$$\mathbf{S} = \{+1, -1\}^n$$

$$P(\mathbf{S}) = \frac{1}{Z} e^{-\beta E(\mathbf{S})}$$



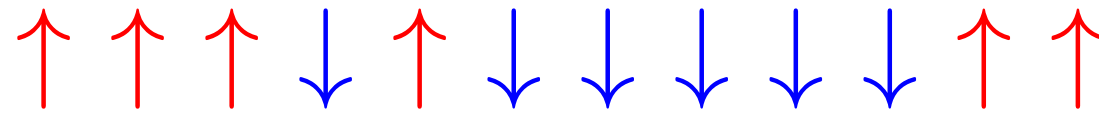
$$Z = \sum_{\mathbf{s}} e^{-\beta E(\mathbf{S})}$$

- Estimating the free energy
- Computing observables / order parameters
- Sampling



# Tensor network for Statistical Mechanics

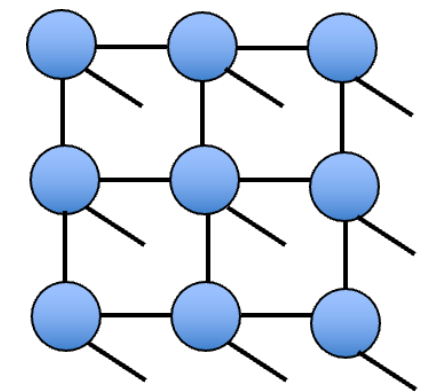
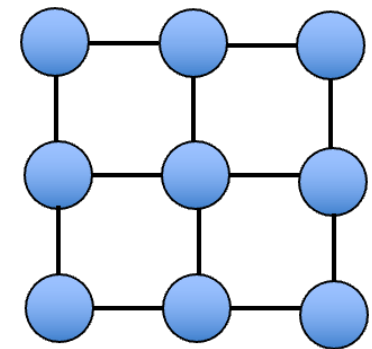
$$\mathbf{S} = \{+1, -1\}^n$$



Any **discrete probability distribution** is a tensor,

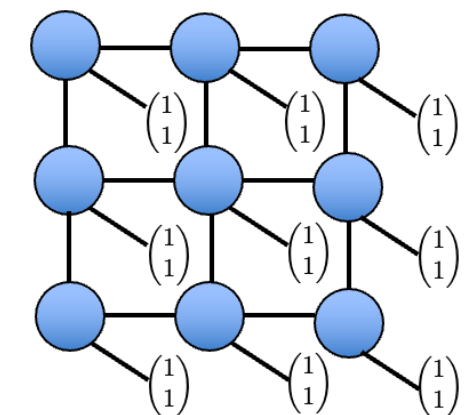
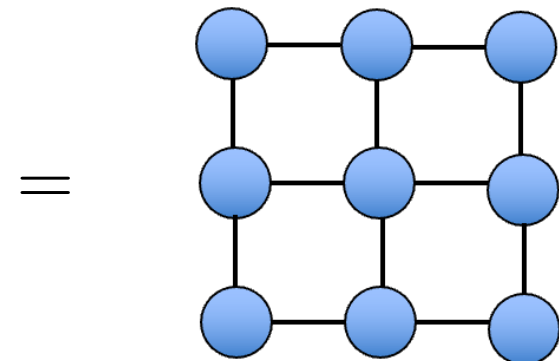
$$P(\mathbf{S}) = \frac{1}{Z} e^{-\beta E(\mathbf{S})} = \frac{1}{Z} \tilde{P}$$

decomposed using tensor networks.



Computing normalization of a **discrete probability distribution**

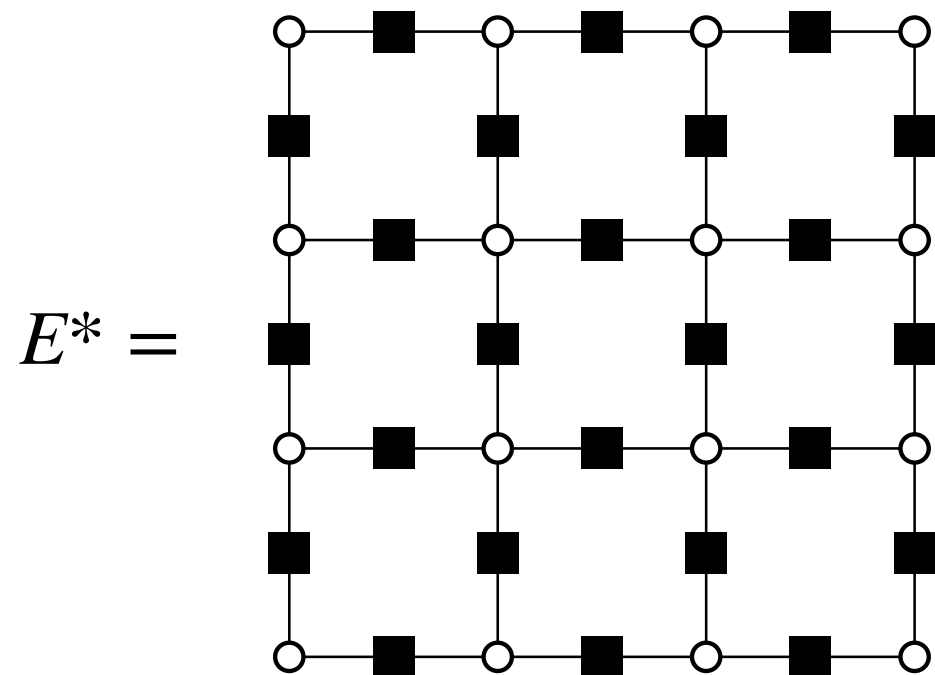
$$Z = \left\| \tilde{P} \right\|_1 = \tilde{P} \cdot \mathbf{1}_{2^n}^\top = \tilde{P} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix}}_n,$$





# Tensor networks for the Ising spin glasses

$$E(\mathbf{S}) = - \sum_{(ij)} J_{ij} S_i S_j - \sum_i h_i$$

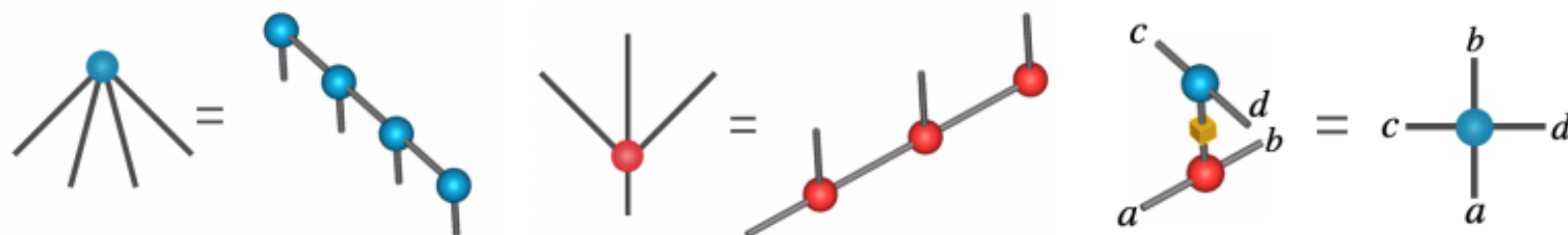
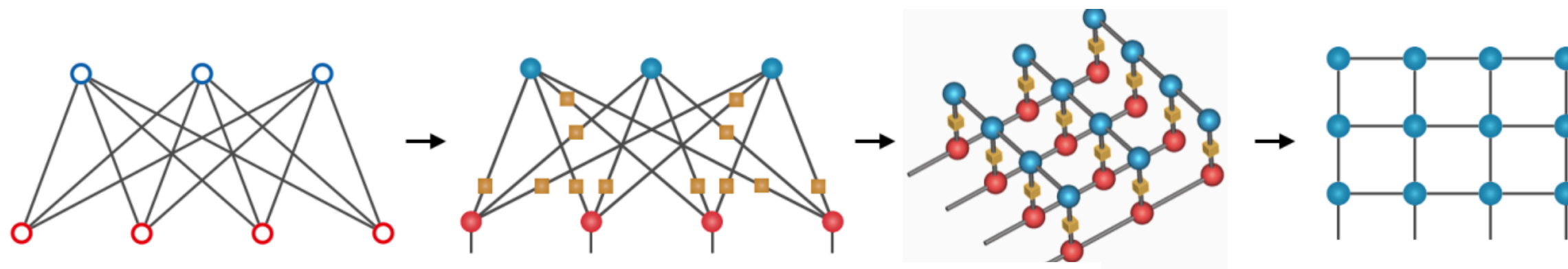
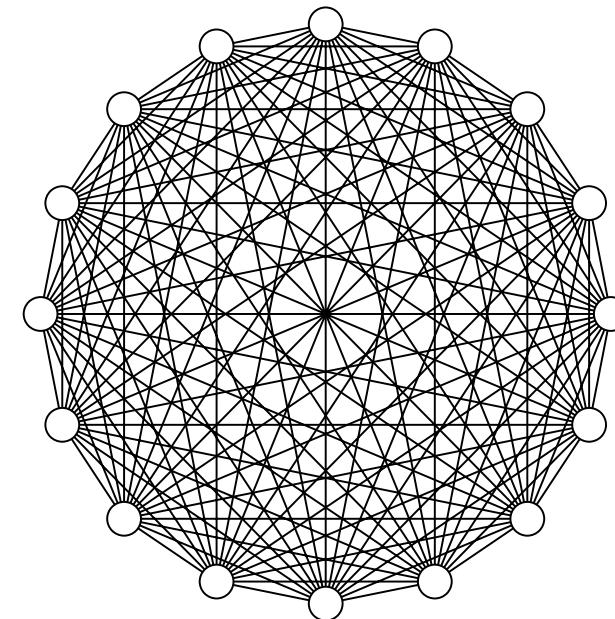
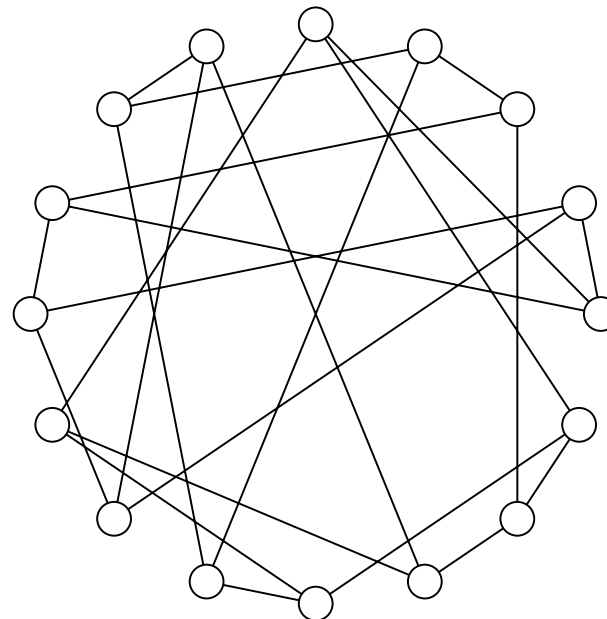
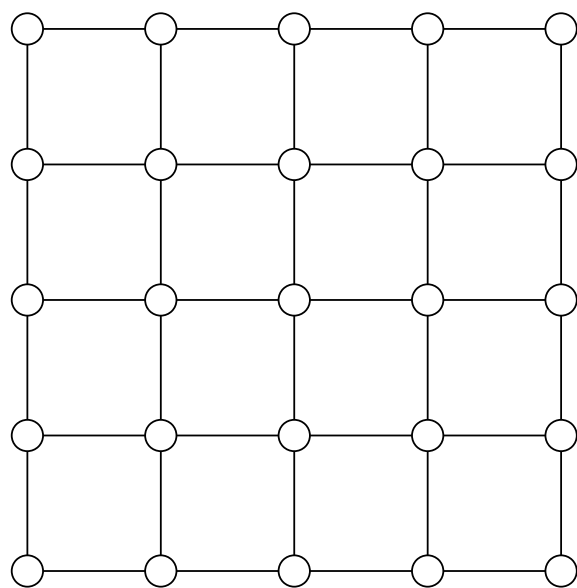


$$\text{---} \blacksquare \text{---} = \begin{pmatrix} e^{J_{ij}} & e^{-J_{ij}} \\ e^{-J_{ij}} & e^{J_{ij}} \end{pmatrix}$$

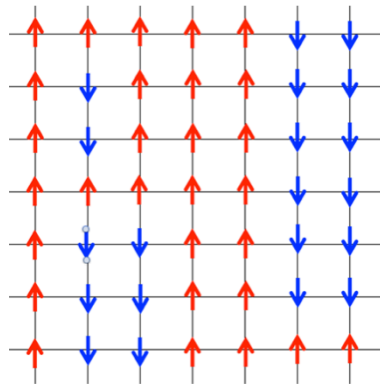
$$\begin{array}{c} 1 \\ | \\ 1 \text{---} \bigcirc \text{---} 1 \\ | \\ 1 \end{array} = e^{\beta h_i} \qquad \begin{array}{c} 2 \\ | \\ 2 \text{---} \bigcirc \text{---} 2 \\ | \\ 2 \end{array} = e^{-\beta h_i}$$

all other elements are 0

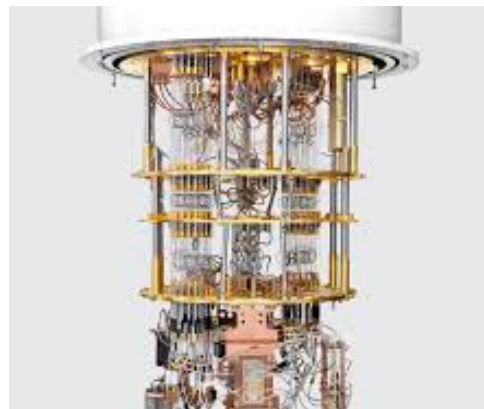
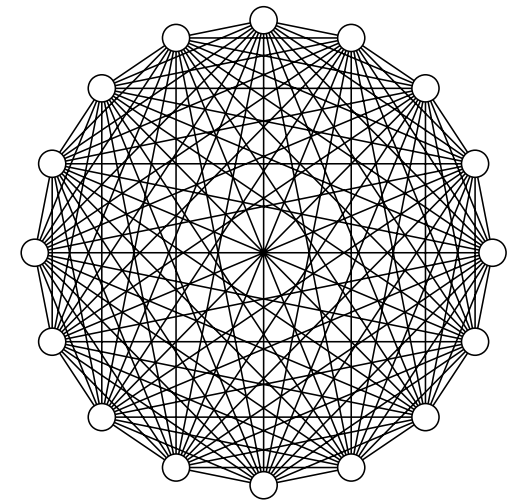
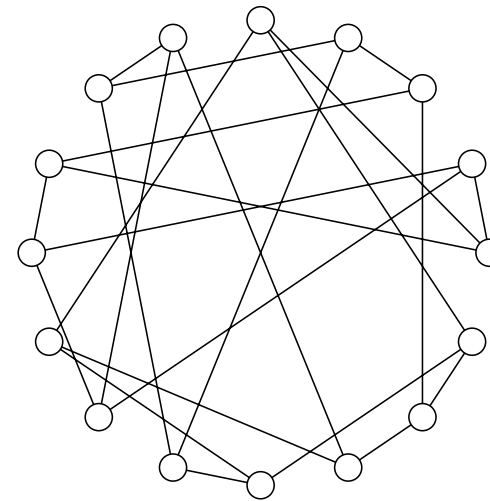
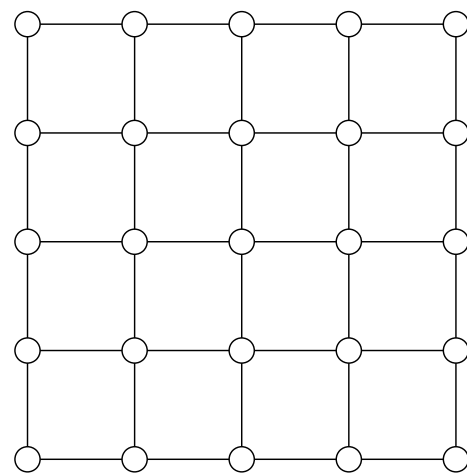
# TN Contraction $\longrightarrow Z(\beta)$



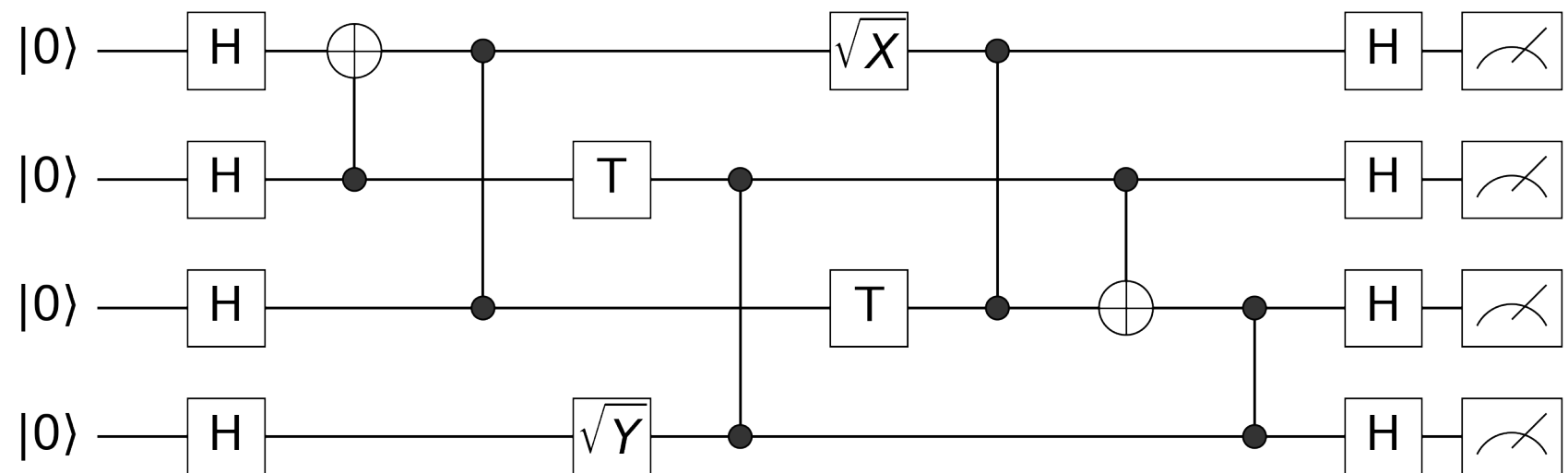
# Stat. Mech. $\longrightarrow$ Quantum Computer Simulation



Statistical Mechanics



Quantum Circuits



Partition function of stat. mech. with **complex** interactions

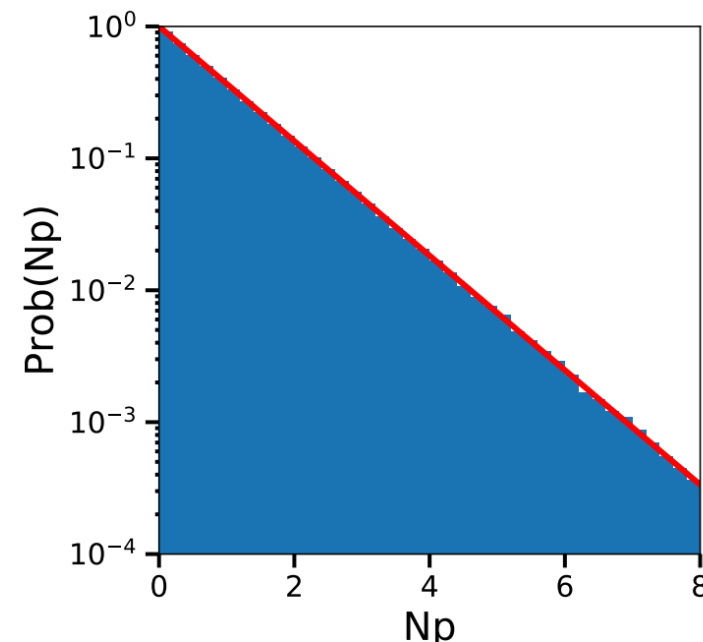
=

single amplitude computation of quantum circuit

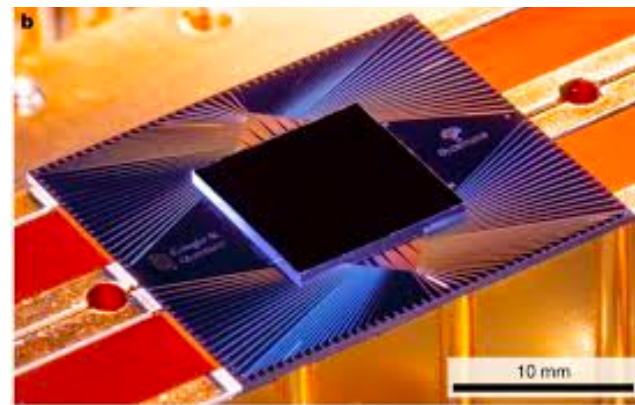
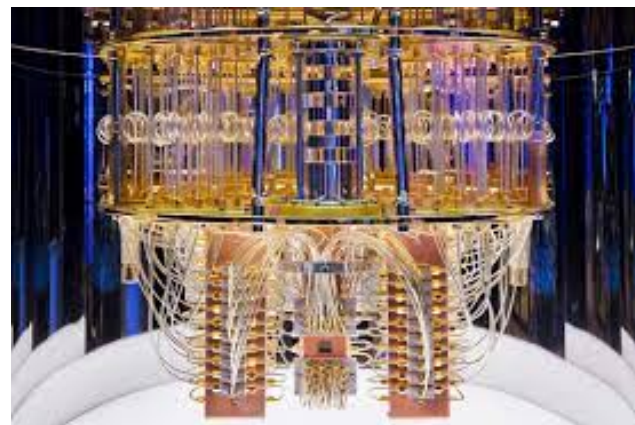
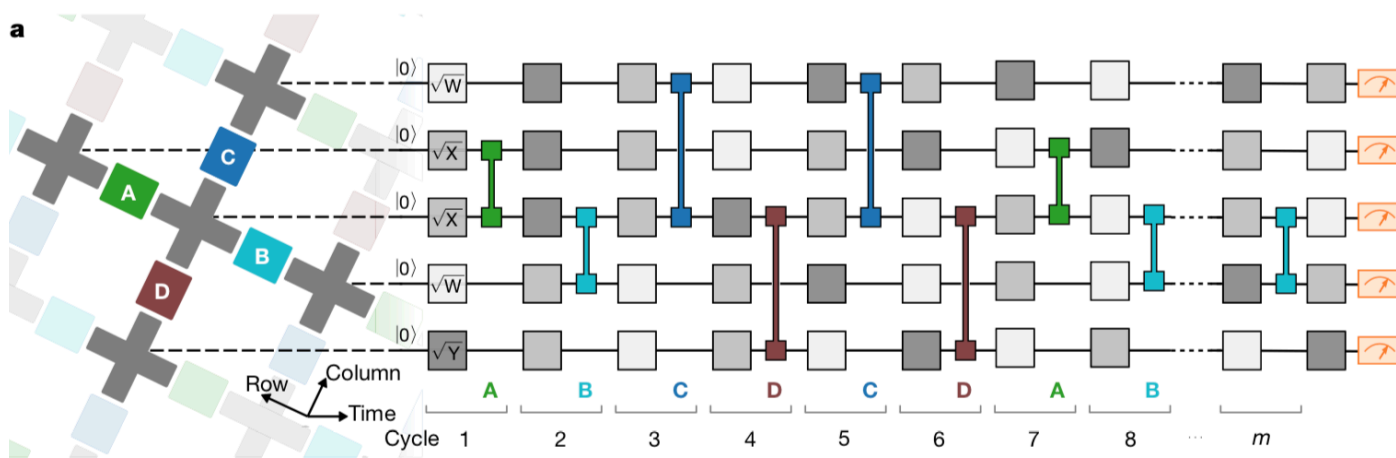
# Random circuits

For a random circuit  $U$  with gates drawn randomly from universal set

- Final state  $|\psi\rangle = U|0\rangle = \sum_{i=1}^{2^n} \psi(s_i) |s_i\rangle$
- Probability distribution  $P_U(s_i) = |\langle s_i | \psi \rangle|^2 = |\psi(s_i)|^2$
- Both real and imaginary part of  $\psi(s_i)$  are uniform random variables on a  $2^n$  Hilbert space with mean 0 and variance  $2^{-n}$
- Porter-Thomas distribution  
 $\text{Prob}(Np) = e^{-Np}$  with  $N = 2^n$



# Google's Quantum Supremacy experiments



**nature**

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [articles](#) > [article](#)

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

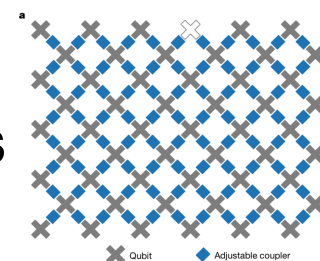
## Quantum supremacy using a programmable superconducting processor

[Frank Arute](#), [Kunal Arya](#), ... [John M. Martinis](#) [+ Show authors](#)

[Nature](#) **574**, 505–510 (2019) | [Cite this article](#)

**878k** Accesses | **1479** Citations | **6167** Altmetric | [Metrics](#)

- **53 qubits, 20 cycles**

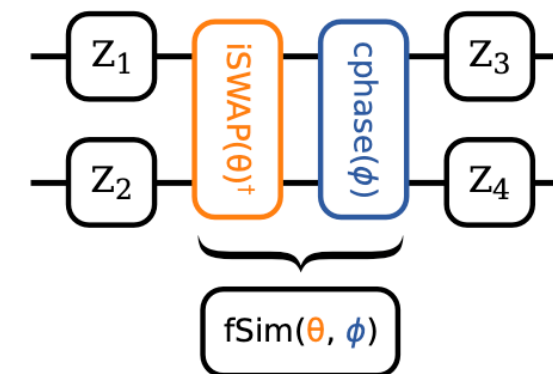


$$\text{fSim}(\theta, \phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -i \sin \theta & 0 \\ 0 & -i \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{bmatrix}$$

- **1 million samples in 200 Sec.**

- **Linear Cross Entropy Fidelity (XEB)  $\approx 0.002$**

- **Classic algorithm requires 10,000 years on Summit**



$$\begin{aligned} F_{\text{XEB}} &= 2^n \sum_{s \in \{1,0\}^n} q(s) p_U(s) - 1 \\ &= 2^n \langle p_U(s) \rangle_q - 1 \\ &\approx \frac{2^n}{m} \sum_{s \sim q} p_U(s) - 1 \end{aligned}$$

