QCMB, November 2022, IJCLab

A density-matrix renormalization group algorithm for simulating quantum circuits with a finite fidelity

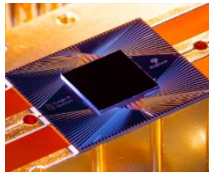
Thomas Ayral
Atos Quantum Laboratory

TA, Louvet, Zhou, Lambert, Stoudenmire, Waintal, 2207.05612

Closing the supremacy gap

Fall 2019:

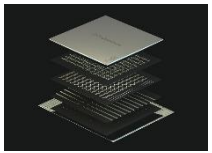
Google claims quantum “supremacy”



+ others!

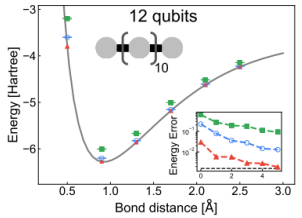
Today:

Even larger machines…



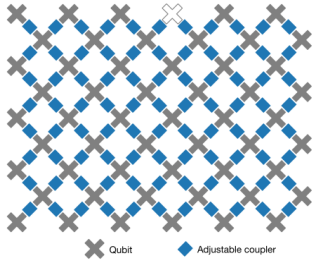
But no useful quantum advantage (yet).

Arute et al ‘20



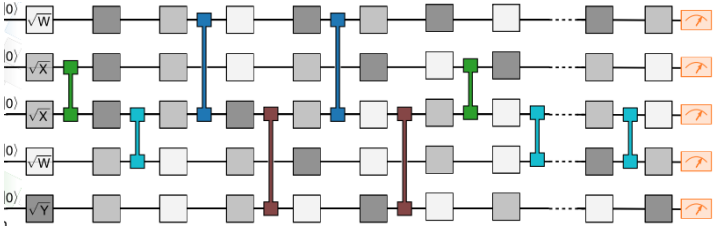
What do we (really) need to beat to reach advantage?

The Google ‘supremacy’ game



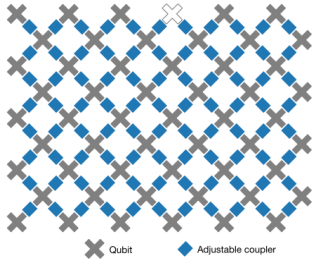
Arute et al ‘19

Execute circuit (53 qubits, 230 two-qubit gates)



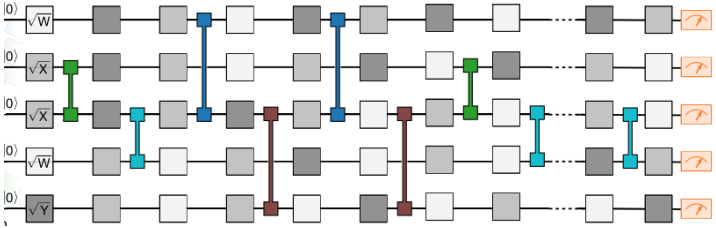
Return bitstrings x = 00101, …

The Google ‘supremacy’ game



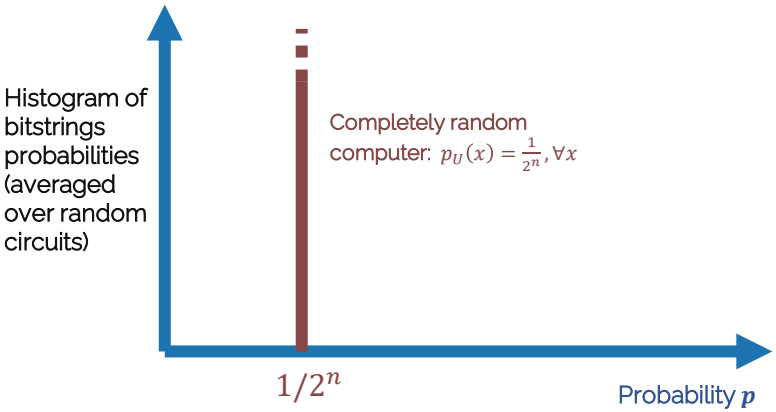
Arute et al ‘19

Execute circuit (53 qubits, 230 two-qubit gates)



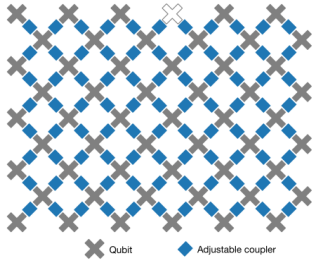
Return bitstrings x = 00101, …

Expectations



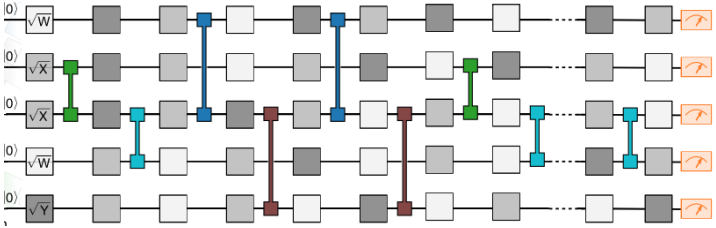
Atos

The Google ‘supremacy’ game



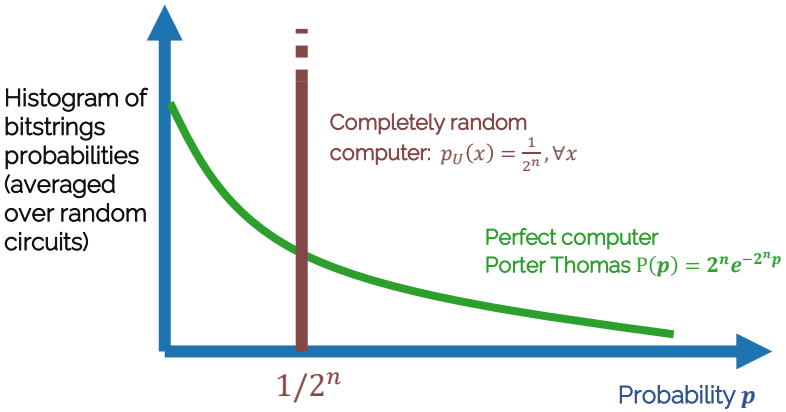
Arute et al ‘19

Execute circuit (53 qubits, 230 two-qubit gates)



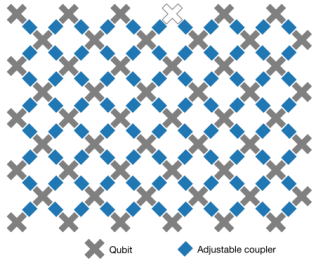
Return bitstrings x = 00101, …

Expectations



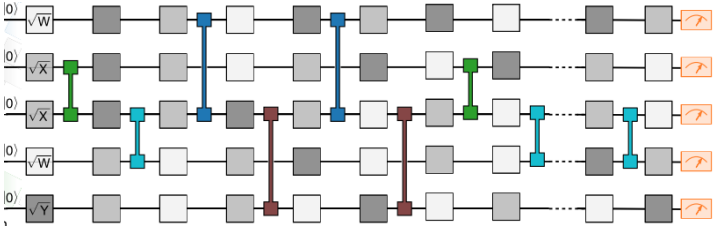
Atos

The Google ‘supremacy’ game



Arute et al ‘19

Execute circuit (53 qubits, 230 two-qubit gates)



Return bitstrings x = 00101, …

Quantum task: Compute “Cross Entropy Benchmarking”

$$\mathrm{XEB} = \left\langle \sum_{x=0}^{2^n-1} Q_U(x)\,(P_U(x) - 1) \right\rangle_U$$

averaged over random circuits

(~ ‘distance’ to perfect distribution)

In practice: sample bitstrings x_i

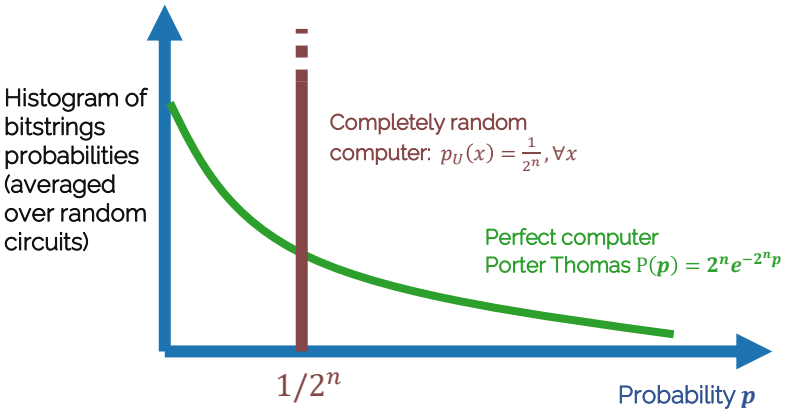
$$\mathrm{XEB} \approx \left\langle \frac{1}{N_{\mathrm{samples}}} \sum_{i=1}^{N_{\mathrm{samples}}} P_U(x_i) - 1 \right\rangle_U$$

Perfect ($Q = P$) XEB = 1

Random: XEB = 0

(need lots of samples when XEB ≪ 1!)

