Random circuit sampling has become a popular means for demonstrating the superiority of quantum computers over classical supercomputers. While quantum chips are evolving rapidly, classical sampling algorithms are also getting better and better. The major challenge is to generate bitstrings exhibiting an XEB fidelity above that of the quantum chips. Here we present a classical sampling algorithm for producing the probability distribution of any given random quantum circuit, where the fidelity can be rigorously bounded. Specifically, our algorithm performs rejection sampling after the introduced very recently multi-tensor contraction algorithm. We show that the fidelity can be controlled by partially contracting the dominant paths in the tensor network and by adjusting the number of batches used in the rejection sampling. As a demonstration, we classically produced 1 million samples with the fidelity bounded by 0.2%, based on the 20-cycle circuit of the Sycamore 53-qubit quantum chip. Though this task was initially estimated to take 10,000 years on the Summit supercomputer, it took about 14.5 days using our algorithm on a relatively small cluster with 32 GPUs (Tesla V100 16GB). Furthermore, we estimate that for the Zuchongzhi 56-qubit 20-cycle circuit one can produce 1M samples with fidelity 0.066% using the Selene supercomputer with 4480 GPUs (Tesla A100 80GB) in about 4 days.

One of the main motivations for quantum computing is a belief, shared by many researchers, that quantum systems are very hard to simulate classically. In the language of complexity theory, it is usually expressed as a plausible but currently unproven conjecture that BPP \( \subseteq \) BQP, where the complexity class BPP is, informally speaking, the class of problems efficiently solvable by classical computers, and BQP is the corresponding class for quantum computers. If this conjecture were true, it would imply, at least theoretically, that quantum computers have a computational advantage (or quantum supremacy) over the classical ones in the asymptotic regime, as the problem size grows to infinity. In fact, to demonstrate such a quantum advantage on a real quantum hardware one should also deal with the decoherence problem and other imperfections. While the general consensus is that, under some reasonable assumptions on the noise scaling, this problem can be solved by applying a fault-tolerant protocol [1], there are some researchers who still doubt these assumptions [2].

From the practical point of view, a convincing argument in favor of the quantum advantage would be an experiment demonstrating that some well-defined problem can be efficiently solved on a real quantum computer, while the solution of the same problem using the state-of-the-art algorithms cannot be obtained within a reasonable amount of time even with the best classical supercomputers in the world [3–5]. At the current stage, a leading candidate for such a quantum advantage experiment, which was already performed on a real quantum hardware [6–8], is the task of sampling from the output probability distributions of random quantum circuits (RQCs). Though initially this task was estimated [6] to take thousands of years on the fastest classical supercomputers, later the simulation time was significantly reduced due to the recent progress on tensor network (TN) based quantum simulation algorithms [9–15]. An interesting idea, which was used recently to significantly reduce the computational cost of RQCs simulation, is to compress the wave function using matrix product states [16].

Usually TN based simulators for RQCs calculate single amplitudes by contracting the tensor network representing the circuit. In fact, it was shown in [17, 18] that TN contraction can also be used to produce not only single amplitudes but also large batches of amplitudes (i.e., the amplitudes for a collection of bitstrings that share some fixed bits), and the computational cost of finding a batch of amplitudes is usually similar to the cost of finding a single amplitude. This idea has been applied recently [13] to spoof the linear cross-entropy benchmarking (linear XEB) test, which was used in Google’s quantum
advantage experiment [6] as a way to certify the fidelity. Explicitly, the linear XEB fidelity $F_{\text{XEB}}$ for a sequence of bitstrings $s_1, \ldots, s_k$, produced in an experiment is defined as

$$F_{\text{XEB}} = \frac{2^n}{k} \sum_{i=1}^{k} p_C(s_i) - 1,$$

where $p_C(\cdot)$ is the theoretical output probability distribution for the circuit $C$ used in the experiment. To spoof the linear XEB test and produce $N$ samples with $F_{\text{XEB}} \geq f$ one can choose $fN$ samples $s$ with the highest probabilities $p_C(s)$ from a large batch of amplitudes found with 100% fidelity and then add $(1-f)N$ uniformly random bitstrings. At first, it may seem that this very simple way of spoofing is quite easy to detect since the produced samples are highly correlated (in $f$ fraction of samples the values for some bit positions are fixed). However, if one uses a sufficiently large number of batches, the spoofing algorithm is much harder to distinguish from the real simulation since the bit positions are no longer fixed.

