達摩院

ALIBABA DAMO ACADEMY

Classical Simulation of Quantum Supremacy Circuits

Cupjin Huang, Alibaba Quantum Laboratory
June 2020

arXiv:2005.06787 : [HZN+20]
arXiv:1805.01450 : [CZH+18]
arXiv:1907.11217 : [ZHN+19]

Classical Simulation of Quantum Supremacy Circuits

Cupjin Huang, Fang Zhang, Michael Newman, Junjie Cai, Xun Gao, Zhengxiong Tian, Junyin Wu, Haihong Xu, Huanjun Yu, Bo Yuan, Mario Szegedy, Yaoyun Shi, Jianxin Chen

Classical Simulation of Intermediate-Size Quantum Circuits

Jianxin Chen, Fang Zhang, Cupjin Huang, Michael Newman, Yaoyun Shi

Alibaba Cloud Quantum Development Platform: Large-Scale Classical Simulation of Quantum Circuits

Fang Zhang, Cupjin Huang, Michael Newman, Junjie Cai, Huanjun Yu, Zhengxiong Tian, Bo Yuan, Haihong Xu, Junyin Wu, Xun Gao, Jianxin Chen, Mario Szegedy, Yaoyun Shi

Main results

- Google [Arute'19]: 200s quantum vs. 10,000 years classical
- Our result: 10,000 years -> < 20 days
- Core contribution: an efficient algorithm for **tensor network contraction**

Quantum supremacy is a process, without an unequivocal "first" demonstration

Contents

CONTENTS

達摩院

ALIBABA DAMO ACADEMY

01 | Introduction

Google's claim of quantum supremacy

Quantum Supremacy

“quantum” regime

classical regime

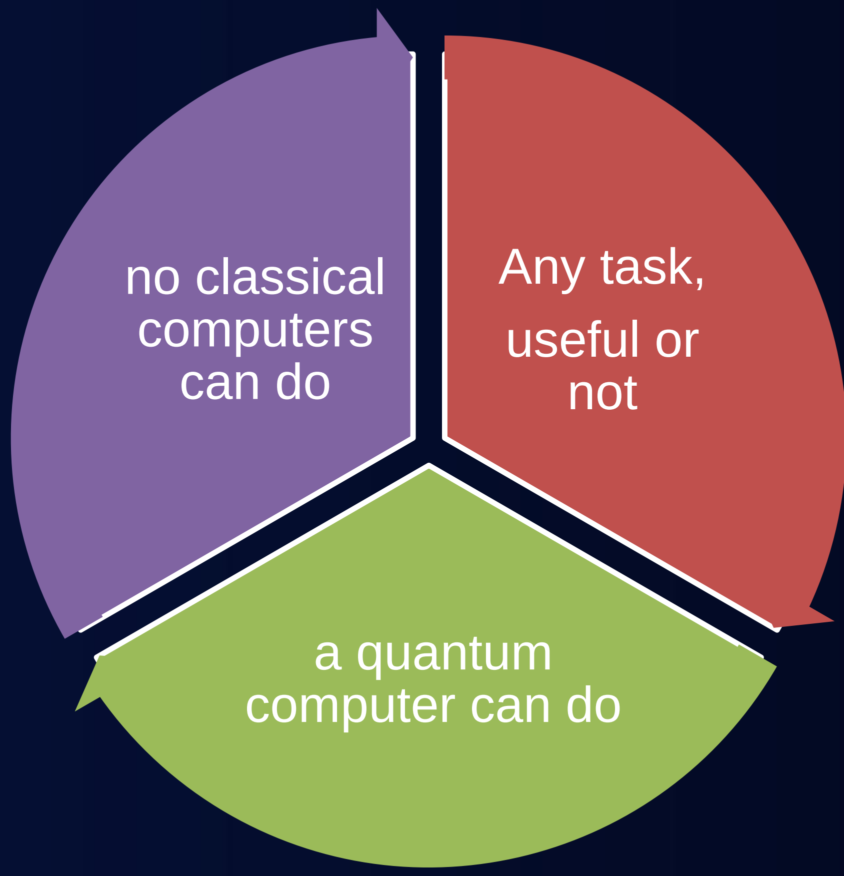
Quantum “supremacy” ~100 qubits

An early demonstration of the usefulness of quantum computing

Definitely before fault-tolerance;
even before NISQ

Quantum supremacy: 3 components

Quantum computers doing *some task* that classical computers *cannot* do



Quantum supremacy proposals

Theoretical proposals:

- Boson Sampling [AA`10]
- QAOA [FH `16]
- IQP circuits [BJS`11]
- Random circuit sampling [BFNV`18]
- …

Experiment(s):

Google RCS [Arute+19]

Google random circuit sampling: basics

達摩院
ALIBABA DAMO ACADEMY

A distribution of quantum circuits $\mathcal{D}$ over a circuit family $\mathcal{C}$

Execute circuit $U \leftarrow \mathcal{D}$; sample on the computational basis

Ideal distribution p_U : $\Pr[X = x] = |\langle x|U|0\rangle|^2$

In reality: sample from a distribution $\tilde{p}_U$ that is "close" to p_U

Multiplicative error? Additive error? Linear XEB

Linear cross entropy benchmarking [Arute+, 19]

$$F(\tilde{p}_U, p_U) \coloneqq 2^n \cdot \mathbb{E}_{X\sim\tilde{p}_U}[p_U(X)] - 1 = 2^n \left(\sum_{x\in\{0,1\}^n} p_U(x) \cdot \tilde{p}_U(x) \right) - 1$$

Replace expectation by sample mean:

