Solving the sampling problem of the Sycamore quantum supremacy circuits

Feng Pan,[1,2] Keyang Chen,[1,3] and Pan Zhang[1,*]
[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]Yuanpei College, Peking University, Beijing 100871, China.

We study the problem of generating independent samples from the output distribution of Google's Sycamore quantum circuits with a target fidelity, which is believed to be beyond the reach of classical supercomputers and has been used to demonstrate quantum supremacy. We propose a new method to classically solve this problem by contracting the corresponding tensor network just once, and is massively more efficient than existing methods in obtaining a large number of *uncorrelated* samples with a target fidelity. For the Sycamore quantum supremacy circuit with 53 qubits and 20 cycles, we have generated one million *uncorrelated* bitstrings $\mathbf{s}$ which are sampled from a distribution $\widehat{P}(\mathbf{s}) = |\widehat{\psi}(\mathbf{s})|^2$, where the approximate state $\widehat{\psi}$ has fidelity $F \approx 0.0037$. The whole computation has cost about 15 hours on a computational cluster with 512 GPUs. The obtained one million samples, the contraction code and contraction order are made public. If our algorithm could be implemented with high efficiency on a modern supercomputer with ExaFLOPS performance, we estimate that ideally, the simulation would cost a few dozens of seconds, which is faster than Google's quantum hardware.

I. INTRODUCTION

The sampling problem of the quantum circuits has been proposed recently as a specific computational task to demonstrate whether programmable quantum devices are able to surpass the ability of classical computations, also known as *quantum supremacy* (or quantum advantage)[1–10]. As a milestone, in 2019, Google released the Sycamore quantum circuits to realize this approach for the first time [1]. The Sycamore quantum supremacy circuits contain 53 qubits and 20 cycles of unitary operations. Google has demonstrated that the noisy sampling task with fidelity $f \approx 0.002$ can be achieved experimentally using the quantum hardware in about 200 seconds, while it would cost 10,000 years on modern supercomputers.

However, the computational time estimated by Google relies on a specific classical algorithm, the Schrödinger-Feynman algorithm [1, 2, 11], rather than a theoretical bound that applies to all possible algorithms. So in principle there could exist algorithms that perform much better than the algorithm used by Google, rejecting the quantum supremacy claim. Indeed, in this article, we provide such an algorithm based on the tensor network method.

After the release of the Sycamore quantum circuits, There have been great efforts put on developing more efficient classical simulation algorithms. IBM has estimated that the 53-qubit state-vector of the Sycamore circuits can be stored and evolved if one could employ all the RAM and hard disks of the Summit supercomputer. However, it is apparently unrealistic to do such numerical experiment. Recently a variety of methods have been proposed for this problem based on computing a single amplitude or a batch of amplitudes [5, 12–15] using tensor network contractions. In particular, [15] proposed contracting the corresponding tensor network for 2000 times to obtain 2000 batches of amplitudes (each batch contains 64 correlated bitstrings), then sample 2000 perfect samples from the batches and mix them with 998,000 random bitstrings to obtain samples with linear cross entropy benchmark (XEB) around 0.002. However the computational cost of contracting the tensor networks for 2000 times is still too large, and the experiment has not been realized yet.

Another attempt to pass the XEB test on the Sycamore quantum supremacy circuits is the recently proposed *big-head* approach [16], which can obtain a large number of correlated samples. Using 60 GPUs for 5 days, the authors of [16] generated one million correlated samples with XEB 0.739, passed the XEB test. We also noticed that very recent works [17, 18] implemented this approach on a supercomputer, and heavily reduced the running time for obtaining a batch of correlated samples. However, if the target of the simulation is not only passing the XEB test but also satisfying the constraint of obtaining *uncorrelated* samples, as in the Sycamore experiments, then one needs to repeat the contraction thousands of times, making the computation cost unaffordable in practice.

In this article, we propose a tensor network approach to solve the uncorrelated sampling problem for the Sycamore quantum supremacy circuits. Our method is based on contractions of the three-dimensional tensor network $\widehat{\mathcal{G}}$ (see Fig. 1) which is converted from the quantum circuit. A single contraction of $\widehat{\mathcal{G}}$ produces $\{\widehat{\psi}_i^\mu\}$ with $i = 1, 2, \cdots L$ and $\mu = 1, 2, \cdots l$, representing amplitudes of L (randomly chosen) uncorrelated groups of bitstrings with each group containing l bitstrings. Since $\{\widehat{\psi}_i^\mu\}$ contains a small portion of entries of a approximate state $\widehat{\psi}$ with fidelity F, so we term it as *sparse-state*. Based on the sparse-state, we do importance sampling to obtain one sample from a group, finally generating L uncorrelated samples from the approximate probability $\widehat{P} = |\widehat{\psi}|^2$, i.e. L approximate samples from the output distribution of the quantum circuit with fidelity F.

Our algorithm is massively more efficient than existing algorithms in producing a large number of uncorrelated approximate samples. On the Sycamore circuits with $n = 53$ qubits and $m = 20$ cycles, we have successfully generated $L = 2^{20}$ approximate samples with fidelity $F \approx 0.0037$ in about 15 hours using 512 GPUs. We remark that to the best of our

* panzhang@itp.ac.cn

knowledge this is the first time that the sampling problem of the Sycamore quantum supremacy circuits (with fidelity larger than Google's hardware samples) with $n = 53$ qubits and $m = 20$ cycles is solved in practice classically.

Our proposed simulation method has several significant features that distinguish our method from existing methods:
(1) Introducing a specific Pauli error $E = \frac{1}{2}I + \frac{1}{2}\sigma_z$ to several positions in the circuit, behaving as drilling holes in the corresponding three-dimensional tensor network, as illustrated in Fig. 1. The hole-drilling method decreases the fidelity to a target value, but reduces the time complexity of the computation, i.e. trading fidelity with computational complexity in contractions.
(2) Exploring low-rank structures in the tensor network in cooperation with drilling holes and tensor slicing. Using special features of the fSim gates of the Sycamore circuits, we can break many edges in the network and heavily reduce the computation cost while only slightly decreases the fidelity.
(3) The contraction method we term as the sparse-state method with a *zig-zag* contraction order. It allows us to use only a single contraction with a restricted space complexity (2^{30} in this work) to obtain amplitudes of a large number ($L\times l$) of bitstrings, i.e. the sparse-state $\{\widehat{\psi}_i^\mu\}$.