In fact, one can use a large number of batches not only for spoofing but also for the sampling task, where we want to produce $N$ independent random samples according to the probability distribution $p_C(\cdot)$. A rather standard approach here is to apply the frugal rejection sampling algorithm [9] to generate random bitstrings by calculating their probabilities and accepting each bitstring with probability proportional to the calculated value. This approach works for random quantum circuits where all bitstrings have probabilities of the same order of magnitude. Unfortunately, for a large number of samples the computational cost of this approach is quite high if the batches are calculated independently one by one. At the same time, it was shown recently [19] that by applying the multi-tensor contraction algorithm, which reuses the partial contraction results, one can reduce the computational cost in this case by several orders of magnitude. In fact, it is shown in [19] that not only the sampling task but also the much harder verification task for RQCs, where one needs to find the exact amplitudes for a large collection of uncorrelated bitstrings, can also be solved in several days on a modern supercomputer though it was initially estimated in [6] to take millions of years.

In the current work, we propose a slight modification of the frugal rejection sampling from [9], which uses more batches than the actual number of the random samples we need to produce. This allows us to give a rigorous analysis of the proposed algorithm in terms of the fidelity and the statistical variation distance to the ideal probability distribution. To produce $m$ samples we calculate $am$ random small batches (e.g., of size $64$), and then use our variant of the frugal rejection sampling algorithm to produce $m$ samples out of these $am$ batches. Our analysis indicates that if $\alpha = 2$, then the statistical variation distance between the probability distributions of our algorithm and the ideal sampling is negligible, and therefore in all our simulations we assume that $\alpha = 2$. Note that the computational cost grows less than linearly as $\alpha \to \infty$, and for $\alpha = 2$ the simulation time is less then two times larger than for $\alpha = 1$.

Moreover, to further reduce the computational cost of the simulations we also apply the partial slicing summations [9] in the tensor-network contraction. This allows us to simulate RQCs with a target fidelity $f$. The partial slicing is similar to the gate decomposition used in the Shrödinger-Feynman algorithm from [6], where some 2-qubit gates are decomposed into sums of pairs of 1-qubit gates. If the target fidelity $f$ in a simulation is less than 1, one can speed up by skipping some of the terms in the summation [6, 9]. In our case, if we have $k$ sliced vertices in the tensor network, then we can sum only over $f^{2^k}$ out of $2^k$ slices and obtain the result with the fidelity approximately equal to $f$. Compared with the method in Refs [6, 9], the key feature of the current work is that our method can accurately predict the obtained fidelity for a given set of slices and choose slices in order to maximize the fidelity. Moreover, we will show that this fidelity can be found by a contraction of some specifically designed tensor network, which gives a new general way to control the fidelity when we apply the partial slicing method.

In the current work, we demonstrate our experimental results using these new algorithms, where we:

1. produce samples for Google’s supremacy (ABCD) circuits from [6] up to 20 cycles;

2. show (using an approach similar to [13]) how to spoof the linear XEB test for the hardest case in the Google’s experiment with fidelity 0.2% in just 4 hours on a personal computer with only one GPU.

All the experimental data produced in these experiments can be found in [20].

Update. Recently, after all the experiments in the current paper were already finished we became aware of the work [15], where an approach, very similar to the multi-tensor contraction algorithm from [19], was used in combination with other techniques to significantly reduce the simulation time. Though the computational cost of our algorithm is similar to the one from [15], the former comes with a rigorous analysis of the fidelity, while the latter is only justified by empirical estimates. Moreover, in the current paper, we also independently confirm our analytical estimates of the fidelity using the verification algorithm from [19].

I. SIMULATION WITH TARGET FIDELITY

In this section, we present a general method for simulating RQCs with given target fidelity $f$. In general, for a quantum circuit $C$, by a standard procedure proposed in [6], to produce multiple independent random samples, one needs to calculate multiple independent batches of amplitudes, which is usually a difficult computing task. However, the multi-tensor simulator from [19] provides
a much more efficient way to accomplish this task by utilizing a global cache which can reuse some intermediate tensors to significantly save the computing time. The algorithm from [19] also uses the simulated annealing method to optimize the contraction tree and the list of sliced variables. Let us remind that a contraction tree [10, 21, 22] encodes a particular way we perform the contraction for a given tensor network. At the same time, the sliced variables (also called the projected variables) correspond to the variables (i.e., the tensor legs) that we sum over at the very last step, which allows us to control the memory budget of the contraction.