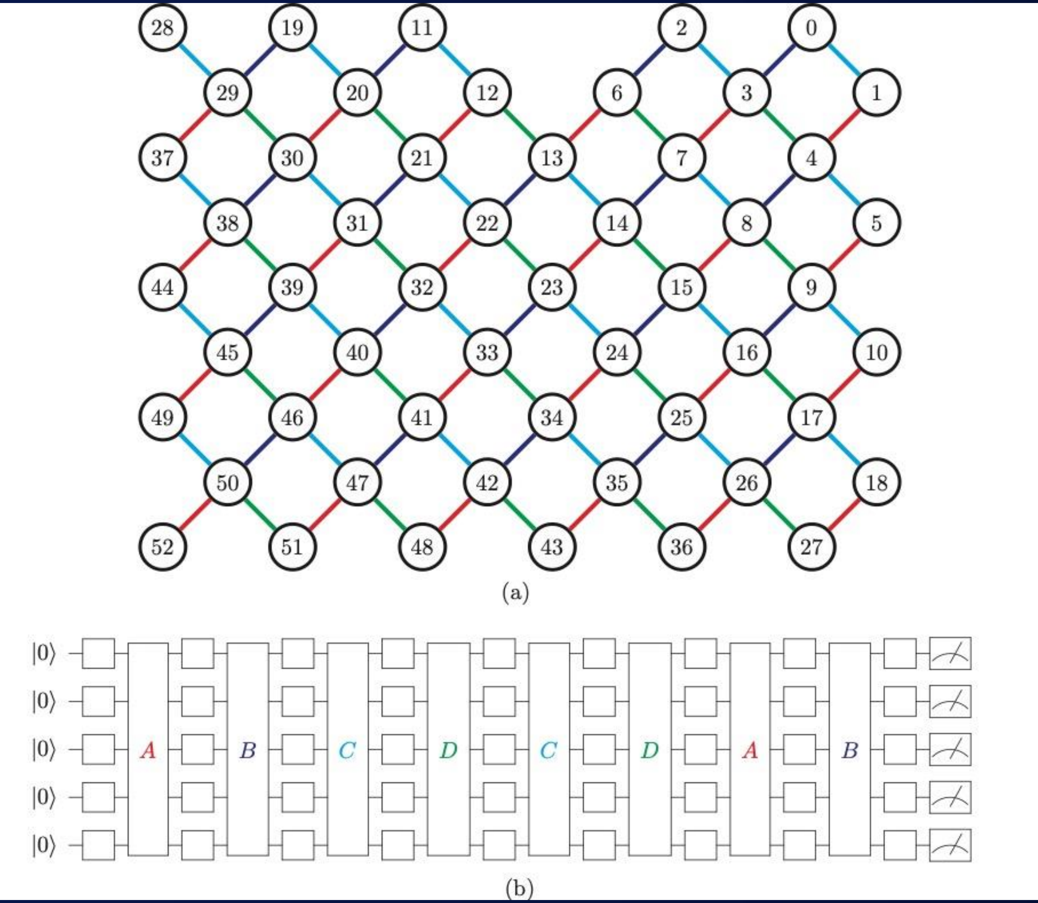
$$F_{\mathcal{X}} \coloneqq 2^n \left(\frac{1}{|\mathcal{X}|} \sum_{x\in\mathcal{X}} p_U(x) \right) - 1$$

- Linear w.r.t. $\tilde{p}_U$
- $F(U_n, p_U) = 0$;
- $F(p_U, p_U) = 1$, under Porter-Thomas assumption
- $F(\cdot, p_U)$ DOES NOT range in $[0,1]$

Task: output samples $\mathcal{X}$ such that $F_{\mathcal{X}} > 0$

Google's experiment [Arute+, 19]

$n = 53$ qubits; $m = 20$ layers of 2-qubit gates
2-qubit gates calibrated to the best; one qubit gate chosen randomly