Compared with the big-head method in [16], the algorithm proposed in this work can produce a large number of *uncorrelated* samples using a single contraction, rather than a large batch of correlated samples in [16]. In other words, with one contraction, the big-head method in [16] simulated a small sub-space of the circuit output distribution exactly, while the proposed method simulate the whole space approximately with a target fidelity. We also remark that although in [16] the method can pass the XEB test, its performance is sensitive to the definition of the XEB. While here we work directly with fidelity, without needing to introduce XEB as a proxies for fidelity.

II. METHOD

The quantum circuits U with n qubits can be regarded as special unitary tensor networks $\mathcal{G}$ with matrices (corresponding to single-qubit gates) and four-way tensors (corresponding to two-qubit gates) connecting to each other. For the Sycamore circuits where the qubits are put on a two-dimensional layout, the corresponding $\mathcal{G}$ is a three-dimensional tensor network as illustrated in Fig. 1. The initial state (the leftmost layer in Fig. 1) and the final state (shown as the rightmost layer in Fig. 1) act as two boundary conditions to $\mathcal{G}$. The initial state is always a product state so acts as a set of vectors; while the final state is represented as either a giant tensor or a set of small tensors (including vectors) depending on how many amplitudes we request in one contraction of $\mathcal{G}$.

If we request all the amplitudes of the final state in the contraction, then the final state acts as a giant tensor with size 2^n. This is essentially a full amplitude simulation requiring a storage space exponential to the number of qubits. If we request only one amplitude of the final state, then the boundary is a product state and acts as a set of vectors. Another case considered in the literature is the batch contraction [15, 16], which requests amplitudes for l correlated bitstrings and gives a tensor with size l as the final boundary condition for $\mathcal{G}$.

In this work our target is a little different: we request a large number of amplitudes for uncorrelated bitstrings, from single contraction of $\widehat{\mathcal{G}}$, a slightly perturbed version of $\mathcal{G}$.

A. Trading off fidelity for computational complexity: hole-drilling in the three-dimensional tensor network

Tensor network $\widehat{\mathcal{G}}$ is created by breaking (removing) K edges (connections) in $\mathcal{G}$. The edge-breaking is implemented by inserting $E = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix}$ in-between the two tensors that the edge is connecting. In this work, we select K edges from input indices of $K/2$ two-qubit gates. Pictorially it represents as drilling $K/2$ holes in the three-dimensional graphical representation of $\widehat{\mathcal{G}}$ as shown in Fig. 1. The positions of holes are determined carefully such that contracting $\widehat{\mathcal{G}}$ is much easier than contracting $\mathcal{G}$, but with the price of decreasing the fidelity.

The amount of fidelity that decreases can be estimated using the expression of E as a specific Pauli error matrix $E = \frac{1}{2}I + \frac{1}{2}\sigma_z$, with $I = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$ and $\sigma_z = \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}$. The effect of the edge-breaking can be understood as breaking the system into a summation of two sub-networks. The first sub-network is a copy of the original one which preserves the information of the original final state, while the second sub-network with the action of σ_z completely destroys the information of the original final state. Since the weight of each sub-network is $1/2$, one then estimates that each edge-breaking decreases the fidelity F by a factor of $1/2$. After breaking K edges in $\mathcal{G}$, we arrive at $\widehat{\mathcal{G}}$. If we contract $\widehat{\mathcal{G}}$ and obtain a full amplitude state vector $\widehat{\psi}$, it would be an approximation to the final state ψ of $\mathcal{G}$, with fidelity estimated as $F_K \approx 2^{-K}$.

The simulation method based on tensor network contractions can be also regarded as Feynmann's path-integral approach, in the sense that the tensor contractions effectively sum over an exponential number of paths. In the case of the Sycamore circuits, the paths are considered to be orthogonal to each other hence contributing equally to the amplitudes obtained as the contraction results. Under this viewpoint, the hole-drilling in $\mathcal{G}$ can be understood as omitting some paths in the path-integral approach, summing over only a fraction of 2^{-K} paths, giving fidelity $F_K \approx 2^{-K}$.

B. Contraction of the 3-dimensional tensor network to obtain the *sparse-state*

If we request all the amplitudes of the final state, it would be an impossible task that requires a space complexity exponential in the number of qubits. In this work we only request the sparse-state, the amplitudes for $L \times l$ bitstrings which are grouped into L groups with each group containing l bitstring

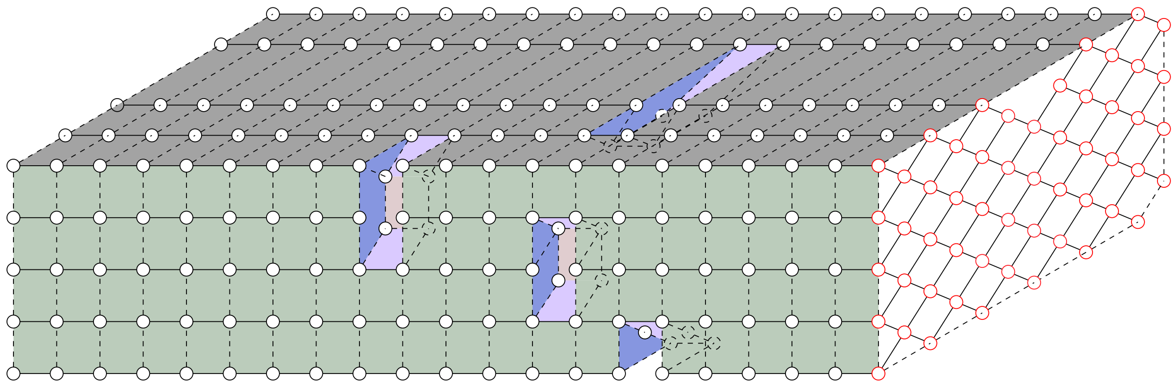

FIG. 1. Pictorial representation of the 3 dimensional tensor network corresponding to the Sycamore quantum circuit with $n = 53$ qubits and $m = 20$ cycles. The leftmost layer represents the initial state, and the rightmost layer represents the final state with red circles representing 53 qubits on the two-dimensional layout. There are 4 holes in the tensor network, designed to reduce heavily the computational complexity of the contraction. Each hole is created by breaking two edges in a selected 2-qubit gate and the companion edges, i.e. removing the entire 2-qubit gate (see Sec. II C), as described in the main text. The result of contracting the 3 dimensional tensor network using the sparse-state method (see Sec. II B) is $L = 2^{20}$ groups of amplitudes, each group contains $l = 2^6 = 64$ bitstring amplitudes. That is, we have computed approximate amplitudes for $2^{26} = 67,108,864$ bitstrings and finally sampled 2^{20} uncorrelated bitstrings from them.

amplitudes (in the practical computations we choose $L = 2^{20}$ and $l = 2^6$). They are given according to a generation process (e.g. randomly generated) in advance and kept fixed during the contraction.