Expectations



Classical task: Draw N_{samples} with same XEB

Atos

What to expect?

The product law for the fidelity

In random circuits,

Zhou et al '20

$$F = \prod_{k \in \text{operations}} f_k \approx (1-\epsilon_1)^{n_1}(1-\epsilon_2)^{n_2}\,(1-\epsilon_{RO})^{n_{\text{RO}}} = 0.15\% \text{ !}$$

1116 430 53

0.16% 0.62% 3.8%

Exponential decay!

Atos

What to expect?

The product law for the fidelity

In random circuits,

Zhou et al '20

$$F = \prod_{k \in \text{operations}} f_k \approx (1-\epsilon_1)^{n_1}(1-\epsilon_2)^{n_2}\,(1-\epsilon_{RO})^{n_{\text{RO}}} = 0.15\% \text{ !}$$

1116 430 53

0.16% 0.62% 3.8%

Exponential decay!

+ with depolarizing noise: $\text{XEB} \approx F$

Experimentally, Google 'measured' $\text{XEB} = 0.2\%$ Quite a low fidelity!

Arute et al '19

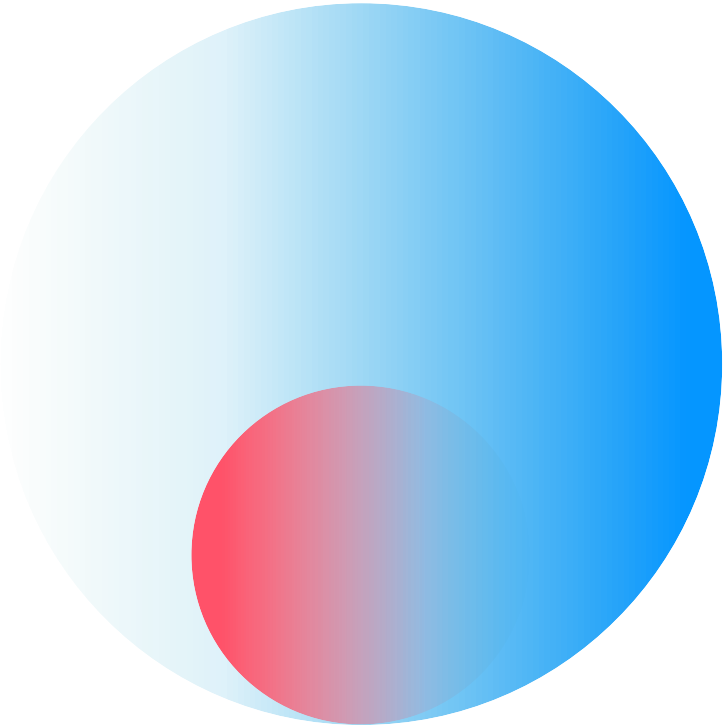
- How did Google measure XEB?

$$\text{XEB} \approx \left\langle \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} P_U(x_i) - 1 \right\rangle_U$$

???

Atos

1. Previous classical simulation strategies



Classical simulation of quantum circuits

From a tensor-network perspective

Goal: compute $P_U(x) = |\langle x|\Psi\rangle|^2 = |\langle x|U|0\rangle|^2$

Tensor network primer:

Matrix: 

Vector: 

Tensor: 

Matrix-matrix: 

$= A_{ij}B_{jk}$

Contraction: 

$= C_{ik}$

Atos

Classical simulation of quantum circuits

From a tensor-network perspective

Goal: compute $P_U(x) = |\langle x|\Psi\rangle|^2 = |\langle x|U|0\rangle|^2$

Tensor network primer:

Matrix: 

Vector: 

Tensor: 

Matrix-matrix: 

$= A_{ij}B_{jk}$

Contraction: 

$= C_{ik}$

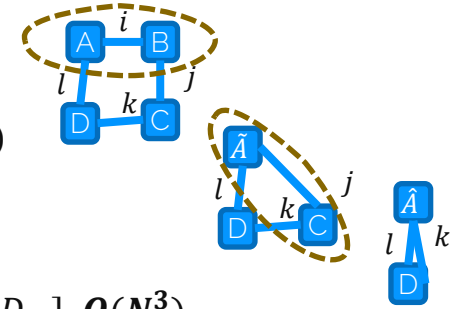
Order matters:

- Naive contraction:

$s = \sum_{ijkl} A_{li}B_{ij}C_{jk}D_{kl}$, $\boldsymbol{O(N^4)}$

- Clever contraction:

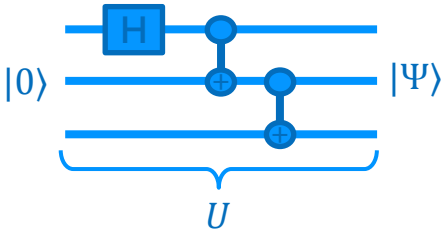
$s = \sum_l [\sum_k \{\sum_j (\sum_i A_{li}B_{ij})\, C_{jk}\}\, D_{kl}]$, $\boldsymbol{O(N^3)}$



Atos

Classical simulation of quantum circuits

From a tensor-network perspective



Goal: compute $P_U(x) = |\langle x|\Psi\rangle|^2 = |\langle x|U|0\rangle|^2$

Computing $\langle x|U|0\rangle$

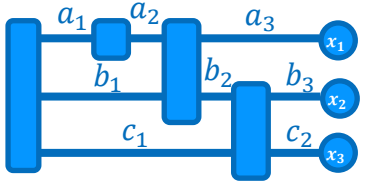
Tensor network primer:

Matrix: A Vector: v Tensor: T

Matrix-matrix: i A j B k $= A_{ij}B_{jk}$

Contraction: i C k $= C_{ik}$

- Corresponding tensor network



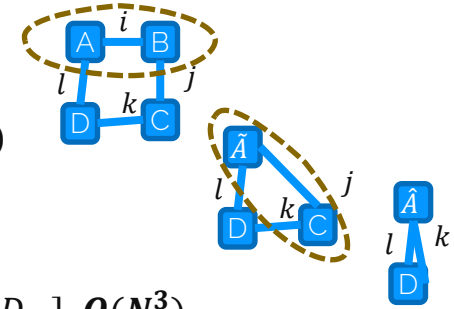
$x = (x_1, x_2, x_3)$

- Find $\langle x|U|0\rangle$: contract the tensor network.

Order matters:

- Naive contraction:

$s = \sum_{ijkl} A_{li}B_{ij}C_{jk}D_{kl}$, $\boldsymbol{O(N^4)}$



- Clever contraction:

$s = \sum_l [\sum_k \{\sum_j (\sum_i A_{li}B_{ij})\, C_{jk}\}\, D_{kl}]$, $\boldsymbol{O(N^3)}$

Note: storage cost

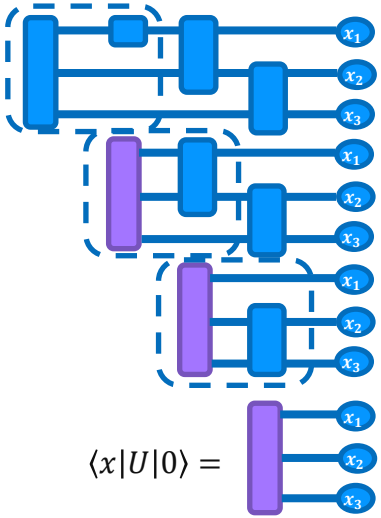
- 3 qubits: 2x2x2 = 8
- n qubits: 2^n!

Atos

Three main classical simulation methods

… as three contraction strategies!