Note that if we do the full summation for all the sliced vertices, the target fidelity of the obtained amplitudes is equal to 1. However, in many cases we cannot afford this, and our aim is to find the amplitudes with some target fidelity \( f \ll 1 \). In this case, we divide the sliced variables into two parts: the partially sliced variables where we perform a partial summation (i.e., sum only over \( f \) fraction of slices), and the remaining fully sliced variables, where perform the full summation. The former aim to find a balance on the sampling time complexity and the fidelity of the produced samples, while the latter only aim to reduce the intermediate memory of the tensor contraction. Next, we will show how to choose the partially sliced vertices and the particular \( f \) fraction of slices for them.

### A. Achieving target fidelity by choosing slices of maximal norms

Let us first show how to choose the \( f \) fraction of slices we sum over on the chosen partially sliced vertices. Let \( C = C_2C_1 \) be a quantum circuit divided into the subcircuits \( C_1 \) and \( C_2 \). Moreover, we assume that the partially sliced vertices are already chosen in the cut of \( C \) splitting it into parts \( C_1 \) and \( C_2 \) (see Fig. 1). For the vector \( |\psi\rangle \equiv |C|0\rangle \), we have

\[
|\psi\rangle = C_2C_1|0\rangle = \sum_{i \in \{0,1\}^k} |\psi_i\rangle = \sum_{i \in \{0,1\}^k} \sum_{j \in \{0,1\}^{n-k}} |\psi_i^j\rangle,
\]

where \( n \) is number of qubits, \( k \) is the number of the partially sliced vertices in the cut of \( C \). Here and below in this section we will assume that \( i, i', i'' \in \{0,1\}^k \), and \( j, j' \in \{0,1\}^{n-k} \). The quantum state \( |\psi\rangle \) can be expressed as:

\[
|\psi\rangle = C_2C_1|0\rangle = \sum_{i \in \{0,1\}^k} \sum_{j \in \{0,1\}^{n-k}} \frac{|\psi_i^j\rangle}{|\psi_i^j\rangle},
\]

where \( |\psi_i^j\rangle \) is number of qubits, \( k \) is the number of the partially sliced vertices in the cut of \( C \). Here and below in this section we will assume that \( i, i', i'' \in \{0,1\}^k \), and \( j, j' \in \{0,1\}^{n-k} \). The quantum state \( |\psi\rangle \) can be expressed as:

\[
|\psi\rangle = C_2C_1|0\rangle = \sum_{i \in \{0,1\}^k} \sum_{j \in \{0,1\}^{n-k}} \frac{|\psi_i^j\rangle}{|\psi_i^j\rangle}.
\]

Note that when \( i \neq i' \), the vectors \( |\psi_i\rangle \) and \( |\psi_i'\rangle \) are orthogonal:

\[
\langle \psi_i | \psi_i' \rangle = \sum_{j, j'} \langle j | C_i^\dagger | j' \rangle \langle j' | C_i | 0 \rangle = \begin{cases} 1 & \text{if } i = i' \\ 0 & \text{else} \end{cases}.
\]

For every set \( X \subset \{0,1\}^k \) we define the vector \( |\psi_X\rangle = \sum_{i \in X} |\psi_i\rangle \). It is the orthogonal projection of the vector \( |\psi\rangle \) on the vector \( |\psi_X\rangle \). The fidelity between 2 mixed states defined by density matrices \( \rho \) and \( \sigma \) is defined by formula:

\[
F(\rho, \sigma) = \left( \text{tr} \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \right)^2.
\]

For the pure states \( \rho = |\psi_p\rangle \langle \psi_p| \) and \( \sigma = |\psi_a\rangle \langle \psi_a| \) this formula can be simplified:

\[
F(\rho, \sigma) = |\langle \psi_p | \psi_a \rangle|^2.
\]

The vector \( |\psi_X\rangle \) has unit norm, hence it can be interpreted as a pure state. Then we can calculate the fidelity between the states \( |\psi_X\rangle \) and \( |\psi\rangle \) as

\[
F = |\langle \psi_X | \psi \rangle|^2 = \frac{\sum_{i \in X} ||\psi_i||^2}{\sum_{i \in X} ||\psi_i||^2}.
\]

Assume we know all the norms \( ||\psi_i|| \), then we can define \( X \) to be the set of the indices \( i \in \{0,1\}^k \) of the vectors \( |\psi_i\rangle \) with the maximal norms. In this case, we get

\[
F = \sum_{i \in X} ||\psi_i||^2 \geq \frac{|X|^2}{2^k} \sum_{i \in X} ||\psi_i||^2 = \frac{|X|^2}{2^k}.
\]