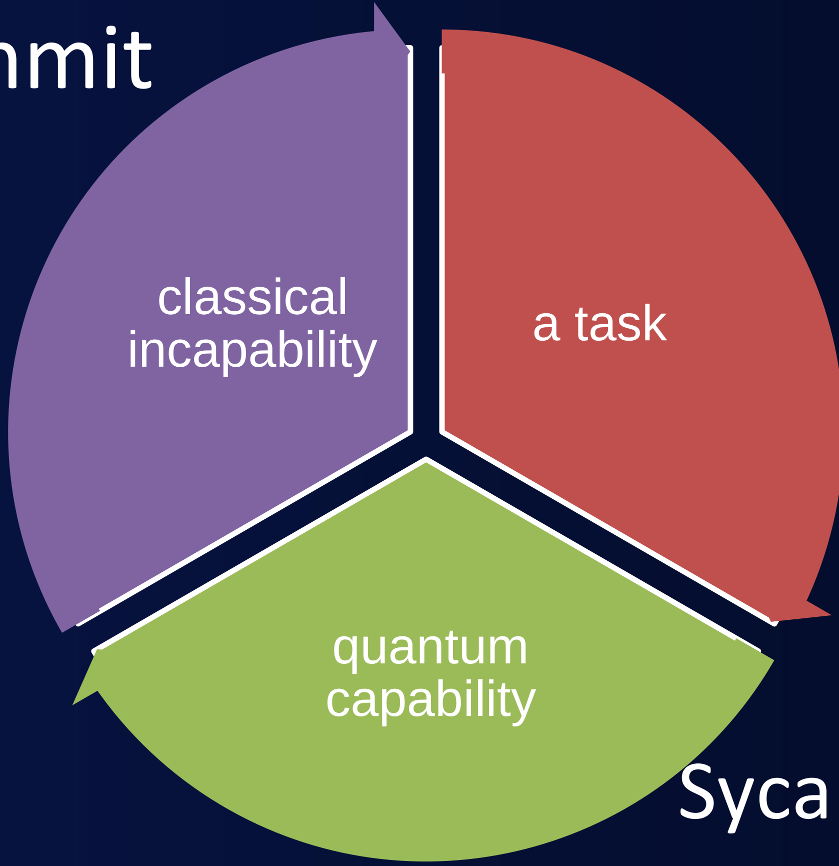


1 million samples; F = 0.2% in 200s

Google's claim of quantum supremacy

10,000 years on Summit

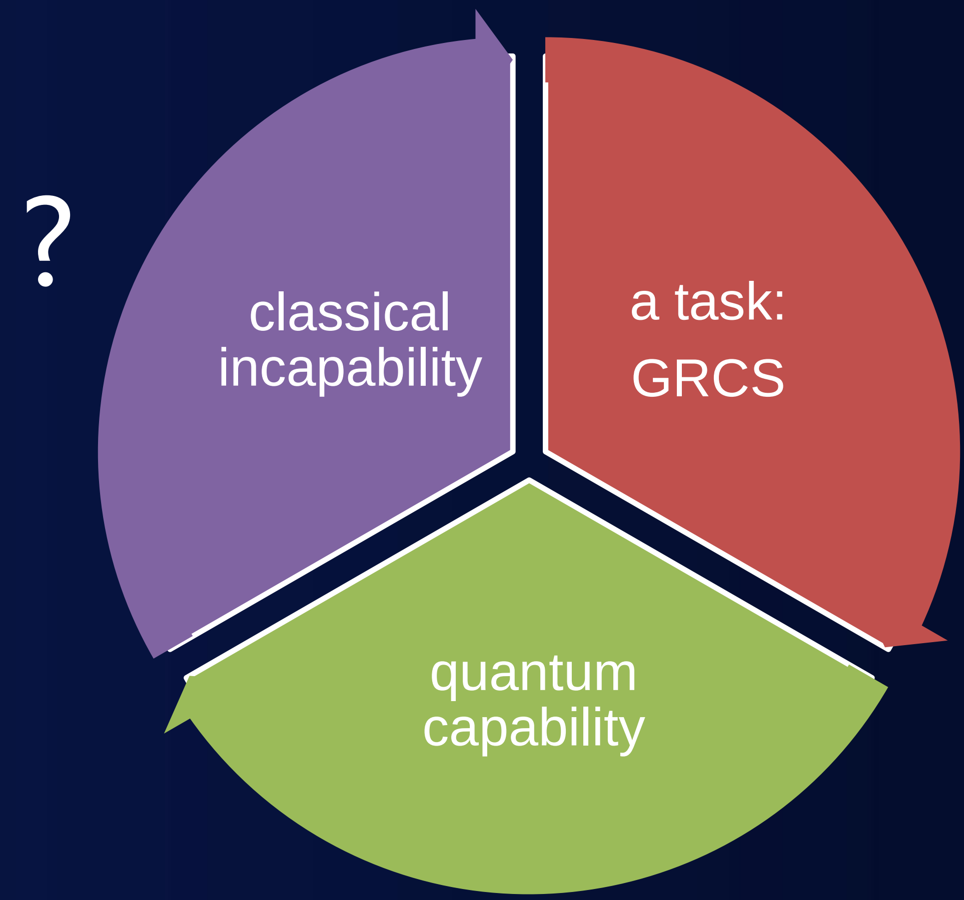
GRCS/ linear XEB



Sycamore chip: 53 qubits, 200s

Our result

10,000 years -> < 20 days

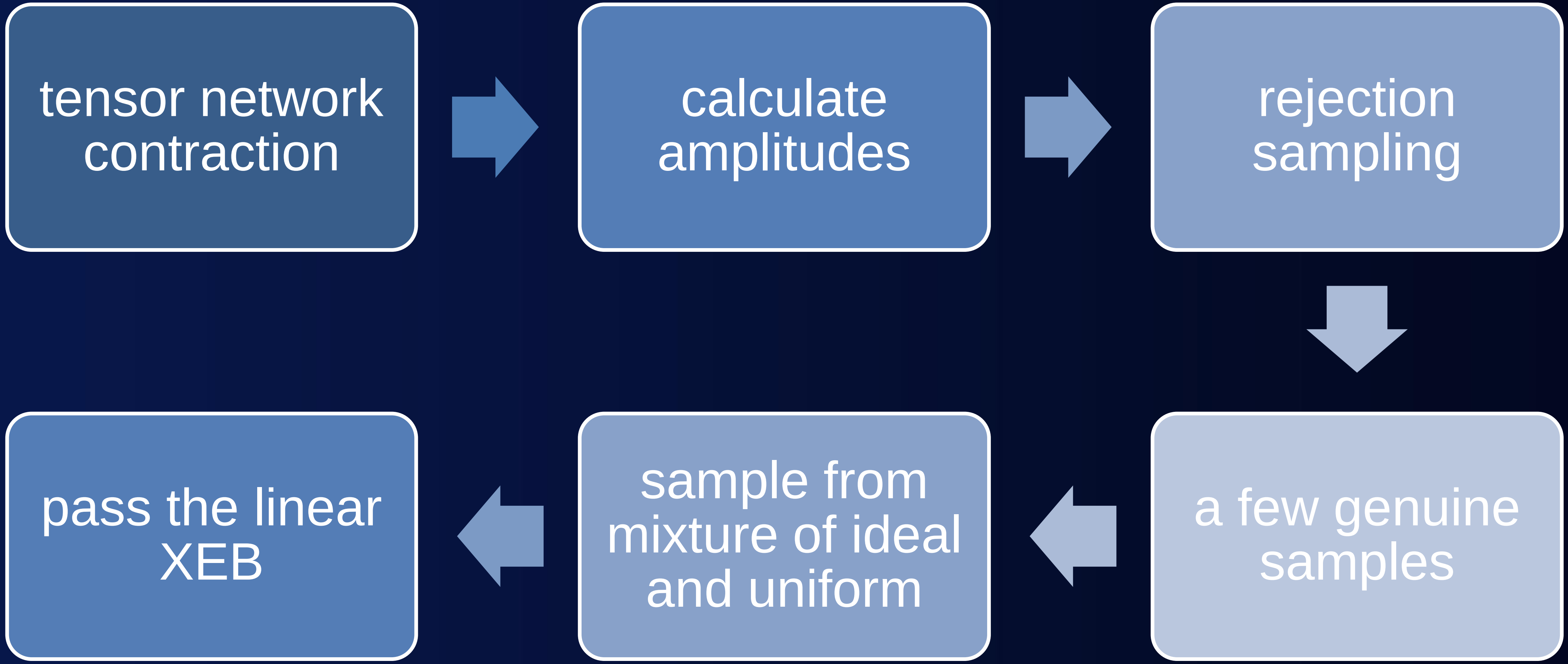


02

Results

Our result: 20 days

Reducing linear XEB to tensor network contraction [MFIB`18]



How many samples are needed?

- **Sample from mixture** of p_U and U_n: $p_f = f \cdot p_U + (1-f) \cdot U_n$ [Villalonga+, 19]
- Linear XEB = f
- With M samples, only need $f \cdot M$ "genuine" samples; uniformly chosen rest

Sample $10^6 \times 0.2\% \; = \; 2000$ "genuine" samples

Reduction to probability calculation

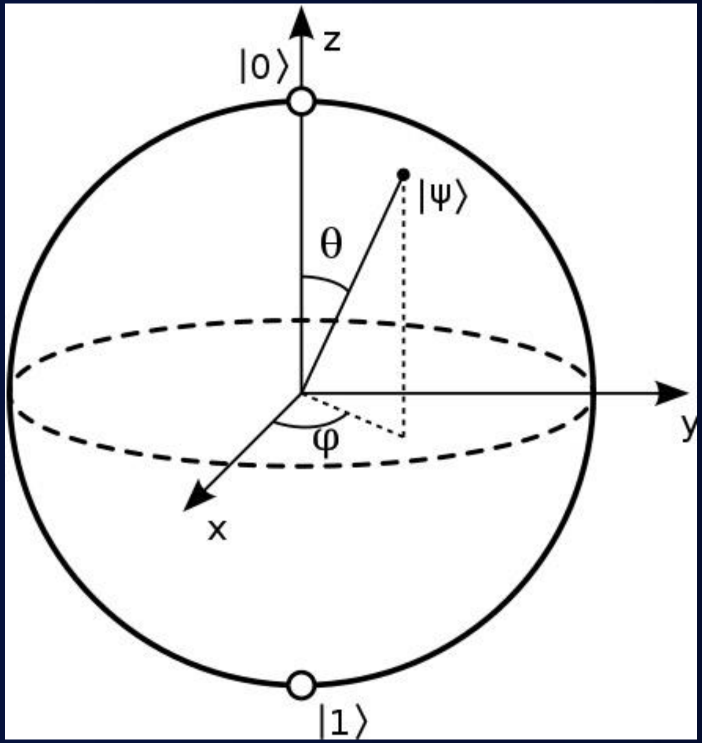
Rejection sampling:

1. Uniformly sample $X \sim \{0,1\}^n$
2. Accept X with probability $\frac{p_U(X)}{\max_{x\in\{0,1\}^\wedge n} p_U(x)}$
3. Allowing distortion: accept with probability $\frac{\max\{p_U(X),K*2^{-n}\}}{K*2^{-n}}$
4. Compute $M \gg K$ samples per batch for certainty