# Google's Sycamore circuits

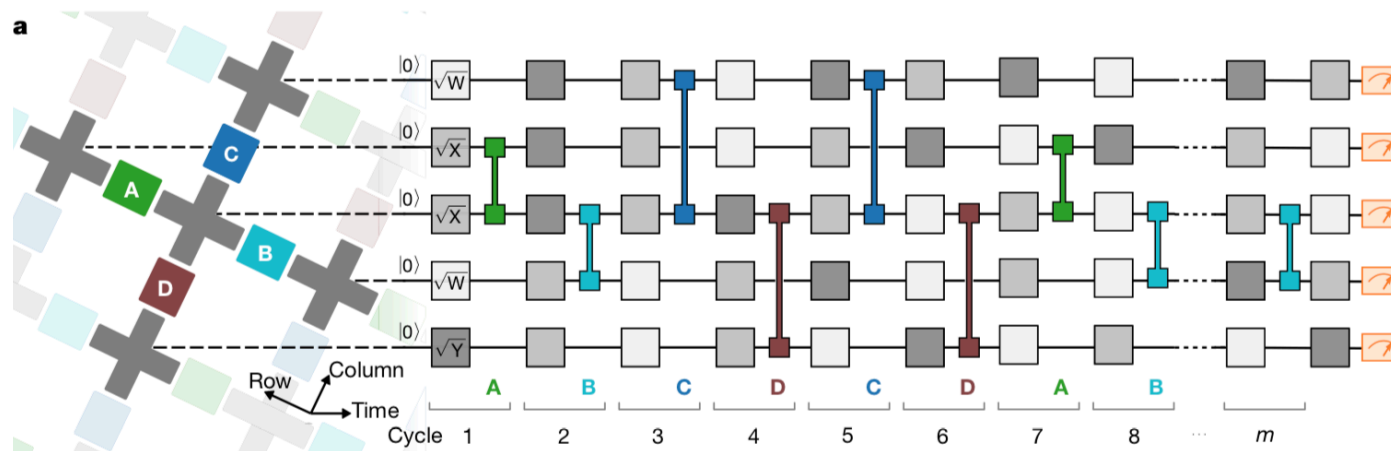
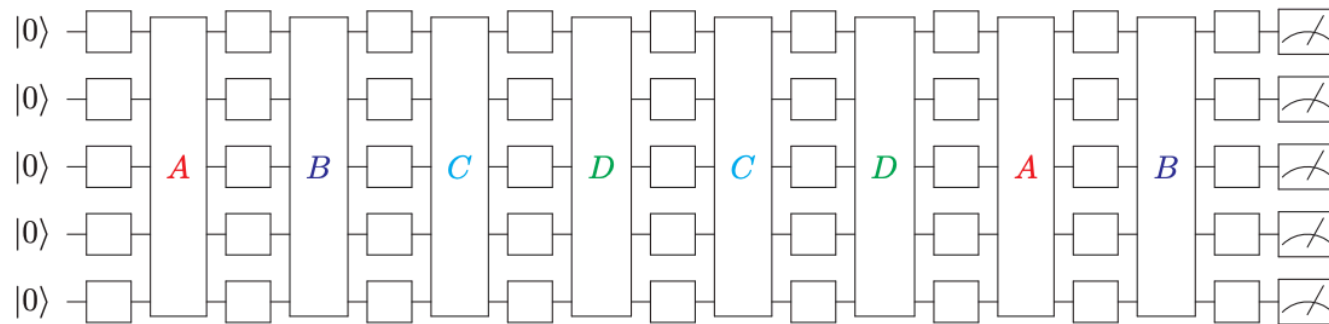
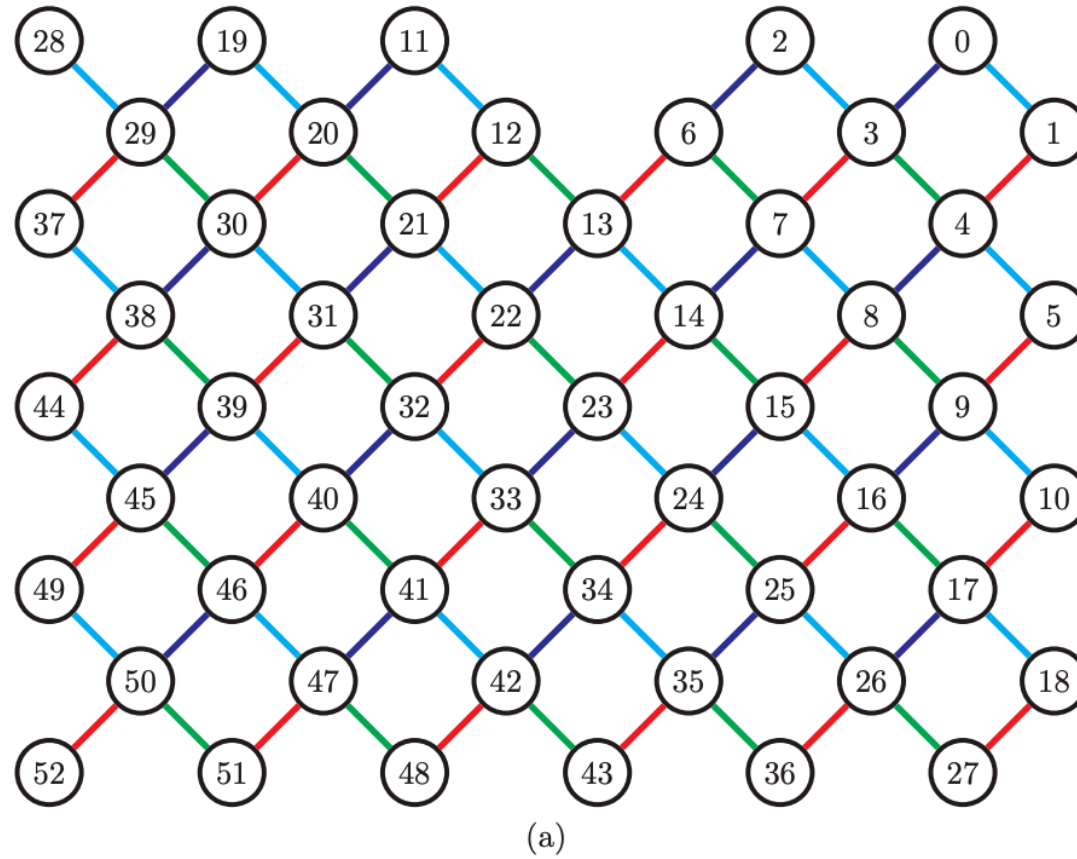
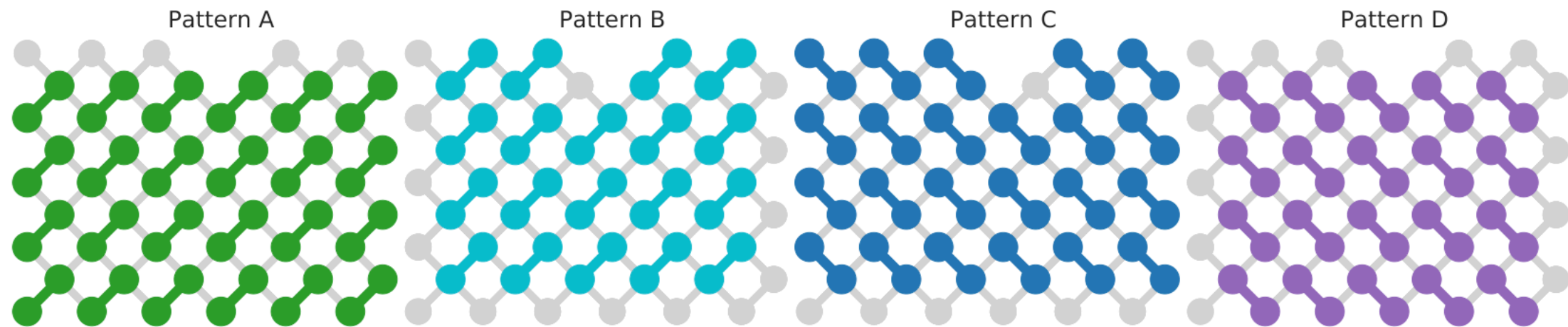


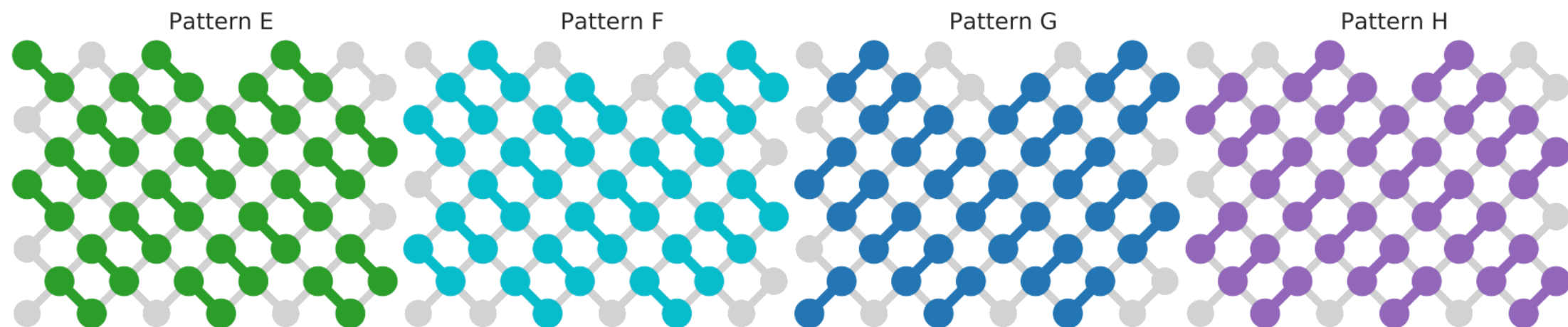
Fig courtesy: Huang et al 2020



# Two kinds of circuits



**Pattern of Supremacy circuits**



**Pattern of Simplifiable circuits**

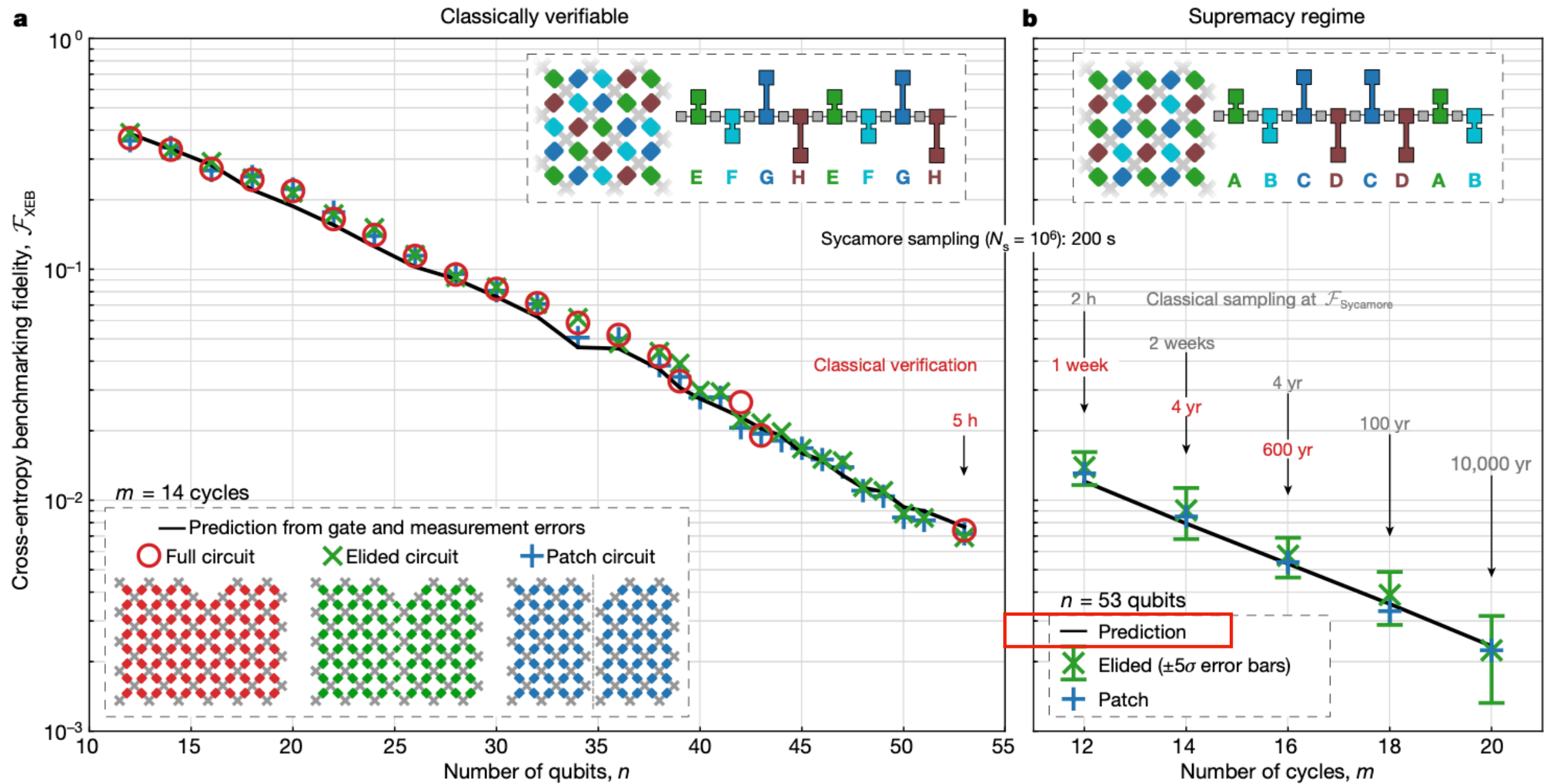
# Error model of Google's circuits

- all errors in the evolving quantum state may be characterized by a set of localized Pauli errors (bit-flips or phase-flips)
- discrete and probabilistic
- system fidelity is well predicted by a simple model in which the individually characterized fidelities of each gate are multiplied together

⇒ a predictive uncorrelated error model up to a Hilbert space of size  $2^{53}$



# Error model of Google's circuits



# Fidelity measure

Not possible to output the (noisy) final state  $|\phi\rangle$

- Quantum device can only sample from it
- Not possible to compute the fidelity  $|\langle\psi|\phi\rangle|^2$

Approximate estimates:

- KL divergence
- Cross entropy
- Logarithm cross entropy
- Linear cross entropy

# Entropy

$|\psi\rangle = U|0\rangle$  is the final state of the circuit  $U$

$P_U(s) = |\langle s | \psi \rangle|^2 = |\psi(s)|^2$  is the distribution of bitstring  $s$

$m$  samples  $S = \{s_1, s_2, \dots, s_m\}$  are drawn from  $P_U(s)$ .

The joint probability of generating  $S$  is

$$P(S) = \prod_{i=1}^m P(s_i)$$

And  $\log P(S) = \sum_{i=1}^m \log P(s_i) = -mH(P_U) + O(m^{1/2})$

where

$$H(P_U) = - \sum_{i=1}^{2^n} P_U(s_i) \log P_U(s_i) \text{ is the entropy of } P_U.$$

# Cross Entropy of the true distribution

For the true output distribution of the random circuit  $U$  with a sufficient depth.

The prob. Follows the Porter-Thomas distribution

$$\text{Prob}(Np) = e^{-Np}$$

The entropy is computed as

$$H(P_U) = - \sum_{i=1}^{2^n} P_U(s_i) \log P_U(s_i) = - \int_0^\infty dp \log p N^2 e^{-Np} = \log N - 1 + \gamma$$

$\gamma \approx 0.577$  is the Euler's gamma constant.

# Cross Entropy of the generation distribution

For a generation distribution  $q(s)$  (given e.g. by a classic algorithm) where  $m$  samples  $S = \{s_1, s_2, \dots, s_m\}$  are drawn from.

The probability of observing samples  $S$  on the circuit  $U$

$$\text{Prob}(S) = \prod_{i=1}^m P_U(s_i)$$

$$\log \text{Prob}(S) = \sum_{i=1}^m \log P_U(s_i) = -mH(q, P_U) + O(m^{1/2})$$

$$H(q, P_U) = - \sum_{i=1}^{2^n} q(s_i) \log P_U(s_i) : \text{cross entropy between } q \text{ and } P_U.$$

$$\text{e.g. } H(q_{\text{uni}}, P_U) = - \sum_{i=1}^{2^n} 2^{-n} \log P_U(s_i) = \log N + \gamma = H_0$$

$$H(P_U) = \log N - 1 + \gamma$$

Notice that

# Cross Entropy and the KL divergence

$$\begin{aligned} H(q, P_U) &= - \sum_{i=1}^{2^n} q(s_i) \log P_U(s_i) \\ &= - \left( \sum_{i=1}^{2^n} q(s_i) \log P_U(s_i) - \sum_{i=1}^{2^n} q(s_i) \log q(s_i) + \sum_{i=1}^{2^n} q(s_i) \log q(s_i) \right) \\ &= - \left( \sum_{i=1}^{2^n} q(s_i) [\log P_U(s_i) - \log q(s_i)] + \sum_{i=1}^{2^n} q(s_i) \log q(s_i) \right) \\ &= - D_{\text{KL}} + H(q) \\ D_{\text{KL}} &= H(q) - H(q, P_u) \geq 0 \\ &\implies H(q, P_u) \leq H(q) \end{aligned}$$

# Cross Entropy Benchmark

Use difference of Cross Entropy to define how well the generation distribution  $q(s)$  can predict the output of the circuit  $U$

$$\begin{aligned}\Delta H(q) &= H_0 - H(q, P_U) \\ &= \sum_i \left( q(s_i) - \frac{1}{N} \right) \log P_U(s_i)\end{aligned}$$

- 0 for uniform distribution  $q_{\text{uni}}$
- 1 for true distribution  $P_U(s)$

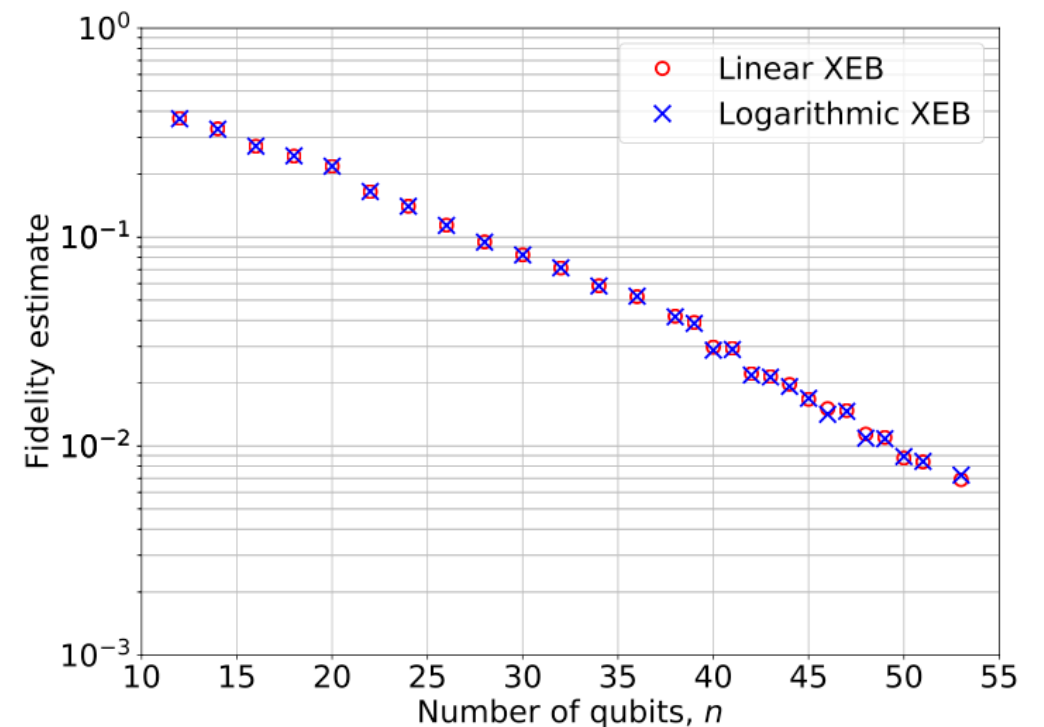
# Cross Entropy Benchmark (XEB)

## Logarithm XEB

$$\begin{aligned} F_{\log\text{XEB}} &= \langle \log N \log P_U(s) \rangle_q + \gamma \\ &= \sum_{\mathbf{s} \in \{1,0\}^n} q(\mathbf{s}) \log N \log p_U(\mathbf{s}) + \gamma \\ &\approx \frac{1}{m} \sum_{s \sim q} \log N \log p_U(\mathbf{s}) + \gamma \end{aligned}$$

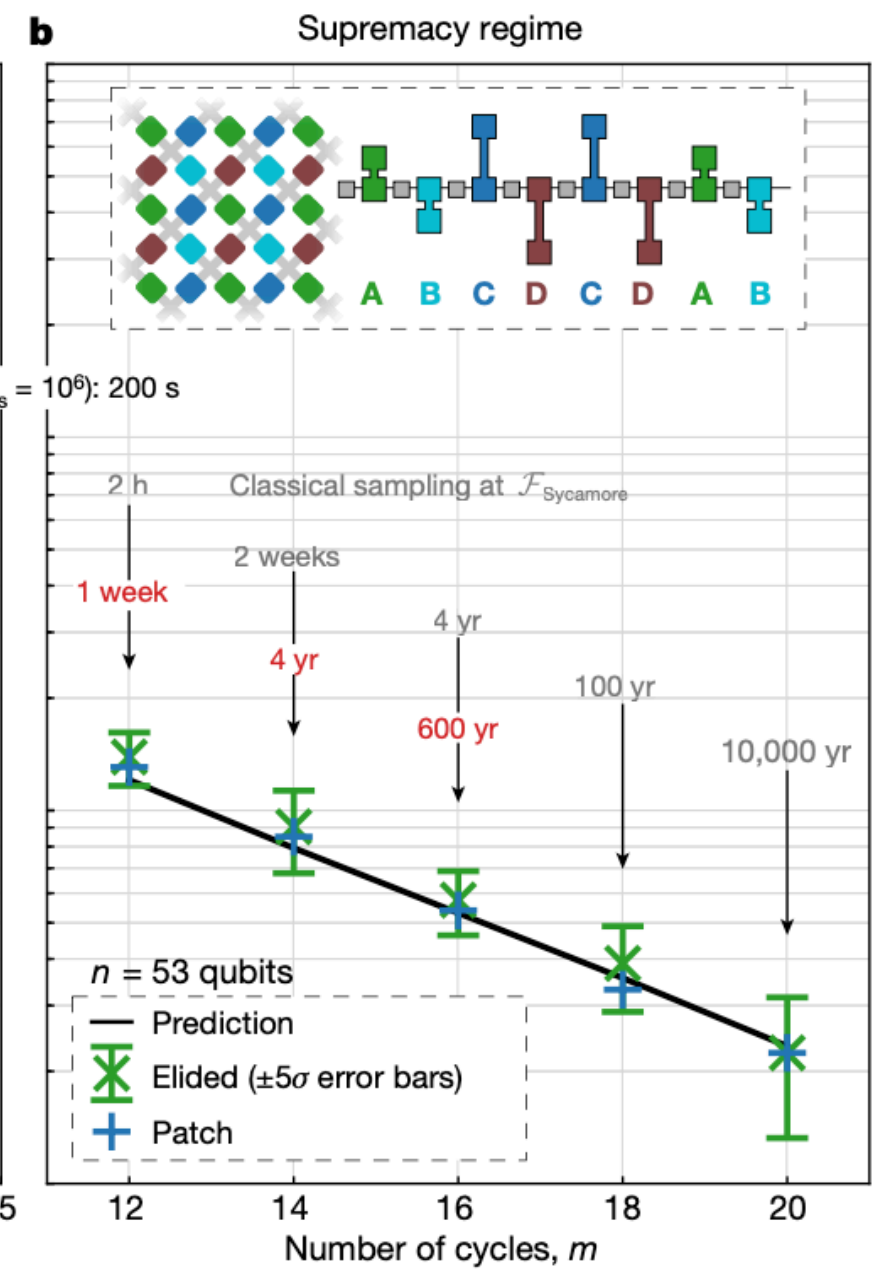
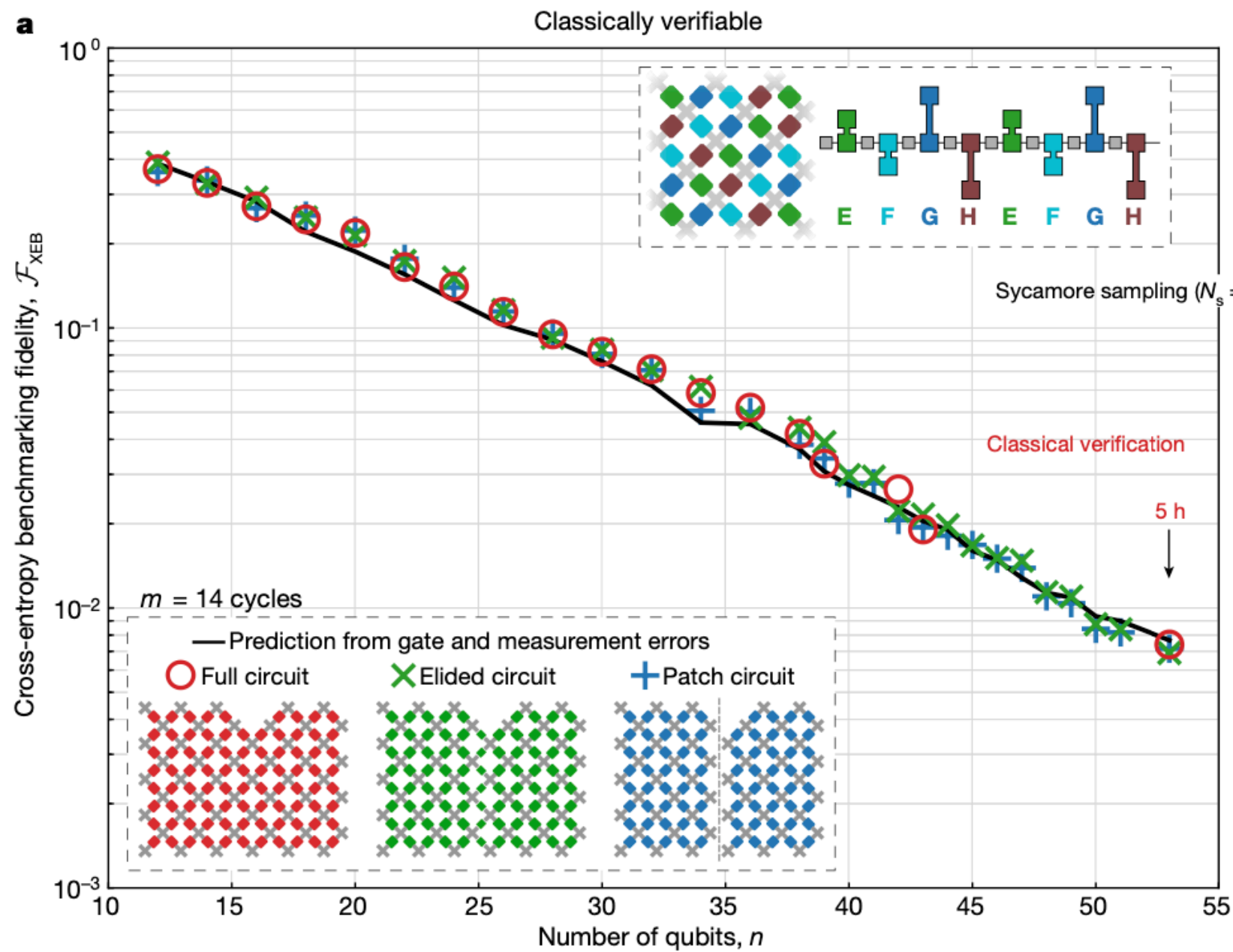
## Linear XEB

$$\begin{aligned} F_{\text{XEB}} &= 2^n \langle p_U(\mathbf{s}) \rangle_q - 1 \\ &= 2^n \sum_{\mathbf{s} \in \{1,0\}^n} q(\mathbf{s}) p_U(\mathbf{s}) - 1 \\ &\approx \frac{2^n}{m} \sum_{s \sim q} p_U(\mathbf{s}) - 1 \end{aligned}$$