1. Schrödinger



width

- CPU cost: $N_{\text{gates}}\exp(n)$
- Storage cost: $\exp(n)$

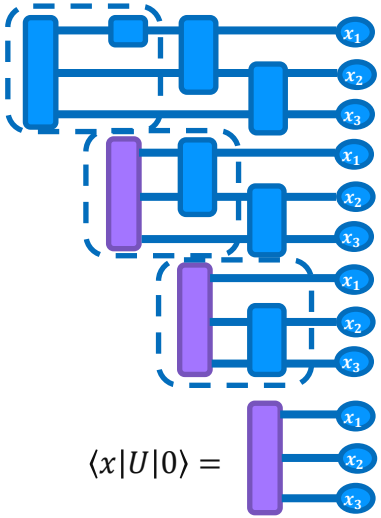
All amplitudes at once: “strong”

Three main classical simulation methods

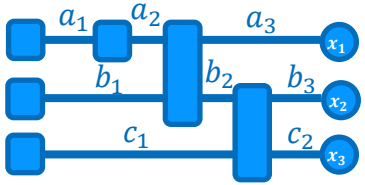
… as three contraction strategies!

Markov & Shi '08

1. Schrödinger



2. "Tensor network"



1. Find (close to) optimal contraction strategy (NP hard problem!)
2. Contract (GPUs, TPUs…)

width

- **CPU cost**: $N_{\text{gates}}\exp(n)$
- **Storage cost**: $\exp(n)$

All amplitudes at once: "strong"

min(depth, width)

- **CPU** : $\exp(\text{Treewidth})$
- **Storage**: $\exp(\text{Treewidth})$

One amplitude: "closed"

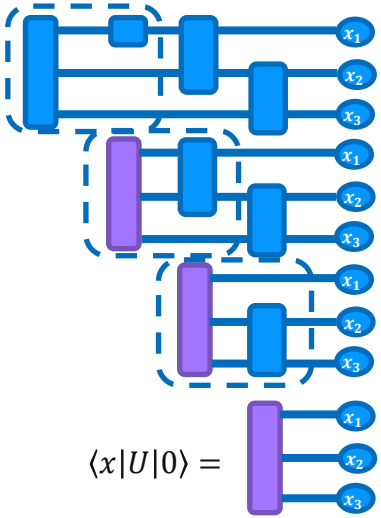
Atos

Three main classical simulation methods

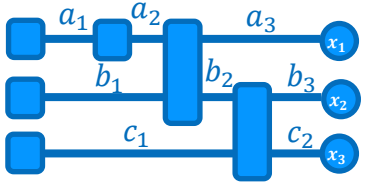
… as three contraction strategies!

Markov & Shi '08

1. Schrödinger

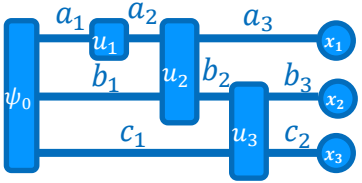


2. "Tensor network"



1. Find (close to) optimal contraction strategy (NP hard problem!)
2. Contract (GPUs, TPUs…)

3. Feynman (sum over paths)



$$\langle x|U|0\rangle = \sum_{\underbrace{a_1,a_2,\ldots c_1,c_2}_{\text{paths}}} [\psi_0]_{a_1b_1c_1}[u_1]_{a_1a_2}\,[u_2]_{a_2b_1,a_3b_2}\,[u_3]_{b_2c_1,b_3c_2}\,\delta_{a_3x_1}\delta_{b_3x_2}\delta_{c_2x_3}$$

width

- **CPU cost**: $N_{\text{gates}}\exp(n)$
- **Storage cost**: $\exp(n)$

All amplitudes at once: "strong"

min(depth, width)

- **CPU** : $\exp(\text{Treewidth})$
- **Storage**: $\exp(\text{Treewidth})$

One amplitude: "closed"

depth

- **CPU**: $\propto N_{\text{paths}} \sim \exp(N_{\text{gates}})$
- **Storage**: const.

One amplitude: "closed"

How did Google 'prove' supremacy?

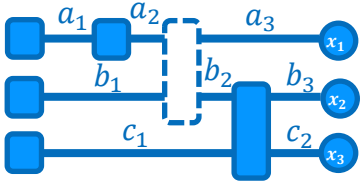
Arute et al '19

STEP 1: Compute experimental XEB

Need $P_U(x)$!

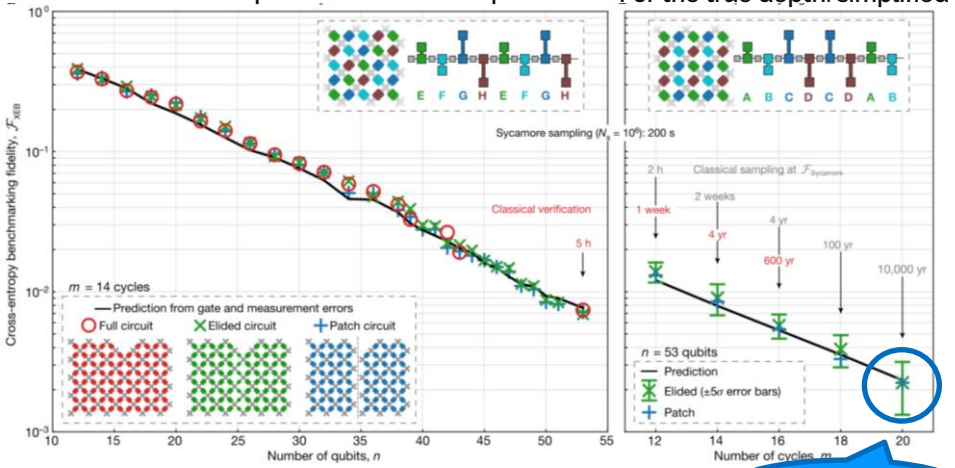
Trick: simplified circuit

- easier contraction
- … but same XEB!



Test: full=simplified for smaller depths

For the true depth: simplified



XEB = 0.2%

Atos

How did Google 'prove' supremacy?

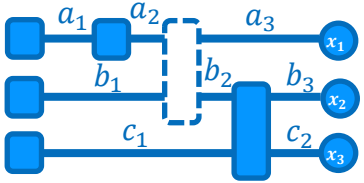
Arute et al '19

STEP 1: Compute experimental XEB

Need $P_U(x)$!

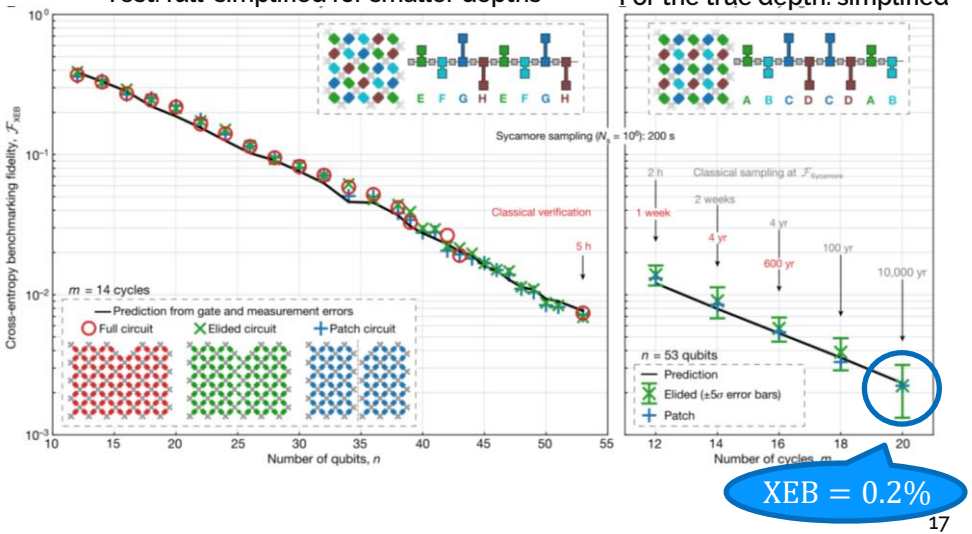
Trick: simplified circuit

- easier contraction
- … but same XEB!



Test: full=simplified for smaller depths

For the true depth: simplified



XEB = 0.2%

STEP 2: Estimate classical time to get 0.2% XEB

- Essentially: Feynman approach.

Claim:

sum only fraction F of all paths => fidelity F

$$t_{\text{one path}} \times N_{\text{paths}} \times F = 10'000 \text{ years}$$

… compare to: 200 secs to sample bitstrings

Goal: can we close this gap?

Atos

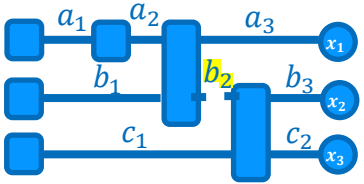
Previous attempts to catch up with Google

Pednault et al '19: Schrödinger, 2.5 days (est.)

Previous attempts to catch up with Google

Pednault et al '19: Schrödinger, 2.5 days (est.)