Sampling <- computing $M = 64$ probabilities per sample

Porter-Thomas statistics

Random circuit sends $|0\rangle$ to a typical point in the hypersphere



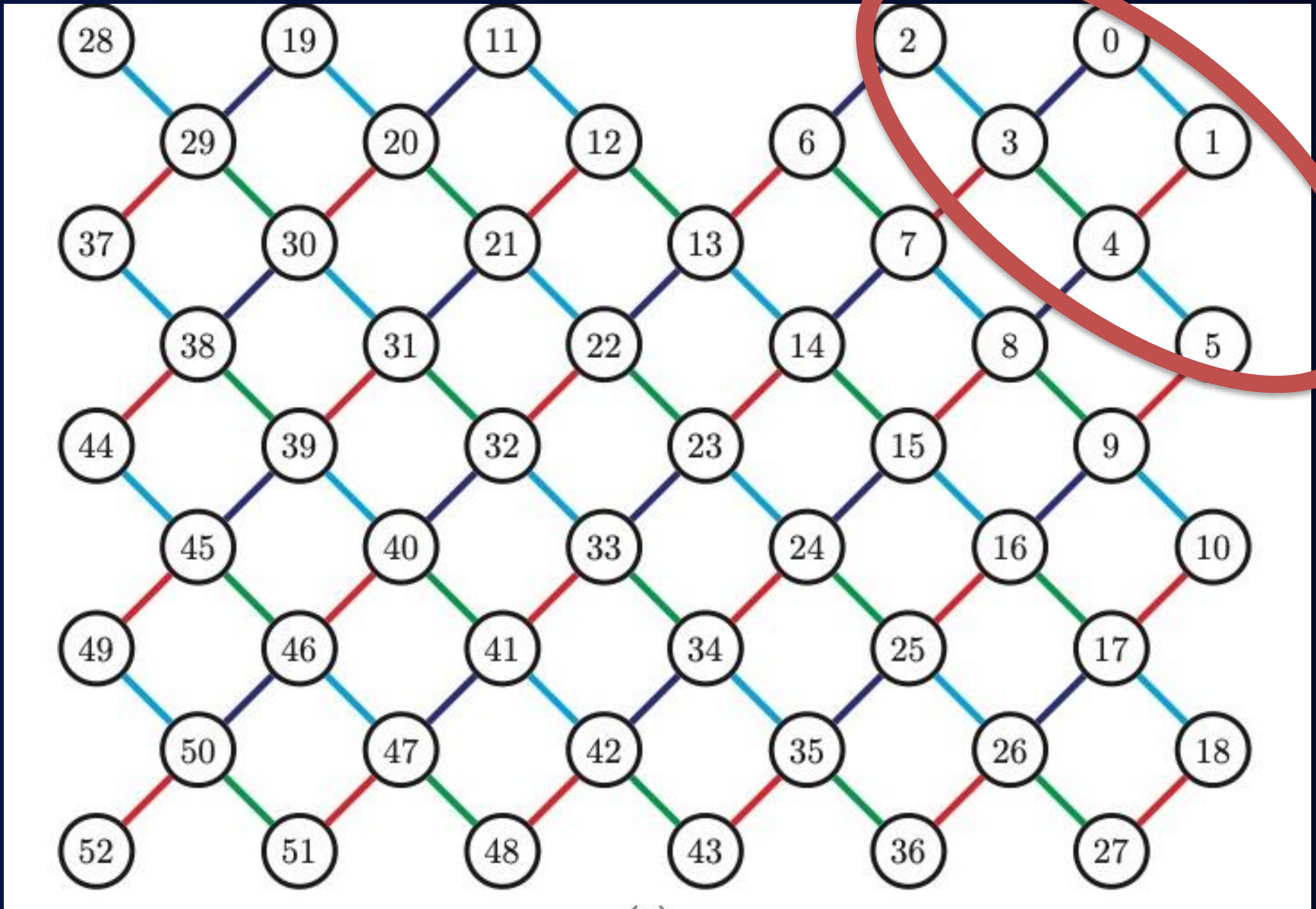
Probability follows the Porter-Thomas statistics

Most bitstrings are around average; enables **rejection sampling**

Reduction to tensor network contraction

Efficiently calculate 64 amplitudes *at once*:

randomly post-select on 47 qubits ->



(a)

<- full amplitudes on 6 qubits

Summary

tensor network contraction → calculate 64 amplitudes → rejection sampling → 2000x genuine samples → sample from mixture of ideal and uniform → pass the linear XEB

03

Techniques

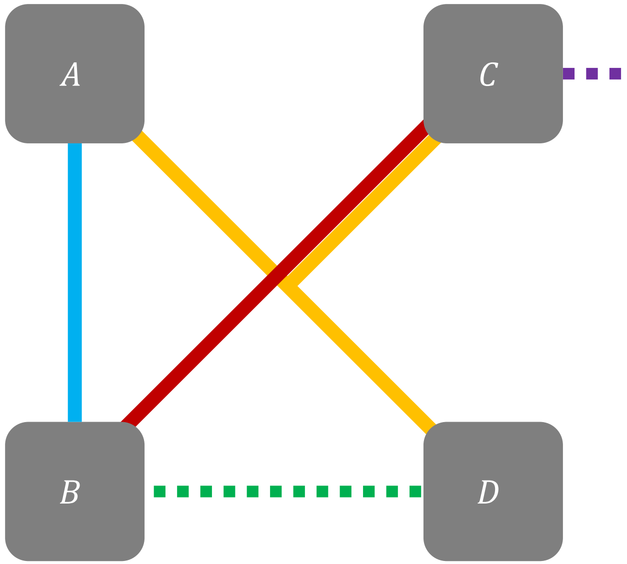
Efficient tensor network contraction

Tensors & tensor networks

Tensors: Multi-dimensional arrays

- Vectors are 1-tensors; matrices are 2-tensors

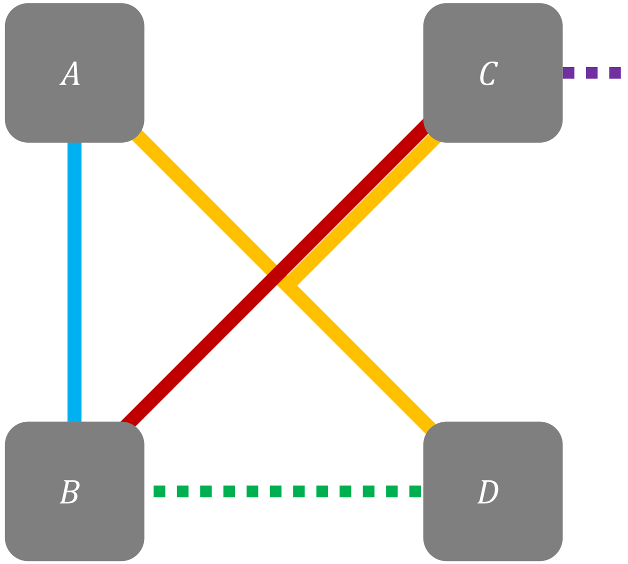
Tensor networks: tensors where indices merged together are identified (sometimes summed up)



$$T_{be} := \sum_{a,c,d} A_{ac} B_{abd} C_{cde} D_{bc}$$

Natural representation for quantum circuits (amplitudes)

