Editors' Suggestion

Simulation of Quantum Circuits Using the Big-Batch Tensor Network Method

Feng Pan[1,2] and Pan Zhang[1,3,4,*]

[1]*CAS Key Laboratory for Theoretical Physics, Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China*
[2]*School of Physical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China*
[3]*School of Fundamental Physics and Mathematical Sciences, Hangzhou Institute for Advanced Study, UCAS, Hangzhou 310024, China*
[4]*International Centre for Theoretical Physics Asia-Pacific, Beijing/Hangzhou, China*

(Received 7 May 2021; revised 30 July 2021; accepted 10 December 2021; published 19 January 2022)

We propose a tensor network approach to compute amplitudes and probabilities for a large number of correlated bitstrings in the final state of a quantum circuit. As an application, we study Google's Sycamore circuits, which are believed to be beyond the reach of classical supercomputers and have been used to demonstrate quantum supremacy. By employing a small computational cluster containing 60 graphical processing units (GPUs), we compute exact amplitudes and probabilities of 2×10^6 correlated bitstrings with some entries fixed (which span a subspace of the output probability distribution) for the Sycamore circuit with 53 qubits and 20 cycles. The obtained results verify the Porter-Thomas distribution of the large and deep quantum circuits of Google, provide datasets and benchmarks for developing approximate simulation methods, and can be used for spoofing the linear cross entropy benchmark of quantum supremacy. Then we extend the proposed big-batch method to a full-amplitude simulation approach that is more efficient than the existing Schrödinger method on shallow circuits and the Schrödinger-Feynman method in general, enabling us to obtain the state vector of Google's simplifiable circuit with $n = 43$ qubits and $m = 14$ cycles using only one GPU. We also manage to obtain the state vector for Google's simplifiable circuits with $n = 50$ qubits and $m = 14$ cycles using a small GPU cluster, breaking the previous record on the number of qubits in full-amplitude simulations. Our method is general in computing bitstring probabilities for a broad class of quantum circuits and can find applications in the verification of quantum computers. We anticipate that our method will pave the way for combining tensor network–based classical computations and near-term quantum computations for solving challenging problems in the real world.

DOI: 10.1103/PhysRevLett.128.030501

An essential question in near-term quantum computation is whether programmable quantum devices are able to perform beyond the ability of classical computations in specific computational tasks. An ideal example is sampling from a random quantum circuit [1–8]. In 2019, Google's quantum computing group released the Sycamore circuits [1] with $n = 53$ qubits and demonstrated the "quantum supremacy" [8,9] by showing that they can experimentally solve the noisy sampling task from the output distribution $P_U(\boldsymbol{s})$ of the Sycamore circuit U in the computational basis $|\boldsymbol{s}\rangle$, which would cost 10 000 years on modern supercomputers.

However, despite the great success of the experiments, we note that the critical basis for the assertion of quantum supremacy, the accuracy of sampling in terms of the linear cross-entropy benchmark (XEB), and the running time of classic simulations still leave some space for further discussions. First, Google was only able to compute the exact XEB values for circuits for the simplifiable circuits (with EFGH sequence) with $m = 14$ cycles and estimated the XEB values for the supremacy circuits (with ABCDCDAB sequence) using extrapolations [1]. This means that the fidelity of samples generated from the quantum supremacy circuits is not verified. Second, the estimate of computational time was based on a classic simulation method: the Schrödinger-Feynman algorithm [1,2,10]. Potentially there could be new algorithms that are more efficient than the algorithm used by Google, demanding much less computational time than the estimate.