Gray & Kourtis '20, Huang et al '20: Index slicing: tensor network+ Feynman:



$$\langle x|U|0\rangle = \sum_{b_2} T_1(x_1; b_2) T_2(x_2, x_3; b_2)$$

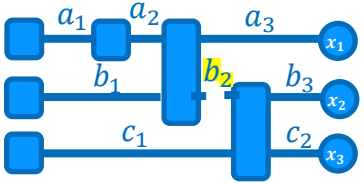
… 19.3 days (massively parallel)!

Atos

Previous attempts to catch up with Google

Pednault et al '19: Schrödinger, 2.5 days (est.)

Gray & Kourtis '20, Huang et al '20: Index slicing: tensor network+ Feynman:



$$\langle x|U|0\rangle = \sum_{b_2} T_1(x_1; b_2) T_2(x_2, x_3; b_2)$$

… 19.3 days (massively parallel)!

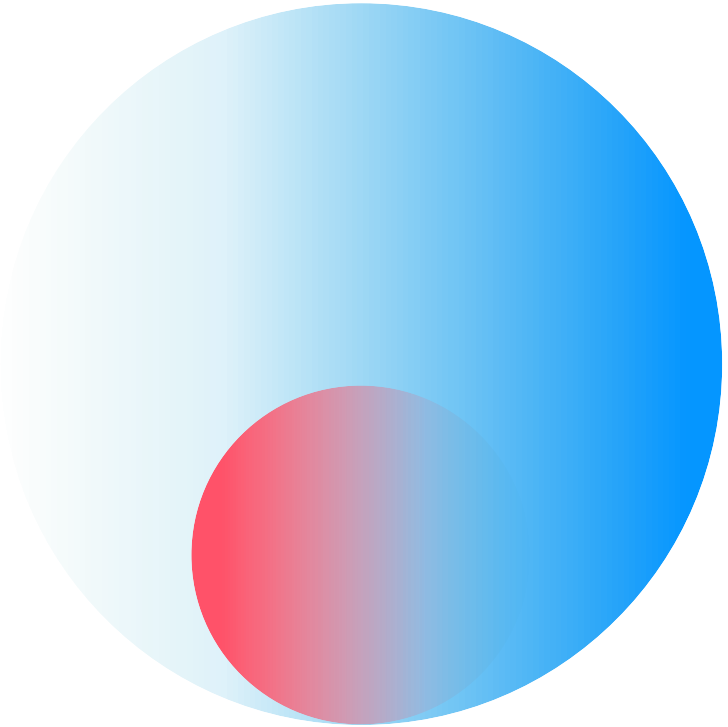
See also Pan & Zhang '21 (5 days [GPUs]), Pan et al '21.

- … down to 304 seconds (Liu et al '21)… on a very large HPC system (42 million cores)!

Main issue: exponential scaling! Very expensive to add a few qubits / gates…

Atos

2. Leveraging (low) entanglement



Beating the exponential with a finite fidelity

Matrix Product States (MPS)

See e.g
Schollwöck '11

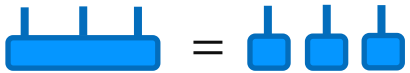
- Previous attempts:

Surrender fidelity by summing fewer Feynman paths.

- New idea: use key quantum property: entanglement

Trivial case: Product states

$$[\psi_0]_{a_1 b_1 c_1} = [\psi_0^1]_{a_1} [\psi_0^2]_{b_1} [\psi_0^3]_{c_1}$$



Atos

Beating the exponential with a finite fidelity

Matrix Product States (MPS)

See e.g Schollwöck '11

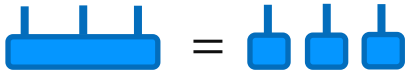
- Previous attempts:

Surrender fidelity by summing fewer Feynman paths.

- New idea: use key quantum property: entanglement

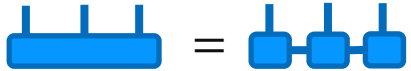
Trivial case: Product states

$$[\psi_0]_{a_1 b_1 c_1} = [\psi_0^1]_{a_1} [\psi_0^2]_{b_1} [\psi_0^3]_{c_1}$$



- Example: an entangled state:

$$[\psi_0]_{a_1 b_1 c_1} = \frac{1}{\sqrt{2}}[\psi_0^1]_{a_1} [\psi_0^2]_{b_1} [\psi_0^3]_{c_1} + \frac{1}{\sqrt{2}} [\chi_0^1]_{a_1} [\chi_0^2]_{b_1} [\chi_0^3]_{c_1}$$



= "Matrix product state".

Beating the exponential with a finite fidelity

Matrix Product States (MPS)

See e.g Schollwöck '11

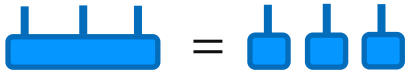
- Previous attempts:

Surrender fidelity by summing fewer Feynman paths.

- New idea: use key quantum property: entanglement

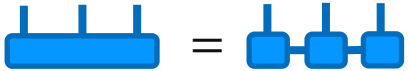
Trivial case: Product states

$$[\psi_0]_{a_1 b_1 c_1} = [\psi_0^1]_{a_1}\,[\psi_0^2]_{b_1}\,[\psi_0^3]_{c_1}$$



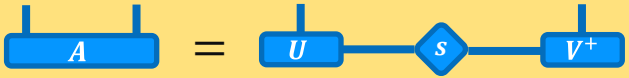
- Example: an entangled state:

$$[\psi_0]_{a_1 b_1 c_1} = \tfrac{1}{\sqrt{2}}[\psi_0^1]_{a_1}\,[\psi_0^2]_{b_1}\,[\psi_0^3]_{c_1} + \tfrac{1}{\sqrt{2}}\,[\chi_0^1]_{a_1}\,[\chi_0^2]_{b_1}\,[\chi_0^3]_{c_1}$$

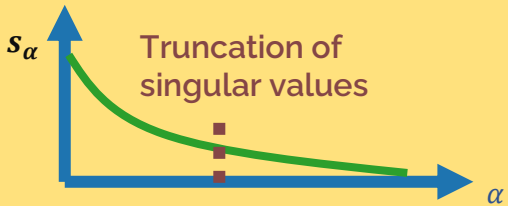


= "Matrix product state".

Compressing any state? Singular value decomposition (SVD)



$$A_{ij} = U_{i\alpha} s_\alpha V^+_{\alpha j}$$



Key result:

$$f = \left|\langle\psi|\psi_{\text{compressed}}\rangle\right|^2 = \sum_{\alpha<\chi} s_\alpha^2$$

Can estimate fidelity!