Tensors network contraction



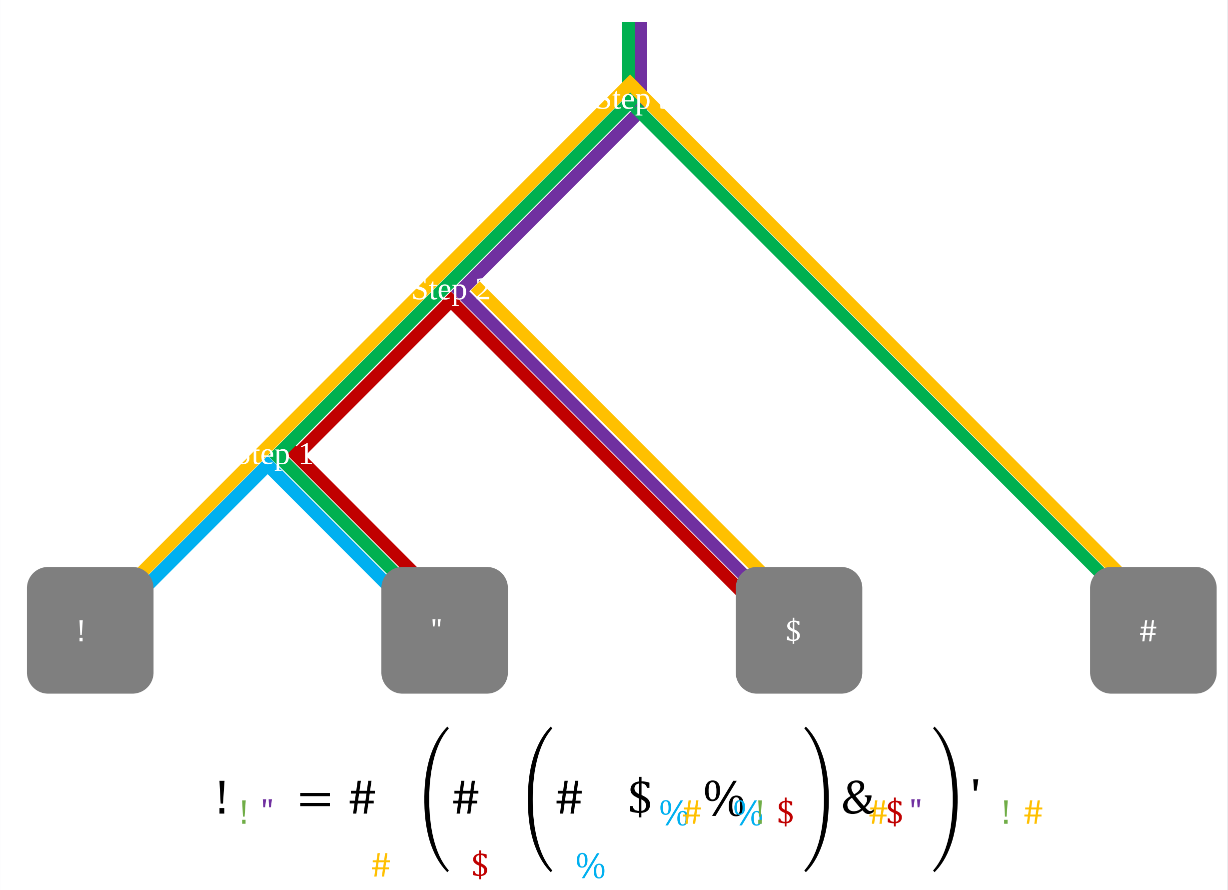
$$T_{be} := \sum_{a,c,d} A_{ac} B_{abd} C_{cde} D_{bc}$$

- Counting problem; #P-complete
- Exponential time/space in worst case

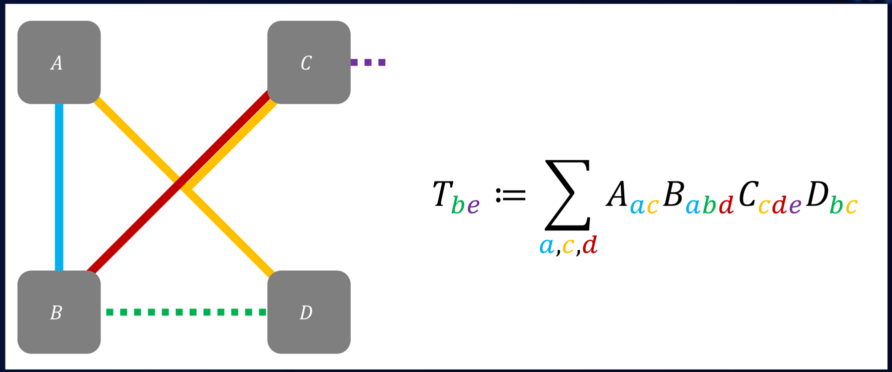
Parallel & space/time-efficient algorithm

Experiment on GPU cluster

Contraction of tensor networks: the Schrödinger way



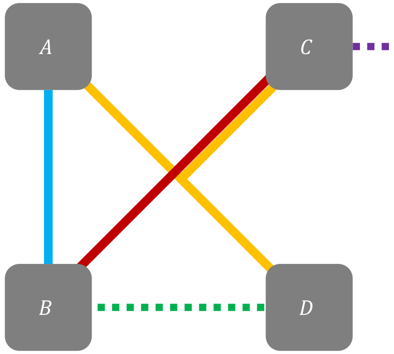
State-vector update ->
Sequential pairwise multiplication



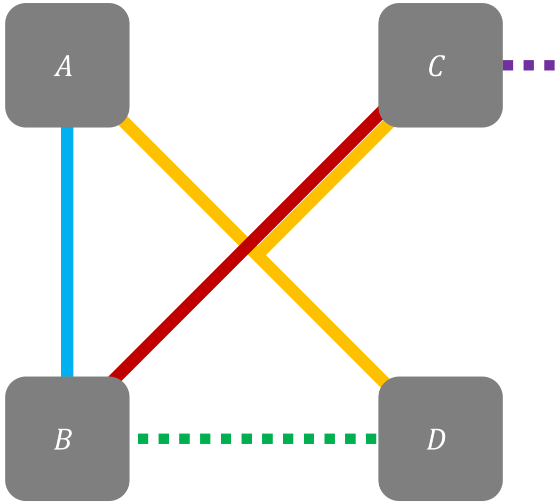
Sequential pairwise contraction: Binary contraction tree

- Significant improvement for shallow quantum circuits
- Hard to find a good tree
- Sequential algorithm
- Hard space lower bound

Contraction of tensor networks: the Feynman way



$$T_{be} := \sum_{a,c,d} A_{ac} B_{abd} C_{cde} D_{bc}$$



$$T_{be} := \sum_{a,c,d} A_{ac} B_{abd} C_{cde} D_{bc}$$

Feynman path integral ->
Enumeration over indices

Complete Feynman path integral:

- Time complexity: 2^m
- Space complexity: $poly(n, m)$

- Good space complexity; highly parallel

- Prohibitive time complexity

Contraction scheme: hybriding Schrödinger and Feynman

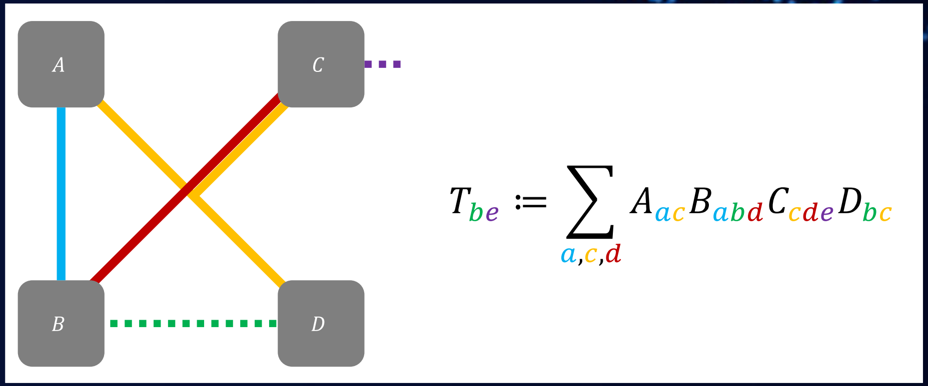
達摩院
ALIBABA DAMO ACADEMY

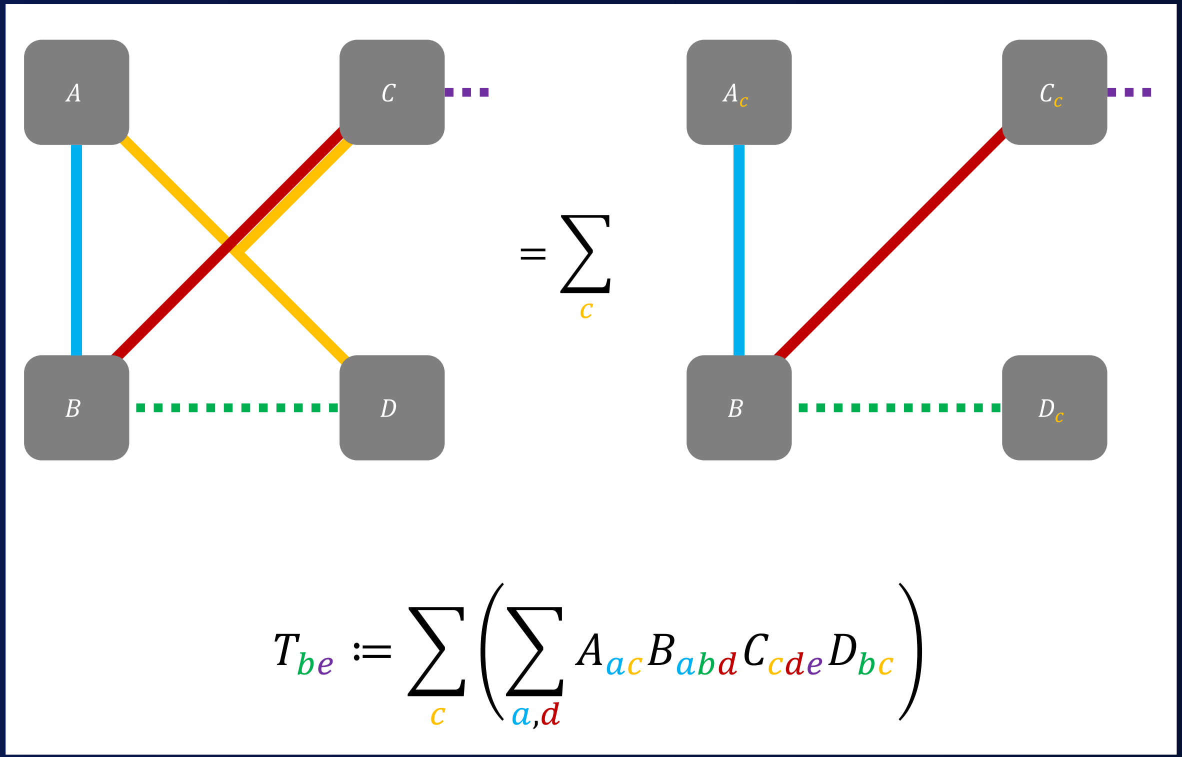
Sequential → Parallel

Method	Schrödinger	???	Feynman
Time	$O(n \cdot 2^n)$	???	$\Omega(2^m)$
Space	$\Omega(2^{cw(T)})$	???	$O^*(1)$

partial parallelization

Contraction scheme: hybriding Schrödinger and Feynman [CZH18]





- Each assignment yields a subtask
- subtasks have identical structures
- results summed up at the end