(2)

Since the vectors \( |\psi_i^j\rangle \) are also orthogonal, we have

\[
||\psi_i^j||^2 = \sum_j ||\psi_i^j||^2 = \sum_j ||C_2 | j i^j ||^2 = \sum_j \sum_j \langle j | C_1 | 0 \rangle^2 \equiv \sum_j \sum_j \langle j | C_1 | 0 \rangle^2.
\]

Let us mention that \( ||\psi_i||^2 \) can be interpreted as the probability to obtain the state \( |i\rangle \) after we measure the qubits corresponding to the partially sliced vertices on the output of the subcircuit \( C_1 \). Moreover, the norms of all vectors \( |\psi_i\rangle \) can be calculated simultaneously as the result of a contraction for the tensor network shown on the right part of Fig. 1. It is not hard to see that the result of the contraction for this tensor network corresponds to

\[
||\psi_i||^2 = \sum_{i', i''} \sum_j \delta_{i, i', i''} \langle 0 | C_1^\dagger | j i'\rangle \langle j i'' | C_1 | 0 \rangle,
\]

where \( \delta_{i, i', j, k} = \begin{cases} 1, & \text{if } i = j = k, \\ 0, & \text{else} \end{cases} \). Recall that the contraction with the tensor \( \delta_{i, i, j, k} \) is equivalent to the identification of the variables \( i, j \) and \( k \).
Select the set $S \subseteq I$ of $k$ partial summed sliced vertices such that no vertex from $S$ inside the lightcone of another vertex from $S$ (i.e., all $k$ vertices are outputs of some subcircuit of $C$). Moreover, vertices from $S$ should be as close as possible to the inputs of the circuit $C$, and $\sum_{i \in S} \sum_{j \in \text{input}} |a_{ij}|^2$ should be minimal. The value of $k$ can be selected in each case depending on the preliminary estimate of the total algorithm complexity.

2. Let $C_1$ be the minimal subcircuit, with the same inputs as the inputs of $C$ and the outputs containing the whole set $S$, i.e., the subcircuit $C_1$ consists of the union of the lightcones of all vertices from $S$.

3. Calculate the norms $\|\psi_i\|$ by contracting the tensor network shown on Fig. 1.

4. Select a set $X \subseteq \{0, 1\}^k$ of minimal size such that $\sum_{i \in X} \|\psi_i\|^2 \geq f$. It is enough to choose $X$ consisting of $i \in \{0, 1\}^k$ with maximal norms $\|\psi_i\|$.

It is easy to see that $|X| \leq \lceil f 2^k \rceil$. Together with the set $X$ we obtain the fidelity $F = \sum_{i \in X} \|\psi_i\|^2$.

The computational cost of the simulation with fidelity $f$ can be estimated as

$$C_s(f) = \frac{|X|}{2^k} C_s(1) \leq \frac{\lceil f 2^k \rceil}{2^k} C_s(1) < (f + 2^{-k}) C_s(1),$$

where $C_s(1)$ is the corresponding computational cost with 100% fidelity.

In Appendix B, you can find a more detailed variant of the above slice selection algorithm.

C. Partial contraction

Suppose we have one of the following simulation tasks: calculation of a single amplitude, calculation of a batch of amplitudes or calculation of a set of batches of amplitudes. In all these cases we need to calculate some components of the full-state vector $|\psi\rangle = C |0\rangle$. Suppose we also have a contraction tree $T$ for this task, a set of all sliced vertices $I$, a set $S \subseteq I$ of partially sliced vertices, and a set $X \subseteq \{0, 1\}^k$ of slices such that $\|\psi_X\|^2 = F$. To perform our simulation task with fidelity $F$ it is enough to calculate corresponding components of the vector $|\psi_X\rangle$ instead of $|\psi\rangle$. This can be done as follows:

1. Calculate the sum of all slices of the circuit $C$ with contraction tree $T$, corresponding to the indices $X \times \{0, 1\}^{|I| - |S|}$, i.e., when the partially sliced vertices (from $S$) take values from $X$, the fully sliced vertices (from $I \setminus S$) take all possible values.