Arute et al. Nature 2019





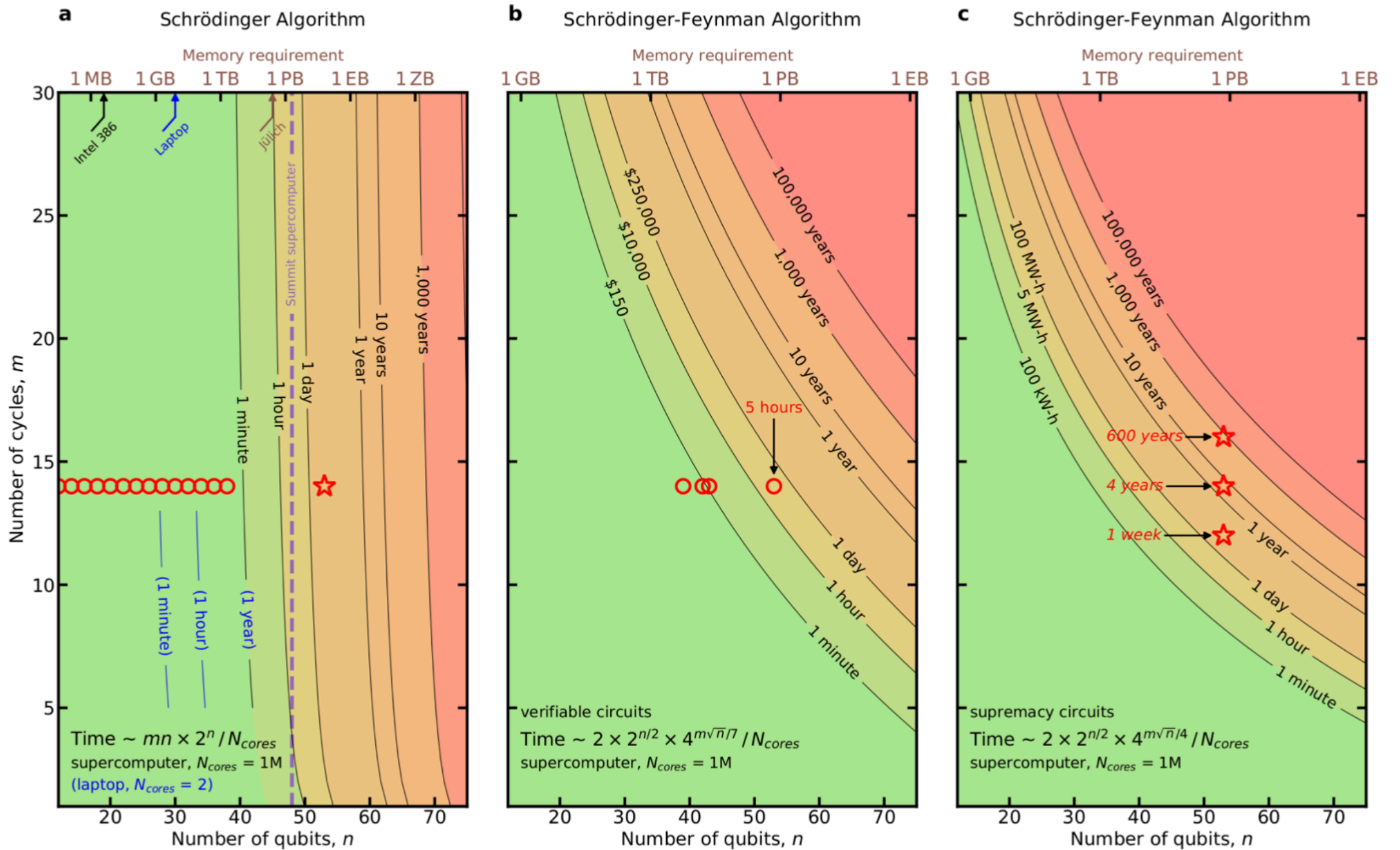
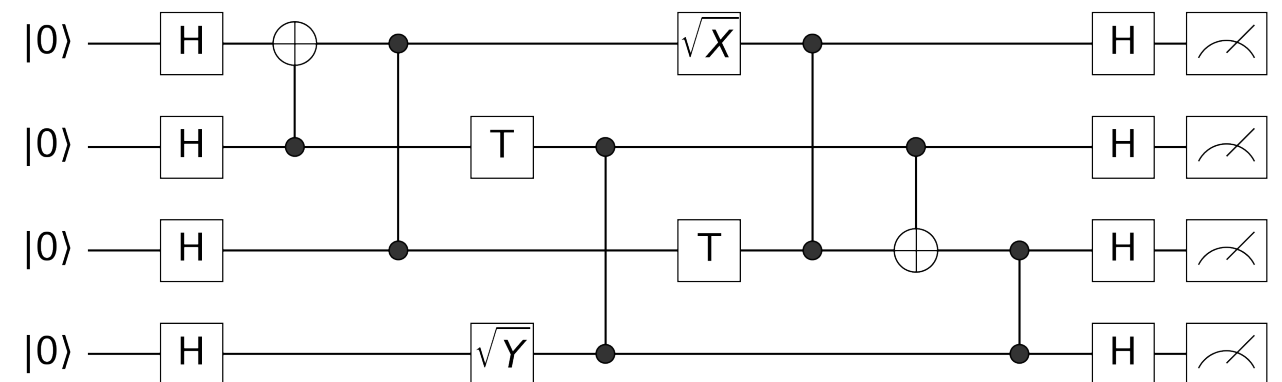


FIG. S50. **Scaling of the computational cost of XEB using SA and SFA.** **a**, For a Schrödinger algorithm, the limitation is RAM size, shown as vertical dashed line for the Summit supercomputer. Circles indicate full circuits with  $n = 12$  to 43 qubits that are benchmarked in Fig. 4a of the main paper. 53 qubits would exceed the RAM of any current supercomputer, and is shown as a star. **b**, For the hybrid Schrödinger-Feynman algorithm, which is more memory efficient, the computation time scales exponentially in depth. XEB on full verifiable circuits was done at depth  $m = 14$  (circle). **c**, XEB on full supremacy circuits is out of reach within reasonable time resources for  $m = 12, 14, 16$  (stars), and beyond. XEB on patch and elided supremacy circuits was done at  $m = 14, 16, 18$ , and 20.

# Simulation methods

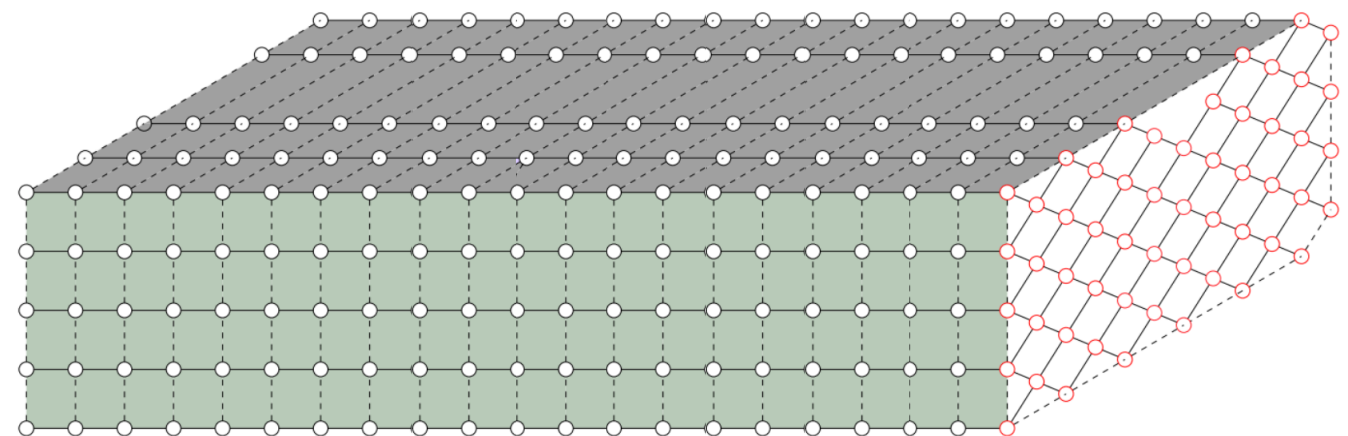
## Full amplitudes:

- Storing full state-vector [**Yao.jl**, Qiskit, Qulacs, Cirq...]
- Schrödinger-Feynmann
- MPS [PRX 10, 041038 (2020)]



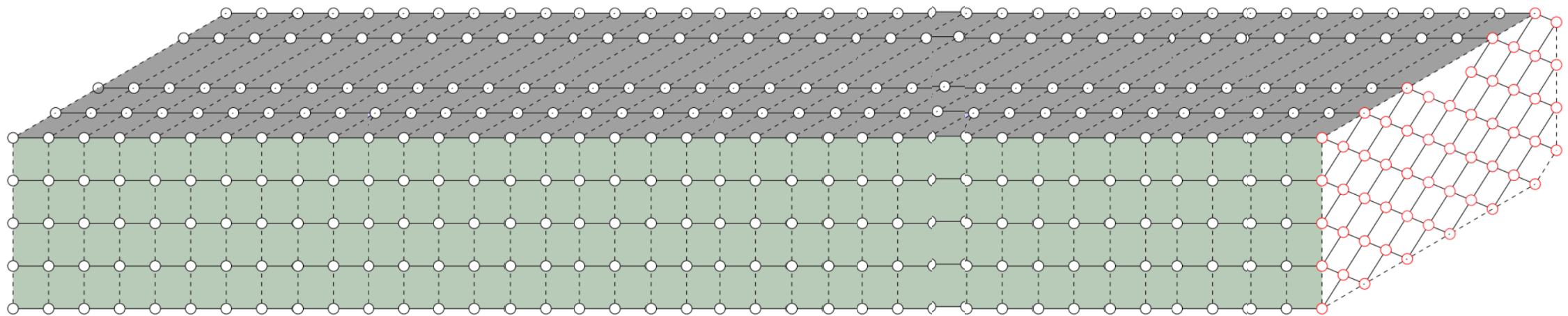
## Single/batch amplitudes:

- PEPS based / QuickBB order (single amplitude)
- Cotengra (single amplitude)
- Alibaba ACQDP (64-amplitude batch)
- Big-batch method [arXiv:2103.03074]
- Recursive multi-tensor contraction [arXiv:2108.05665]
- Sparse-state method [arXiv:2111.03011]



# Full amplitude simulations

Once you can store the state-vector, you can simulate the circuit with an arbitrary depth.



**Because the tree width of the 3D graph  
is influenced heavily by the size of the boundary**

Quantum circuit diagram for the Quantum Fourier Transform (QFT) on 4 qubits. The circuit consists of four horizontal qubit lines, each starting with a  $|0\rangle$  state and an H gate. The circuit includes various gates: a CNOT from qubit 1 to qubit 2, a CNOT from qubit 2 to qubit 3, a CNOT from qubit 3 to qubit 4, a CNOT from qubit 4 to qubit 3, a CNOT from qubit 3 to qubit 2, and a CNOT from qubit 2 to qubit 1. Additionally, there are phase gates: a  $\sqrt{X}$  gate on qubit 1, a T gate on qubit 2, a T gate on qubit 3, and a  $\sqrt{Y}$  gate on qubit 4. The circuit ends with four H gates and four measurement symbols.

## 019

Tensor	Disk trasfers per disk slice	All-to- alls per disk slice	5Q kernels per disk slice	Tensor ranks per socket	Num gates	Contraction cost tot. FLOPs	Compute time (days)	% of total time	Achieved PFLOPS	
1		0.000977	28	28	84	1.181·10 <sup>21</sup>	0.002082	0.08%	0.0308	
2		0.000977	25	27	84		0.001859	0.07%	0.0173	
Contraction				31			0.117058	4.59%	116.7304	
3.3			16	32	63		0.010658	0.42%	18.4865	
3.4		1	6	32	23		0.003997	0.16%	17.9975	
3.5		1	8	32	26		0.005329	0.21%	15.2587	
Disk write	1	1								
Disk read	1	1								
4.4			11	32	49		0.007327	0.29%	20.9141	
4.5		1	10	32	45		0.006661	0.26%	21.1275	
Disk write	1	1								
Disk read	1	1								
5.5			9	32	35	0.005995	0.24%	18.2583		
5.6		1	7	32	21	0.004663	0.18%	14.0850		
Disk write	1	1								
Subtotals						1.181·10 <sup>21</sup>	0.165631	6.50%	87.4462	
Compute			120				0.487725	19.13%		
All-to-alls		9.001953					1.896296	74.37%		
Disk I/O	5									
Total	5	9.001953	120	32.67243	430		2.549652	100.00%	87.4462	



# Tensor network methods

## Simulating quantum computation by contracting tensor networks

Igor L. Markov<sup>1</sup> and Yaoyun Shi<sup>2</sup>

Department of Electrical Engineering and Computer Science  
The University of Michigan  
2260 Hayward Street  
Ann Arbor, MI 48109-2121, USA  
E-mail: {imarkov|shiyy}@eecs.umich.edu

### Abstract

The treewidth of a graph is a useful combinatorial measure of how close the graph is to a tree. We prove that a quantum circuit with  $T$  gates whose underlying graph has treewidth  $d$  can be simulated deterministically in  $T^{O(1)} \exp[O(d)]$  time, which, in particular, is polynomial in  $T$  if  $d = O(\log T)$ . Among many implications, we show efficient simulations for log-depth circuits whose gates apply to nearby qubits only, a natural constraint satisfied by most physical implementations. We also show that *one-way quantum computation* of Raussendorf and Briegel (*Physical Review Letters*, 86:5188–5191, 2001), a universal quantum computation scheme with promising physical implementations, can be efficiently simulated by a randomized algorithm if its quantum resource is derived from a small-treewidth graph.

-ph/0511069v7 12 Jul 2009

**Markov, Shi arXiv:quant-ph/0511069**

Treat tensor networks as graphical models

# TN and graphical model

## Simulation of low-depth quantum circuits as complex undirected graphical models

Sergio Boixo,<sup>1</sup> Sergei V. Isakov,<sup>1</sup> Vadim N. Smelyanskiy,<sup>1</sup> and Hartmut Neven<sup>1</sup>

<sup>1</sup>*Google Inc., Venice, CA 90291, USA*

(Dated: January 23, 2018)

Near term quantum computers with a high quantity (around 50) and quality (around 0.995 fidelity for two-qubit gates) of qubits will approximately sample from certain probability distributions beyond the capabilities of known classical algorithms on state-of-the-art computers, achieving the first milestone of so-called quantum supremacy. This has stimulated recent progress in classical algorithms to simulate quantum circuits. Classical simulations are also necessary to approximate the fidelity of multiqubit quantum computers using cross entropy benchmarking. Here we present numerical results of a novel classical simulation algorithm to calculate output probabilities of universal random circuits with more qubits and depth than previously reported. For example, circuits with  $5 \times 9$  qubits of depth 40,  $7 \times 8$  qubits of depth 30, and  $10 \times (\kappa > 10)$  qubits of depth 19 are all easy to sample by calculating around one thousand measurements in a single workstation. Cross entropy benchmarking with around one million measurements for these circuits is now also possible in a computer cluster. The algorithm is related to the “Feynman path” method to simulate quantum circuits. For low-depth circuits, the algorithm scales exponentially in the depth times the smaller lateral dimension, or the treewidth, as explained in Boixo et. al. [1], and therefore confirms the bounds in that paper. In particular, circuits with  $7 \times 7$  qubits and depth 40 remain currently out of reach. Follow up work on a supercomputer environment will tighten this bound.

- Treat tensor networks as graphical models
- Treat tensor network contraction as node elimination in graphical models
- Use heuristics (e.g. QuickBB) for finding an elimination order

# Slicing

## Classical Simulation of Intermediate-Size Quantum Circuits

Jianxin Chen,<sup>1,\*</sup> Fang Zhang,<sup>2,3,†</sup> Cupjin Huang,<sup>2,3</sup> Michael Newman,<sup>2,4</sup> and Yaoyun Shi<sup>2</sup>

<sup>1</sup>*Aliyun Quantum Laboratory, Alibaba Group, Hangzhou, Zhejiang 311121, China*

<sup>2</sup>*Aliyun Quantum Laboratory, Alibaba Group, Bellevue, WA 98004, USA*

<sup>3</sup>*Department of Electrical Engineering and Computer Science,  
University of Michigan, Ann Arbor, MI 48109, USA*

<sup>4</sup>*Department of Mathematics, University of Michigan, Ann Arbor, MI 48109, USA*

(Dated: May 8, 2018)

We introduce a distributed classical simulation algorithm for general quantum circuits, and present numerical results for calculating the output probabilities of universal random circuits. We find that we can simulate more qubits to greater depth than previously reported using the cluster supported by the Data Infrastructure and Search Technology Division of the Alibaba Group. For example, computing a single amplitude of an  $8 \times 8$  qubit circuit with depth 40 was previously beyond the reach of supercomputers. Our algorithm can compute this within 2 minutes using a small portion ( $\approx 14\%$  of the nodes) of the cluster.

Furthermore, by successfully simulating quantum supremacy circuits of size  $9 \times 9 \times 40$ ,  $10 \times 10 \times 35$ ,  $11 \times 11 \times 31$ , and  $12 \times 12 \times 27$ , we give evidence that noisy random circuits with realistic physical parameters may be simulated classically. This suggests that either harder circuits or error-correction may be vital for achieving quantum supremacy from random circuit sampling.

PACS numbers: 03.65.Ud

- Treat tensor networks as graphical models
- Use slow-heuristics (e.g. QuickBB) for find a elimination order
- **Dynamic slicing**



# Partitioning-based contraction order

SciPost Physics

Submission

## Fast counting with tensor networks

S. Kourtis<sup>1\*</sup>, C. Chamon<sup>1</sup>, E. R. Mucciolo<sup>2</sup>, A. E. Ruckenstein<sup>1</sup>

<sup>1</sup> Physics Department, Boston University, Boston, Massachusetts 02215, USA

<sup>2</sup> Department of Physics, University of Central Florida, Orlando, Florida 32816, USA

\* kourtis@bu.edu

October 23, 2019

## Hyper-optimized tensor network contraction

Johnnie Gray<sup>1,2</sup> and Stefanos Kourtis<sup>1,3,4</sup>

<sup>1</sup>Blackett Laboratory, Imperial College London, London SW7 2AZ, United Kingdom

<sup>2</sup>Division of Chemistry and Chemical Engineering, California Institute of Technology, Pasadena, California 91125, USA

<sup>3</sup>Department of Physics, Boston University, Boston, MA, 02215, USA

<sup>4</sup>Institut quantique & Département de physique, Université de Sherbrooke, Québec J1K 2R1, Canada

March 12, 2021

# Amplitudes to samples

## Quantum Supremacy Is Both Closer and Farther than It Appears

Igor L. Markov<sup>1</sup>, Aneeqa Fatima<sup>1</sup>, Sergei V. Isakov<sup>2</sup>, and Sergio Boixo<sup>3</sup>

<sup>1</sup> University of Michigan, 2260 Hayward St, Ann Arbor, MI 48109

<sup>2</sup> Google Inc., 8002 Zürich, Switzerland

<sup>3</sup> Google Inc., Venice, CA, 90291

September 28, 2018

- Small uncorrelated batch (about 10) for one perfect sample
- Frugal sampling

# Approximate simulation using MPSs

PHYSICAL REVIEW X **10**, 041038 (2020)

Featured in Physics

## What Limits the Simulation of Quantum Computers?

Yiqing Zhou<sup>1,2</sup>, E. Miles Stoudenmire<sup>2</sup>, and Xavier Waintal<sup>3</sup>

<sup>1</sup>*Department of Physics, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA*

<sup>2</sup>*Center for Computational Quantum Physics, Flatiron Institute, New York, New York 10010, USA*

<sup>3</sup>*Univ. Grenoble Alpes, CEA, IRIG-Pheligs, 38054 Grenoble, France*



(Received 19 February 2020; revised 22 September 2020; accepted 5 October 2020; published 23 November 2020)

An ultimate goal of quantum computing is to perform calculations beyond the reach of any classical computer. It is therefore imperative that useful quantum computers be very difficult to simulate classically, otherwise classical computers could be used for the applications envisioned for the quantum ones. Perfect quantum computers are unarguably exponentially difficult to simulate: the classical resources required grow exponentially with the number of qubits  $N$  or the depth  $D$  of the circuit. This difficulty has triggered

# Amplitudes to samples

ARTICLE OPEN

## A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware

Benjamin Villalonga<sup>1,2,3</sup>, Sergio Boixo<sup>4</sup>, Bron Nelson<sup>2,5</sup>, Christopher Henze<sup>2</sup>, Eleanor Rieffel<sup>2</sup>, Rupak Biswas<sup>2</sup> and Salvatore Mandrà<sup>2,6\*</sup>

Here we present qFlex, a flexible tensor network-based quantum circuit simulator. qFlex can compute both the exact amplitudes, essential for the verification of the quantum hardware, as well as low-fidelity amplitudes, to mimic sampling from Noisy Intermediate-Scale Quantum (NISQ) devices. In this work, we focus on random quantum circuits (RQCs) in the range of sizes expected for supremacy experiments. Fidelity  $f$  simulations are performed at a cost that is  $1/f$  lower than perfect fidelity ones. We also present a technique to eliminate the overhead introduced by rejection sampling in most tensor network approaches. We benchmark the simulation of square lattices and Google's Bristlecone QPU. Our analysis is supported by extensive simulations on NASA HPC clusters Pleiades and Electra. For our most computationally demanding simulation, the two clusters combined reached a peak of 20 Peta Floating Point Operations per Second (PFLOPS) (single precision), i.e., 64% of their maximum achievable performance, which represents the largest numerical computation in terms of sustained FLOPs and the number of nodes utilized ever run on NASA HPC clusters. Finally, we introduce a novel multithreaded, cache-efficient tensor index permutation algorithm of general application.

npj Quantum Information (2019)5:86

; <https://doi.org/10.1038/s41534-019-0196-1>

- Small **correlated** batch (about 10) for one perfect sample
- Frugal sampling

# Combining partition-order and Frugal sampling

## Classical Simulation of Quantum Supremacy Circuits

Cupjin Huang,<sup>1</sup> Fang Zhang,<sup>2</sup> Michael Newman,<sup>3</sup> Junjie Cai,<sup>4</sup>  
Xun Gao,<sup>1</sup> Zhengxiong Tian,<sup>5</sup> Junyin Wu,<sup>4</sup> Haihong Xu,<sup>5</sup> Huanjun Yu,<sup>5</sup>  
Bo Yuan,<sup>6</sup> Mario Szegedy,<sup>1</sup> Yaoyun Shi<sup>1</sup>, Jianxin Chen<sup>1</sup>

<sup>1</sup>Alibaba Quantum Laboratory,

Alibaba Group USA, Bellevue, WA 98004, USA

<sup>2</sup>Department of Electrical Engineering and Computer Science,

University of Michigan, Ann Arbor, MI 48109, USA

<sup>3</sup>Departments of Physics and Electrical and Computer Engineering,

Duke University, Durham, NC 27708, USA

<sup>4</sup>Alibaba Cloud Intelligence,

Alibaba Group USA, Bellevue, WA 98004, USA

<sup>5</sup>Alibaba Cloud Intelligence,

Alibaba Group, Hangzhou, Zhejiang 310000, China

<sup>6</sup>Alibaba Infrastructure Service,

Alibaba Group, Hangzhou, Zhejiang 310000, China

- Small batch (64) for one perfect sample
- Frugal sampling
- Single tensor-network contraction for each batch



# Simulated annealing for order/slicing finding

## Recursive Multi-Tensor Contraction for XEB Verification of Quantum Circuits

Gleb Kalachev,<sup>1,2,\*</sup> Pavel Panteleev,<sup>1,2,†</sup> and Man-Hong Yung<sup>1,3,‡</sup>

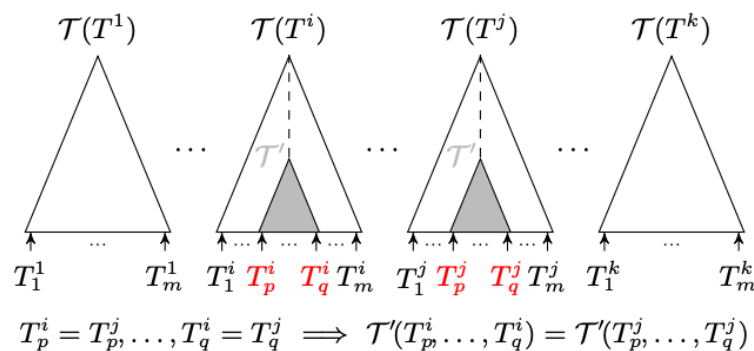
<sup>1</sup>*Huawei 2012 Lab*

<sup>2</sup>*Lomonosov Moscow State University.*

<sup>3</sup>*Institute for Quantum Science and Engineering, and Department of Physics,  
Southern University of Science and Technology, Shenzhen, 518055, China*

(Dated: August 13, 2021)

The computational advantage of noisy quantum computers have been demonstrated by sampling the bitstrings of quantum random circuits. An important issue is how the performance of quantum devices could be quantified in the so-called “supremacy regime”. The standard approach is through the linear cross entropy (XEB), where the theoretical value of the probability is required for each bitstring. However, the computational cost of XEB grows exponentially. So far, random circuits of the 53-qubit Sycamore chip was verified up to 10 cycles of gates only; the XEB fidelities of deeper circuits were approximated with simplified circuits instead. Here we present a multi-tensor contraction algorithm for speeding up the calculations of XEB of quantum circuits, where the computational cost can be significantly reduced through a recursive manner with some form of memoization. As a demonstration, we analyzed the experimental data of the 53-qubit Sycamore chip and obtained the exact values of the corresponding XEB fidelities up to 16 cycles using only moderate computing resources (few GPUs). If the algorithm was implemented on the Summit supercomputer, we estimate that for the 20-cycles supremacy circuits, it would only cost 7.5 days, which is several orders of magnitudes lower than previously estimated in the literature.