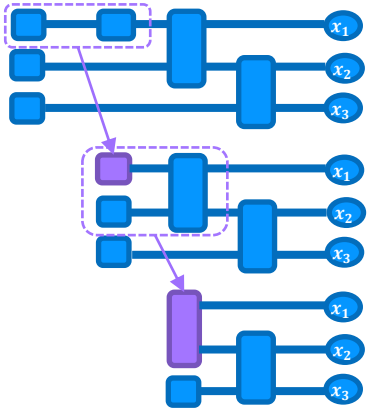
Atos

A first step towards reproducing the experiment

with “grouped” Matrix Product States

Zhou, Stoudenmire, Waintal PRX 20

Algorithm to compute $\langle x|U|0\rangle$: Vidal ‘04



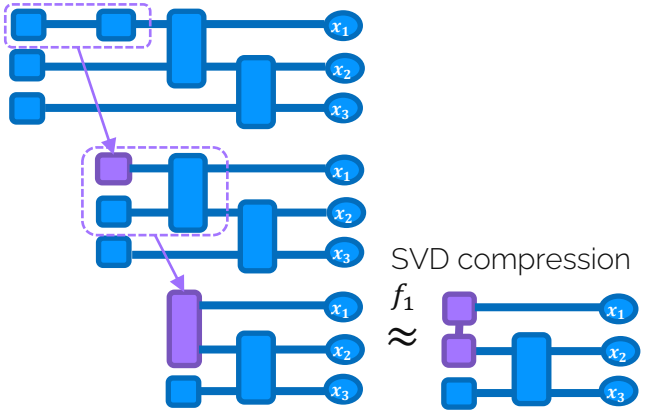
Atos

A first step towards reproducing the experiment

with “grouped” Matrix Product States

Zhou, Stoudenmire, Waintal PRX 20

Algorithm to compute $\langle x|U|0\rangle$: Vidal ‘04



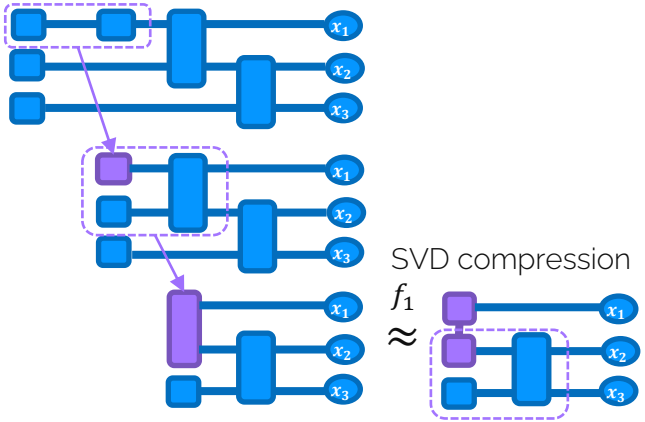
Atos

A first step towards reproducing the experiment

with “grouped” Matrix Product States

Zhou, Stoudenmire, Waintal PRX 20

Algorithm to compute $\langle x|U|0\rangle$: Vidal ‘04

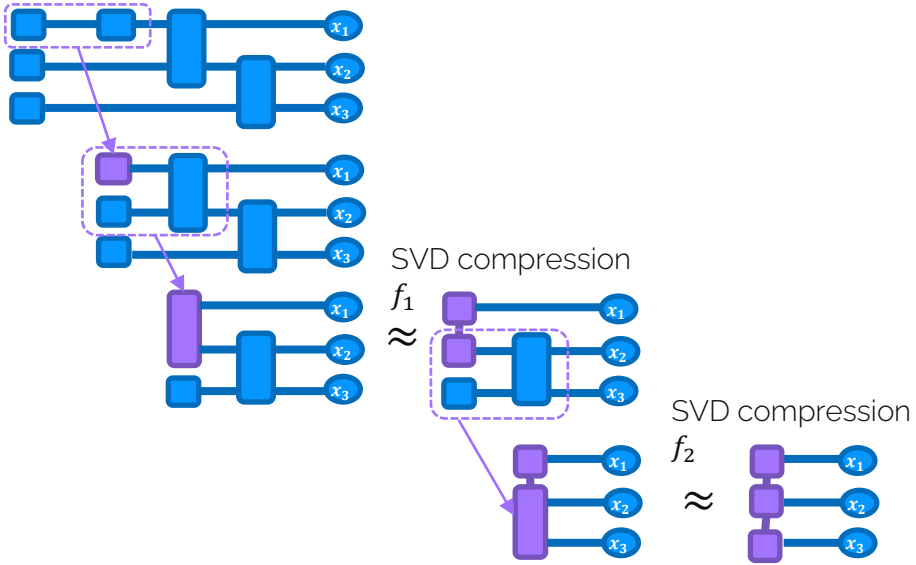


Atos

A first step towards reproducing the experiment
with "grouped" Matrix Product States

Zhou, Stoudenmire, Waintal PRX 20

Algorithm to compute $\langle x|U|0\rangle$: Vidal '04

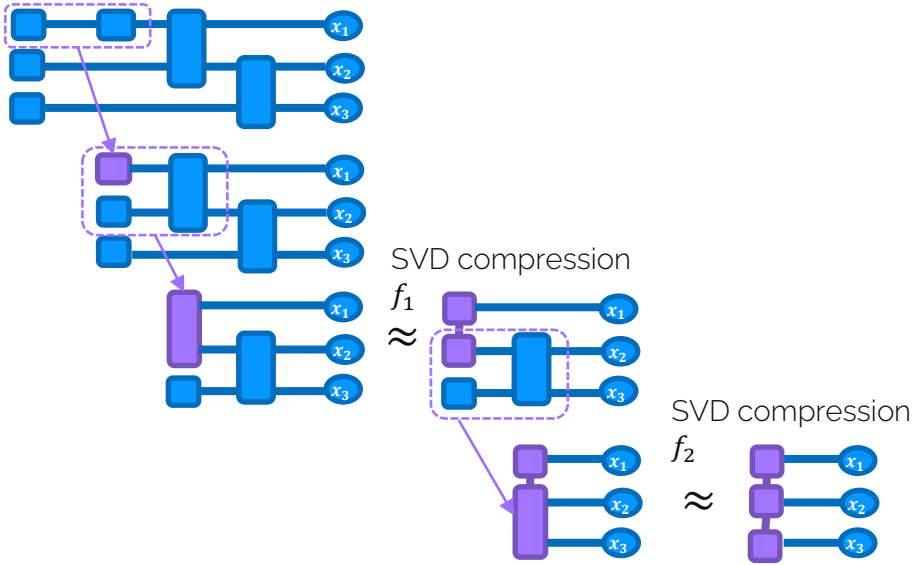


Atos

A first step towards reproducing the experiment
with "grouped" Matrix Product States

Zhou, Stoudenmire, Waintal PRX 20

Algorithm to compute $\langle x|U|0\rangle$: Vidal '04



- Final fidelity: $F = f_1 f_2$
- Works… but not enough to reproduce Google

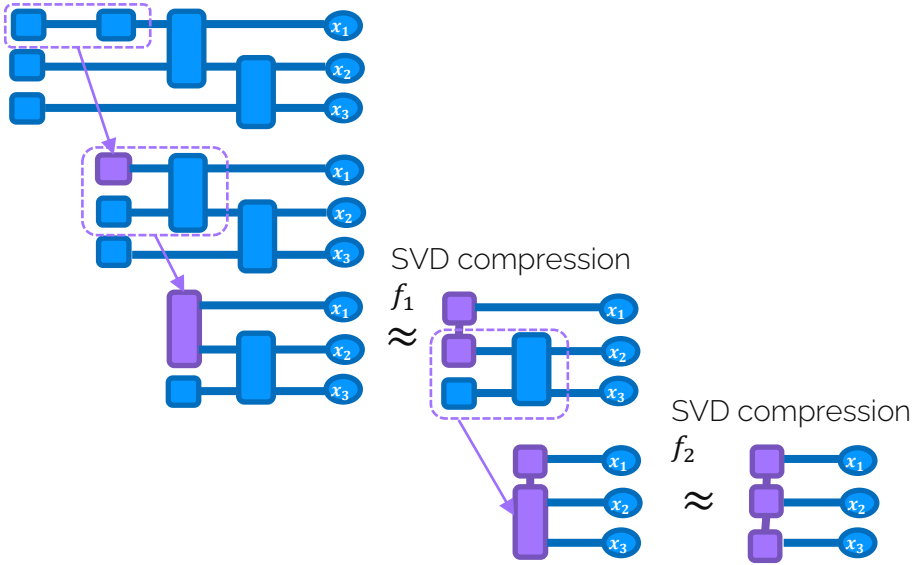
Atos

A first step towards reproducing the experiment

with "grouped" Matrix Product States

Zhou, Stoudenmire, Waintal PRX 20

Algorithm to compute $\langle x|U|0\rangle$: Vidal '04



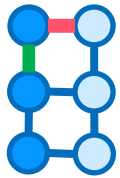
- Final fidelity: $F = f_1 f_2$
- Works… but not enough to reproduce Google

Improvement: Schrödinger + MPS

- "Group tensors together"

Useful for 2D qubit grids:

- Vertical gates: "exact"
- Horizontal gates: "compressed"



Improves fidelity… but still not enough!

Atos

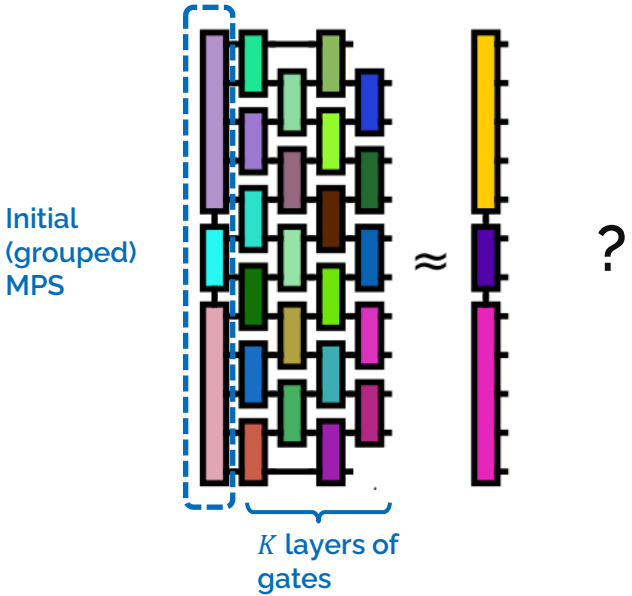
This work: a triply hybrid strategy

TA, Louvet, Zhou, Lambert, Stoudenmire, Waintal, 2207.05612

MPS + Schrödinger + tensor networks via a Density Matrix Renormalization Group method