2. Normalize the result dividing by $\sqrt{F}$.

II. SAMPLING ALGORITHM

A. Modified rejection sampling for RQCs

In the previous section, we described how to calculate some amplitudes or batches of amplitudes for the state $|\psi\rangle = |\psi_X\rangle$ that approximates the exact state $|\psi\rangle = C |0\rangle$ with the fidelity at least $f$. Here we describe the algorithm that we use to simulate the RQCs from Google’s supremacy experiment. In this algorithm, we assume that the output probability distribution does not have bitstrings with very high probability. In this case, we can calculate some amplitudes and use the modification of the frugal rejection sampling algorithm described below.

If target fidelity is $\ll 1$ as in Google’s experiment, then we cannot use the approach proposed in [28] because it does not guarantee enough precision to maintain...
the fidelity when it is already small. So, we need another approach to reduce the number of batches needed to produce the given number of samples. We suppose that the bitstring $b$ is sampled according to the distribution corresponding to the measurement of the state $|\psi\rangle$. We denote the corresponding measurement operation by $M$. Let us separate the qubits into 2 parts $A$ and $B$, $N_A = 2^{|A|}$, $N_B = 2^{|B|}$. Then each bitstring $b$ can be represented as a pair $(b_A, b_B)$, where $b_A \in [N_A]$, $b_B \in [N_B]$. Denote the probability of the bitstring $(i, j)$ by $p_{i,j} = P\{M(\psi') = (i, j)\}$. Let $p_j$ be the probability that $M_B(\psi') = j$, then $p_j = \sum_{i \in [N_A]} p_{i,j}$.

At each step, we do the following.

1. Select at random $j \in [N_B]$ and calculate the batch of amplitudes that gives us the probabilities $p_{i,j}$ and hence $p_j$; $i \in [N_A]$, $j \in [N_B]$.

2. We accept this batch with probability $t_j = \min\left(1, \frac{p_{i,j} N_B}{\alpha}\right)$, where $\alpha > 1$ is a parameter.

3. If the batch is accepted, we sample one bitstring from this batch according to the distribution where a bitstring $(i, j)$ has the conditional probability $P\{b = (i, j) \mid b_B = j\} = P\{b_A = i \mid b_B = j\} = p_{i,j}/p_j$.

Since the average batch probability is $1/N_B$, the average number of batches we need to get one sample is approximately equal to $\alpha$. If we want to sample $k$ bitstrings, we should calculate approximately $\alpha k$ batches and then sample from them using the described algorithm. Note that we should choose the parameter $\alpha$ using the knowledge about the bitstrings distribution to be sure that $p_j < \alpha/N_B$ for all but a few.

The probability $p'_{i,j}$ that the sample is produced in one step and it is the bitstring $b$ is the product of the probabilities $1/N_B$ ($j = b_B$ at first step), $t_j$ (we accept batch) and $p_{i,j}/p_j$ ($b = (i, j)$ conditioned on $b_B = j$), i.e.,

$$p'_{i,j} = \frac{1}{N_B} \min\left(1, \frac{p_{i,j} N_B}{\alpha}\right) \frac{p_{i,j}}{p_j} = \frac{1}{\alpha} \min\left(\frac{\alpha}{N_B}, p_j\right) \frac{p_{i,j}}{p_j}.$$  

Let $p_j' = \min(p_j, \alpha/N_B)$, $\epsilon_j = p_j - p_j'$, $\epsilon = \sum_{j \in [N_B]} \epsilon_j$. The probability that a sample is produced in one step is

$$t = \sum_{i,j} p'_{i,j} = \frac{1}{\alpha} \sum_{j \in [N_B]} p'_j = \frac{1 - \epsilon}{\alpha}.$$  

When some sample is produced, the probability that it is a bitstring $(i, j)$ is

$$\tilde{p}_{i,j} = \frac{1}{\epsilon} p_j' \frac{p_{i,j}}{p_j}.$$  

### B. Statistical variational distance

Define $\tilde{p}_j = \sum_{i = 1}^{N_A} \tilde{p}_{i,j} = p_j'(1 - \epsilon)$. The statistical variational distance between $p$ and $\tilde{p}$ is

$$D(p, \tilde{p}) = \frac{1}{2} \sum_{j=1}^{N_B} \left| p_j - \tilde{p}_j \right| = \frac{1}{2} \sum_{j=1}^{N_B} \left| p_j - \tilde{p}_j \right|.$$  

Since $\sum p_j = 1 = \sum \tilde{p}_j$ and $|x - y| = y - x + 2 \max(0, x - y)$, we have

$$\sum_{j=1}^{N_B} p_j - \tilde{p}_j = 2 \sum_{j=1}^{N_B} \max(0, p_j - \tilde{p}_j).$$  