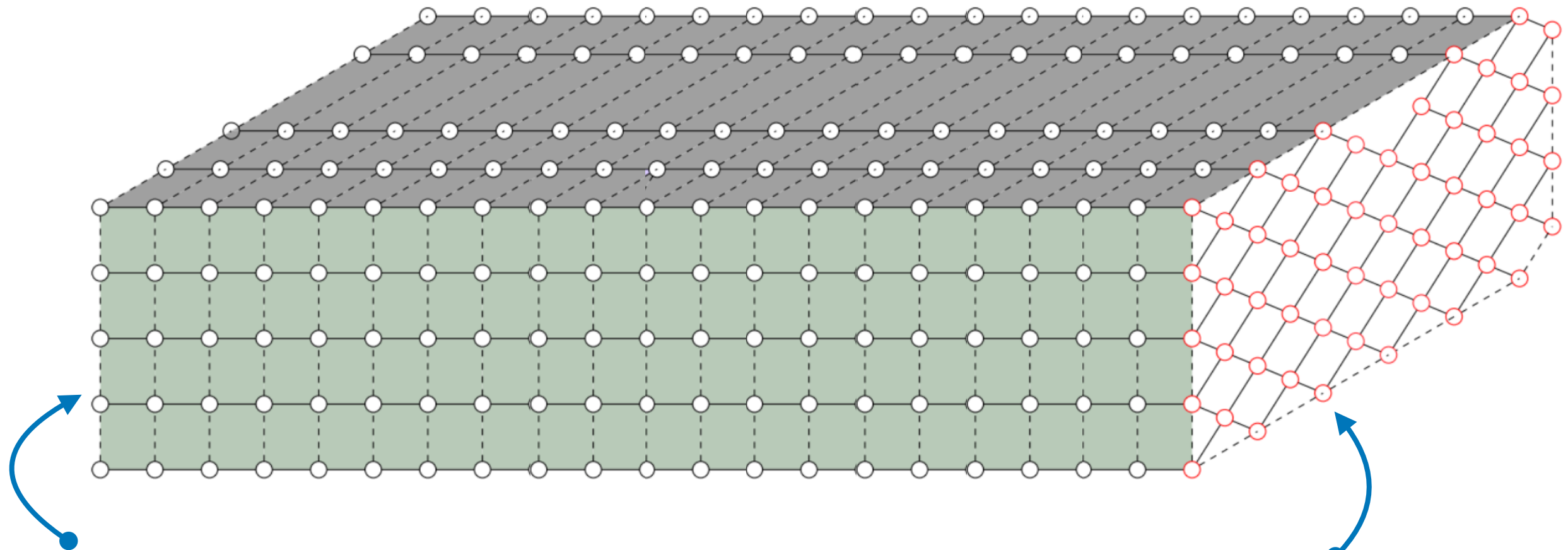
**Local moves**

$m$	$k$	Contraction cost		Efficiency		Time (days or years)		
		1 amp (S)	$k$ amps (M)	S	M	S	M	gain
12	0.5M	$1.8 \cdot 10^{13}$	$2.8 \cdot 10^{17}$	61%	43%	94 d	4.3 d	22x
14	0.5M	$1.0 \cdot 10^{14}$	$1.9 \cdot 10^{18}$	60%	60%	538 d	21 d	25x
16	2M	$8.9 \cdot 10^{16}$	$1.4 \cdot 10^{19}$	63%	48%	5000 y	0.5 y	10000x

(b) Verification complexity

**Verifying the Sycamore circuits up to 16 cycles**






# The big-batch method








**Left boundary condition:**

**Right boundary condition:**

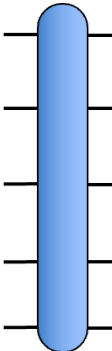
**Product state**

$|0\rangle$    
 $|0\rangle$    
 $|0\rangle$    
 $|0\rangle$    
 $|0\rangle$  



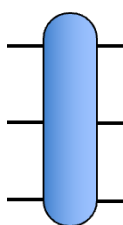
**Single-amplitude**

**Full-amplitude**



**Big-batch**

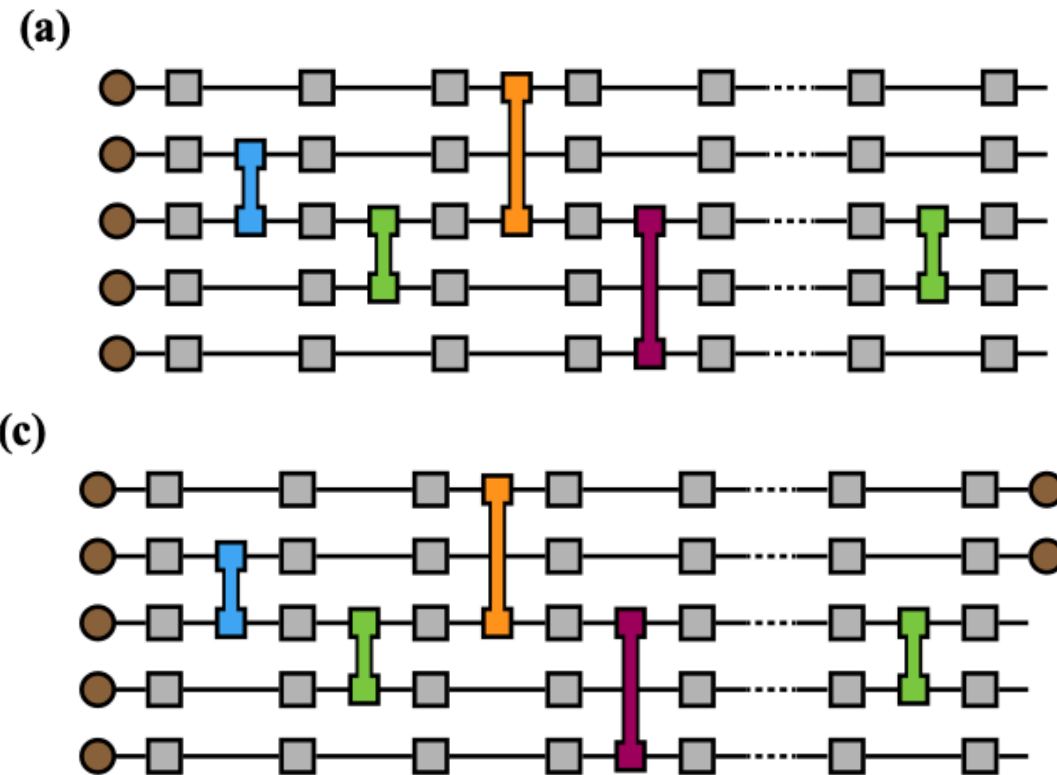
  
  


F. Pan and PZ, arXiv:2103.03074 (2021)

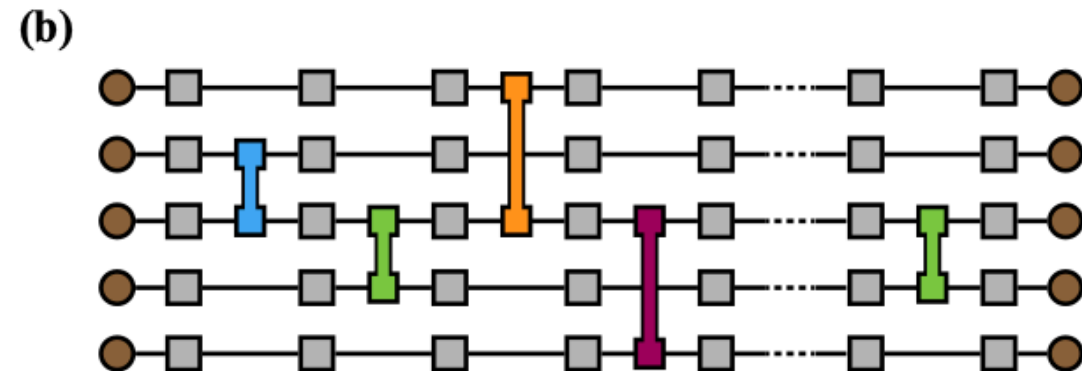
F. Pan and PZ, Phys. Rev. Lett. 128, 030501 (2022)

# 4 kinds of boundary conditions

**Full-amplitude**



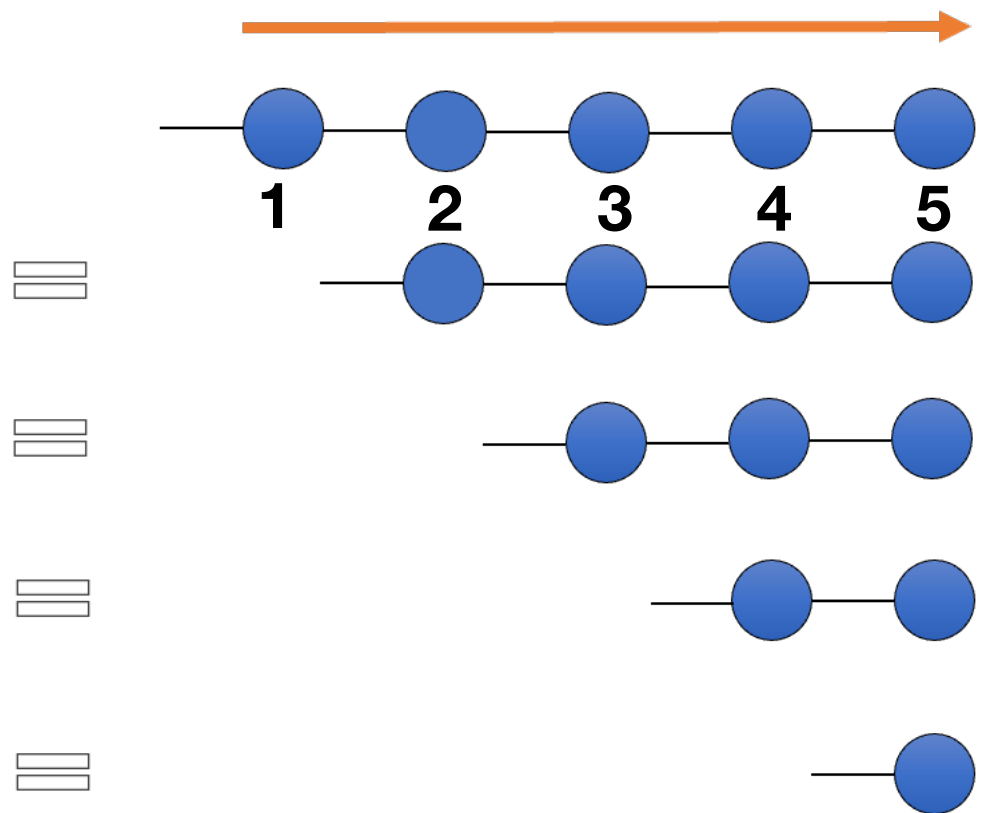
**Single-amplitude**



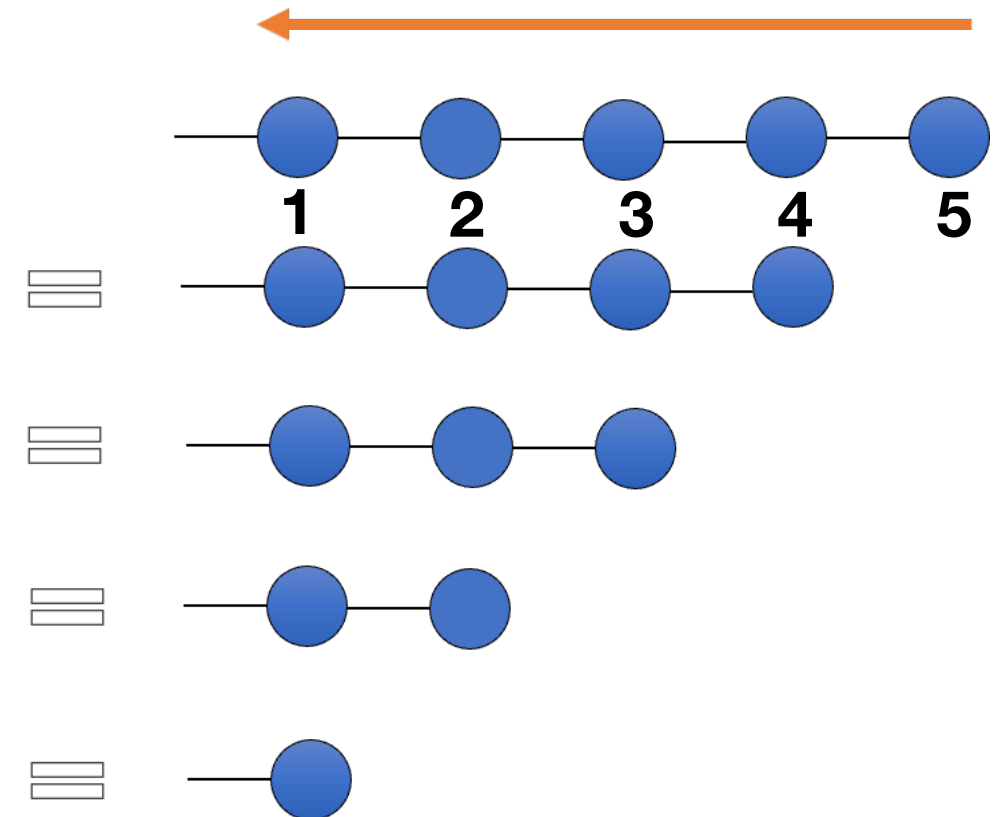
**Batch-amplitude**



# Contraction order

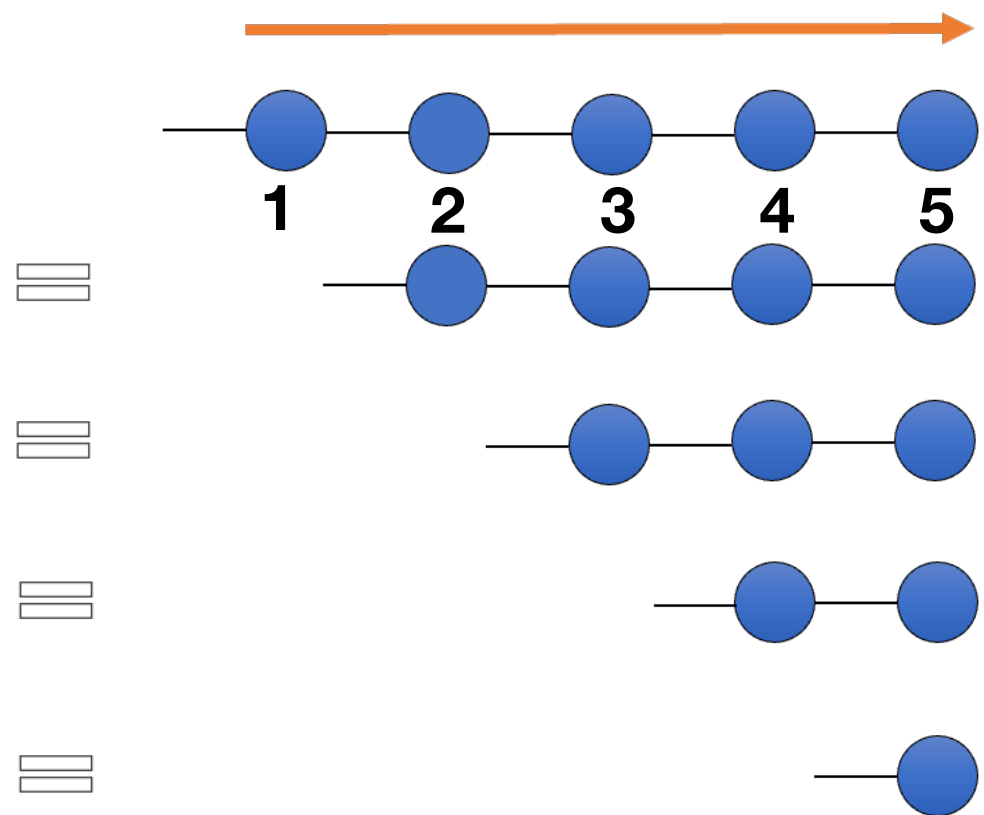


**Time complexity:**  $d^3 + d^3 + d^3 + d^2$

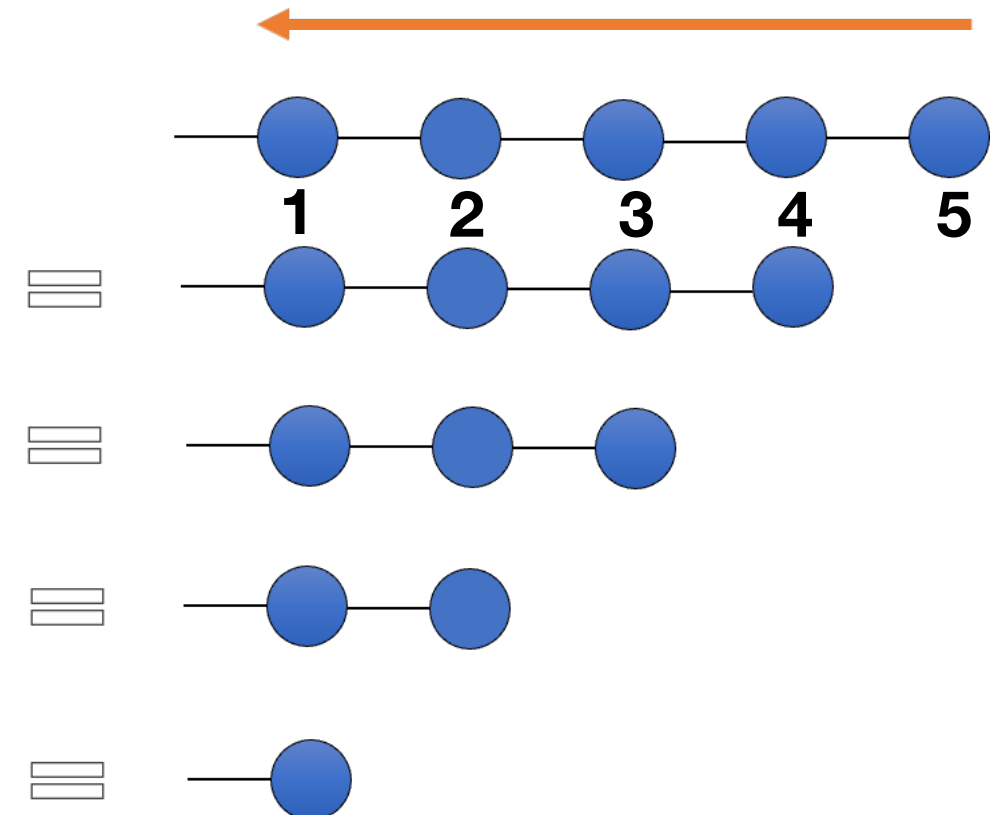


**Time complexity:**  $d^2 + d^2 + d^2 + d^2$

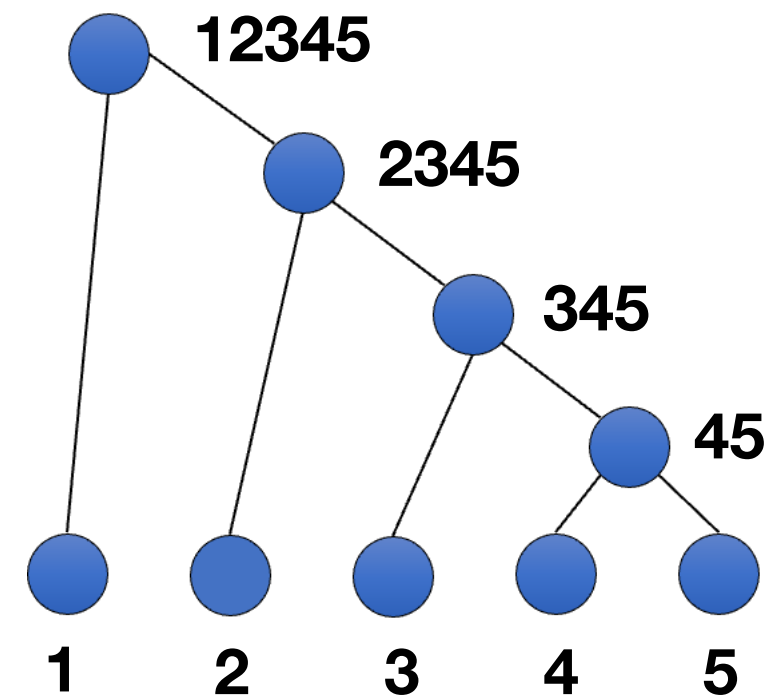
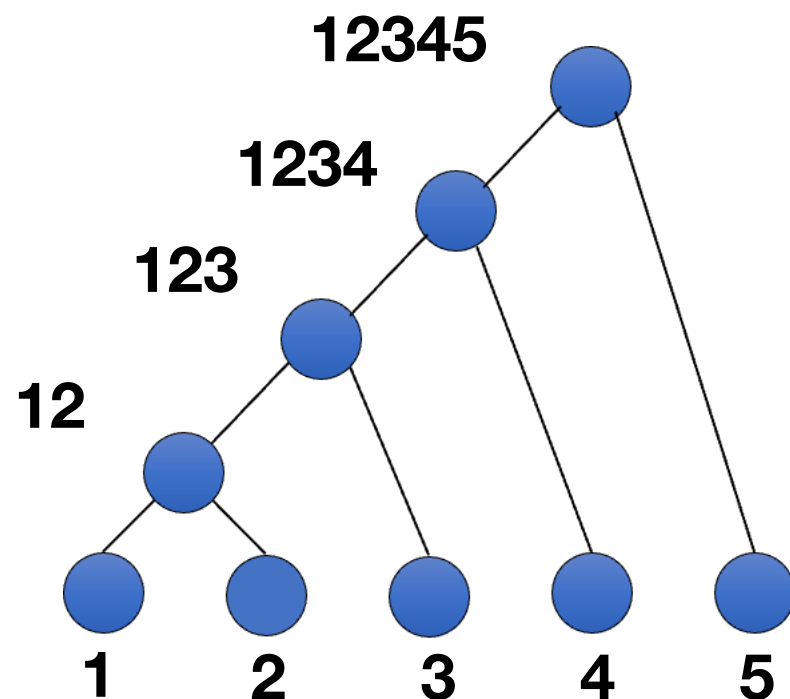
# Contraction order and contraction tree



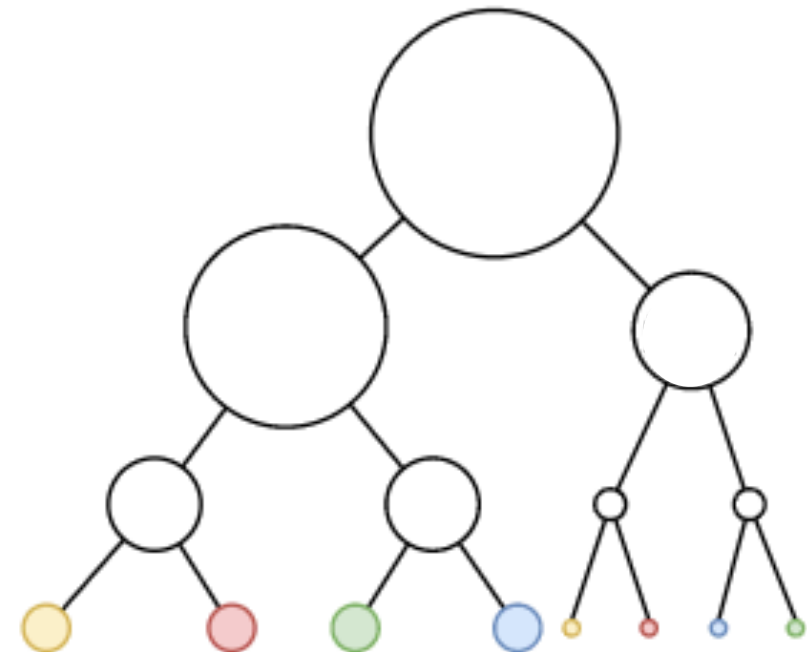
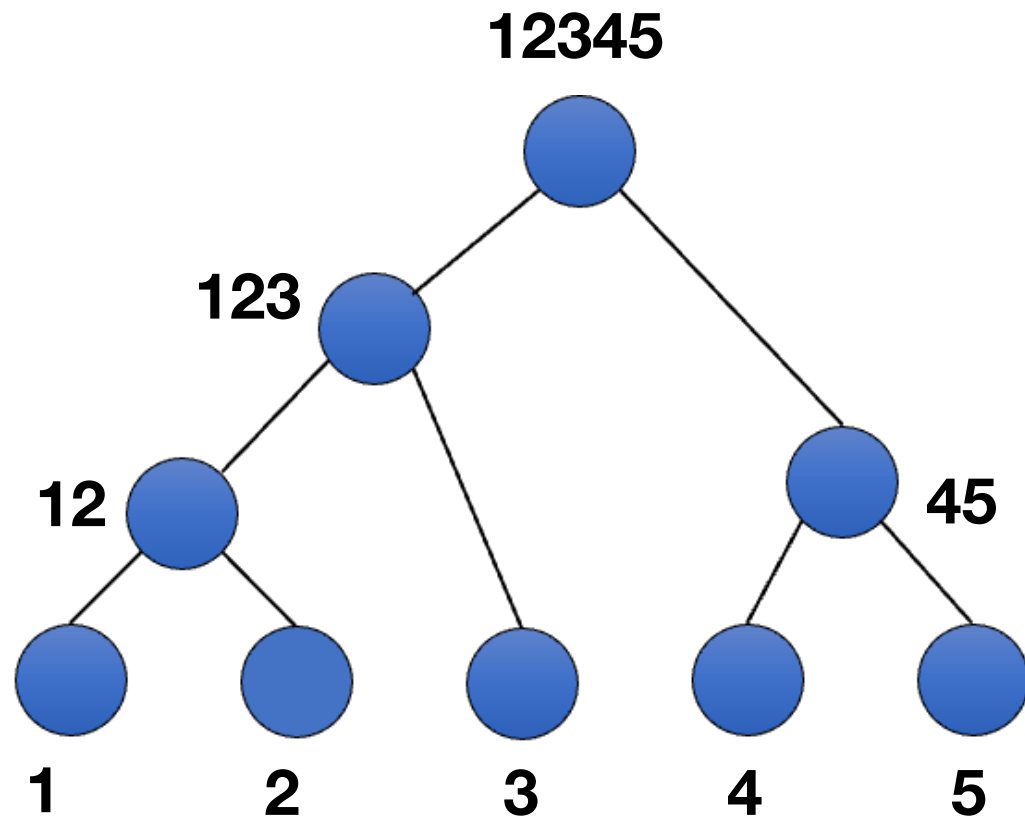
**Time complexity:**  $d^3 + d^3 + d^3 + d^2$