Previous approach: apply 1 gate and compress

Here: apply several layers of gates,
… and find "optimal" MPS:



Atos

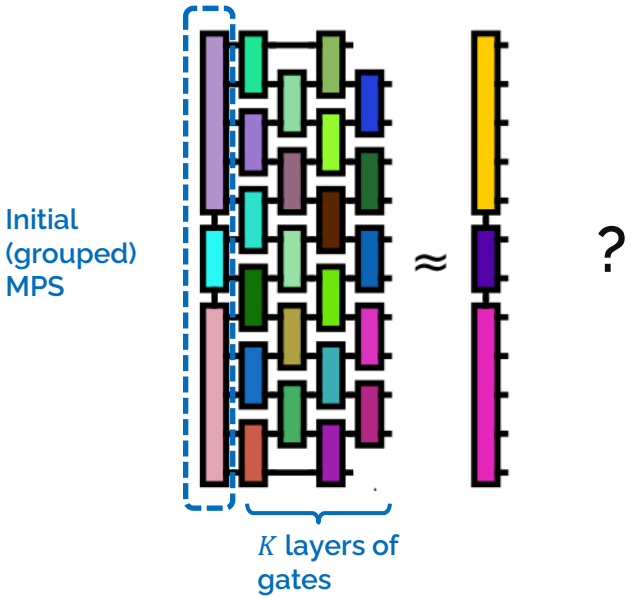
This work: a triply hybrid strategy

TA, Louvet, Zhou, Lambert, Stoudenmire, Waintal, 2207.05612

MPS + Schrödinger + tensor networks via a Density Matrix Renormalization Group method

Previous approach: apply 1 gate and compress

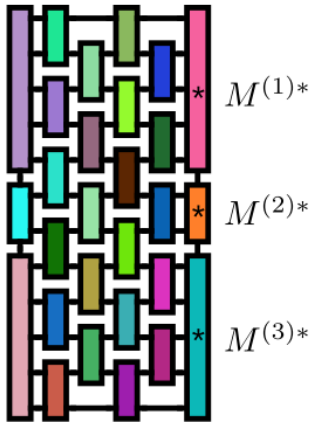
Here: apply several layers of gates,
… and find "optimal" MPS:



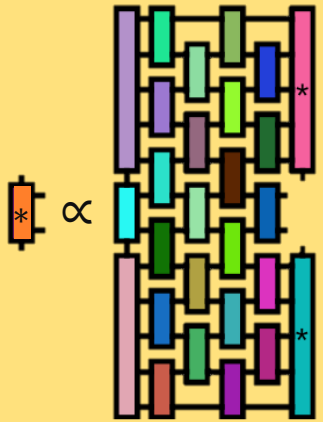
How to find optimal MPS?

- DMRG: find MPS with maximal overlap
- Tensor-by-tensor optimization: n_s "sweeps"

Overlap:



Best $M^{(2)}$ tensor:

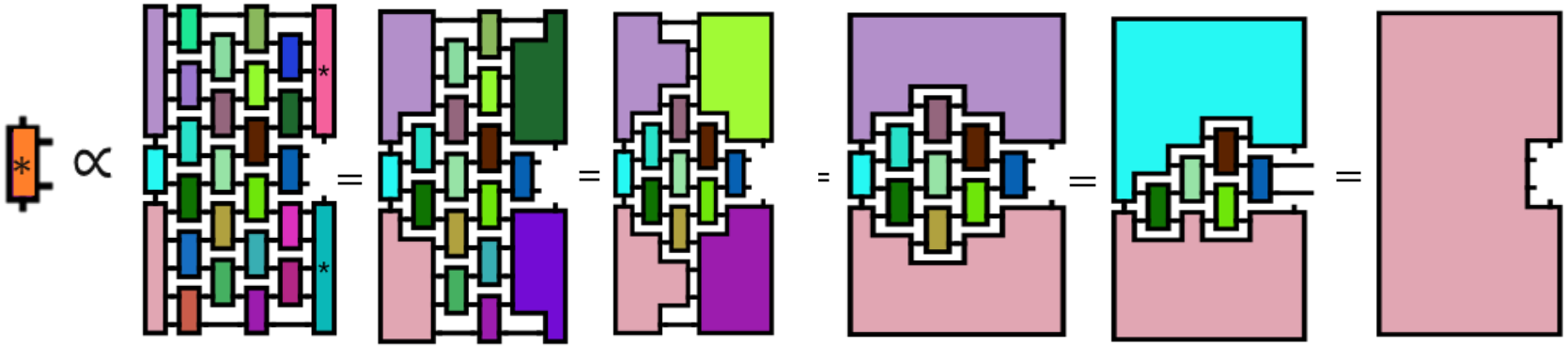


A tensor network!

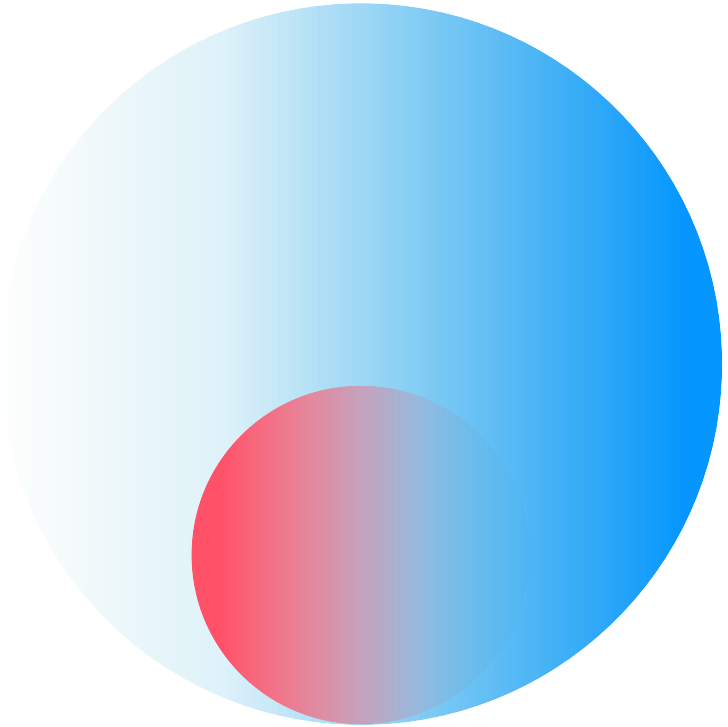
Atos

Contracting the tensor network

Mix of automatic contraction with “manual” guidance:



3. Results



A different relation between F and XEB

Reminder: in noisy random circuits:

$$F \approx \mathbf{XEB}$$

In our method:

$$\mathbf{XEB} \approx \sqrt{F}$$

- Strange?

A different relation between F and XEB

Reminder: in noisy random circuits:

$$F \approx \mathrm{XEB}$$

In our method:

$$\mathrm{XEB} \approx \sqrt{F}$$

- Strange?

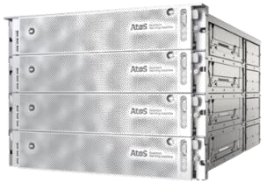
In the chaotic limit (infinite depth):

$$F = \frac{1}{2^n}$$

$$\mathrm{XEB} = \frac{1}{2^{n/2}}$$

A different relation between F and XEB

Atos QLM



Reminder: in noisy random circuits:

$$F \approx \mathrm{XEB}$$

In our method:

$$\mathrm{XEB} \approx \sqrt{F}$$

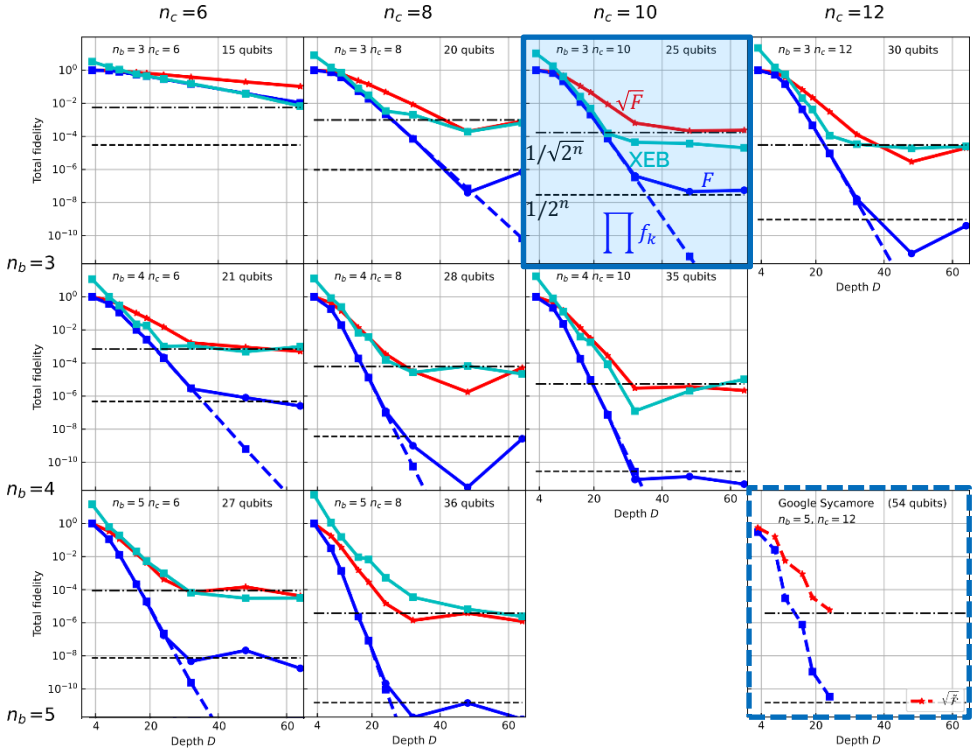
- Strange?