There are basically two kinds of methods for simulating quantum circuits. The first kind stores and evolves the full quantum state vector ψ and is known as the Schrödinger method. Since bitstring probabilities $P_U(\boldsymbol{s}) = |\langle\psi|\boldsymbol{s}\rangle|^2$ are known, sampling from the bitstring space is easy and the computational complexity is linear in the number of depth m; thus, it is very efficient for quantum circuits with a small number of qubits. Google used this method for simulating circuits up to 43 qubits [1] with the largest instance run on the Jülich supercomputer with 100 000 cores and 250 terabytes memory. However, for a large number of qubits, the method suffers from an exponential space complexity. The largest instance that has been simulated has 49 qubits [11], beyond which the size of total RAM becomes the bottleneck even with supercomputers. IBM has justified theoretically that the 53-qubit state vector of the Sycamore circuits can be stored and evolved if one could employ not only all the RAM but also all the hard disks of the Summit supercomputer. However, the experiment has not been done yet. To address the issue of exponential space complexity, Google used the Schrödinger-Feynman algorithm [2,10] which breaks the circuits into two parts, connected using Feynman path integrals, and each part is simulated using the Schrödinger method. Based on this method, Google estimated that simulating the Sycamore circuits with 20 cycles requires a 10 000-year running time on the Summit supercomputer.

The second kind of method does not store 2^n bitstring probabilities in memory but computes one bitstring probability or a small batch of them based on tensor networks [5,8,10,12–21]. Quantum circuits can be treated as particular tensor networks with unitary constraints. Given an initial state and the bitstring representing the measurements at the end, contraction of the tensor network gives the amplitude of the output bitstring. The space complexity of the tensor network method is controlled by the size of the largest tensor encountered during the contraction, which equals the exponential of treewidth of the line graph corresponding to the tensor network [12]. For shallow circuits where the treewidth is small, the tensor network method is very efficient even for circuits with a large number of qubits [5,14–16,21]. However, the tensor network method is not scalable with the circuit depth because the complexity is usually exponential to the depth and hence very expensive for large circuits with sufficient depth. For the Sycamore circuits with 20 cycles, a recent work [19] estimated that computing probabilities for a batch of 64 bitstrings requires about 833 seconds using a Summit-compatible supercomputer. To the best of our knowledge, so far no work has ever successfully obtained the probability of even one bitstring for the Sycamore circuit with 20 cycles. More seriously, sampling from the bitstring space is difficult for the tensor network method because the computational complexity is proportional to the number of samples one demands. In order to obtain enough bitstrings, e.g., for reaching a XEB value that is comparable to Google's hardware samples, the tensor network contraction has to be repeated many times, making the overall computation intractable.

In this Letter, we propose a tensor network method to obtain a large number of correlated bitstring amplitudes and probabilities at once based on the careful design of subspace to enumerate (or sample) from that we coin as the "big-batch method." It can be regarded as an intermediate between the full-state vector method and the single (or small-batch) amplitude method. It is more efficient than the Schrödinger-Feynman algorithm and the existing tensor-network methods for computing probabilities of a large number of correlated samples from the Sycamore circuits and can be extended to full-amplitude simulations.

Big-batch simulation of quantum circuits.—Our idea is to combine the advantages of the full-state vector method and the single-amplitude tensor network method using a subspace simulation. As depicted in Fig. 1, we separate n qubits in the final state (at the rightmost layer in the figure) into two groups with group sizes n_1 (blue) and n_2 (red), respectively. An arbitrary bitstring $\boldsymbol{s}$ is then represented as a concatenation of partial bitstrings $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, and the probability for the bitstring can be expressed as $P_U(\boldsymbol{s}) = P_U(\boldsymbol{s}_1; \boldsymbol{s}_2)$, the joint probability of $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. With $n_2 = 0$, the method is identical to the single-amplitude estimation approach using tensor networks. With n_2 small, it is essentially the idea that has been explored in [1,10,19], where several qubits at the final state are selected manually and kept open, giving a small batch (typically 64) of amplitudes by a single contraction. However, there are several difficulties for using a large batch size n_2. First, using more open qubits significantly increases the contraction