However, contracting $\widehat{\mathcal{G}}$ to arrive at the $L \times l$ size sparse-state is a very difficult task, and the space complexity of the contraction would be much larger than $L \times l$. To solve the problem we extend the *big-head* algorithm proposed in [16]. In the big-head algorithm, the three-dimensional tensor network is cut into two parts, $\widehat{\mathcal{G}}_{\text{head}}$ whose contraction cost dominates the whole computation, and $\widehat{\mathcal{G}}_{\text{tail}}$ which contains all the qubits in the final state and can completely reuse the contraction results of $\widehat{\mathcal{G}}_{\text{head}}$ for computing all the requested amplitudes. In this work, the big-head method is extended to work with the sparse-state (rather than a batch of correlated bitstrings in [16]). To this end, we need to balance the computation cost of $\widehat{\mathcal{G}}_{\text{head}}$ and the cost of $\widehat{\mathcal{G}}_{\text{tail}}$. The contraction results of $\widehat{\mathcal{G}}_{\text{head}}$ is a vector $\mathbf{v}_{\text{head}}$, with size much larger than our storage limit, so in practice, we enumerate k entries in $\mathbf{v}_{\text{head}}$, that is, making 2^k slices of the $\mathbf{v}_{\text{head}}$, each slice has size 2^{29}. Given each slice of $\mathbf{v}_{\text{head}}$, the $\widehat{\mathcal{G}}_{\text{head}}$ is contracted with a good contraction order and local dynamic slicing, similar to [16].

Contacting $\widehat{\mathcal{G}}_{\text{tail}}$ is more difficult than contracting $\widehat{\mathcal{G}}_{\text{head}}$ because the boundary condition given by the sparse-state is heavier to deal with than the boundary conditions of $\widehat{\mathcal{G}}_{\text{head}}$. So we proposed a new *zig-zag* method for finding a good contraction order. The method starts at the beginning boundary of $\widehat{\mathcal{G}}_{\text{tail}}$, contracting neighboring tensors in a complexity-greedy manner all the way towards the boundary of the sparse-state, then turns around to contract greedily the tensors and come back to the beginning boundary. The process is repeated until all the tensors in $\widehat{\mathcal{G}}_{\text{tail}}$ are contracted, and the sparse-state $\{\psi_i^\mu\}$ is obtained. The spirit of the zig-zag contraction order is to make use of both boundaries to reduce the space and time complexity of contraction.

C. Exploring low-rank structures

In the Sycamore circuits, two-qubit unitary transformations are parameterized using the fSim gates

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

Specifically, the parameters in Google's experiments [1] are tuned to $\theta \approx \pi/2$ in order to keep the decomposition rank equal to 4 with a near-flat spectrum. This setting increases significantly the cost of classical simulations when compared with Controlled-Z gates which has decompositional rank 2, in exact simulations as well as in approximate simulations [19, 20]. It has been reported in [1] that due to the flat spectrum of the fSim gates, there is almost no space to make use of the spectrum imbalances for speeding up the simulation in e.g. the Schrödinger-Feynmann algorithm.

However we observe that in our approach there are two situations that we can explore the low rank structures.
(1) The first situation is in the hole drilling, when the two input indices (α and β) of the fSim gate are cut, i.e. applying two Pauli errors gate as $A = \left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\right) \cdot \text{fSim}(\theta,\phi)$, as illustrated in Fig. 2 top. It evaluates to $B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, which is a rank-one matrix and can be equivalently written as $C = 1$ in the contraction process, and it does not decrease fidelity.
(2) The second situation is in the enumeration process of k entries of $\mathbf{v}_{\text{head}}$ as well as in the local slicing process, where one input index of the fSim gate is enumerated. In this case, fixing an index is regarded as breaking one input edge in the tensor diagram as illustrated in Fig. 2 bottom (e.g. the top left edge γ of tensor D is cut), giving a three-way tensor E. It is straightforward to check that although the decompositional rank of

E on the bottom right index ω is 2, the corresponding singular values, $\left(\sqrt{\sin^2(\theta)+1}, \cos(\theta)\right)$, are heavily imbalanced in the Sycamore circuits with $\theta \approx \pi/2$. In this way we can do a rank-one approximation by dropping the singular vectors corresponding to the singular value $\cos(\theta)$. This rank-one approximation decreases the fidelity approximately by a factor $(\sin^2(\theta)+1)/2$, while effectively break another edge ω, which we term as the *companion edge* in the tensor network. For totally k enumerations in $\mathbf{v}_{\text{head}}$ and slicing edges in the tensor network , we do the rank-one approximation for all associated fSim gates, cutting k associated companion edges. This decreases the fidelity F by a factor $\prod_{i=1}^{k}(\sin^2(\theta_i)+1)/2$, while at the same time removes k edges from the tensor network.

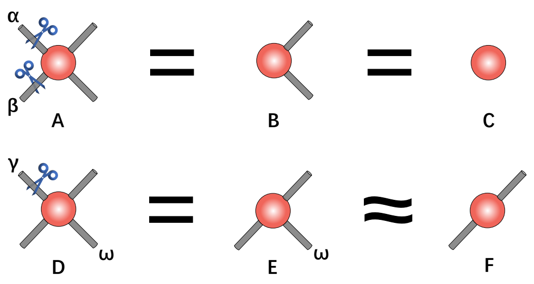

FIG. 2. Two situations that we can detect the low-rank structures in out method. (Top:) When two indices α, β are pinned to 0 for the fSim gate, the result is a rank-one matrix B, effectively equals to a scalar $c = 1$ in tensor network contractions. (Bottom:) When one index γ of a fSim gate is pinned, the resulting tensor E has decomposition rank 2 but very imbalanced singular value spectrum $\left(\sqrt{\sin^2(\theta)+1}, \cos(\theta)\right)$ with $\theta \approx \pi/2$ in Sycamore circuits. Thus pinning the index ω gives a slightly decrease to the fidelity.

D. Sampling from the sparse-state

After contracting $\widehat{\mathcal{G}}$ using the methods we have introduced in the previous sections, the sparse-state $\{\widehat{\psi}_i^{\mu}\}$ is obtained, which are uniform samples from 2^n entries of a state $\widehat{\psi}$, with fidelity to the true state estimated as

$$F_{\text{estimate}} \approx 2^{-K} \prod_{i=1}^{k} (\sin^2(\theta_i)+1)/2. \tag{2}$$

Since the sparse-state $\{\widehat{\psi}_i^{\mu}\}$ is composed of L groups, each group contains l amplitudes, we use a Markov chains to sample one bitstring out of l amplitudes in each group using the Metropolis rule [21], producing L samples which is considered as unbiased samples from $\widehat{\psi}$. We also notice that if $|\widehat{\psi}|^2$ follows the Porter-Thomas distribution [8, 22, 23] (as we verify empirically in Sec. III), we can use the frugal sampling [1, 11] which is much faster and guarantee to give perfect samples with $l = 64$ with the Porter-Thomas distribution [15]. We also remark that to obtain L uncorrelated samples, we only need to generate groups independently and randomly, there is no necessary to maintain uncorrelated bitstrings in each group.