In the chaotic limit (infinite depth):

$$F = \frac{1}{2^n}$$

$$\mathrm{XEB} = \frac{1}{2^{n/2}}$$

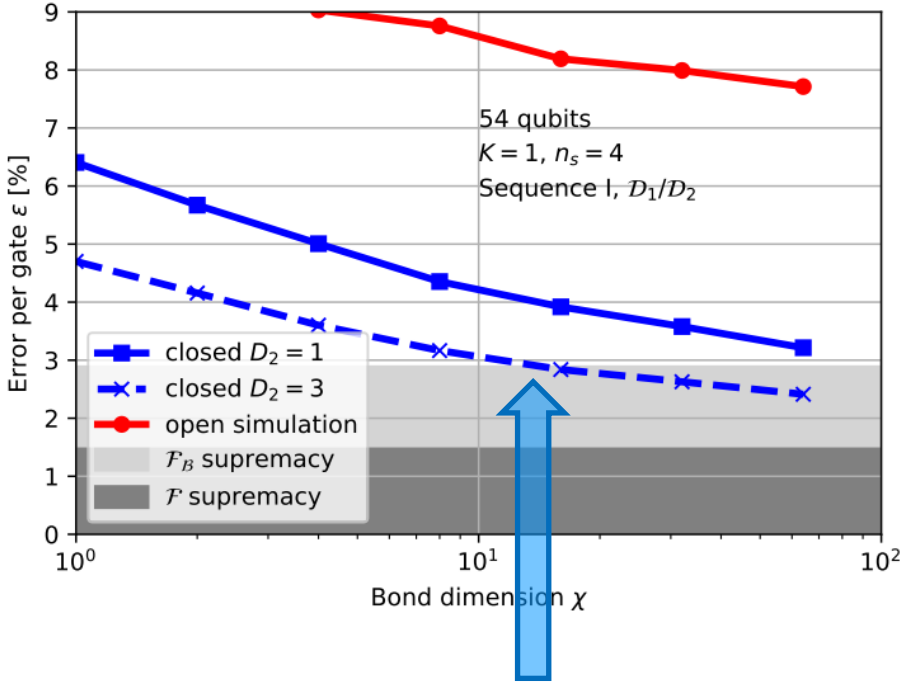
Numerical evidence for finite depths:



Atos

Closing the supremacy gap

Error per gate: $\varepsilon = 1 - \tilde{F}^{1/N_{2g}}$

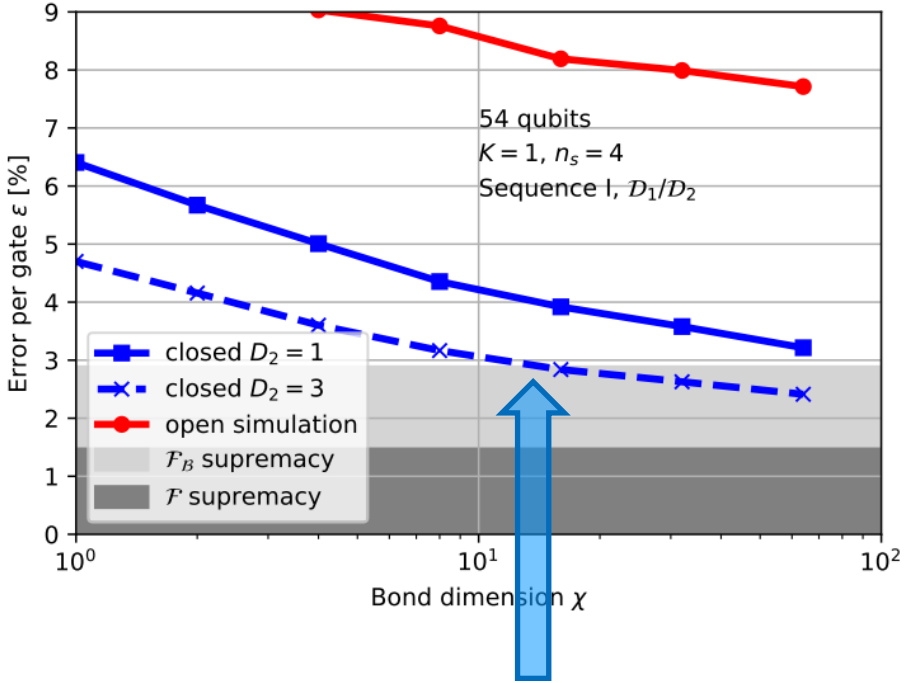


Better XEB error rate than Google Sycamore

Atos

Closing the supremacy gap

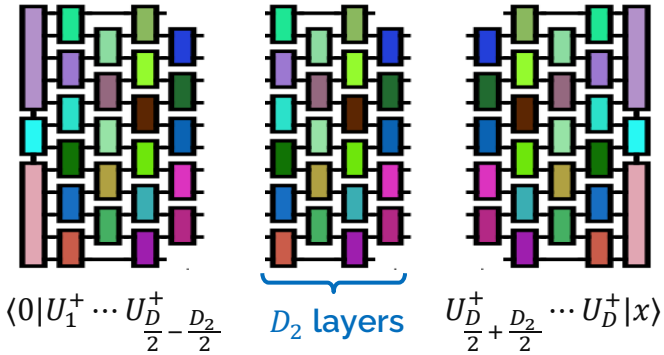
Error per gate: $\varepsilon = 1 - \tilde{F}^{1/N_{2g}}$



Better XEB error rate than Google Sycamore

Closed (one amplitude) vs open (all):

- Open: most powerful. **Strong simulation**.
- Closed: can contract from "both sides". **Lower compression losses!**



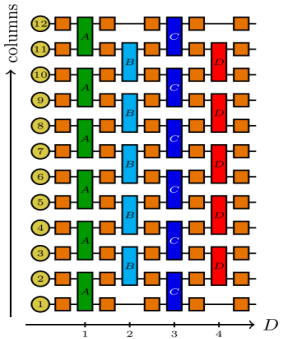
"Closed" easily usable for "weak" simulation: Bravyi et al '21.

Here (near-chaotic distribution): Could use Metropolis-Hastings: 70% acceptance rate.