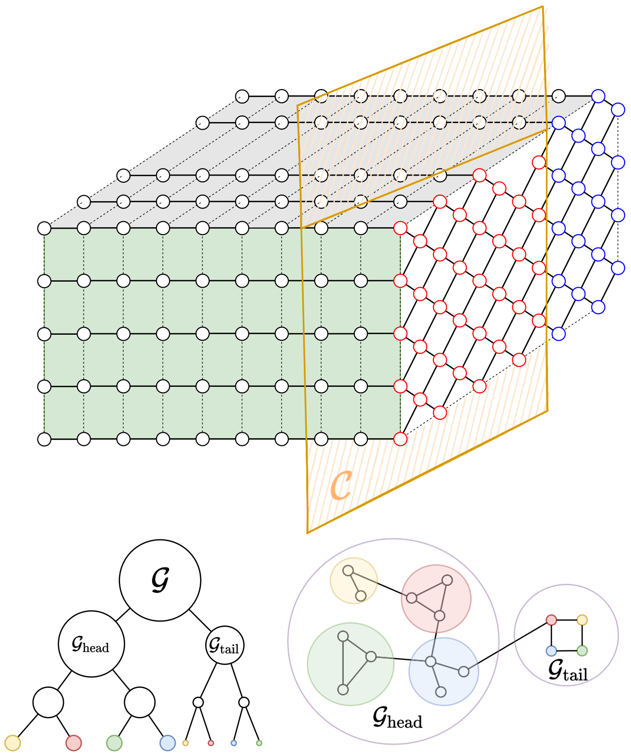
$\mathcal{C}$
$\mathcal{G}$
$\mathcal{G}_{\text{head}}$
$\mathcal{G}_{\text{tail}}$
$\mathcal{G}_{\text{head}}$
$\mathcal{G}_{\text{tail}}$

FIG. 1. The top panel presents a pictorial representation of the 3-dimensional tensor network corresponding to a quantum circuit. The leftmost layer represents the initial state, and the rightmost layer represents the final state, where the blue circles represent measured (closed) qubits, which fix the entries in the final bitstring $\boldsymbol{s}$, and the red circles represent open qubits, where the corresponding entries in $\boldsymbol{s}$ can vary. The yellow plane $\mathcal{C}$ cuts into the tensor network and separates the network into two parts, $\mathcal{G}_{\text{head}}$ and $\mathcal{G}_{\text{tail}}$, depicted in the bottom left panel. The $\mathcal{G}_{\text{head}}$ contains all closed qubits, and $\mathcal{G}_{\text{tail}}$ includes all the open qubits. Both are further partitioned into subgraphs hierarchically until the size of the subgraph is smaller than 60. The bottom right panel displays the bottleneck between $\mathcal{G}_{\text{head}}$ and $\mathcal{G}_{\text{tail}}$ given by $\mathcal{C}$.

complexity over single-amplitude contraction. In the literature, only a small number (typically 6) of open qubits are selected to control the complexity increase. Second, finding an optimal combination of n_2 qubits is a hard combinatorial optimization problem, particularly when the cost for evaluating the quality of the combinations is very high.

In this work, we aim to select a large number of open qubits n_2 without significantly increasing the overall computational cost. To this end, rather than selecting open qubits manually, we first find a (special) contraction order $\mathcal{O}$ (which indicates a sequence of pairwise contractions to perform) with a good space and time complexity and then choose the open qubits based on the order.