III. RESULTS ON THE SYCAMORE SUPREMACY CIRCUITS

In this section we report the simulation results we have obtained on the Sycamore quantum supremacy circuits. We focus on the hardest circuits with $n = 53$ qubits, $m = 20$ cycles, and sequence ABCDCDAB, which has been used to demonstrate the quantum supremacy based on the estimated 10,000-year running time of the Schrödinger-Feynmann algorithm [1]. The circuit files are retrieved from [24], and the circuits are loaded with Cirq [25] script contained in the data repository and converted to the tensor network $\mathcal{G}$.

A. Tensor network contraction

We first simplify the tensor network since there are a lot of contraction steps one can do in advance without interfering the following procedure. In the simplification, all tensors with two or fewer indices are contracted into their neighbors, resulting in a tensor network with $n = 455$ tensors. To arrive at $\widehat{\mathcal{G}}$, we chose $K = 8$ edges to break, they are associated with 4 fSim gates. Using the low-rank structure described in Sec. II C, we completely remove the two-qubit gates by introducing proper Pauli error gates. This gives 4 holes marked in Fig. 1. This approximation decreases the fidelity by a factor 2^{-8}. Then the tensor network is divided into two parts, the head part $\widehat{\mathcal{G}}_{\text{head}}$ and the tail part $\widehat{\mathcal{G}}_{\text{tail}}$, as illustrated in Fig. 4.

We introduce 6 local slicing edges (local means that the slicing does not influence the contraction of $\widehat{\mathcal{G}}_{\text{tail}}$) in contraction of $\widehat{\mathcal{G}}_{\text{head}}$. The space and time complexity are 2^{30} and 2.3816×10^{13} respectively. Contracting the $\widehat{\mathcal{G}}_{\text{head}}$ results in a tensor $\mathbf{v}_{\text{head}}$ of size 2^{45}, which we can not store, so we enumerate 16 entries of the $\mathbf{v}_{\text{head}}$, creating 2^{16} sub-tasks of tensor network contraction, each of which corresponding to a configuration of 16 binary variables.

In each sub-task, $\mathbf{v}_{\text{head}}$ is sliced to a tensor with size 2^{29}, which works as a boundary for $\widehat{\mathcal{G}}_{\text{tail}}$. For the Sycamore circuits with $n = 53$ qubits and $m = 20$ cycles, we set $L = 2^{20}$ and $l = 2^6$, i.e. organizing the requested bitstrings to 2^{20} independent groups, each of which contains 2^6 bitstrings. It acts as another boundary of $\widehat{\mathcal{G}}_{\text{tail}}$. In contracting $\widehat{\mathcal{G}}_{\text{tail}}$, we introduces 7 local slicing edges, and the space and time complexity in our sparse-state contraction scheme are 2^{30} and 2.9425×10^{13} respectively. The overall time complexity of the entire computation (for finishing 2^{16} sub-tasks) is 3.489×10^{18}, which is slightly lower than the previous work [16] in computing a large batch of correlated bitstring amplitudes, and [15] in computing a small batch of correlated bitstrings.

In the contraction of $\widehat{\mathcal{G}}_{\text{tail}}$, there are 5 slicing edges associated with a companion edge. Together with the 16 companion edges in enumerating $\mathbf{v}_{\text{head}}$, there are totally $k = 21$ companion edges. As introduced in Sec. II C, we do further low-rank approximations on the $k = 21$ associated fSim gates, decreasing the fidelity by a factor $\prod_{i=1}^{21}(\sin^2(\theta_i) + 1)/2 \approx 0.9565$, where θ_i in the equation denotes the parameters of involved fSim gates. Together with the fidelity decreasing introduced in hole-drilling, the final fidelity is estimated as

$$F_{\text{estimate}} = 2^{-8} \times 0.9565 \approx 0.0037. \quad (3)$$

The head-tail splitting of the circuits is also revealed in the 2-dimensional qubit layout in Fig. 6, labeled with tensor IDs used in our actual contraction code. The actual contraction order are marked in Fig. 6, also listed explicitly in Appendices.

To increase the GPU efficiency, the branch merge strategy [16, 26] was adopted during the contraction. After branch merging, the GPU efficiency is 31.76% for $\widehat{\mathcal{G}}_{\text{head}}$ and 14.27% for $\widehat{\mathcal{G}}_{\text{tail}}$, the overall efficiency is 18.85%. The detailed data about the complexity, estimated fidelity, and GPU efficiency are listed in Table I. We use the *Complex64* as data-type in contraction. The contraction time of $\widehat{\mathcal{G}}_{\text{head}}$ for one sub-task is around 112 seconds and that of $\widehat{\mathcal{G}}_{\text{tail}}$ is around 315 seconds, summing to 427 seconds for completing a single sub-task. The entire simulation with 2^{16} sub-tasks is finished in about 15 hours using a computational cluster with 512 NVIDIA Tesla V100 SXM3 GPUs with 32GB memory. Our contraction code is implemented using *Pytorch* (version 1.7.2) with *cudatoolkit* (version 10.1).

B. Results

When the contraction of $\widehat{\mathcal{G}}$ is completed, we have generated 2^{20} independent groups, each group corresponds to a partial bitstring $\mathbf{x} \in \{1, 0\}^{47}$ that are uniformly and randomly generated. For each group we assign $2^6 = 64$ correlated bitstrings, corresponding to 6 open qubits. So finally we have computed 2^{26} bitstrings amplitudes by contracting $\widehat{\mathcal{G}}$ by summing over 2^{16} paths. As a sanity check, we compute the squared norm $\mathcal{N} = \sum_{i=1}^{2^{20}} \sum_{\mu=1}^{64} |\widehat{\psi}_i^{\mu}|^2$ of the sparse-state by summing only a fraction of total paths, and compare to the expected fidelity with partial summation (i.e. the fraction of the paths). The result are shown in Fig. 3 right, where we can see that they coincide to each other.

Using the norm of the sparse-state we can estimate the normalization factor of the approximate distribution as $2^{27}\mathcal{N}$, and compute the approximate probability of bitstrings. The histogram of the probability is plotted in Fig. 3 left, where we can see that it fits very well to the Porter-Thomas distribution [8, 22, 23].