**Time complexity:**  $d^2 + d^2 + d^2 + d^2$



# Complexity of the contraction tree



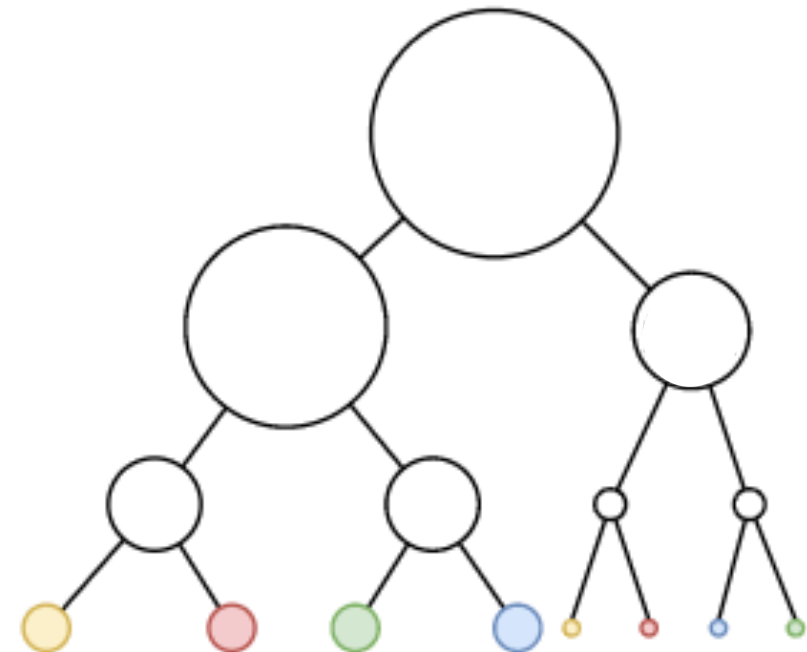
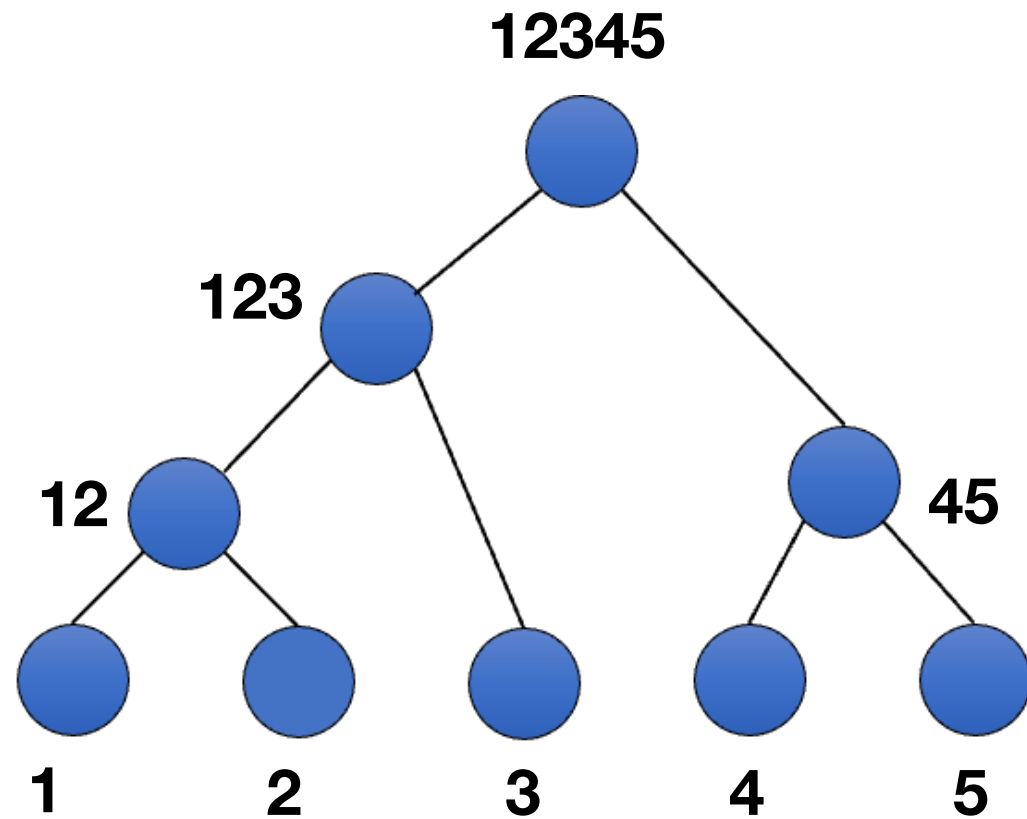
Space complexity:

- size of the largest tensor in the contraction tree
- Optimal space complexity:  $\exp(W)$ ,  $W$  is the tree width of the graph

Time complexity:

- Time complexity of each node is the product of dimensions associated with edges
- Time complexity of the contraction tree is the time complexity of each node (usually **dominated by the largest one**)

Find a contraction tree to minimize the complexity

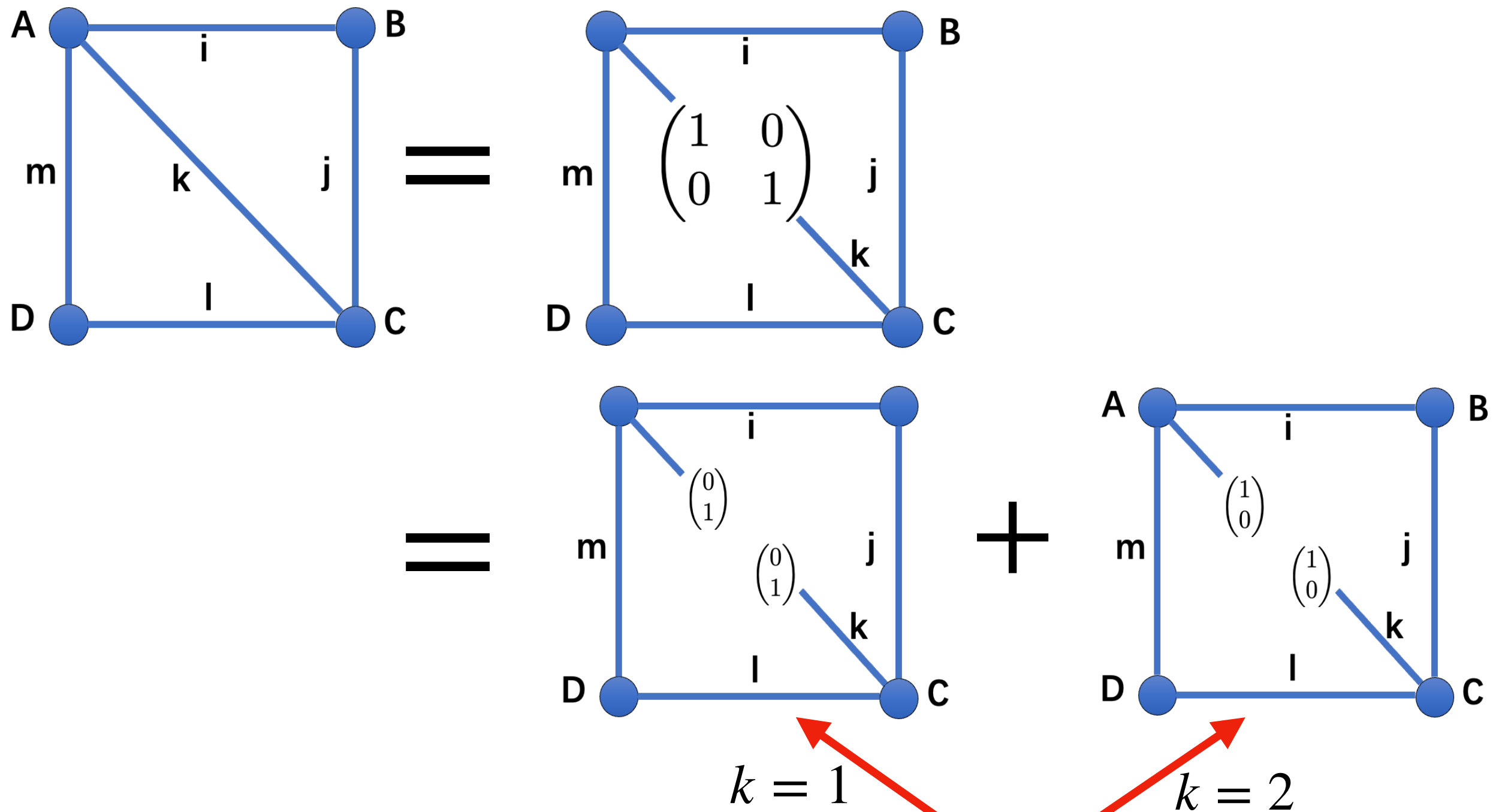


Greedy algorithms

Partitioning based algorithms [Gray/Kourtis 2021]

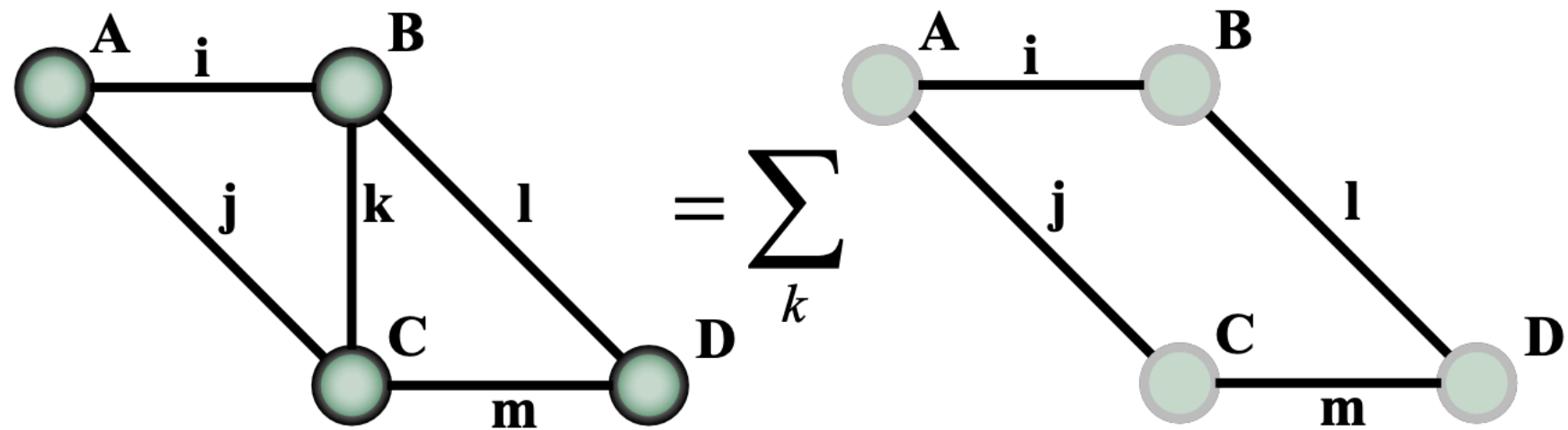
Simulated-annealing-based algorithm [Kalachev et al 2021]

# Slicing



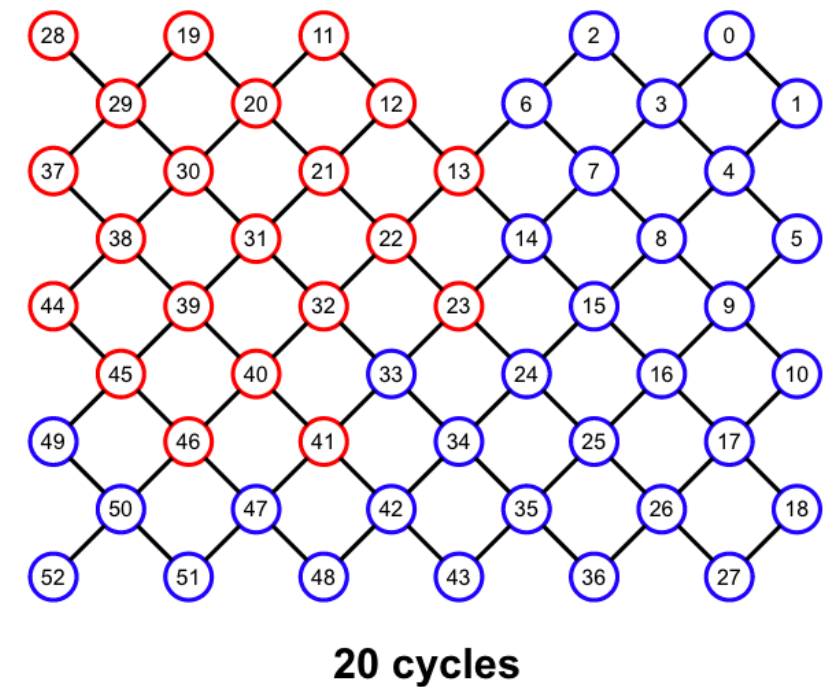
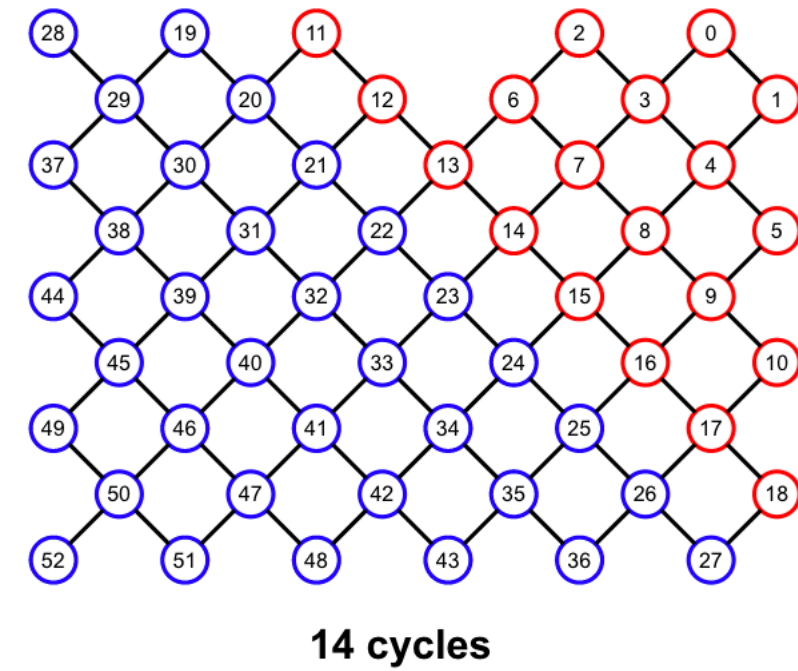
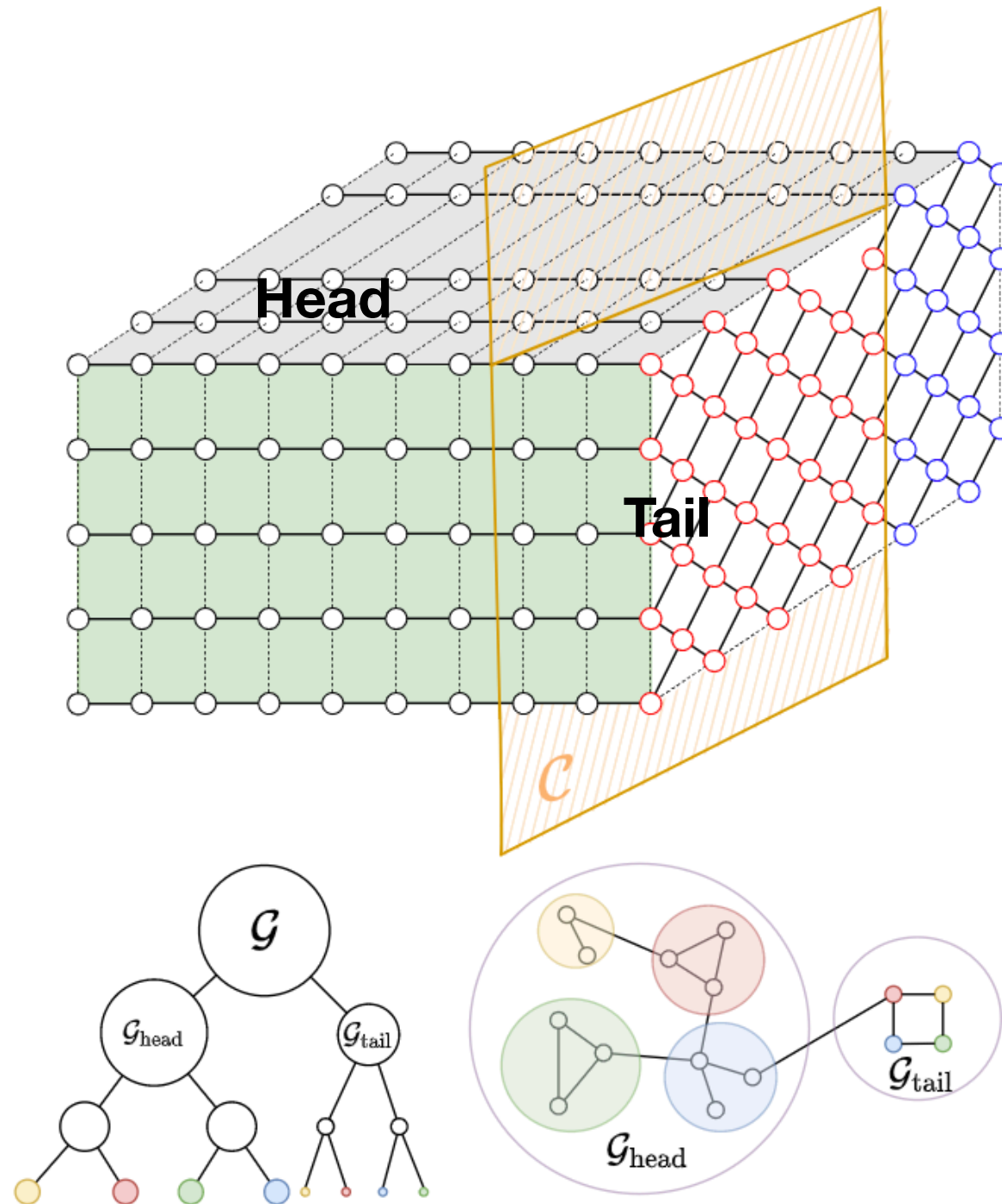
$$\sum_i \sum_j \sum_k \sum_l \sum_m A_{ikm} B_{ij} C_{jkl} D_{lm} = \sum_{k=1}^2 \left( \sum_i \sum_j \sum_l \sum_m A_{ikm} B_{ij} C_{jkl} D_{lm} \right)$$

# Slicing



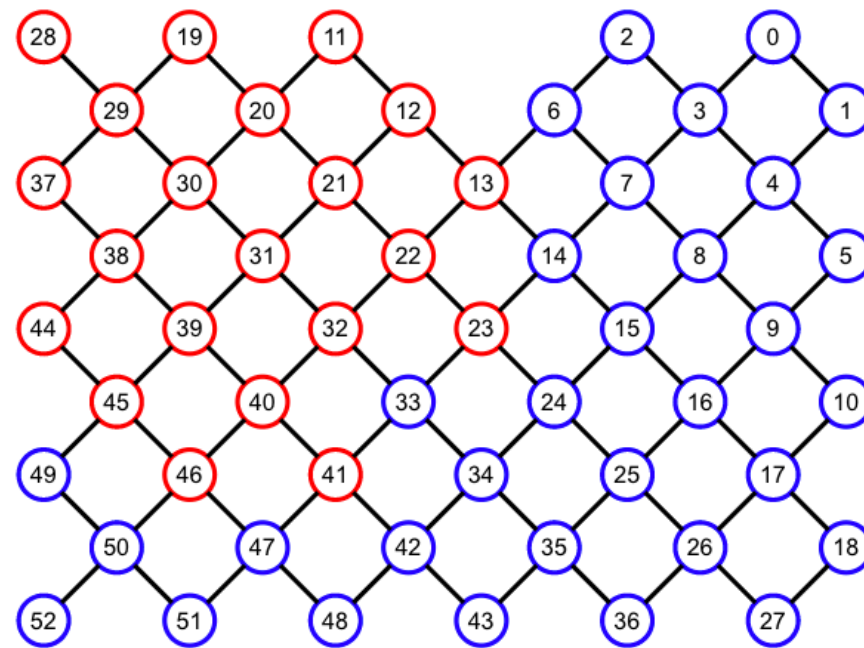
The index  $k$  is sliced and the contraction of the original tensor network becomes summation of  $d_k$  sub-tasks

# big-batch of amplitudes using the big-head algorithm



# Correlated bitstrings

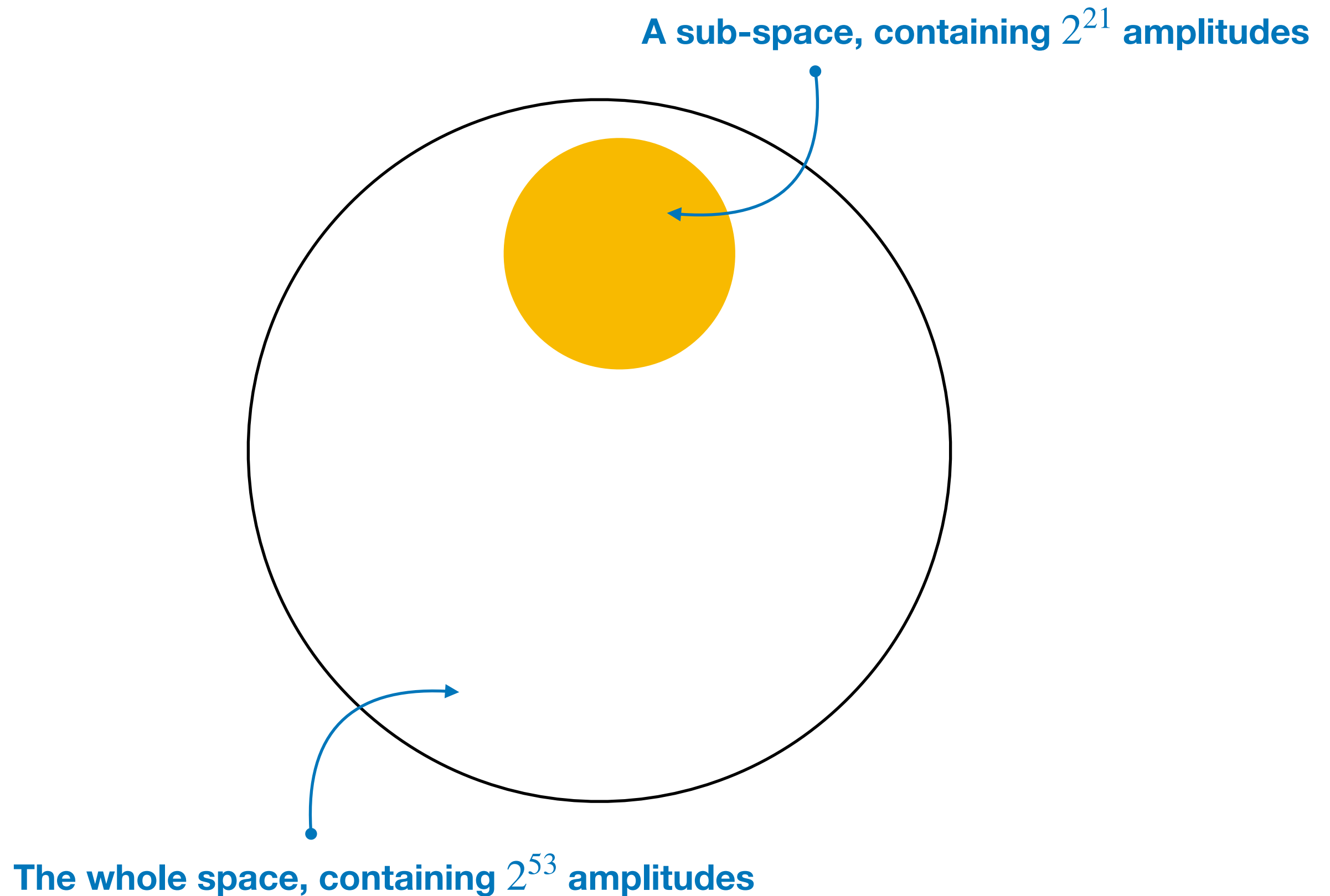
bitstring	amplitude	probability
00000000000000000000000001000000110100000010000100000000	$-2.97 \times 10^{-8} + 2.06 \times 10^{-8}i$	$1.31 \times 10^{-15}$
000000000000000000000000000000000000001000010000000	$1.50 \times 10^{-8} + 3.85 \times 10^{-9}i$	$2.39 \times 10^{-16}$
000000000000111000001111100001111100001111100110000000	$-3.17 \times 10^{-9} - 5.45 \times 10^{-9}i$	$3.97 \times 10^{-17}$
000000000000111000001111100001111100001111000101000000	$-1.89 \times 10^{-10} + 3.13 \times 10^{-9}i$	$9.86 \times 10^{-18}$
0000000000000000000000000010000000000111100010000000	$8.07 \times 10^{-10} + 4.35 \times 10^{-10}i$	$8.41 \times 10^{-19}$