For large and deep circuits, e.g., the Sycamore circuits, the space complexity for contracting the full tensor network with a large n_2 open indices are typically out of reach even with a near-optimal contraction order. So, we proceed by enumerating all possible partial bitstrings s_2, that is, trading off space complexity using time complexity by enumerating all possible ways for closing the open qubits. However, this approach causes a serious issue on the time complexity: one needs to repeat the contraction for 2^{n_2} times, which is intractable. To resolve this issue, we demand the contraction order found in the first place to satisfy a special constraint: detecting a "big-head structure" of the tensor network by identifying a bottleneck, as illustrated in Fig. 1. Given a final-state bitstring $\boldsymbol{s}$, the identified bottleneck separates the whole network into two parts: the head network $\mathcal{G}_{\text{head}}$ containing all the closed end qubits, and the tail tensor network $\mathcal{G}_{\text{tail}}$ containing all the open end qubits, connected by n_c edges. In the contraction order $\mathcal{O}$, $\mathcal{G}_{\text{head}}$ and $\mathcal{G}_{\text{tail}}$ are contracted independently, resulting in two vectors $\mathbf{v}_{\text{head}}(\boldsymbol{s}_1)$ and $\mathbf{v}_{\text{tail}}(\boldsymbol{s}_2)$. The amplitude of $\boldsymbol{s}$ is computed at the final step of contraction in $\mathcal{O}$ as the inner product of the two vectors $\psi(\boldsymbol{s}) = \mathbf{v}_{\text{head}}(\boldsymbol{s}_1)\cdot\mathbf{v}_{\text{tail}}(\boldsymbol{s}_2)$. Since all n_2 open qubits are located at $\mathcal{G}_{\text{tail}}$, by fixing the partial bitstring $\boldsymbol{s}_1$, one only needs to compute $\mathbf{v}_{\text{head}}(\boldsymbol{s}_1)$ once and then reuse it to obtain amplitude $\psi(\boldsymbol{s})$ for every $\boldsymbol{s}_2$. By carefully designing the contraction order $\mathcal{O}$, we can put dominating computational cost to $\mathcal{G}_{\text{head}}$, with the bottleneck n_c small enough such that $\mathbf{v}_{\text{head}}(\boldsymbol{s}_1)$ can be stored and reused. In this way, amplitudes of 2^{n_2} correlated bitstrings can be computed with computational complexity almost identical to that of computing one bitstring.

The key component of the big-head simulation is finding a contraction order $\mathcal{O}$ satisfying the constraints described in the last section. We design the order-finding algorithm relying on a partitioning algorithm that splits the whole tensor network into two parts, $\mathcal{G}_{\text{head}}$ and $\mathcal{G}_{\text{tail}}$, using the "first cut" $\mathcal{C}$ cutting n_c edges. In Fig. 1, we give a pictorial illustration of the first cut made on a 3-dimensional tensor network. The partition algorithm tries to minimize the cut size n_c, given constraints on the group sizes $\{n_1, n_2\}$. To find a first cut and the top partition (as illustrated in Fig. 1), we need to make sure the computational complexity of contracting two networks is acceptable. We achieve this by hierarchically partitioning [13,19,22] each subgraph into two smaller subgraphs using a clustering algorithm until every subgraph is small enough (set to 60 tensors in this work). The constraint for hierarchical partitioning is that both the time and space complexity of the individual contraction must be smaller than target values. The hierarchical partitioning also gives a contraction order O_{coarse} to the coarse-grained graph, treating the finest subgraphs as vertices. After the partitioning, we contract all the finest subgraphs using a greedy contraction order and then contract the coarse-grained graph according to O_{coarse}.

For deep quantum circuits, the space complexity of the found order is still too large, so we employ the dynamic slicing method [13,15,17,19,20,23], which selects a set of n_e edges from the tensor network and enumerates the indices associated with the edges. This breaks the overall contraction task into 2^{n_e} subtasks, each of which has much smaller space and time complexity and hence can be contracted independently.

We focus on Google's Sycamore circuits with $m = 20$ cycles, which have been used for demonstrating quantum supremacy. The Sycamore circuits have $n = 53$ qubits locating on a 2-dimensional layout (see Fig. 1). Each cycle of operations contains a layer of single-qubit gates (randomly sampled from $\{\sqrt{X}, \sqrt{Y}, \sqrt{W}\}$) and two-qubit fSim gates with different parameters ϕ and θ. The fSim gates have decompositional rank 4 and hence are believed to be much harder to simulate than the controlled-Z gates even in the approximation level [24].