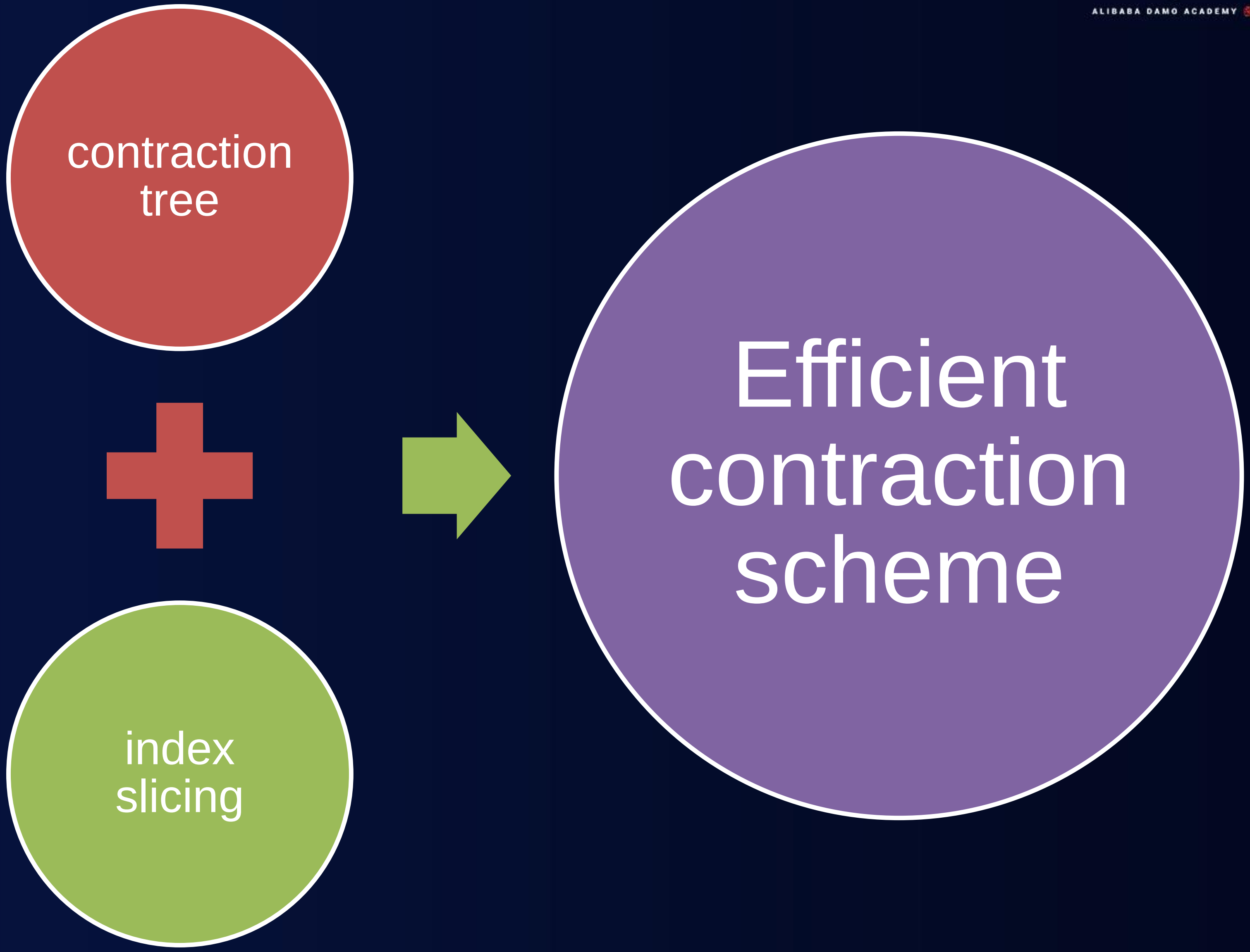
- Trade space for time; no hard limit

Single-time tree constuction + embarrassing parallelism

Goal: minimize time complexity, s.t. space constraint

Putting it all together

Combinatorial optimizations



Tensor network contraction : timeline

[MS'08]
- Tree-decomposition
- Minimizing highest-cost tree node

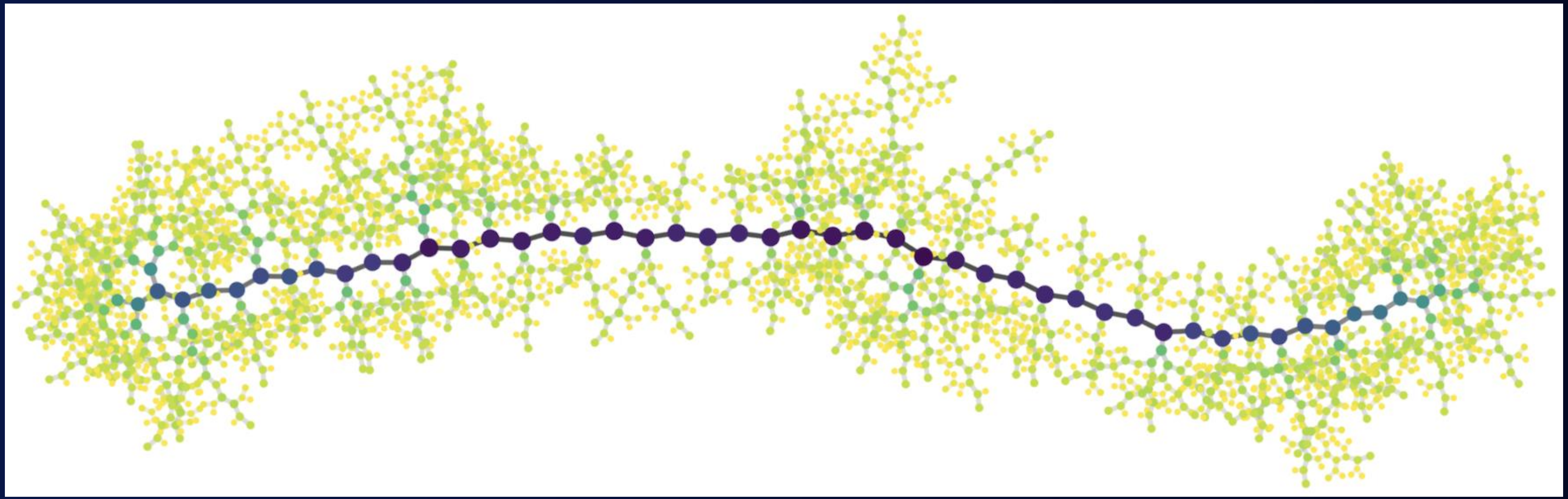
[BIS+'17]
- Hypergraph formulation
- Simplifying diagonal gates

[CZH+'18]
- Dynamic slicing
- Parallelization by index slicing

[GK'20]
- Hypergraph decomposition
- Minimizing overall cost of the tree

[HZN+'20]
- Stem optimization
- Reducing parallelization overhead on stem

Finding good contraction schemes: stems & branches



Dominating nodes come in a short path

Putting it all together

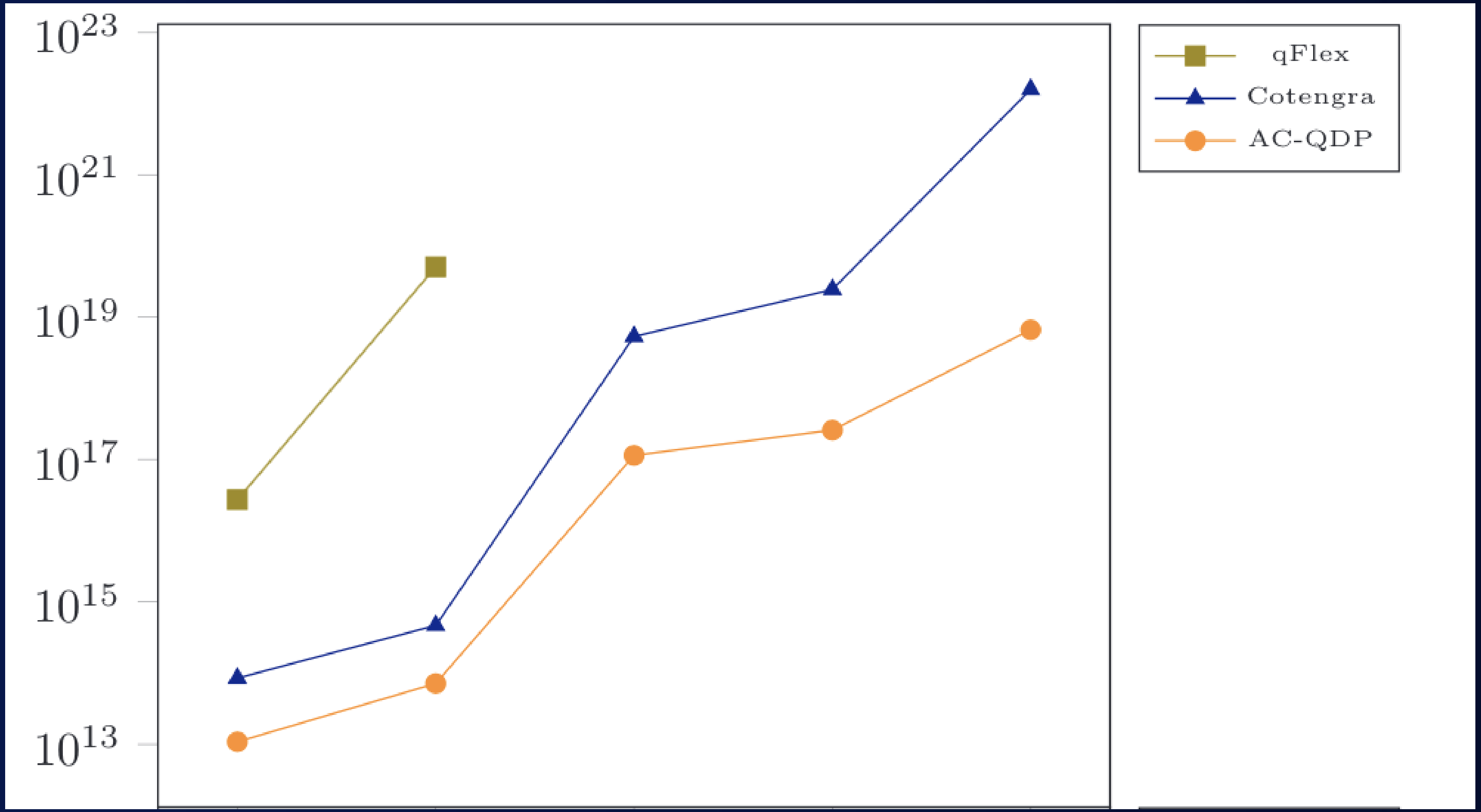
Find a contraction tree → Efficient parallelization