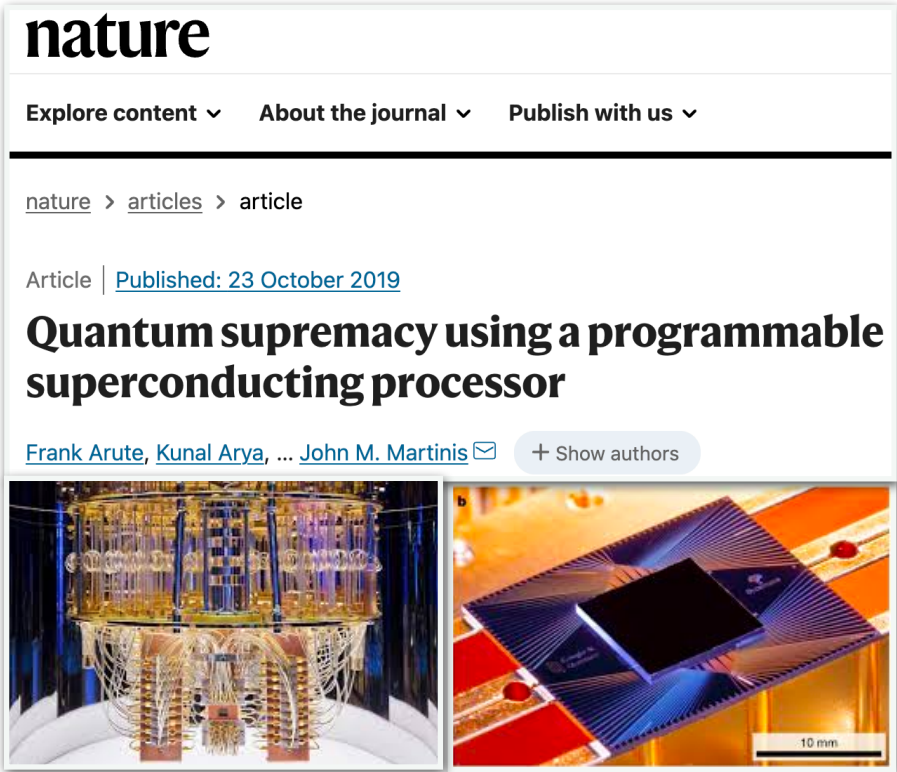
**20 cycles**



# A big batch of amplitudes



# Big-batch → full amplitude simulation



43-qubit Sycamore circuit



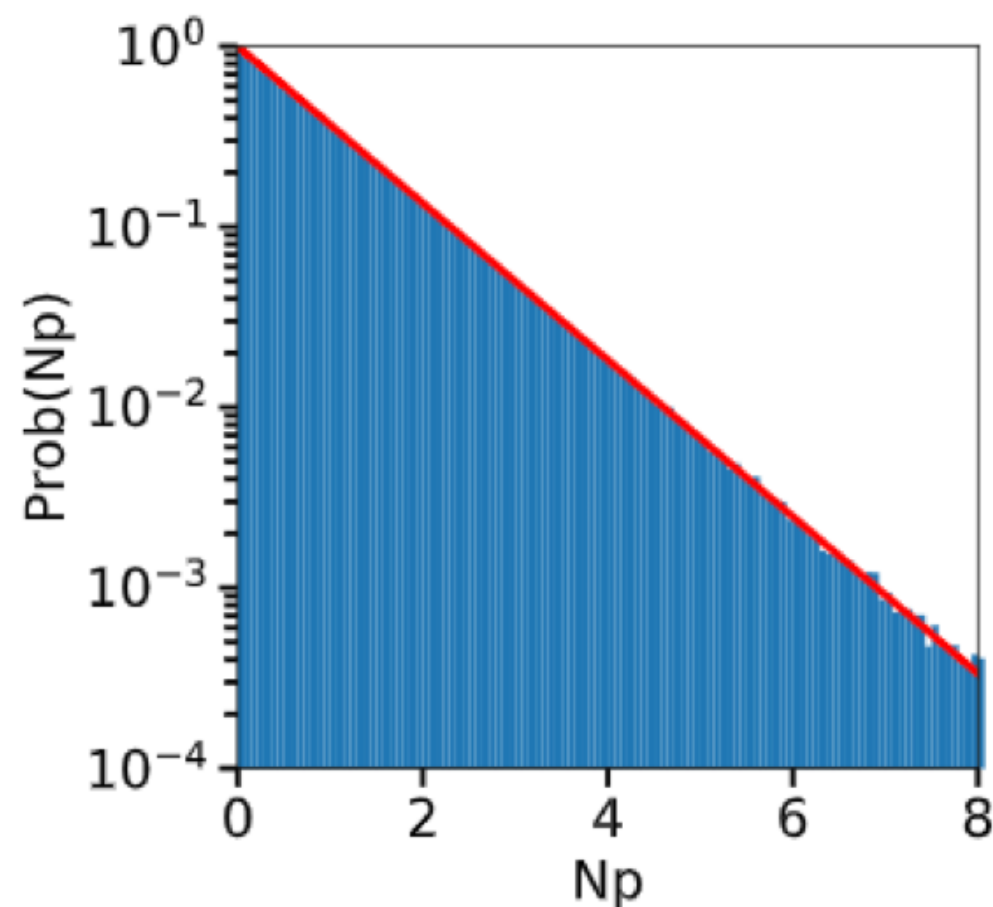
# qubits to simulate

Google's paper:	Julich Supercomputer 100,000 cores 250T meomory	
	1 GPU	
Big-batch algorithm	Full amplitudes 49 qubits	
	Taihu light supercomputer [Li et al. IEEE PDS 2019]	
Big-batch algorithm	50 qubits	
	100 GPU's	

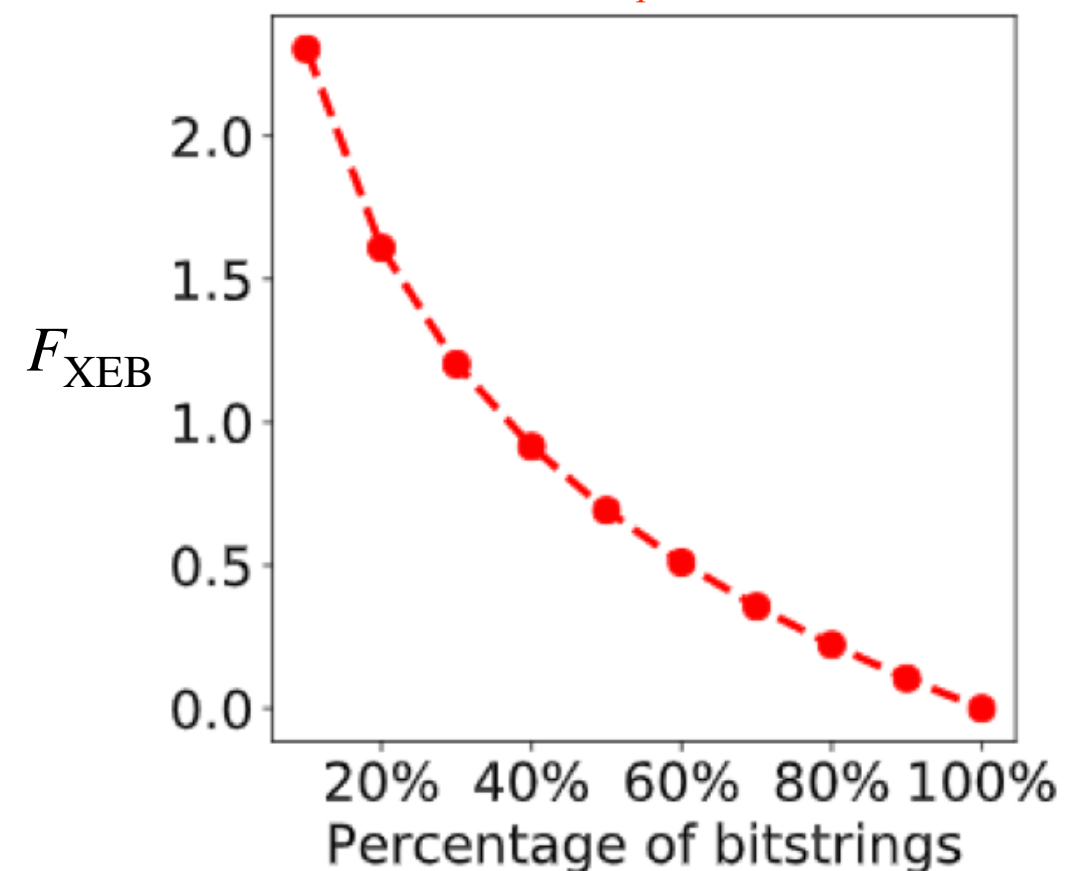
# Spoofing the Linear Cross Entropy Benchmark (XEB)

For supremacy circuits with  
53 qubits and 20 cycles

$$\begin{aligned} F_{\text{XEB}} &= 2^n \sum_{\mathbf{s} \in \{1,0\}^n} q(\mathbf{s}) p_U(\mathbf{s}) - 1 \\ &= 2^n \langle p_U(\mathbf{s}) \rangle_q - 1 \\ &\approx \frac{2^n}{m} \sum_{\mathbf{s} \sim q} p_U(\mathbf{s}) - 1 \end{aligned}$$



$2^{21}$  exact probabilities  $P_U(\mathbf{s})$   
verifying the Porter-Thomas distribution



**Post-sampling**  
Selecting 1 million bitstrings, XEB=**0.739**

# Computational cost

	Computational complexity	Computation hardware	Time
Google [Arute et. al., 2019] ( <b>Estimated</b> )	— — —	Summit Super Computer	10,000 Years
IBM [Pednault et. al., 2019] ( <b>Estimated</b> )	$1.18 \times 10^{21}$	Summit Super Computer (all disks)	2.5 days
Alibaba [Huang et. al., 2020] ( <b>Estimated</b> )	$1.33 \times 10^{22}$	Summit Super Computer	20 days
Ours [arXiv:2103.03074] ( <b>Computed</b> ) Correlated sampling	$4.51 \times 10^{18}$	60 <b>NVIDIA GPUs</b>	5 days

# Parallel Computation with GPUs



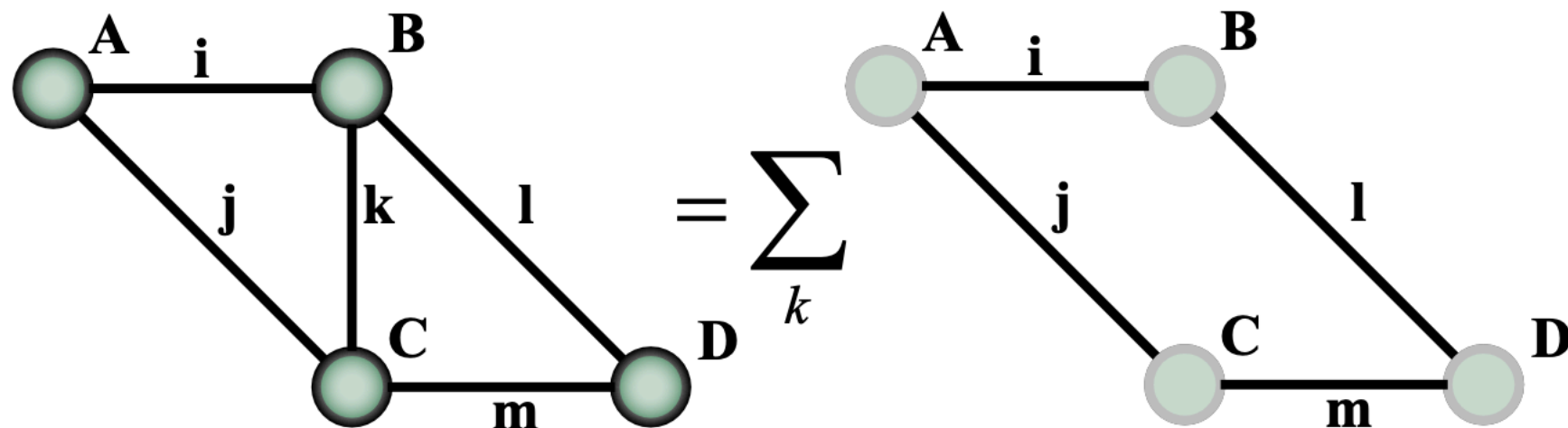
**CPU**

- Intel CPUs: 3 GFlops / core , 32 Cores



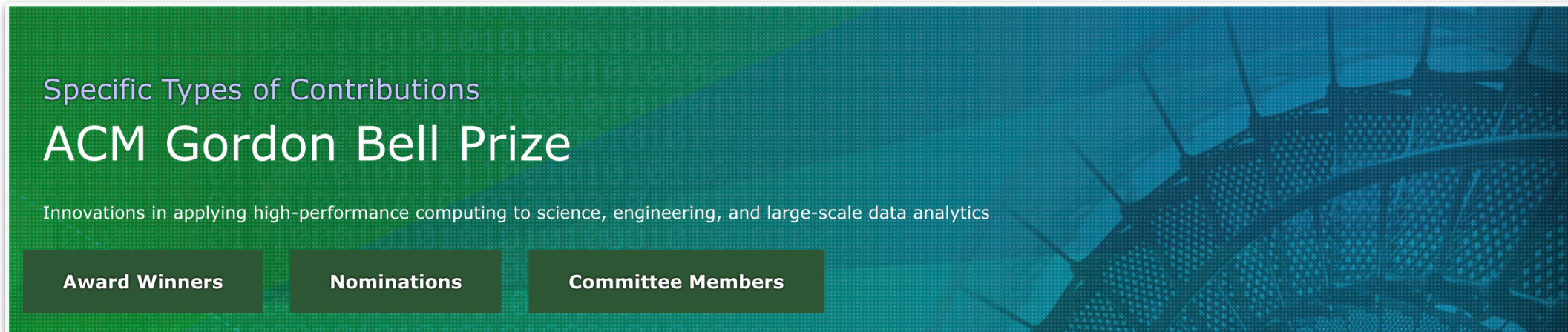
**GPU**

- NVIDIA Tesla V100 FP32 performance: 14.13 TFLOPS
- NVIDIA Tesla A100 FP32 performance: 19.5 TFLOPs





# Implementing the big-batch approach on a super computer reduces the computation time from 5 days to 314 seconds

A banner for the ACM Gordon Bell Prize. The background is a dark green and blue abstract pattern. The text is white and green.

Specific Types of Contributions

## ACM Gordon Bell Prize

Innovations in applying high-performance computing to science, engineering, and large-scale data analytics

[Award Winners](#) [Nominations](#) [Committee Members](#)

## Closing the “Quantum Supremacy” Gap: Achieving Real-Time Simulation of a Random Quantum Circuit Using a New Sunway Supercomputer

Yong (Alexander) Liu<sup>1,3</sup>, Xin (Lucy) Liu<sup>1,3</sup>, Fang (Nancy) Li<sup>1,3</sup>, Haohuan Fu<sup>2,3</sup>, Yuling Yang<sup>1,3</sup>  
Jiawei Song<sup>1,3</sup>, Pengpeng Zhao<sup>1,3</sup>, Zhen Wang<sup>1,3</sup>, Dajia Peng<sup>2,3</sup>, Huarong Chen<sup>1,3</sup>  
Chu Guo<sup>4</sup>, Heliang Huang<sup>4</sup>, Wenzhao Wu<sup>3</sup>, Dexun Chen<sup>2,3</sup>

1. Zhejiang Lab, Hangzhou, China

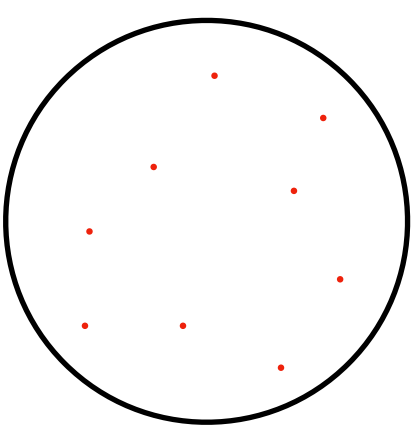
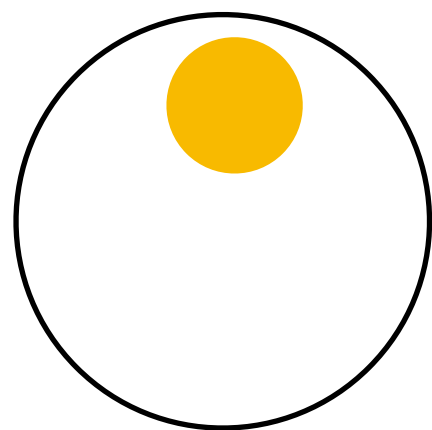
2. Tsinghua University, Beijing, China

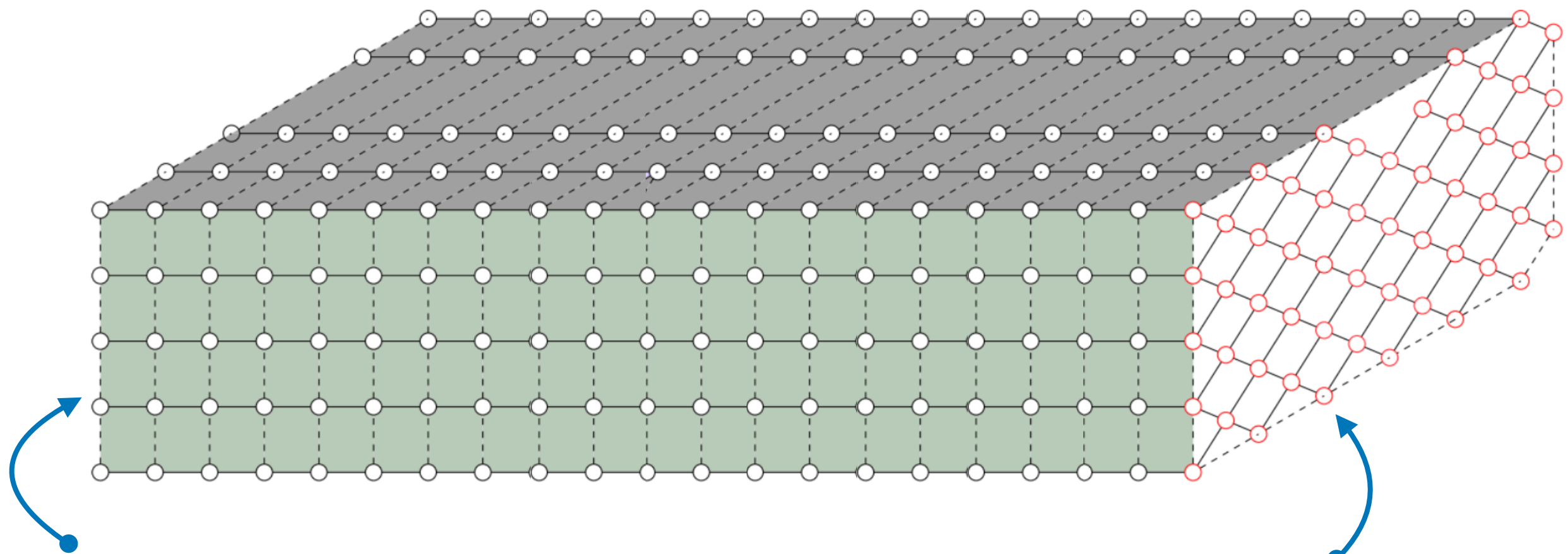
3. National Supercomputing Center in Wuxi, Wuxi, China

4. Shanghai Research Center for Quantum Sciences, Shanghai China

# Differences

Big-batch simulation + sampling	Sycamore hardware sampling
Correlated samples	Uncorrelated samples
Exact sub-space simulation	Noisy full-space simulation