Computing bitstring probabilities in the subspace.—We first simplify the quantum circuit by absorbing single-qubit gates into two-qubit gates, resulting in a tensor network with $n = 381$ nodes, and then find a contraction order for the network that partitions the tensors into two subgraphs ($\mathcal{G}_{\text{head}}$ and $\mathcal{G}_{\text{tail}}$) with sizes $n_{\text{head}} = 345$ and $n_{\text{tail}} = 36$, respectively. We determined 21 open qubits in $\mathcal{G}_{\text{tail}}$. We assign 32 entries of $\boldsymbol{s}_1$, i.e., the other part of the bitstring corresponding to closed qubits, to 0. In contracting $\mathcal{G}_{\text{head}}$ for obtaining the $\mathbf{v}_{\text{head}}$ vector, we use the dynamic slicing method [13,15,17,19,20,23] and divide the $\mathcal{G}_{\text{head}}$ contraction task into 2^{23} subtasks, each of which has space complexity 2^{30} to fit into 32 G memory of a graphical processing unit (GPU). The time complexity of contracting $\mathcal{G}_{\text{head}}$ is 4.51×10^{18}, which dominates the overall computational complexity for obtaining 2^{21} bitstring amplitudes. The computational cost for obtaining a different number of bitstrings in different algorithms is compared in Table I. We remark that our computational cost for obtaining 2×10^6 bitstring probabilities is much lower than the estimated computational cost of the Shrödinger-Feynman method used by Google [1] and is also slightly lower than the computational complexity for obtaining 64 amplitudes in the state-of-the-art tensor network method [19]. The algorithm can be trivially parallelized on multiple GPUs. So, in

TABLE I. Computational cost of different methods for obtaining bitstring probabilities of the Sycamore circuit with 53 qubits and 20 cycles.

	# bitstrings	Time complexity	Space complexity	Computational time	Computational hardware
Google [1]	10^6	...	...	10 000 years	Summit supercomputer
Cotengra [13]	1	3.10×10^{22}	2^{27}	3088 years	One NVIDIA Quadro P2000
Alibaba [19]	64	6.66×10^{18}	2^{29}	267 days	One V100 GPU
Ours	**2 097 152**	4.51×10^{18}	2^{30}	149 Days	One A100 GPU

our experiments, we employed a small computational cluster composed of 60 NVIDIA GPUs. The overall computation cost was about 5 days. More details about the algorithm and the computations can be found in the Supplemental Material [25].

In this work, we fix 32 entries as $s_1 = \{0, 0, 0, \ldots, 0\}$, and enumerate all possible combinations of the other 21 entries in the bitstring. This produces a set of 2^{21} correlated bitstrings, denoted as $\Omega = \{s_1; s_2^{(i)}\}_{i=1}^{21}$. Some bitstrings samples can be found in the Supplemental Material [25]. We plot the histogram of the obtained 2^{21} bitstring probabilities in Fig. 2, where we can see that the obtained distribution fits perfectly to the Porter-Thomas distribution [8,26,27]. The minimum and the maximum probability of bitstrings are $P_{\min} = 8.04 \times 10^{-8} \times 2^{-53} \approx 0$ and $P_{\max} = 16.1 \times 2^{-53}$, respectively. Since the exact probabilities of 2^{21} bitstrings are stored, we can postselect a subset of bitstrings such that the total probability is very high. This gives a way to generate a large number of (e.g., 1×10^6) bitstrings with high probability, achieving a high XEB value. We emphasize that a high XEB value does not necessarily indicate a high fidelity here because the postselection only samples from a subspace of the target distribution. So this can be treated as a "spoofing" to the XEB benchmarking. The quantum threshold assumption