Taking into account $\tilde{p}_j \geq p_j'$ and $p_j \geq p_j'$, from (3) and (4) we obtain

$$D(p, \tilde{p}) \leq \sum_{j=1}^{N_B} \max(0, p_j - \tilde{p}_j) \leq \sum_{j=1}^{N_B} \max(0, p_j - p_j') = \epsilon.$$  

We should choose the parameter $\alpha$ in order to make $\epsilon$ small enough to obtain the given fidelity. Suppose we sample from state $|\psi\rangle$ such that $F(|\psi\rangle, |\psi\rangle) = |\langle \psi | \psi \rangle|^2 \geq f$. Distance $D(p, \tilde{p})$ corresponds to trace distance between the density matrix $M(|\psi\rangle)$ and the density matrix $M(\tilde{\psi})$. We need to estimate the fidelity $f' = F(M(|\psi\rangle), M(\tilde{\psi}))$. Using the triangle inequity for Bures metric $D_B(p, \sigma) = \sqrt{2(1 - \sqrt{F(p, \sigma)})}$, we have

$$D_B(M(|\psi\rangle), \tilde{p}) \leq D_B(M(|\psi\rangle), p) + D_B(p, \tilde{p}).$$  

Since $\text{diag}(p) = M(|\psi\rangle)$ and

$$F(M(|\psi\rangle), M(\tilde{\psi})) \geq F(|\psi\rangle, |\tilde{\psi}\rangle) \geq f,$$  

we have

$$D_B(M(|\psi\rangle), \tilde{p}) \leq \sqrt{2(1 - \sqrt{f'})}.$$  

We will use following relation between fidelity and trace distance: $1 - \sqrt{F(p, \tilde{p})} \leq D(p, \tilde{p})$. For short denote $d = D(p, \tilde{p})$, then

$$D_B(p, \tilde{p}) = \sqrt{2(1 - \sqrt{F(p, \tilde{p})})} \leq \sqrt{2d}.$$  

Substituting (7), (8), and the definition of $D_B(M(|\psi\rangle), \tilde{p})$ into (6) and dividing by $\sqrt{2}$ we have

$$\sqrt{1 - \sqrt{f'}} \leq \sqrt{1 - \sqrt{f} + \sqrt{d}}.$$  

To obtain nontrivial estimation on $f'$, we require that $d < f/16$. In this case

$$\sqrt{f'} \geq \sqrt{f} - d - 2 \sqrt{d(1 - \sqrt{f})} > \sqrt{f} - 2\sqrt{d},$$
we compare actual distribution of bitstring and Table 1. Finally, we have an estimate
\[
f' \geq f(1 - 4\sqrt{d/f}) ,
\]
where \( d \) is the sampling algorithm error in terms of the trace distance. Using (5) we can estimate \( d \) from above by \( \epsilon \) which we can estimate in different ways.

For example, if we want to sample with fidelity \( f' = 1\% \), then we can set \( f = 1.1 \) and choose \( \alpha \) such that \( \epsilon < 5 \cdot 10^{-6} \). Note that when we estimate \( \epsilon \), we always assume something about distribution \( p \) (for example, for random quantum circuit with enough depth components of \( p \) have Porter-Thomas distribution).

C. Estimation of \( \epsilon \)

One way to estimate \( \epsilon \) is to use the assumption that the vector \( |\psi'\rangle \) has Porter-Thomas distribution. In this case, the probabilities \( p_{i,j} \) are almost independent and have exponential distribution \( p_{i,j} \sim \text{Exp}(2^{n}) \). The probability \( p_{j} \) of a batch of size \( N_{A} \) is sum of \( N_{A} \) independent exponentially distributed random variables and has gamma-distribution \( p_{j} \sim \text{Gamma}(N_{A}, 2^{n}) \). We can estimate error expectation
\[
\mathbb{E}\epsilon_{j} = \frac{\Gamma(N_{A}, 2^{n}\alpha/N_{B})}{\Gamma(N_{A})} = \frac{\Gamma(N_{A}, \alpha N_{A})}{\Gamma(N_{A})}
\]
where \( \Gamma(s, x) = \int_{x}^{\infty} t^{s-1} e^{-t} \, dt \) is the upper incomplete gamma function. Hence,
\[
\mathbb{E}\epsilon = \sum_{j=1}^{N_{A}} \mathbb{E}\epsilon_{j} = N_{B} \frac{\Gamma(N_{A}, \alpha N_{A})}{\Gamma(N_{A})} .
\]

On figure 2 we compare actual distribution of bitstring probabilities with exponential distribution and also compare distribution of batch probabilities with gamma distribution.