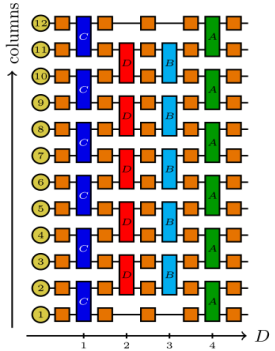
Atos

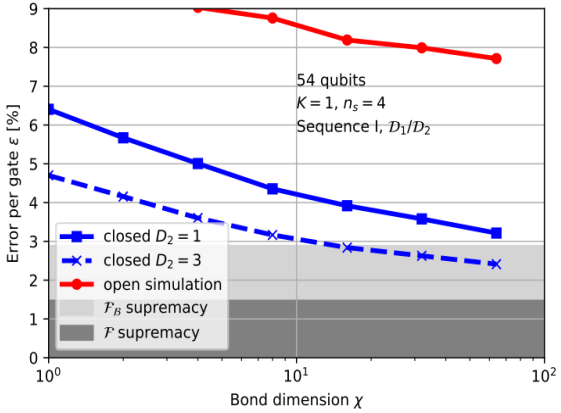
The influence of the type of circuit

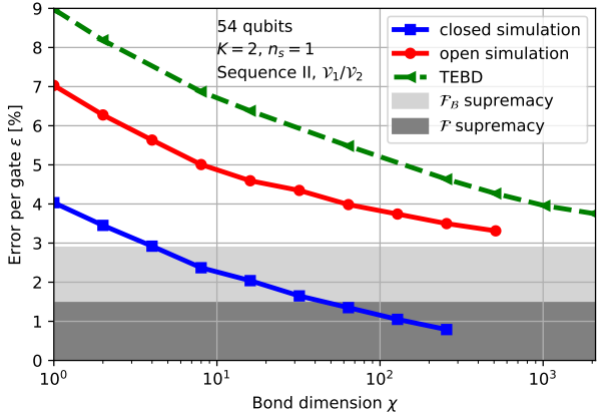
The supremacy sequency



An easier sequence

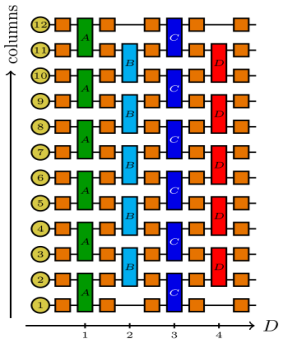




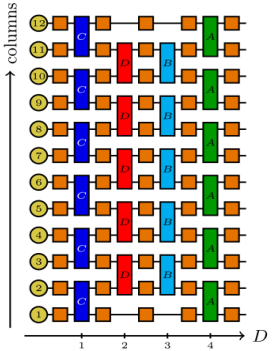


The influence of the type of circuit

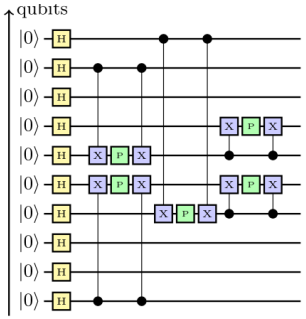
The supremacy sequency

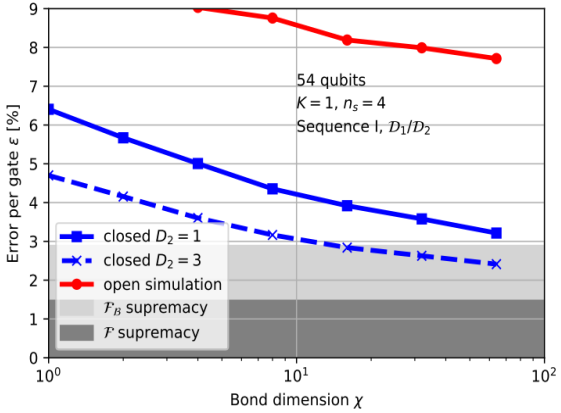


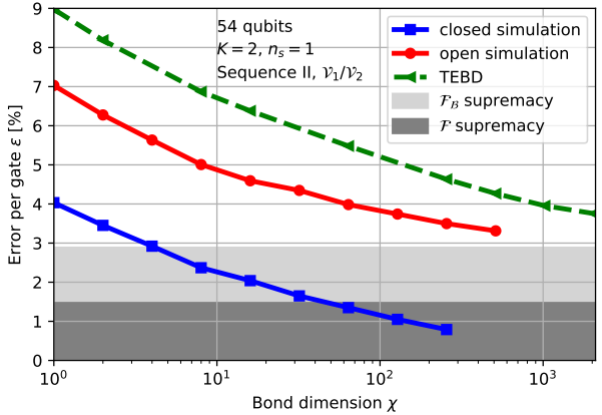
An easier sequence

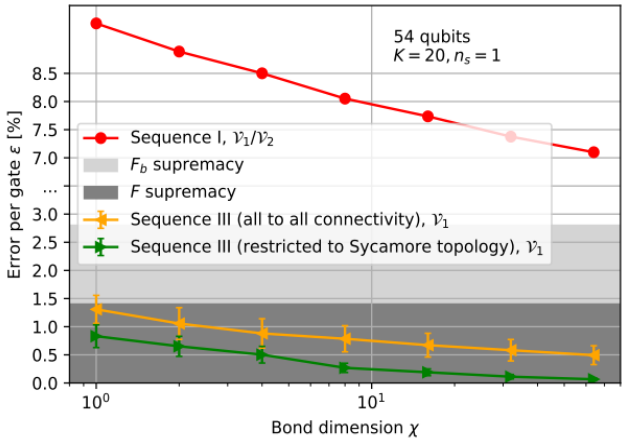


A “useful” sequence





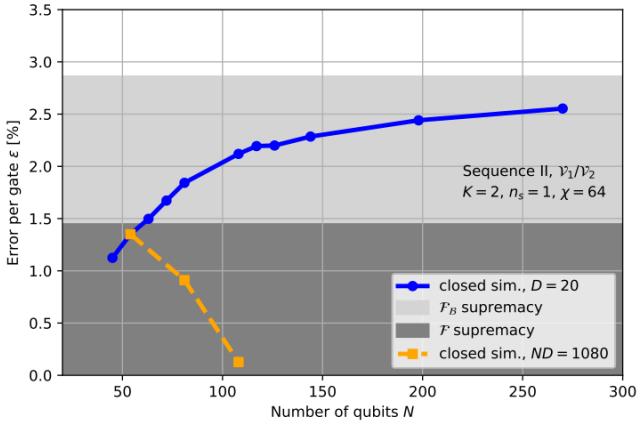




A scalable method: what happens when increasing the qubit count?

Fixed depth D:

- Error per gate increases… then stagnates:



But more gates, XEB decreases…: must increase N_{samples} to reduce variance.

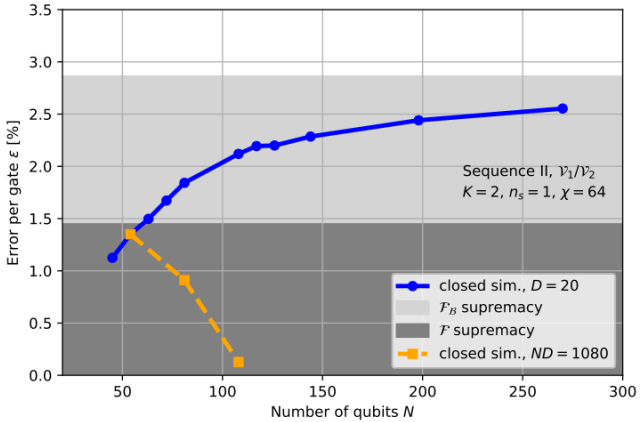
Keep nD fixed (fixed XEB: fixed experimental time!):

- better and better error per gate!

A scalable method: what happens when increasing the qubit count?

Fixed depth D:

- Error per gate increases… then stagnates:



But more gates, XEB decreases…: must increase N_{samples} to reduce variance.

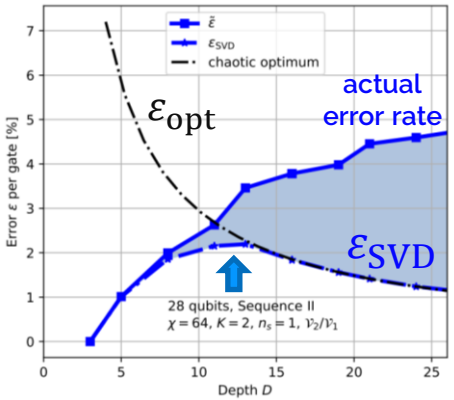
Keep nD fixed (fixed XEB: fixed experimental time!):

- better and better error per gate!

How to understand the stagnation?

Can compute "optimal" error rate after SVD compression:

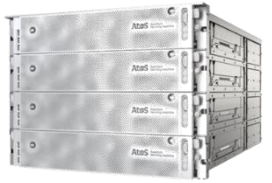
$$\varepsilon_{\text{opt}} = \frac{1}{D}\left(\log 2 - \frac{\log 4\chi}{2N}\right)$$



when ε_{SVD} reaches ε_{opt} [~chaotic limit], actual error rate deviates from ε_{SVD}

Conclusions

Available on
Atos QLM
("QPEG")



Google's game can be won by a classical simulator

- Key: leverage 3 different simulations methods:
  Schrödinger, tensor-network, Matrix Product State
- Very basic implementation (no sophisticated contraction)

Beware of "large Hilbert space fallacy": Structure matters!

- MPS: capture true difficult quantity: entanglement
- This is the target for quantum advantage!

Qubit quality is crucial! Recent progress in QEC @ETH, Google, Quantinuum…

XEB flaws: not scalable, not 'useful' task

- Q-score (Martiel, TA, Allouche '20): use QAOA MaxCut as benchmark

Atos