Construction of the stem
Hypergraph decomposition
Optimization

Choosing indices to slice
Local optimization on stem

Results

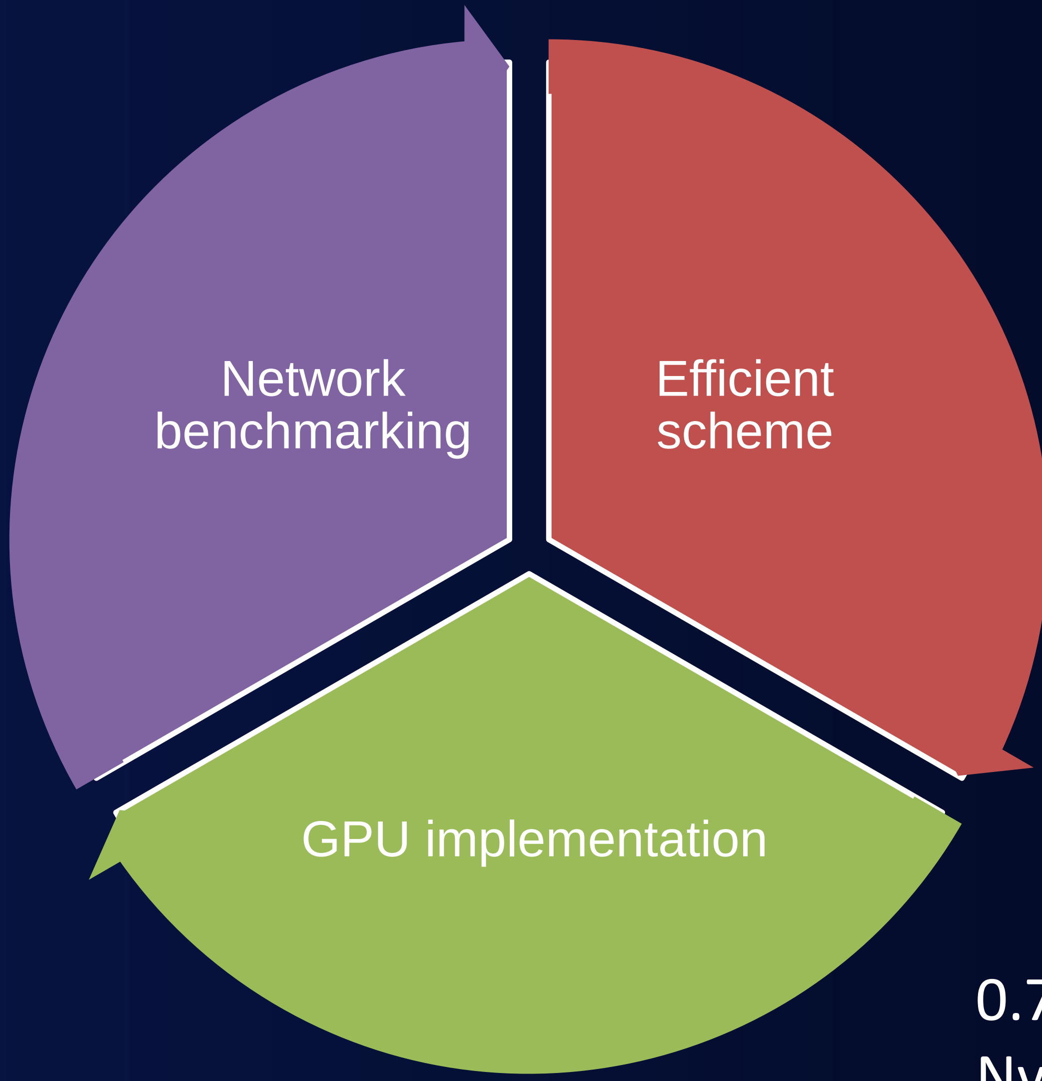
For m=20, total time complexity 6.66e18 FLOPs per sample; space complexity 4GiB
25 sliced indices; individual subtask time complexity 1.98e11 FLOPs



$> 10^3$ improvement
for m=20 w.r.t. [GK'20]

Experimental verification: 19.3 days

No significant latency observed on Alibaba Cloud



2000 * 2**25 subtasks, each with 1.98e11 FLOPs

0.7s per subtask on Nvidia V100;
GPU efficiency ~15%

04

Discussion

Comparison with other results, and discussion on supremacy

	Exact algorithms	Approximate algorithms
Experiments	Few amplitudes [Ours, GK'20, Arute+'19] 20 days	SFA [MFIB+'18] 10,000 years
Proposals	Full state vector [Pednault+'19] 2.5 days	MPS-based [Zhou+'20] ???

The SFA algorithm [MFIB,`18]

Break down the full state into superposition of product states, and pick randomly from there

- Suitable for target fidelity
- One-shot computation; does not scale with number of samples
- Number of subtasks is prohibitively large: 2**66*0.2% >> 2**25*2000

Overall fidelity ~ 0.2% in 10,000 years on Summit

Full state vector approach [Pednault+,`19]

Full state-vector update, done with secondary memory: 2.5 days

- Computation only needs to be done once

- Requires huge storage; hard to extend to more qubits
- Majority of time spent on memory I/O

More of a thought-experiment: no actual experiment / code supporting

MPS-based approach [Zhou+,`20]

Approximately simulate the quantum circuit using MPS

- Cost grows polynomially up to some fidelity threshold, and then exponentially
- Very efficient when fidelity requirement is lower than threshold

Variants of Google circuit considered; the quantum supremacy circuit not yet attempted

How far can our approach go?

Inefficiencies observed in the algorithm & the implementation: over 100x

Flexible trade-off between time and space

Heavy preprocessing to find trees & slices

Scales linearly with number of samples

Summary

Efficient parallelized tensor network contraction;
Quantum supremacy task in < 20 days

The boundary between classical and quantum is moving & blurry

Improvements on both classical & quantum

Algorithmic/ complexity-theoretic understanding of the task

Focusing on making both classical and quantum useful!

thanks