Second way to estimate \( \epsilon \) is to calculate \( \epsilon_{J} = \sum_{j \in J} \epsilon_{j} \) for set \( J \) for which batches were calculated during algorithm. Then calculate estimation \( \hat{\epsilon} = \epsilon_{J}/|J| \). This method doesn’t use assumption about concrete distribution however works in assumption that all probabilities have the same order of magnitude which is true for RQCs of enough depth.

III. EXPERIMENTS

A. Sampling with fidelity

1. Experiment with \( |\psi\rangle \) norm calculation

The complexity of simulation with fidelity depends on how many slices should be contracted to obtain given fidelity. The number of slices depends on how does maximal \( || |\psi\rangle \|| \) norm differ from mean \( || |\psi\rangle \|| \) norm. In Fig. 3 there are calculated norms of \( || |\psi\rangle \||^{2}/2^{k} \) for schedules used for sampling 1M amplitudes for sycamore RQCs. There were 2 settings: \( k = 10 \) and \( k = 15 \) indices involved in partial slicing. In Table I there are complexity of \( || |\psi_{i}\rangle \|| \) calculation, normalized standard deviations and range of \( || |\psi_{i}\rangle \||^{2} \). Normalization here is multiplication by \( 2^{k} \) to make mean value equal to 1.

From Fig. 3 and Table I we see that the distribution of \( || |\psi_{i}\rangle \||^{2} \) is far from normal in most cases, and sometimes variance is very big, so it cannot be assumed that all \( || |\psi_{i}\rangle \||^{2} \) are close to \( 2^{-k} \), and we should find it directly. In some cases such as \( m = 14 \) maximal value 2 times bigger than average, this allows to calculate almost 2 times less slices to obtain fidelity 0.02 (compare slicing ratio and fidelity \( F \) in Table III). For \( m \geq 16 \) variance is not so big, and slicing ratio is almost equal to fidelity. Probably, this is because sliced vertices for \( m \geq 16 \) are closer to the middle of circuit and subcircuit \( C_{1} \) can be viewed as a random circuit with enough depth.

2. Sampling algorithm validation on elided circuits

In this section, we verify our sampling algorithm on the elided circuits from Google’s experiment [6]. Since our

| \( m \) | \( k \) | Complexity | \( \sqrt{2^{m}D || |\psi_{i}\rangle \||^{2}} \) | range \( (2^{m}|| |\psi_{i}\rangle \||^{2}) \) |
|---|---|---|---|---|
| 20 | 10 | \( 1.2 \times 10^{10} \) | 0.021 | [0.95, 1.05] |
| 20 | 15 | \( 1.6 \times 10^{11} \) | 0.040 | [0.94, 1.06] |
| 18 | 10 | \( 2.1 \times 10^{12} \) | 0.017 | [0.96, 1.04] |
| 18 | 15 | \( 2.3 \times 10^{12} \) | 0.021 | [0.93, 1.07] |
| 16 | 10 | \( 2.7 \times 10^{11} \) | 0.007 | [0.97, 1.02] |
| 16 | 15 | \( 5.6 \times 10^{14} \) | 0.013 | [0.96, 1.04] |
| 14 | 10 | \( 8.7 \times 10^{10} \) | 0.341 | [0.50, 1.88] |
| 14 | 15 | \( 5.4 \times 10^{10} \) | 0.341 | [0.49, 1.93] |
| 12 | 8 | \( 3.6 \times 10^{12} \) | 0.135 | [0.78, 1.26] |

TABLE I. Complexity, deviation and range of \( || |\psi_{i}\rangle \||^{2} \).
method adopts the frugal rejection sampling, to work well it requires some special properties of the output probability distribution. In particular, it works well when there is a very small number of output bitstrings that have probabilities significantly bigger than $1/2^n$, where $n$ is the number of qubits. Google’s team has already provided some data in [6] that confirms this assumption for supremacy RQCs. In this section, we demonstrate our algorithm on the elided circuits for which we can calculate the exact amplitudes and $F_{\text{XEB}}$. We apply our algorithm with target fidelity 1% and compare the obtained $F_{\text{XEB}}$ with 0.01. The calculated fidelities are shown in Fig. 1.