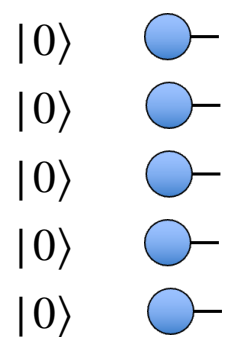




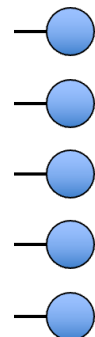
**Left boundary condition:**

**Right boundary condition:**

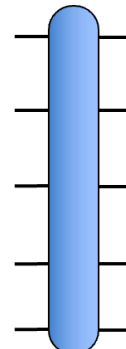
**Product state**



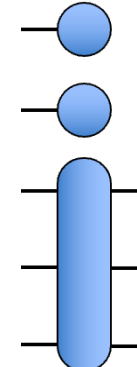
**Single-amplitude**



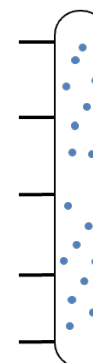
**Full-amplitude**



**Big-batch**



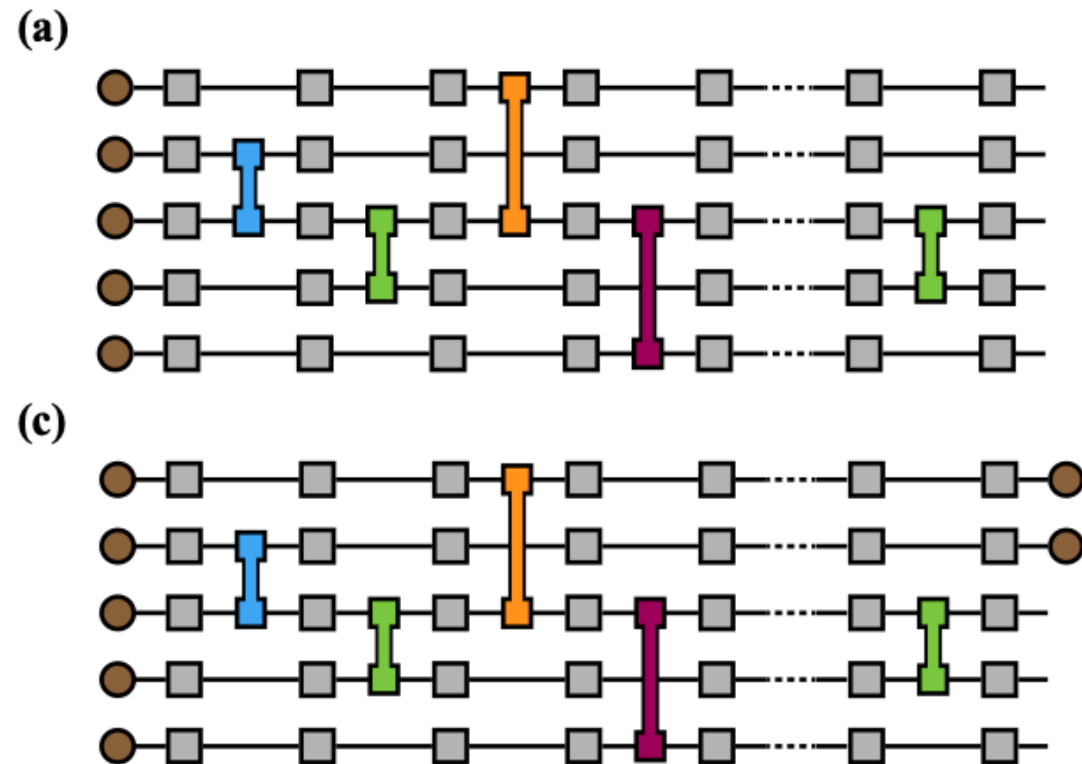
**Sparse state**





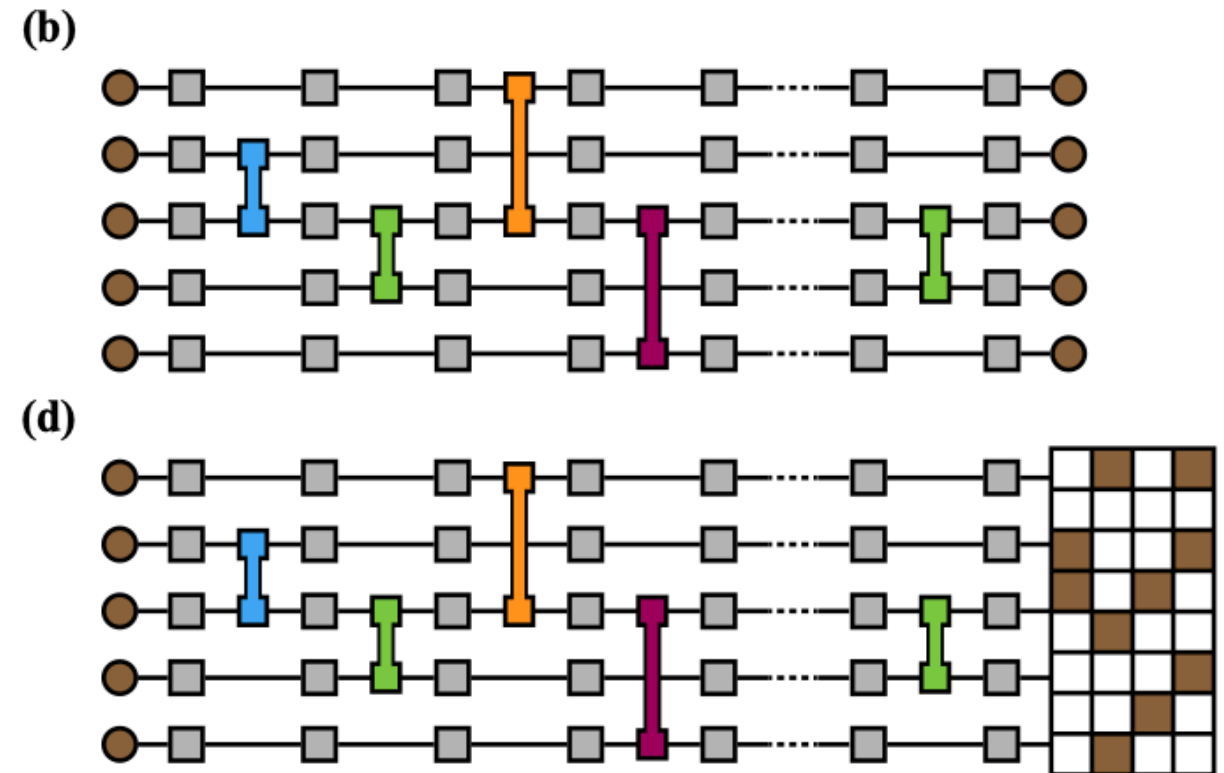
# 4 kinds of boundary conditions

**Full-amplitude**



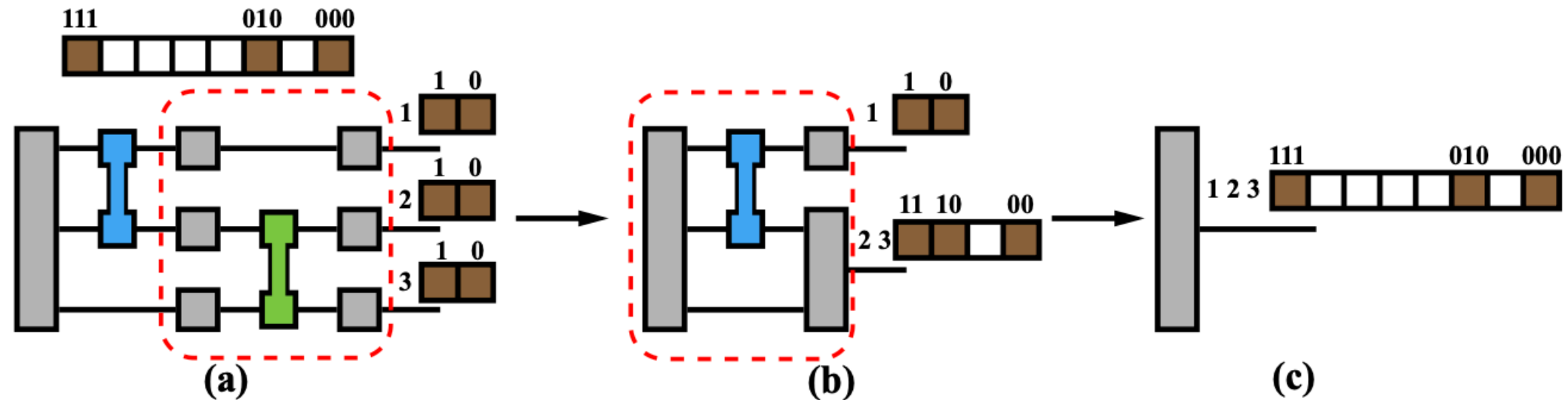
**Batch-amplitude**

**Single-amplitude**



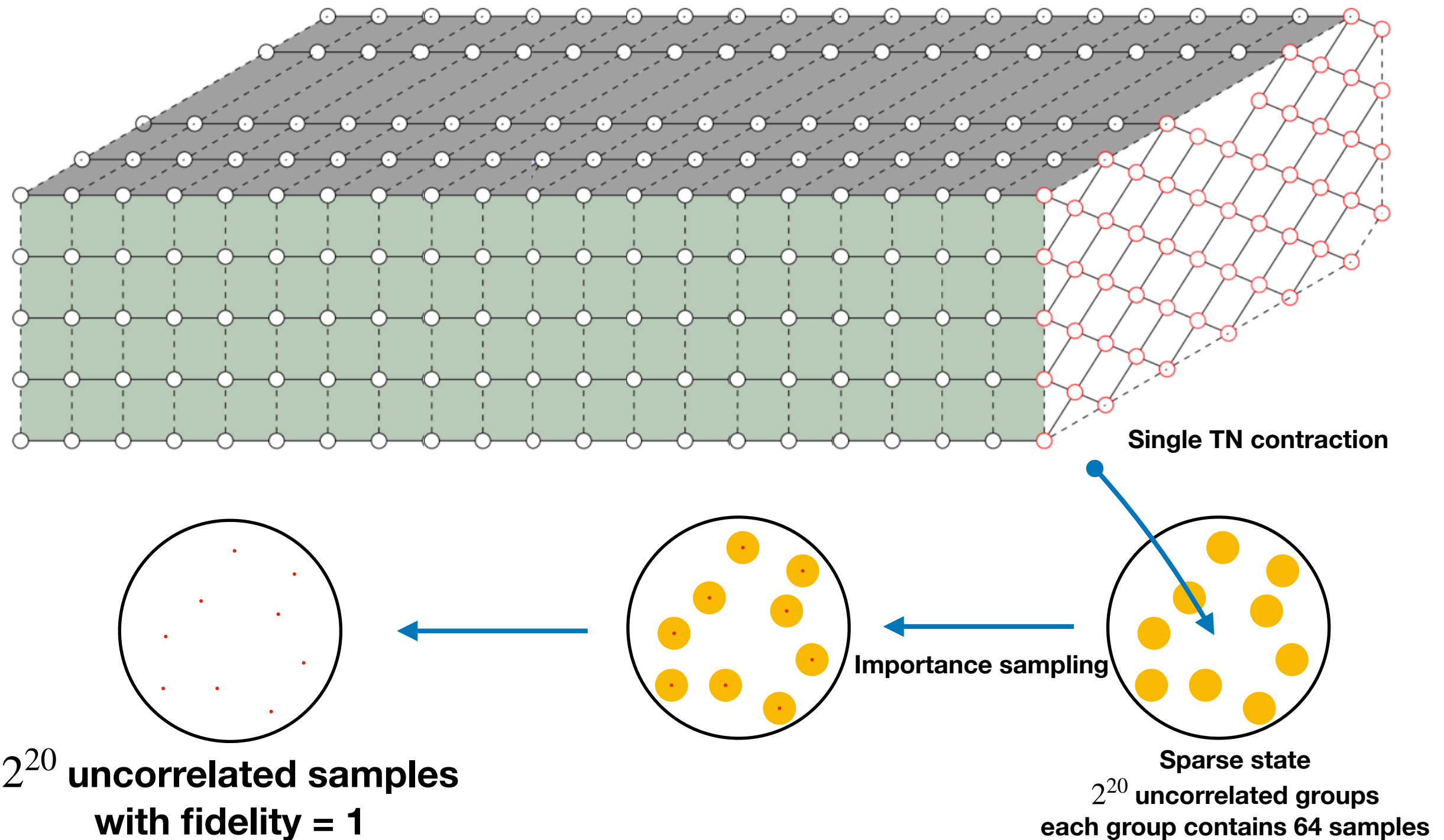
**Sparse-state**

# Contractions with the sparse state



- We want only amplitudes for bitstrings  $\{111, 010, 000\}$
- Contracting the red part in (a) merges qubits 2 and 3, resulting to (b), where only  $\{11, 10, 00\}$  sub-bitstrings are necessary.
- Contracting the red part in (b) gives the final results containing only the required bitstrings.

# Solving the uncorrelated sampling problem



# From group to a representative bitstring

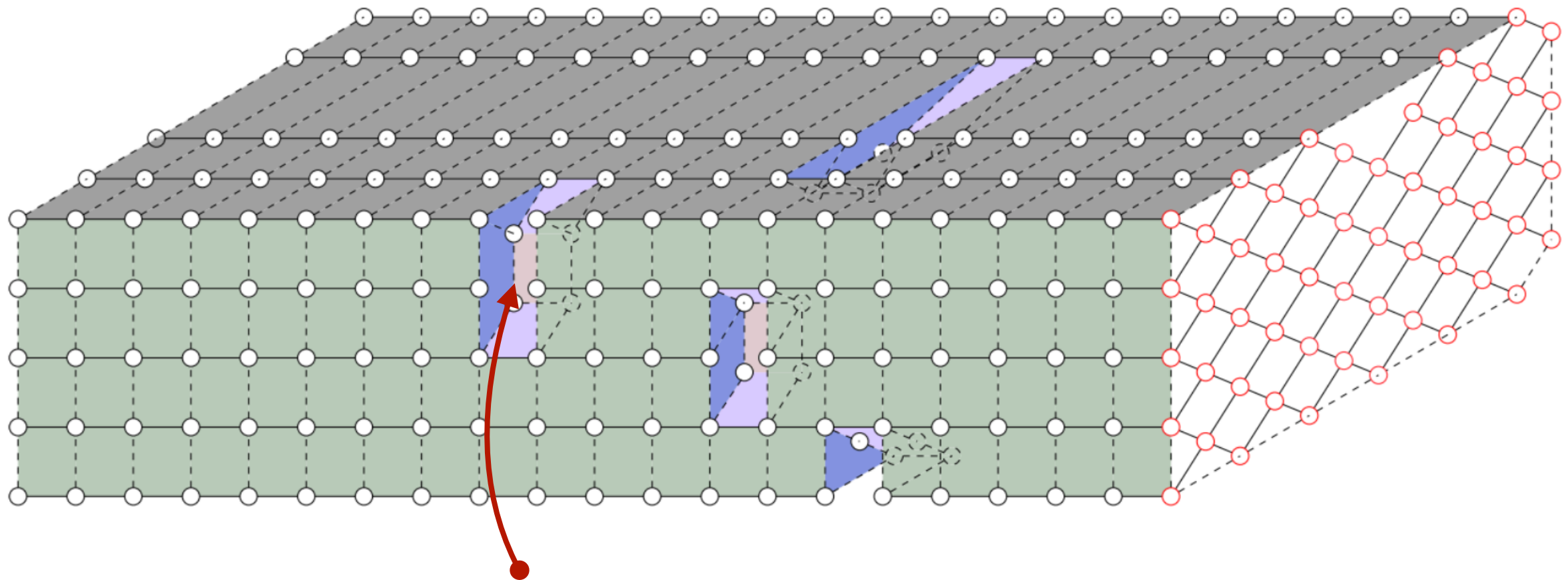


- Each group contains 64 bitstrings  $\{S_i | i = 1, \dots, 64\}$ , with probability  $\{P_i = |\psi_i|^2\}$  associated with the sparse state  $\psi$ .
- Loop over 64 bitstrings from the  $s_1$ . At the  $i$ -th bitstring, replace the current bitstring by  $s_i$  with probability  $\min(1, P_i/P_{\text{current}})$ .
  - It's nothing but Metropolis-Hasting
  - Satisfies Detailed Balance
  - 64 is large enough for the PT distribution

# Trading off fidelity for computational complexity

1. Using approximate state, e.g. MPS
2. Sampling from a mixed distribution ( a small portion from true distribution + a large part from pure noise)
  - Noisy state:  $\rho = f|\psi\rangle\langle\psi| + (1 - f)\frac{1}{2^n}$
  - 2000 bit strings with from  $|\psi\rangle\langle\psi|$  and 998000 from uniform distribution
3. Summing over a fraction of paths in the path-integral representation.

# Trading off fidelity for computational complexity

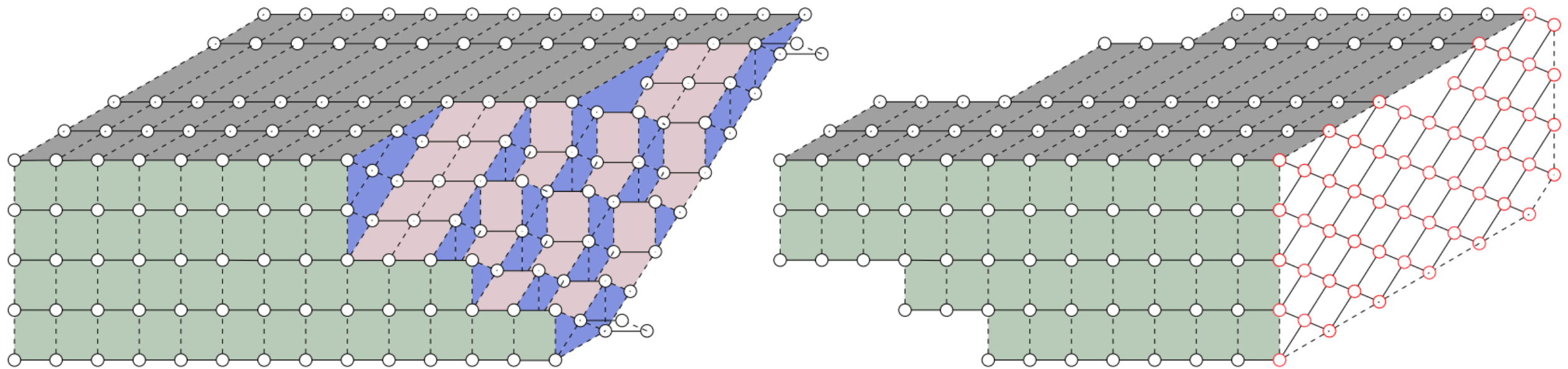


**Drilling a hole**  $\longrightarrow$  **breaks 2 qubit lines**  $\longrightarrow$  **inserting 2 error gates**

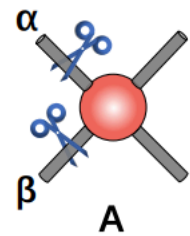
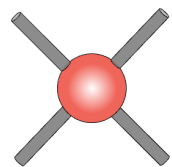
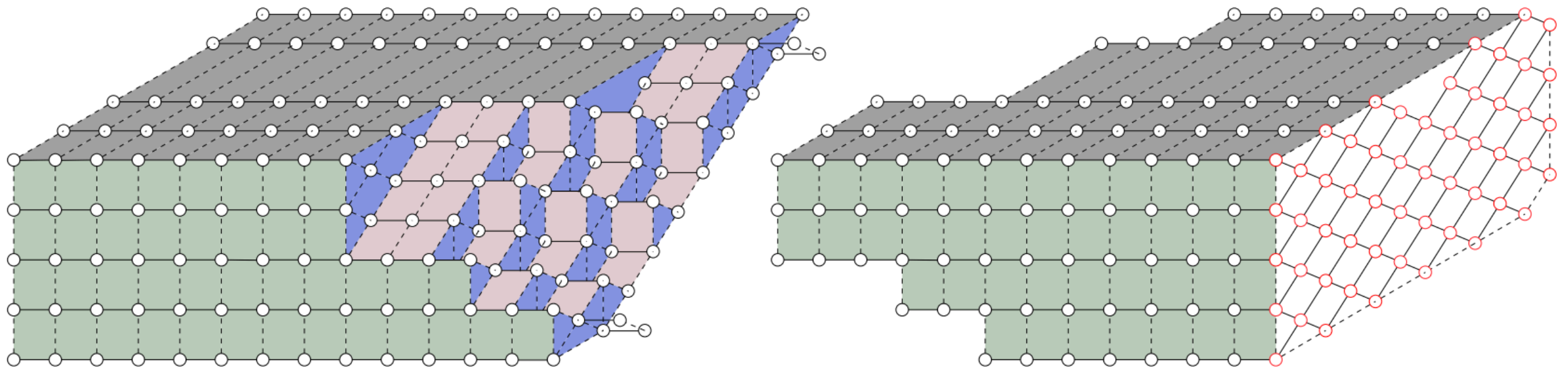
$$E = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \frac{1}{2}I + \frac{1}{2}\sigma_z$$

**Drilling 4 holes**  $\longrightarrow F \approx 2^{-8} = 0.00390625$

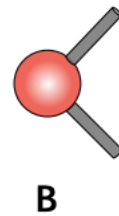
# Head-tail point of view to the hole-drilling



# Exploring low-rank structures in the fSim gates



=

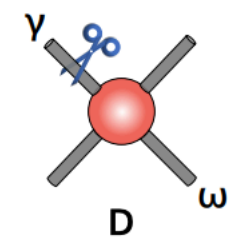


=

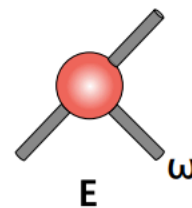


**Keep fidelity**

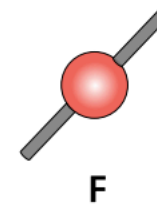
$$\text{fSim}(\theta, \phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -i \sin \theta & 0 \\ 0 & -i \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{bmatrix}$$



=



≈

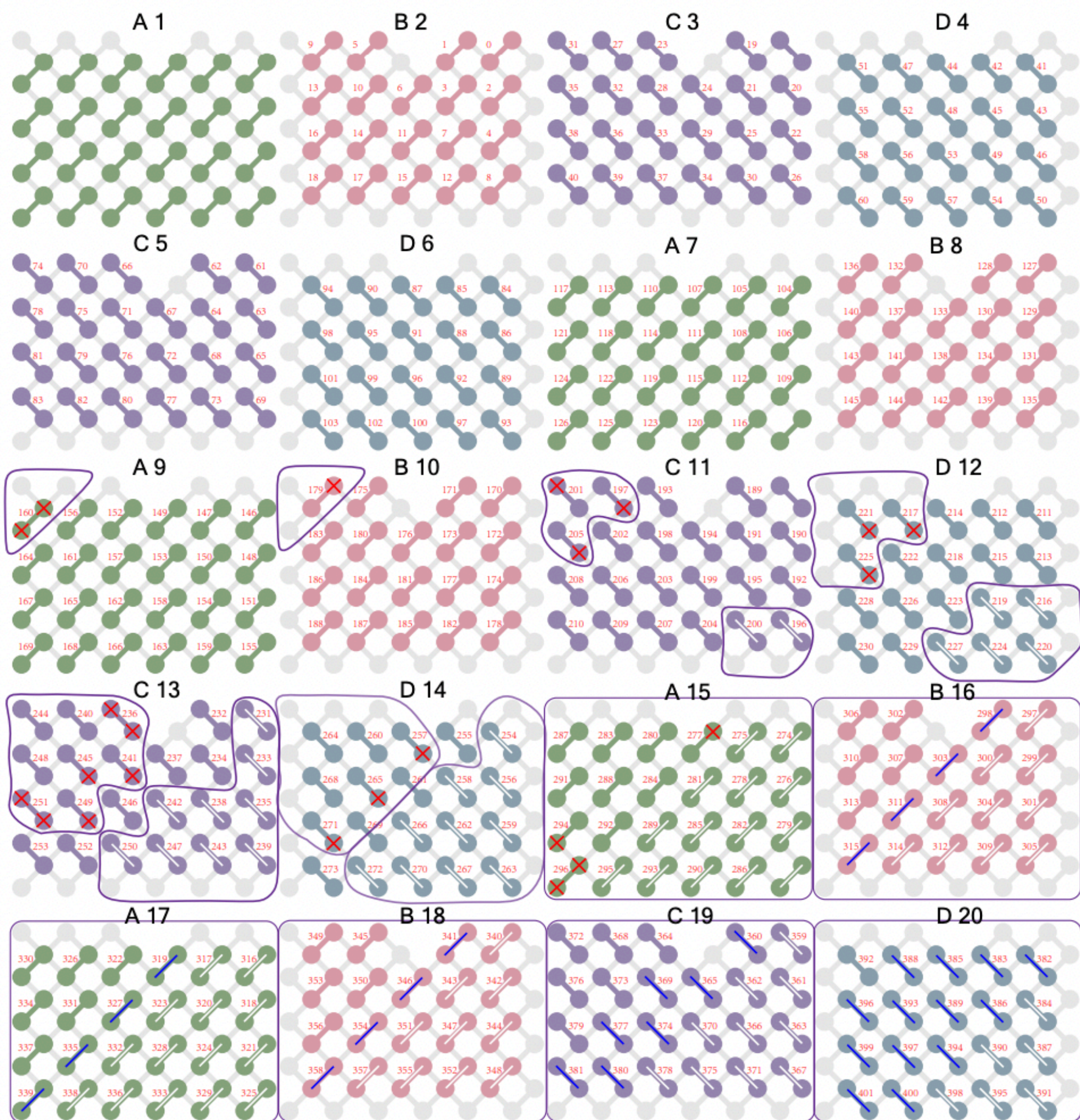


**Decrease fidelity  
by a factor of  
 $(\sin^2(\theta) + 1)/2$**

$$\prod_{i=1}^{21} [(\sin^2(\theta_i) + 1)/2] \approx 0.9565$$

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







# Parallel Computation with GPU



**CPU**



**GPU**


# GPU efficiency

NVIDIA-SMI 470.82.01				Driver Version: 470.82.01		CUDA Version: 11.4	
GPU Fan	Name Temp	Persistence-M Perf Pwr:Usage/Cap	Bus-Id	Disp.A Memory-Usage	Volatile GPU-Util	Uncorr. Compute M. MIG M.	ECC
0 N/A	Tesla V100S-PCI... 32C	Off P0 37W / 250W	00000000:1A:00.0	Off 26024MiB / 32510MiB	0%	0 Default N/A	
1 N/A	Tesla V100S-PCI... 30C	Off P0 25W / 250W	00000000:3D:00.0	Off 13MiB / 32510MiB	0%	0 Default N/A	
2 N/A	NVIDIA A100-PCI... 34C	Off P0 64W / 300W	00000000:88:00.0	Off 31607MiB / 81251MiB	18%	0 Default Disabled	
3 N/A	NVIDIA A100-PCI... 56C	Off P0 199W / 300W	00000000:89:00.0	Off 4684MiB / 81251MiB	100%	0 Default Disabled	
4 N/A	NVIDIA A100-PCI... 30C	Off P0 61W / 300W	00000000:B1:00.0	Off 1921MiB / 81251MiB	18%	0 Default Disabled	
5 N/A	NVIDIA A100-PCI... 32C	Off P0 63W / 300W	00000000:B2:00.0	Off 20670MiB / 81251MiB	15%	0 Default Disabled	

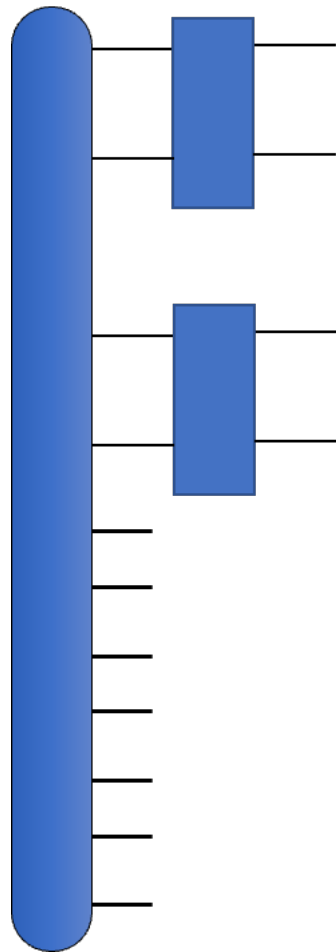
# GPU efficiency

- NVIDIA Tesla V100 FP32 performance: 14.13 TFLOPS
- NVIDIA Tesla A100 FP32 performance: 19.5 TFLOPs
- But with limited Bandwidth: 900GB/s and 1.6TB/s

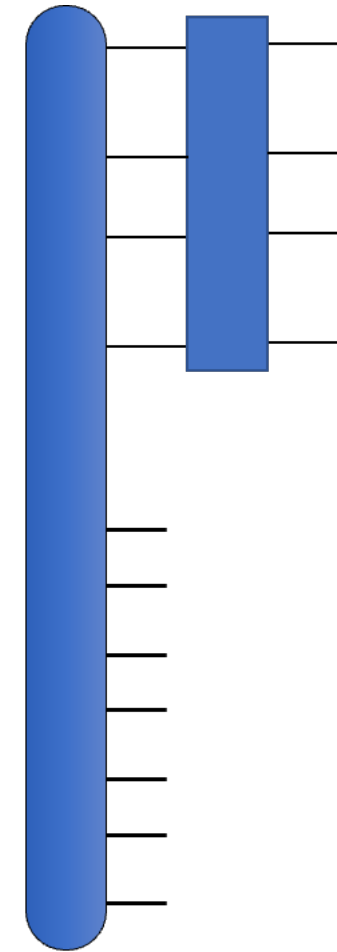
**6 for multiplication + 2 for addition, with Complex64 (FP32 + FP32)**


$$E = \frac{8 \times \text{time complexity}}{\text{GPU FLOPS capacity} \times \text{running time}} .$$