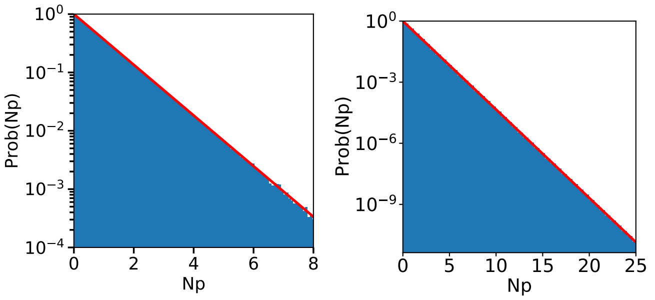

FIG. 2. Left: Histogram of bitstring probabilities $P_U(s) = P_U(s_1; s_2)$ for 2^{21} correlated bitstrings obtained from the Sycamore circuit with $n = 53$ qubits, $m = 20$ cycles, sequence ABCDCDAB, seed 0, with the assignment of partial bitstring s_1 fixed to $\{0, 0, \ldots, 0\}$. In the figure, $N = 2^{53}$, p denotes the probability of bitstring $P_U(s)$. Right: Histogram of probabilities of *all* 2^{50} bitstrings obtained from the Sycamore circuit with $n = 50$ qubits, $m = 14$ cycles, sequence EFGH. The red line in the figures represent the Porter-Thomas distribution [1].

and linear cross-entropy quantum threshold assumption conjectures [2,6] have stated that there cannot be any algorithm that can spoof linear cross entropy unless this algorithm can also simulate circuit amplitudes, that is, the spoofing task was conjectured to be hard asymptotically.

Full-amplitude simulations.—Full-amplitude simulation of quantum circuits is challenging due to the high space complexity induced by storing and evolving the full-state vector. The big-batch method offers an alternative method to compute amplitudes of all 2^n bitstrings by enumerating 2^{n_1} configurations for closed qubits; for each of them, we compute $P_U(s_1; s_2)$ using the big-batch method. In other words, we compute n_1 batches of amplitudes for generating all elements of the state vector sequentially. The benefits of the method are twofold. First, it avoids storing the state vector and hence enjoys a small space complexity; second, by exploiting structures using the big-batch tensor network method, the time complexity could be heavily reduced. To demonstrate its performance, we first compute the exact full amplitudes for the Sycamore circuit with $n = 43$ qubits, $m = 14$ cycles, and the EFGH sequence. In [1], to accomplish this task the Jülich supercomputer with 100 000 cores and 250 terabytes memory has been used. In our simulations, we separate the 43 qubits into $n_1 = 14$ closed qubits and $n_2 = 29$ open qubits and enumerate all 2^{n_1} configurations s_1. For each configuration, we compute $P_U(s_1; s_2)$ using the big-batch tensor network method. The overall computation was performed using a single NVIDIA V100S GPU in 12 hours. We note here that the computational complexity of the full-amplitude computation based on the big-batch tensor network method increases very fast with the circuit depth, while the time complexity of the Schrödinger method is linear in the circuit depth. To further demonstrate the ability of the proposed method, we computed the exact full amplitudes for the Sycamore circuit with $n = 50$ qubits, $m = 14$ cycles, and the EFGH sequence. We separate the 50 qubits into $n_1 = 22$ closed qubits and $n_2 = 28$ open qubits. The space complexity of computing a batch of probabilities is 2^{30}; the time complexity is 5.82×10^{10}. The histogram of all 2^{50} probabilities is shown in the righthand side of Fig. 2, which verifies that the histogram follows perfectly the Porter-Thomas distribution. The whole computation was performed on a small computational cluster with 100 GPUs using about 10 days. We remark that the previous record of the largest full-amplitude simulation of quantum circuits has 49 qubits, computed on the Sunway TaihuLight supercomputer [11]. We also remark that the proposed exact full-amplitude method can be adapted straightforwardly to obtain a noisy state vector with a target fidelity f, where the computational cost can be reduced by the factor of $1/f$.