The final step is generating 2^{20} uncorrelated bitstrings samples from the sparse-state based on the distribution. We have used two methods to generate samples. The first method is the reject sampling based on Markov Chain Monte Carlo, as we have described in Sec. II D. The other method is frugal sampling which is guaranteed to work well because the Fig. 3 indicates that our distribution fits very well to the Porter-Thomas distribution [1, 11]. The 2^{20} bitstrings generated using the reject sampling are available at [27].

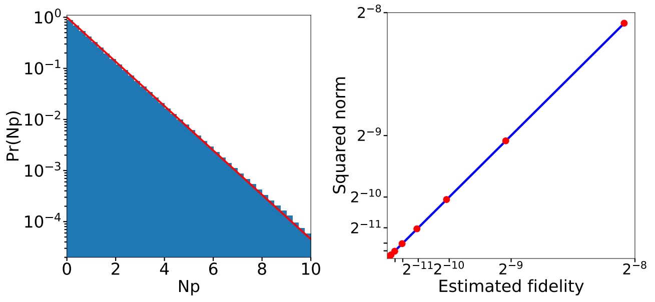

FIG. 3. (Left:) Histogram of approximate bitstring probabilities $p(\mathbf{s}) = |\widehat{\psi}(\mathbf{s})|^2/\mathcal{N}_s$, where $\mathcal{N}_s = 2^{27}\sum_{i=1}^{2^{20}}\sum_{\mu=1}^{64}|\widehat{\psi}_i^{\mu}|^2$ is the norm factor, for 2^{26} bitstrings obtained from the Sycamore supremacy circuits with $n = 53$ qubits and $m = 20$ cycles. $N = 2^n$ is the dimension of the Hilbert space. The approximate final state $\widehat{\psi}$ is obtained by obtained by contracting a tensor network as illustrated in Fig. 1, i.e. summing over 2^{16} paths, each of which corresponds to a sub tensor network. The estimated fidelity $\widehat{\psi}(\mathbf{s})$ to the true final state $\psi(\mathbf{s})$ is $F \approx 0.0037$. The red line denotes the Porter Thomas distribution. (Right:) Comparison between the estimated fidelity (blue lines) and the norm factor of state $\widehat{\psi}_L(\mathbf{s})$ obtained by summing over a fraction of paths.

IV. VALIDATION OF OUR METHOD USING SMALLER SYCAMORE CIRCUITS

The estimated of fidelity $F_{\text{estimate}} \approx 0.0037$ of one million samples generated using our method does not require to define a proxy of fidelity, such as the XEB [1]. But one can use XEB to verify the fidelity value that we claim, e.g. by utilizing the verification method reported in [28]. However verifying one million samples for $n = 53$ qubits, $m = 20$ cycles cost a huge amount of computational resources that we can not afford. So in this section, we show validation of our approximate sampling method using smaller Sycamore circuits where we can compute the exact amplitudes of the original circuit, and easily verify the fidelity of $\widehat{\psi}$ and XEB of generated samples.

We choose a Sycamore circuit with $n = 30$ qubits, $m = 14$ cycles, and sequence EFGH, compute and store the exact final state ψ. Then we remove K edges associated with fSim gates together with K companion edges in the middle of the circuit, mimicking what we did for the circuits with $n = 53$ qubits and $m = 20$ cycles. Then we compute the corresponding ψ_K exactly by evaluating the state vector, and compute the fidelity F. In Fig. 5, we compare the true fidelity F and the estimated fidelity F_{estimate} (using the method in Sec. II C), we can see from the figure that they coincide very well. From the probability distribution associated with the approximate state vector $\widehat{P} = |\widehat{\psi}|^2$, we can obtain a set of independent samples using the reject sampling method described in Sec. II D. Analogous to the sampling procedure we performed for the circuits with $n = 53$ qubits and $m = 20$ cycles. On the circuit with $n = 30$ qubits and $m = 14$ cycles we also computed approximate probabilities $\{\widehat{P}(\mathbf{s}_i)\}$ for 2^{26} bitstrings $\{\mathbf{s}_i\}$ which are grouped into 2^{20} groups. Then we sample one bitstring from each group using the rejection sampling with Markov chains and finally produce 2^{20} uncorrelated bitstrings $\{s_i|i = 1, 2, \cdots 2^{20}\}$.

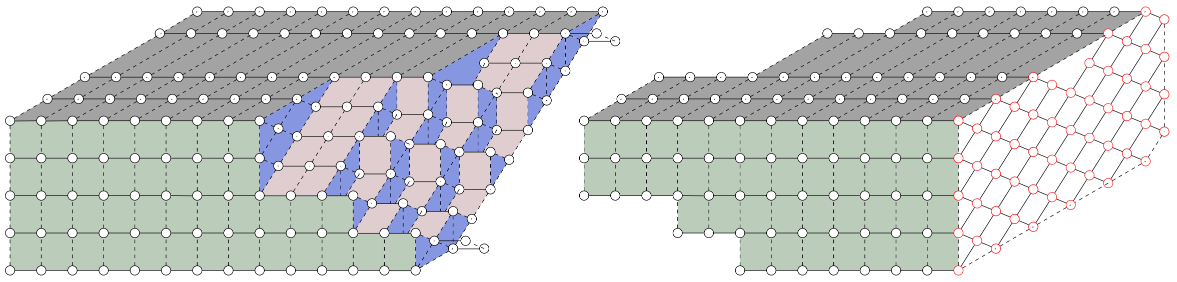

FIG. 4. The 3-dimensional tensor network $\widehat{\mathcal{G}}$ (corresponding to the Sycamore circuit of $n = 53$ qubits, $m = 20$ cycles) is split into two parts, $\widehat{\mathcal{G}}_{\text{head}}$ (left) and $\widehat{\mathcal{G}}_{\text{tail}}$ (right).

Data	Original	Branch merge
T_c head one sub-task	2.3816×10^{13}	6.967×10^{13}
T_c tail one sub-task	2.9425×10^{13}	8.796×10^{13}
Overall T_c (2^{16} sub-tasks)	3.489×10^{18}	1.033×10^{19}
Space complexity	2^{30}	
# 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%

TABLE I. Detailed data in the contractions of the Sycamore circuits with $n = 53$ qubits and $m = 20$ cycles. T_c represents the time complexity. The companion edges in the fSim slicing can be used to decrease the contraction complexity. In the simulation, we include no companion edges in the head, 5 out of 7 in the tail, and all 16 in the interface. Some companion edges are excluded because they do not contribute much to the overall complexity and including them will result in a drop in fidelity. The calculated fidelity comes from 21 companion edges, each of them contributes a factor $(\sin^2\theta + 1)/2$ to the fidelity (θ is the parameter of the associated fSim gate). The GPU efficiencies are calculated by $8 \cdot T_c/(P \cdot t)$ where P is the single-precision performance of GPU, t is the computation time and 8 is the factor of matrix multiplication for complex number ($8 \cdot T_c$ denotes the overall floating-point operations of the contraction).