# Trade GPU efficiency with FLOPs: Branch Merge



**Low time complexity**  
**Low efficiency**



**Low time complexity**  
**Low efficiency**

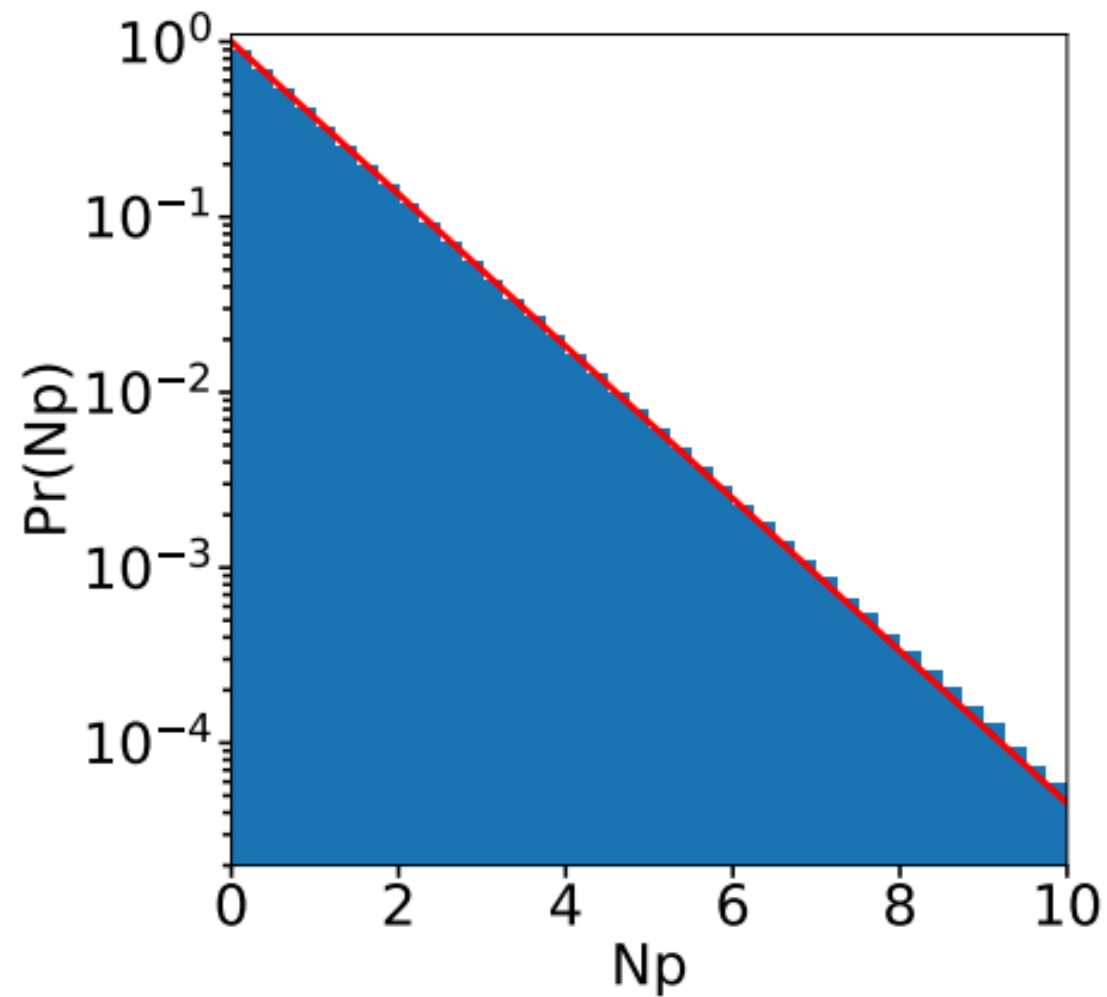


**3.87 x Time Complexity**

**4.4%  $\Rightarrow$  38.4%**

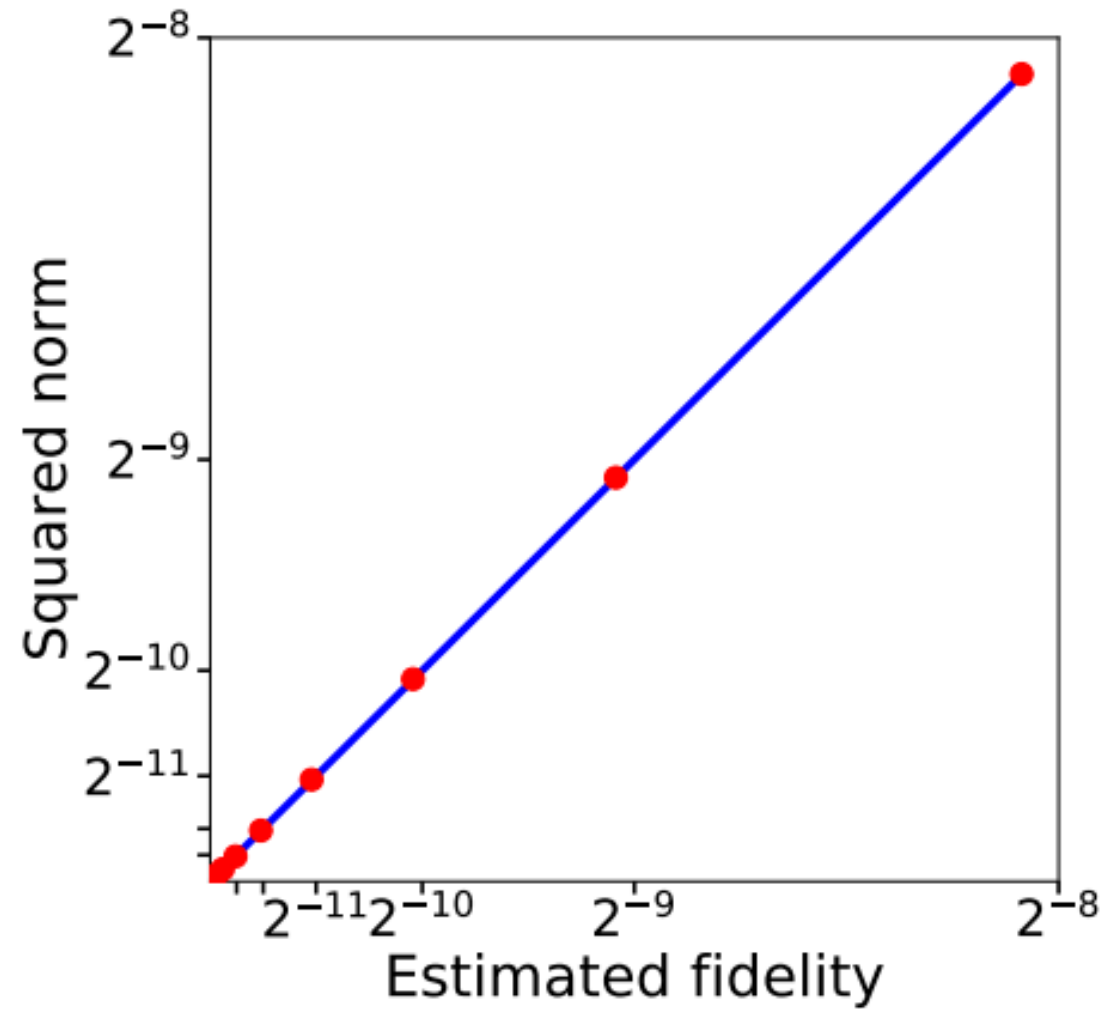
Data	Original	Branch merge
$T_c$ head one sub-task	$2.3816 \times 10^{13}$	$6.967 \times 10^{13}$
$T_c$ tail one sub-task	$2.9425 \times 10^{13}$	$8.796 \times 10^{13}$
Overall $T_c$ ( $2^{16}$ sub-tasks)	$3.489 \times 10^{18}$	$1.033 \times 10^{19}$
Space complexity	$2^{30}$	
# of slicing edges in $\widehat{\mathcal{G}}_{\text{head}}$	6	
# of slicing edges in $\widehat{\mathcal{G}}_{\text{tail}}$	7	
# of slicing edges in the interface	16	
# of companion edges in $\widehat{\mathcal{G}}_{\text{head}}$	0	
# of companion edges in $\widehat{\mathcal{G}}_{\text{tail}}$	5	
# of companion edges in the interface	16	
Fidelity of rank one approximation	0.9564714760983217	
GPU efficiency head	-	31.76%
GPU efficiency tail	-	14.27%
Overall efficiency	-	18.85%

# Histogram of probabilities



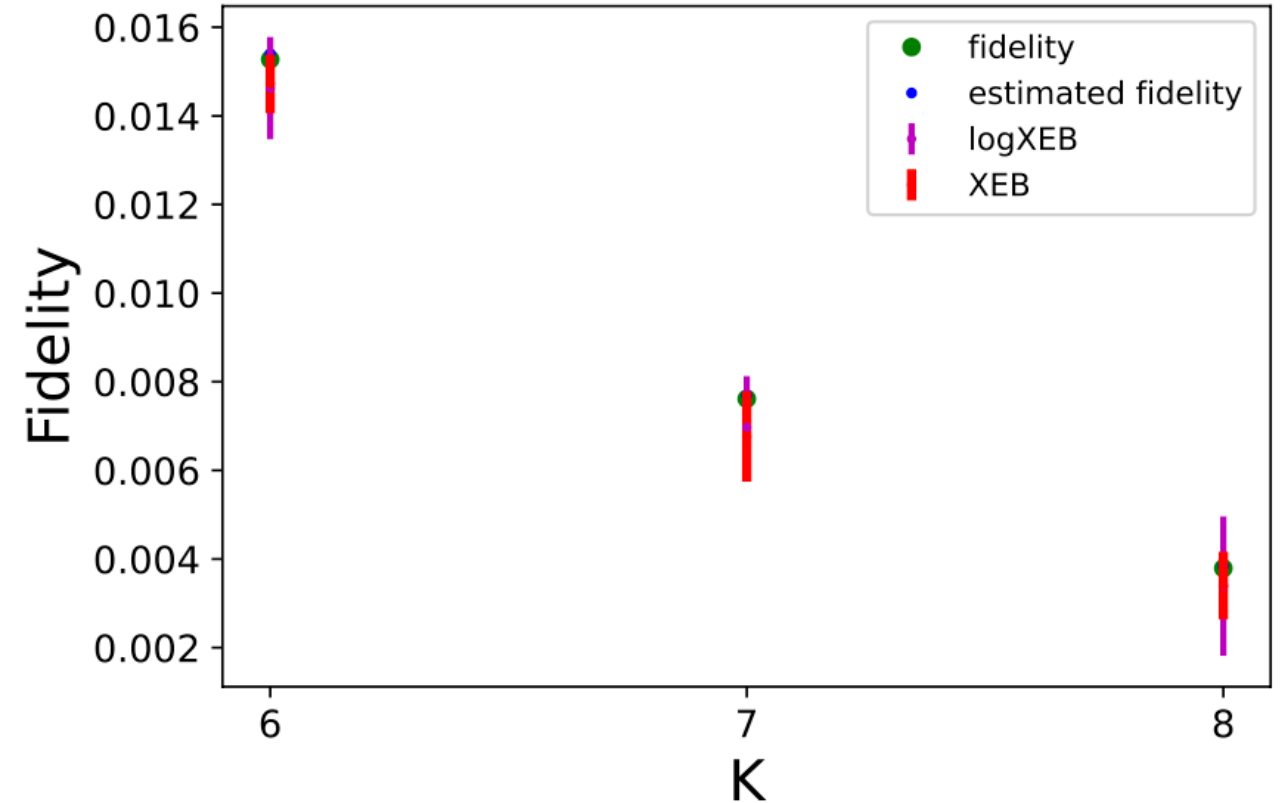
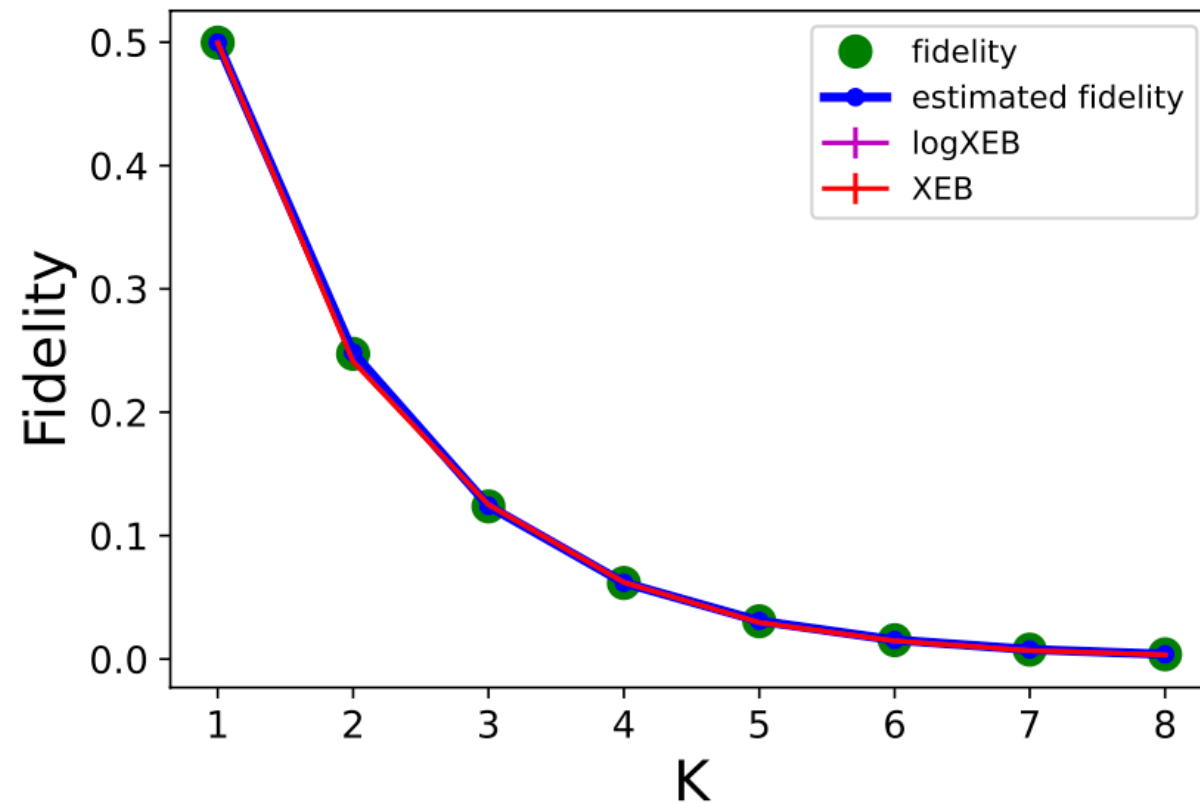
**Histogram of approximate  
bitstring probabilities**

$$p(s) = |\widehat{\psi}(s)|^2 / \mathcal{N}_s$$



**Fidelity vs. norm  
in the path integral interpretation**

# Validation of the approach using smaller Sycamore circuits

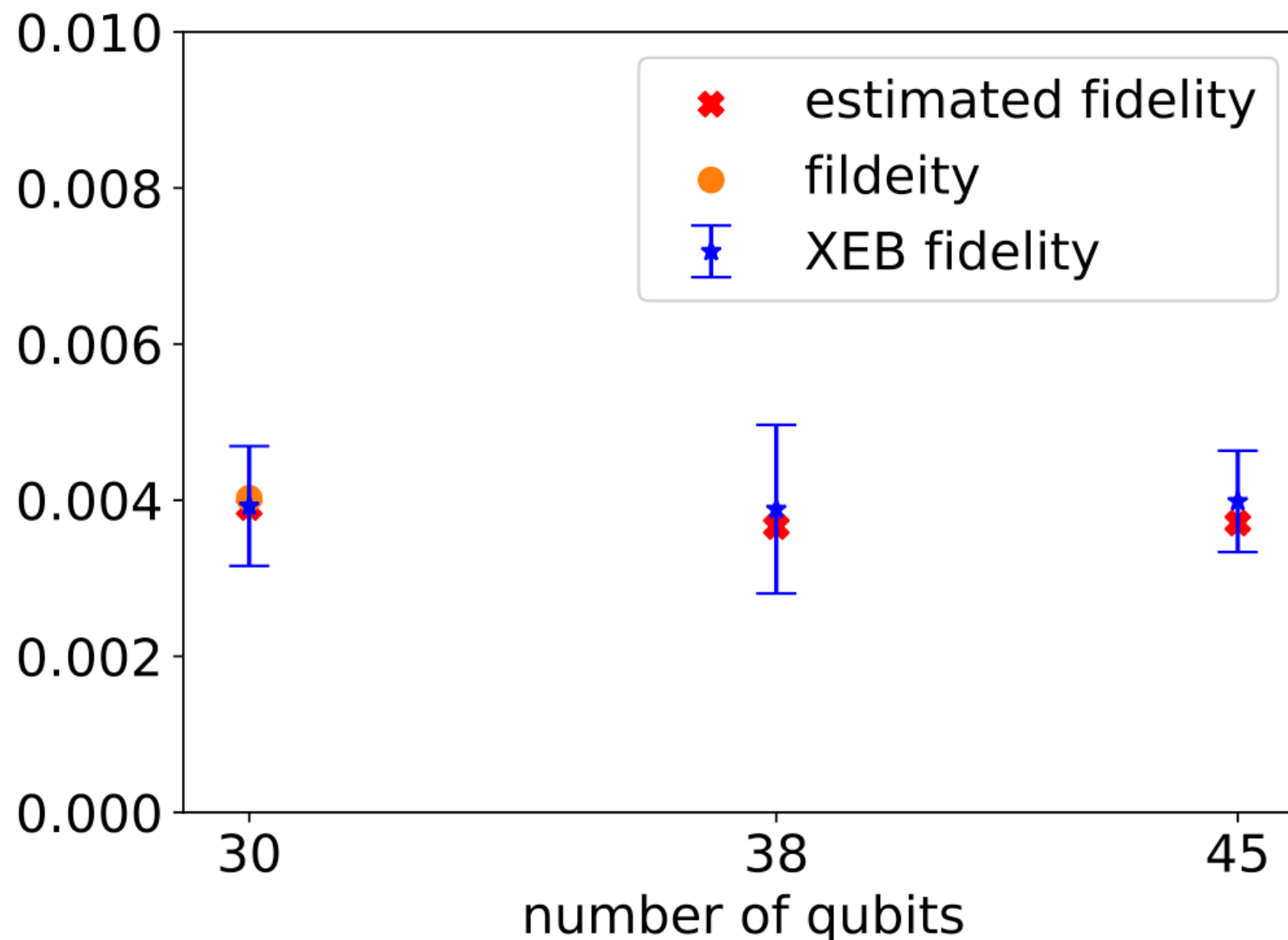


**Sycamore EFGH circuits with 30 qubits, 14 cycles**

**We obtain  $2^{26}$  bitstring probabilities and  $2^{20}$  uncorrelated samples, averaged over 15 sets of samples.**

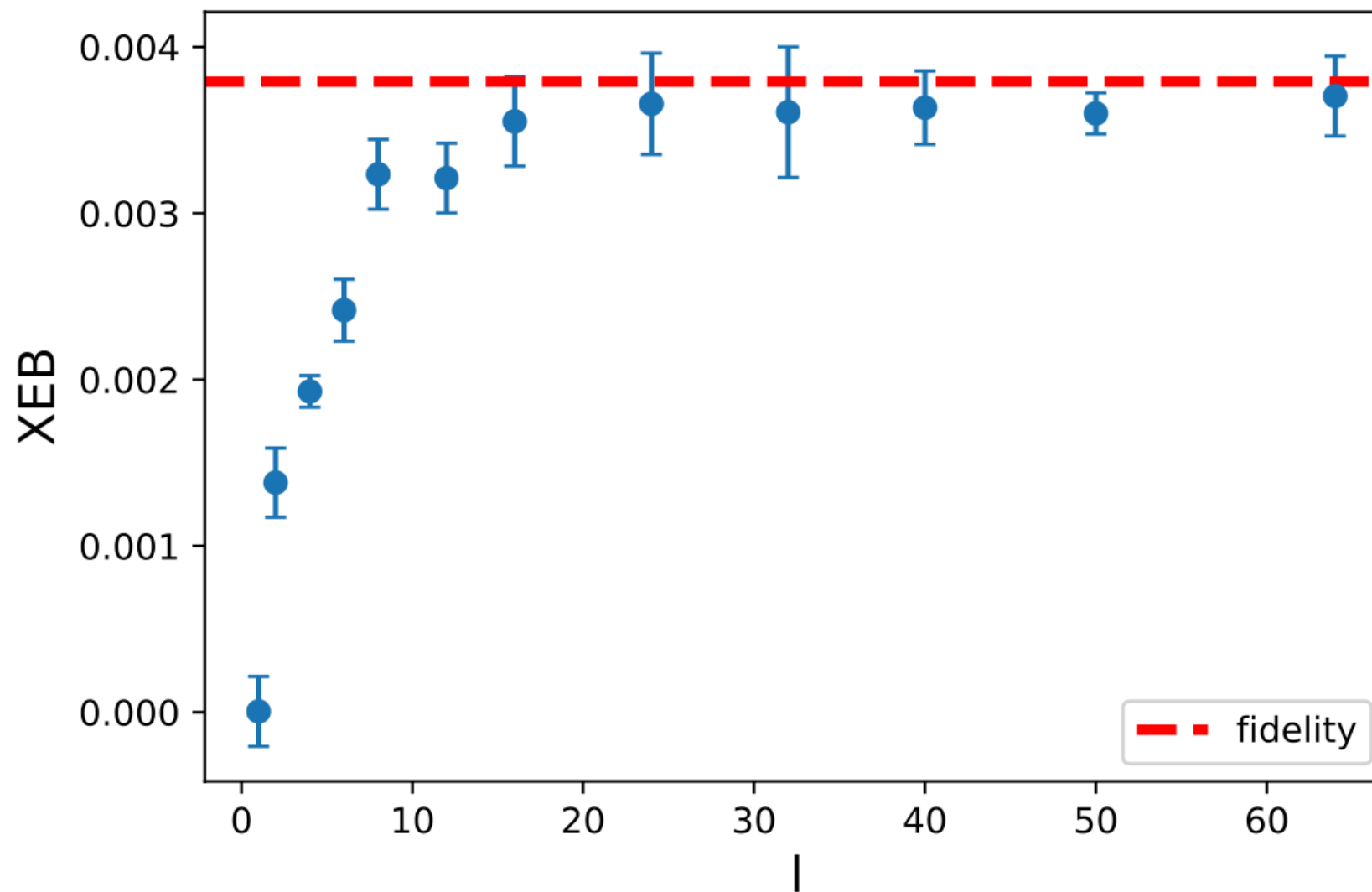
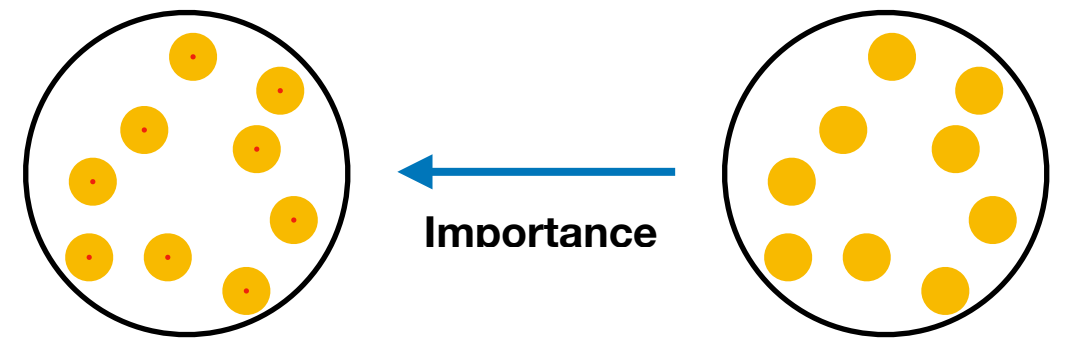


# Verifications with different sizes



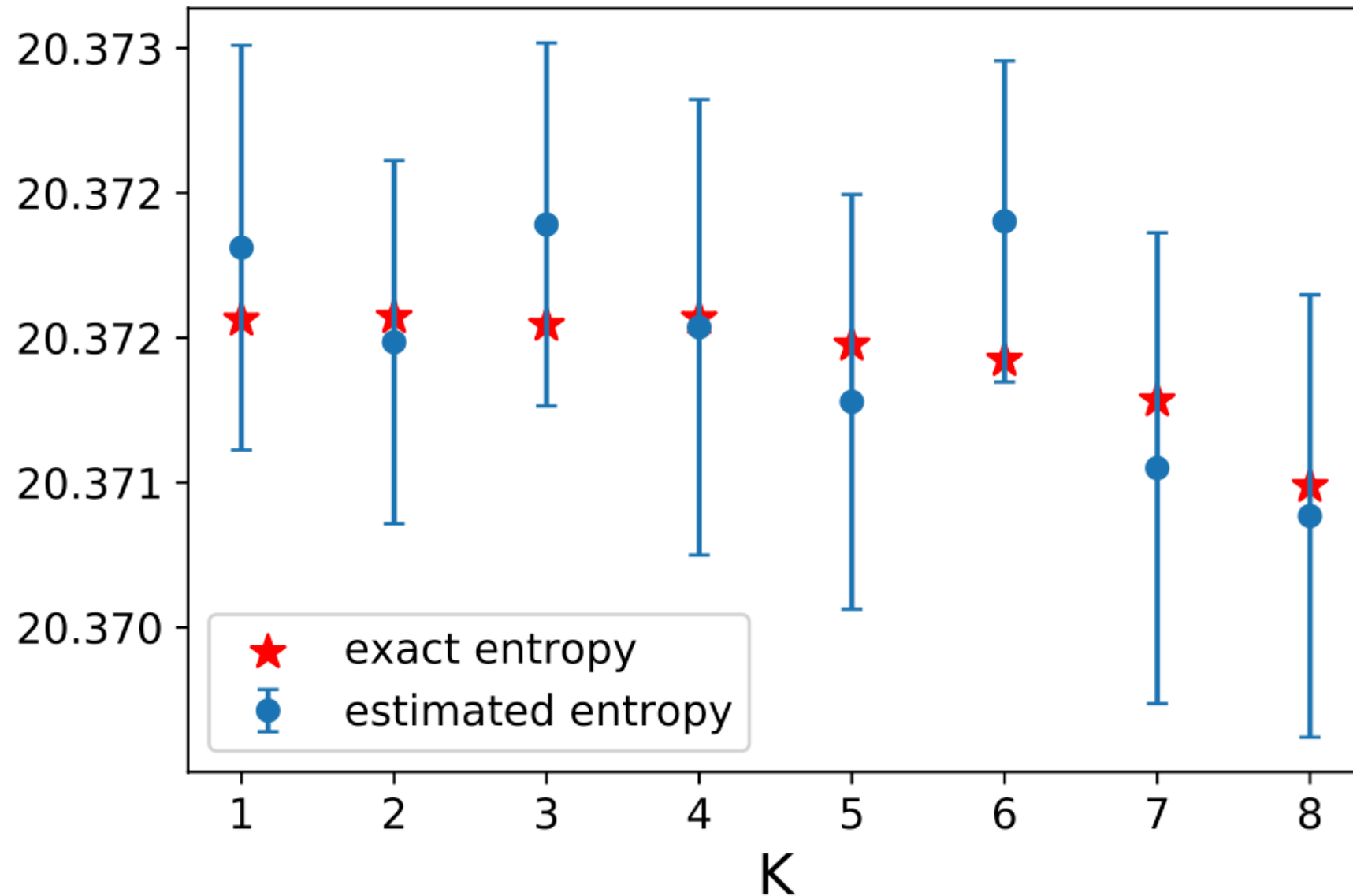
**Sycamore EFGH circuits with 14 cycles, and a different number of qubits.  
We obtain  $2^{26}$  bitstring probabilities and  $2^{20}$  uncorrelated samples, averaged  
over 15 sets of samples.**

# Dependence on the group size



**XEB as a function of group size  $l$  for Sycamore EFGH circuits with 30 qubits, 14 cycles**  
**We obtain  $2^{26}$  bitstring probabilities and  $2^{20}$  uncorrelated samples, averaged over 15 sets of samples.**

# Verification of the entropy



The exact entropy of the approximate state distribution of  $|\psi_K(\mathbf{s})|^2$  compared with the entropy estimated using  $2^{20}$  uncorrelated samples generated using the sampling method, for K cuts in the Sycamore circuits with  $n = 30$  qubits,  $m = 14$  cycles, and EFGH sequence. Each data point is averaged over 15 independent sets of samples of size  $2^{20}$ .

# Computational cost

	Computational complexity	Computation hardware	Time
Google [Arute et. al., 2019] ( <b>Estimated</b> )	— — —	Summit Super Computer	10,000 Years
IBM [Pednault et. al., 2019] ( <b>Estimated</b> )	$1.18 \times 10^{21}$	Summit Super Computer (all disks)	2.5 days
Alibaba [Huang et. al., 2020] ( <b>Estimated</b> )	$1.33 \times 10^{22}$	Summit Super Computer	20 days
Ours [arXiv:2103.03074] ( <b>Computed</b> ) Correlated sampling	$4.51 \times 10^{18}$	60 <b>NVIDIA GPUs</b>	5 days
Ours [arXiv: 2111.03011] ( <b>Computed</b> ) Uncorrelated sampling	$3.49 \times 10^{18}$	512 <b>NVIDIA GPUs</b>	15 hours
		Exa-Supercomputer ( <b>Estimate</b> )	Dozens of seconds

References:

F. Pan, K.Chen, PZ, arXiv:2111.03011 (2021), Phys. Rev. Lett. 128, 030501  
F. Pan, PZ, arXiv:2103.03074 (2021) , Phys. Rev. Lett. 129, 090502