Discussions.—We have presented a tensor network method for computing a large number of correlated bitstring amplitudes and probabilities for quantum circuits. The method explores a specified subspace of the output probability distribution and can be extended to compute the full-state vector. We have demonstrated the performance of the proposed approaches using Google's Sycamore circuits, where we obtained 2×10^6 correlated bitstring probabilities for the supremacy circuits with $n = 53$ qubits and $m = 20$ cycles, and obtained the full-state vector for the simplifiable circuits with $n = 50$ qubits and $m = 14$ cycles, with a small GPU cluster.

Our tensor-network algorithms have several advantages over Google's hardware sampling of the Sycamore circuits, including the exact computation of amplitudes and computing conditional probabilities $P_U(\mathbf{s}_2|\mathbf{s}_1)$ and sample from the distribution accordingly, which is hard for quantum hardware. At the same time, our experiments also reflect that Google's hardware has several advantages over our algorithm. The most significant one is that Google's hardware is much faster, while our algorithm has exponential complexity and hence is not scalable to both depth and qubit number. Although noisy, the samples generated by Google are not correlated to each other. In this work, we generated correlated samples that belong to a subspace of the target output distribution.

We note here that the big-head algorithm is not the only method to achieve the (correlated) big-batch approach. Other methods e.g., Cotengra [13], can also be used to obtain a big batch of amplitudes. Moreover, we note a recent extension [28] of the big-head method that allows sampling 1×10^6 *uncorrelated* bitstrings from the Sycamore circuits with $n = 53$ qubits and $m = 20$ cycles.

Although we have focused on the simulation of Google's Sycamore circuits, our algorithm is generally designed. The big-head shape and bottleneck structure are general phenomenons existing in many tensor networks. The proposed algorithm can be used straightforwardly for simulating and verifying existing and near-future noisy intermediate-scale quantum circuits. We hope the proposed algorithm could inspire more research to use tensor networks as a bridge to combine classical computations and noisy intermediate-scale quantum computations for solving challenging problems in the real world.

We thank Xun Gao, Jinguo Liu, Dingshun Lv, Yibo Yang, and Lei Wang for helpful discussions. We are grateful to Haijun Liao and Lei Wang for offering A100 GPUs. Part of the computation was carried out at the HPC Cluster of ITP, CAS. P. Z. is supported by Key Research Program of Frontier Sciences, CAS, Grant No. QYZDB-SSW-SYS032, and Projects No. 12047503 and No. 11975294 of National Natural Science Foundation of China. The Sycamore circuit files are downloaded from [29]. Our code is implemented using PYTHON. The contraction code, together with the contraction orders, slicing indices, and the obtained bitstrings probabilities, will be available at [30].

[*]Corresponding author.
panzhang@itp.ac.cn

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, Nature (London) **574**, 505 (2019).

[2] S. Aaronson and L. Chen, Complexity-theoretic foundations of quantum supremacy experiments, arXiv: 1612.05903.

[3] A. Bouland, B. Fefferman, C. Nirkhe, and U. Vazirani, On the complexity and verification of quantum random circuit sampling, Nat. Phys. **15**, 159 (2019).

[4] R. Movassagh, Quantum supremacy and random circuits, arXiv:1909.06210.

[5] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and H. Neven, Simulation of low-depth quantum circuits as complex undirected graphical models, arXiv:1712.05384.

[6] S. Aaronson and S. Gunn, On the classical hardness of spoofing linear cross-entropy benchmarking, arXiv:1910 .12085.

[7] A. Zlokapa, S. Boixo, and D. Lidar, Boundaries of quantum supremacy via random circuit sampling, arXiv: 2005.02464.

[8] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, Characterizing quantum supremacy in near-term devices, Nat. Phys. **14**, 595 (2018).

[9] J. Preskill, Quantum computing and the entanglement frontier, arXiv:1203.5813.

[10] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, Quantum supremacy is both closer and farther than it appears, arXiv:1807.10749.

[11] R. Li, B. Wu, M. Ying, X. Sun, and G. Yang, Quantum supremacy circuit simulation on sunway taihulight, IEEE Trans. Parallel Distrib. Syst. **31**, 805 (2019).