Since we have stored the exact state vector, we can compute exact probabilities $P(\mathbf{s}_i)$ for each sample and evaluate the XEB for the samples using

$$F_{\text{XEB}} = \frac{2^{30}}{2^{20}} \sum_{i=1}^{2^{20}} P(s_i) - 1.$$

The sampling and XEB calculation are repeated for 10 times and the average XEB values are shown in Fig. 5 and compared with the estimated fidelity F_{estimate} as well as the true fidelity F, which we can see all of them agree to each other. We have also checked the XEB value of samples generated using the frugal sampling method, which also agrees with the XEB of our reject sampling and fidelities.

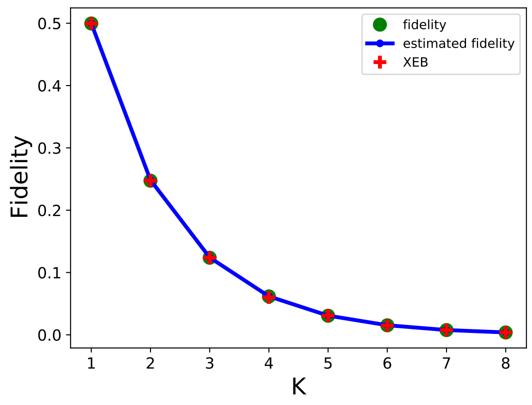

FIG. 5. Comparison between the exact fidelity of approximate state $\widehat{\psi}_K(\mathbf{s})$, estimated fidelity (computed using the method described in Sec. II C), and the XEB value of 2^{20} bitstrings sampled from $\widehat{P}(\mathbf{s}) = |\widehat{\psi}_K(\mathbf{s})|^2$. The approximate state $\widehat{\psi}_K$ is obtained by breaking K edges in the tensor network, for the Sycamore circuits with $n = 30$ qubits, $m = 14$ cycles, and EFGH sequence.

V. CONCLUSION AND DISCUSSIONS

We have presented a new tensor network method for solving the approximate sampling problem of the Sycamore quantum circuits which was thought to be impossible for classical computations. Using our algorithm the simulation for the Sycamore circuits with $n = 53$ qubits and $m = 20$ cycles is completed in about 15 hours using 512 V100 GPUs. There are several places that the proposed algorithms can be further speed up. First, our contraction algorithm is straightforwardly implemented using Pytorch. We expect that using a library that is more suitable for tensor contractions, such as the cuQuantum library [29], the computational efficiency can be greatly increased. Second, in recent days a modern supercomputer could achieve a performance of ExaFLOPS (10^{18} floating-point operations per second). If our simulation of the quantum supremacy circuits (with about 2.79×10^{19} floating-point operations without branch merging) can be implemented in a modern supercomputer with high efficiency, in principle, the overall simulation time can be reduced to a few dozens of seconds, which is faster than Google's hardware experiments.

ACKNOWLEDGMENTS

The computation was carried out at the Cloud Brain I Computing Facility at the Peng Cheng Laboratory and HPC cluster of ITP, CAS. We acknowledge Pengxiang Xu for help in performing computations on the Cloud Brain computers. We also thank Xun Gao, Ying Li, Lei Wang, Song Cheng, and Xiao Yuan for helpful discussions. The samples and the contraction code together with the contraction orders and slicing indices for reproducing our results are available at [27].

[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 **574**, 505 (2019).
[2] S. Aaronson and L. Chen, Complexity-theoretic foundations of quantum supremacy experiments, arXiv preprint arXiv:1612.05903 (2016).
[3] A. Bouland, B. Fefferman, C. Nirkhe, and U. Vazirani, On the complexity and verification of quantum random circuit sampling, Nature Physics **15**, 159 (2019).
[4] R. Movassagh, Quantum supremacy and random circuits, arXiv preprint arXiv:1909.06210 (2019).
[5] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and H. Neven, Simulation of low-depth quantum circuits as complex undirected graphical models, arXiv preprint arXiv:1712.05384 (2017).
[6] S. Aaronson and S. Gunn, On the classical hardness of spoofing linear cross-entropy benchmarking, arXiv preprint arXiv:1910.12085 (2019).
[7] A. Zlokapa, S. Boixo, and D. Lidar, Boundaries of quantum supremacy via random circuit sampling, arXiv preprint arXiv:2005.02464 (2020).
[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, Nature Physics **14**, 595 (2018).
[9] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, *et al.*, Strong quantum computational advantage using a superconducting quantum processor, Physical Review Letters **127**, 180501 (2021).
[10] Q. Zhu, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, M. Gong, *et al.*, Quantum computational advantage via 60-qubit 24-cycle random circuit sampling, Science Bulletin (2021).
[11] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, Quantum supremacy is both closer and farther than it appears, arXiv preprint arXiv:1807.10749 (2018).
[12] J. Chen, F. Zhang, M. Chen, C. Huang, M. Newman, and Y. Shi, Classical simulation of intermediate-size quantum circuits, arXiv preprint arXiv:1805.01450 (2018).
[13] 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, arXiv preprint arXiv:1905.08394 (2019).
[14] J. Gray and S. Kourtis, Hyper-optimized tensor network contraction, arXiv preprint arXiv:2002.01935 (2020).
[15] 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 preprint arXiv:2005.06787 (2020).
[16] F. Pan and P. Zhang, Simulating the sycamore quantum supremacy circuits, arXiv preprint arXiv:2103.03074 (2021).
[17] H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen, C. Guo, H. Huang, W. Wu, *et al.*, Closing the" quantum supremacy" gap: achieving real-time simulation of a random quantum circuit using a new sunway supercomputer, arXiv preprint arXiv:2110.14502 (2021).
[18] X. Liu, C. Guo, Y. Liu, Y. Yang, J. Song, J. Gao, Z. Wang, W. Wu, D. Peng, P. Zhao, F. Li, H.-L. Huang, H. Fu, and D. Chen, Redefining the quantum supremacy baseline with a new generation sunway supercomputer, arXiv preprint arXiv:2111.01066 (2021).
[19] Y. Zhou, E. M. Stoudenmire, and X. Waintal, What limits the simulation of quantum computers?, Physical Review X **10**, 041038 (2020).
[20] 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, Physical Review Letters **125**, 060503 (2020).
[21] M. Newman and G. Barkema, Monte carlo methods in statistical physics chapter 1-4, New York, USA (1999).
[22] C. E. Porter and R. G. Thomas, Fluctuations of nuclear reaction widths, Physical Review **104**, 483 (1956).
[23] T. A. Brody, J. Flores, J. B. French, P. Mello, A. Pandey, and S. S. Wong, Random-matrix physics: spectrum and strength fluctuations, Reviews of Modern Physics **53**, 385 (1981).
[24] Martinis, John M. et al. (2021), Quantum supremacy using a programmable superconducting processor, Dryad, Dataset, `https://doi.org/10.5061/dryad.k6t1rj8`.
[25] C. Developers, Cirq (2021), See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors.
[26] C. Huang, F. Zhang, M. Newman, X. Ni, D. Ding, J. Cai, X. Gao, T. Wang, F. Wu, G. Zhang, H.-S. Ku, Z. Tian, J. Wu, H. Xu, H. Yu, B. Yuan, M. Szegedy, Y. Shi, H.-H. Zhao, C. Deng, and J. Chen, Efficient parallelization of tensor network contraction for simulating quantum computation, Nature Computational Science **1**, 578 (2021).
[27] `https://github.com/Fanerst/solve_sycamore`.
[28] G. Kalachev, P. Panteleev, and M.-H. Yung, Recursive multi-tensor contraction for XEB verification of quantum circuits, arXiv preprint arXiv:2108.05665 .
[29] `https://developer.nvidia.com/cuquantum-sdk`.