[12] I. L. Markov and Y. Shi, Simulating quantum computation by contracting tensor networks, SIAM J. Comput. **38**, 963 (2008).

[13] J. Gray and S. Kourtis, Hyper-optimized tensor network contraction, Quantum **5**, 410 (2021).

[14] C. Guo, Y. Liu, M. Xiong, S. Xue, X. Fu, A. Huang, X. Qiang, P. Xu, J. Liu, S. Zheng *et al.*, General-Purpose Quantum Circuit Simulator with Projected Entangled-Pair States and the Quantum Supremacy Frontier, Phys. Rev. Lett. **123**, 190501 (2019).

[15] J. Chen, F. Zhang, M. Chen, C. Huang, M. Newman, and Y. Shi, Classical simulation of intermediate-size quantum circuits, arXiv:1805.01450.

[16] F. Pan, P. Zhou, S. Li, and P. Zhang, Contracting Arbitrary Tensor Networks: General Approximate Algorithm and Applications in Graphical Models and Quantum Circuit Simulations, Phys. Rev. Lett. **125**, 060503 (2020).

[17] B. Villalonga, S. Boixo, B. Nelson, C. Henze, E. Rieffel, R. Biswas, and S. Mandrà, A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware, npj Quantum Inf. **5**, 86 (2019).

[18] B. Villalonga, D. Lyakh, S. Boixo, H. Neven, T. S. Humble, R. Biswas, E. G. Rieffel, A. Ho, and S. Mandrà, Establishing the quantum supremacy frontier with a 281 pflop/s simulation, Quantum Sci. Technol. **5**, 034003 (2020).

[19] C. Huang, F. Zhang, M. Newman, J. Cai, X. Gao, Z. Tian, J. Wu, H. Xu, H. Yu, B. Yuan *et al.*, Classical simulation of quantum supremacy circuits, arXiv:2005.06787.

[20] R. Schutski, T. Khakhulin, I. Oseledets, and D. Kolmakov, Simple heuristics for efficient parallel tensor contraction and quantum circuit simulation, Phys. Rev. A **102**, 062614 (2020).

[21] C. Guo, Y. Zhao, and H.-L. Huang, Verifying Random Quantum Circuits with Arbitrary Geometry using Tensor Network States Algorithm, Phys. Rev. Lett. **126**, 070502 (2021).

[22] S. Kourtis, C. Chamon, E. R. Mucciolo, and A. d. E. Ruckenstein, Fast counting with tensor networks, SciPost Phys. **7**, 060 (2019).

[23] F. Zhang, C. Huang, M. Newman, J. Cai, H. Yu, Z. Tian, B. Yuan, H. Xu, J. Wu, X. Gao *et al.*, Alibaba cloud quantum development platform: Large-scale classical simulation of quantum circuits, arXiv:1907.11217.

[24] Y. Zhou, E. M. Stoudenmire, and X. Waintal, What Limits the Simulation of Quantum Computers?, Phys. Rev. X **10**, 041038 (2020).

[25] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.128.030501 for detailed algorithm descriptions and additional numerical experiments.

[26] C. E. Porter and R. G. Thomas, Fluctuations of nuclear reaction widths, Phys. Rev. **104**, 483 (1956).

[27] T. A. Brody, J. Flores, J. B. French, P. Mello, A. Pandey, and S. S. Wong, Random-matrix physics: Spectrum and strength fluctuations, Rev. Mod. Phys. **53**, 385 (1981).

[28] F. Pan, K. Chen, and P. Zhang, Solving the sampling problem of the sycamore quantum supremacy circuits, arXiv:2111.03011.

[29] J. M. Martinis *et al.*, Quantum supremacy using a programmable superconducting processor, Dryad, Dataset, 10.5061/dryad (2021).

[30] https://github.com/Fanerst/simulate_sycamore.