Appendix A: Contraction order

Here we put the contraction order that used in the simulation of the Sycamore circuit with $n = 53$ qubits, $m = 20$ cycles. After contracting two tensors at location i and j, the resulting new tensor will be put back to location i. In the following order, the contractions from (47, 70) to (192, 234) belong to the head part, and the rest belong to the tail part.

[(47, 70), (27, 47), (117, 136), (27, 117), (35, 55), (78, 35), (9, 78), (27, 9), (5, 13), (31, 74), (51, 31), (94, 51), (75, 94), (32, 52), (75, 32), (79, 98), (95, 118), (175, 193), (156, 175), (132, 156), (183, 132), (121, 164), (140, 121), (183, 140), (113, 183), (137, 152), (110, 137), (180, 202), (161, 180), (110, 161), (44, 66), (23, 44), (67, 87), (23, 67), (38, 81), (58, 38), (24, 45), (186, 208), (124, 143), (167, 124), (60, 83), (103, 60), (40, 103), (145, 169), (126, 145), (57, 100), (80, 57), (77, 97), (54, 77), (253, 273), (230, 253), (210, 230), (125, 144), (168, 125), (210, 168), (73, 93), (50, 73), (116, 50), (26, 69), (46, 26), (209, 228), (142, 166), (123, 142), (229, 252), (184, 206), (207, 226), (22, 43), (65, 22), (21, 65), (42, 64), (85, 42), (21, 85), (86, 21), (1, 86), (0, 61), (41, 62), (19, 41), (128, 171), (149, 128), (176, 198), (199, 218), (147, 170), (127, 147), (131, 148), (106, 131), (222, 246), (203, 222), (223, 203), (120, 163), (139, 120), (182, 139), (154, 182), (214, 237), (194, 214), (215, 194), (189, 211), (232, 189), (212, 255), (234, 212), (191, 213), (234, 191), (159, 178), (27, 5), (27, 75), (27, 79), (27, 95), (76, 27), (113, 110), (90, 113), (76, 90), (71, 76), (71, 23), (28, 10), (28, 58), (28, 71), (56, 16), (56, 36), (56, 28), (14, 24), (14, 6), (14, 56), (101, 186), (101, 167), (101, 14), (18, 82), (18, 40), (101, 18), (33, 53), (33, 39), (101, 33), (17, 48), (17, 37), (101, 17), (11, 59), (11, 29), (101, 11), (15, 25), (15, 7), (101, 15), (34, 126), (34, 122), (101, 34), (80, 12), (80, 99), (101, 80), (4, 102), (4, 54), (101, 4), (96, 210), (96, 119), (101, 96), (188, 141), (188, 165), (101, 188), (114, 72), (114, 91), (101, 114), (49, 30), (49, 3), (101, 49), (8, 116), (8, 46), (101, 8), (68, 155), (68, 88), (101, 68), (111, 138), (111, 92), (101, 111), (187, 162), (187, 209), (101, 187), (115, 123), (115, 229), (101, 115), (184, 157), (184, 89), (101, 184), (133, 107), (133, 207), (101, 133), (185, 158), (185, 134), (101, 185), (108, 1), (101, 108), (109, 135), (109, 2), (101, 109), (0, 19), (0, 20), (101, 0), (63, 149), (101, 63), (84, 105), (84, 176), (101, 84), (104, 129), (104, 130), (101, 104), (150, 177), (150, 153), (101, 150), (181, 199), (181, 223), (101, 181), (127, 112), (101, 127), (106, 173), (101, 106), (146, 172), (146, 151), (101, 146), (190, 232), (190, 204), (101, 190), (174, 195), (192, 174), (192, 154), (192, 101), (159, 215), (192, 159), (192, 234), (359, 402), (359, 403), (360, 404), (382, 405), (382, 406), (361, 407), (383, 408), (383, 409), (384, 410), (384, 411), (363, 412), (385, 414), (385, 415), (386, 416), (386, 417), (387, 418), (387, 419), (367, 420), (388, 422), (388, 423), (389, 424), (389, 425), (390, 426), (390, 427), (391, 428), (391, 429), (392, 432), (393, 433), (393, 434), (394, 435), (394, 436), (395, 437), (395, 438), (396, 440), (396, 441), (397, 442), (397, 443), (398, 444), (398, 445), (399, 447), (399, 448), (400, 449), (400, 450), (381, 451), (401, 452), (401, 453), (339, 454), (179, 160), (197, 179), (217, 197), (241, 217), (261, 241), (225, 205), (249, 225), (269, 249), (245, 221), (265, 245), (284, 265), (271, 251), (292, 271), (257, 236), (277, 257), (196, 216), (196, 200), (192, 196), (227, 250), (227, 224), (192, 227), (219, 220), (219, 242), (192, 219), (231, 233), (231, 254), (192, 231), (235, 238), (235, 256), (192, 235), (239, 243), (239, 259), (192, 239), (247, 258), (247, 266), (192, 247), (262, 282), (262, 263), (192, 262), (267, 270), (267, 286), (192, 267), (272, 274), (272, 293), (192, 272), (275, 276), (275, 297), (192, 275), (278, 279), (278, 299), (192, 278), (285, 290), (285, 304), (192, 285), (301, 305), (301, 318), (192, 301), (309, 312), (309, 324), (192, 309), (320, 321), (320, 361), (192, 320), (316, 325), (316, 342), (192, 316), (328, 329), (328, 347), (192, 328), (333, 336), (333, 352), (192, 333), (344, 348), (344, 355), (192, 344), (366, 363), (367, 366), (192, 367), (375, 371), (387, 375), (192, 387), (261, 281), (261, 300), (261, 317), (261, 391), (192, 261), (269, 289), (269, 308), (269, 323), (269, 395), (192, 269), (340, 343), (340, 362), (340, 359), (192, 340), (295, 314), (295, 332), (295, 351), (295, 384), (192, 295), (338, 357), (338, 370), (338, 390), (192, 338), (378, 398), (382, 360), (378, 382), (378, 383), (192, 378), (386, 389), (386, 394), (386, 397), (386, 365), (192, 386), (400, 401), (400, 380), (400, 374), (192, 400), (385, 393), (385, 399), (341, 298), (341, 277), (341, 319), (341, 346), (385, 341), (192, 385), (369, 388), (303, 327), (303, 354), (369, 303), (192, 369), (377, 396), (284, 311), (284, 335), (284, 377), (192, 284), (381, 339), (381, 358), (381, 315), (381, 292), (381, 296), (381, 294), (381, 337), (381, 356), (381, 379), (192, 381), (313, 331), (313, 350), (313, 373), (313, 288), (313, 307), (313, 322), (313, 280), (313, 364), (313, 260), (313, 268), (313, 240), (192, 313), (248, 291), (248, 334), (248, 310), (248, 353), (248, 283), (192, 248), (326, 302), (326, 345), (326, 264), (326, 287), (326, 306), (192, 326), (392, 372), (392, 244), (392, 201), (392, 349), (392, 330), (392, 368), (392, 376), (192, 392)].

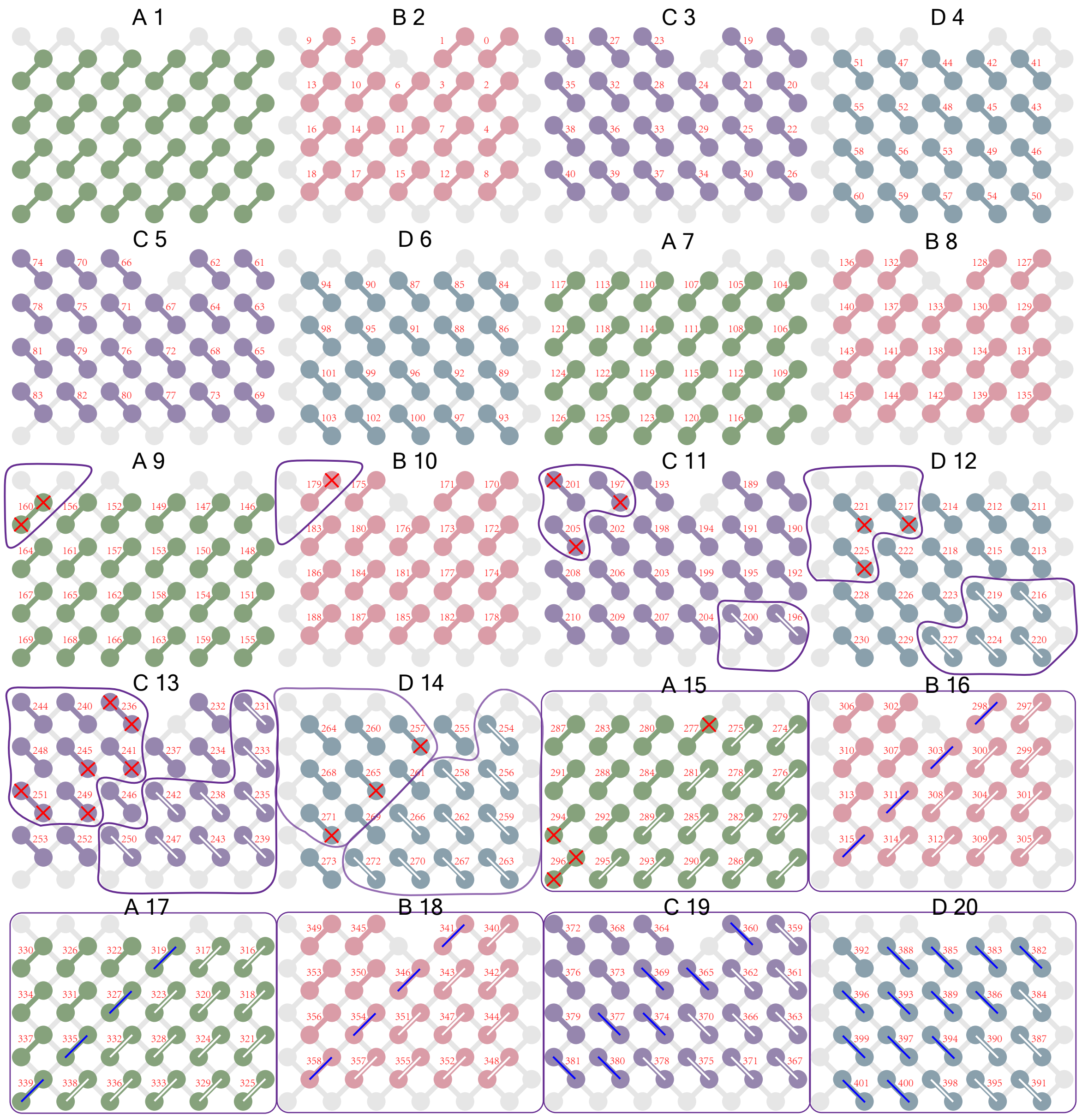

FIG. 6. The head-tail splitting and details about the contraction scheme in the view of two-dimensional qubit layout. The two-qubit gates of all 20 cycles with IDs in our actual contraction code and their sequence patterns in the Sycamore circuit are listed. Note that some gates are contracted out during the simplification stage thus have no corresponding IDs. The gates wrapped by purple lines belong to $\widehat{\mathcal{G}}_{\text{tail}}$ while others belong to $\widehat{\mathcal{G}}_{\text{head}}$. The red crosses over gates represent the slicing indices that connect the former layer to gates in the current layer. Notice that the four gates with two slicing crosses are the miss blocks in Fig. 1. The white and blue lines over the gates denote the zig-zag contraction order of the tail part, white for the first zig and blue for the first zag. The following order can be easily repeated accordingly.