**Experiment description.** For each number of cycles $m = 12, 14, 16, 18, 20$ and each of 10 elided circuit instances provided by Google we calculate $2^m$ random batches of size 64 with target fidelity 1%. Note that the actual fidelity $F = \sum_{i \in X} ||\psi_i||^2$, shown in table II, is slightly higher. Then we apply our modified frugal rejection sampling algorithm to calculated batches and obtain slightly more than $10^6$ samples. After we get all samples, we calculate the amplitudes for these samples with 100% fidelity using the multi-tensor contraction algorithm from [19]. After we get all amplitudes, we calculate $F_{\text{XEB}}$ for all 10 circuit instances. The average $F_{\text{XEB}}$ for each number of cycles is shown in table II.

### 3. Sampling supremacy circuits

In table III shown sampling complexity for Google’s RQC circuits with different number of cycles where time is shown for 1 GPU Tesla V100.

**Experiment description.** For each number of cycles $m = 12, 14, 16, 18, 20$ we take first full circuit instance provided by Google and calculate $2^{16}$ batches of size 64 with target fidelity 2% for $m = 12, 14, 16, 18$ and 0.2% for $m = 20$, which is not less than as Google’s sycamore quantum computer has on the same circuit (see target and actual fidelities in table III). Then 1M samples was generated for each $m$ using proposed modified frugal rejection sampling algorithm. Experimental data [20] contains 5 text files, each contains 1M bitstrings for corresponding $m$.

Simulation run on 4 servers, each has 8 GPU Tesla V100 16GB, total running time for all cases is approximately 14.5 days. In table III there is detailed information about each case complexity and running time normalized for one GPU Tesla V100. Estimated time of generating 1M samples for $m = 20$ on Summit supercomputer is approximately 24 min.

For the recent experiment on Zuchongzhi quantum computer [7] the sampling task is significantly more complex. We prepared a contraction tree for generating 2M batches of size 64 for a 56-qubit circuit with 20 cycles. The full contraction complexity is $2.15 \cdot 10^{25}$ Flops, the contraction requires 80 GB memory. We estimated the time for sampling with target fidelity 0.066% on Selene supercomputer with 4480 Tesla A100 80GB. Note that the contraction schedule has high arithmetic intensity. Hence in this estimation we assume that the computational efficiency of our simulator will be at least 50%.

![FIG. 3. The distribution of $||\psi_i||^2$. The labels on X axis shows the normalized values $2^k||\psi_i||^2$. Note that the mean values with this normalization are equal to 1.](image)

### TABLE II. The fidelities obtained for the Google’s elided circuits when the target fidelity is 1%. For each number of cycles $m$ and for all 10 elided circuits provided by Google 1M samples was generated. The fidelities in the table are averaged over these 10 circuits.

| $m$ | 12  | 14  | 16  | 18  | 20  |
|-----|-----|-----|-----|-----|-----|
| $F$ | 0.0104 | 0.0103 | 0.0117 | 0.0104 | 0.0113 |
| $F_{\text{XEB}}$ | 0.0108 | 0.0099 | 0.0121 | 0.0103 | 0.0114 |

### TABLE III. Complexity of sampling 1M samples with fidelity. $C_s$ is complexity measure, number of complex number multiplications during contraction. Total number of flops is $8C_s$.

| $m$ | target fidelity | $F$ | slicing ratio | $C_s$ | efficiency | time  |
|-----|----------------|-----|--------------|------|------------|-------|
| 12  | 0.02           | 0.0215 | 0.0136 | 1.1 \cdot 10^{16} | 26% | 6.6 hours |
| 14  | 0.02           | 0.0218 | 0.0117 | 5.4 \cdot 10^{16} | 42% | 20.5 hours |
| 16  | 0.02           | 0.0208 | 0.0205 | 4.0 \cdot 10^{17} | 52% | 5 days |
| 18  | 0.02           | 0.0201 | 0.0195 | 1.9 \cdot 10^{18} | 32% | 40 days |
| 20  | 0.002          | 0.0021 | 0.0021 | 2.2 \cdot 10^{19} | 31% | 15 months |
The time estimate in this case is:

\[(2.15 \cdot 10^{25})\text{FLOPs} \times 0.066\% \approx 3.6 \cdot 10^5 \text{s} \approx 4 \text{ days}.\]

Taking into account a relatively high arithmetic intensity, some additional optimization can be done using tensor cores with single precision. Therefore potentially the simulation time can be reduced to 1 day or even less.

### B. Spoofing Linear XEB

In this section, we show our experimental results on the spoofing Linear XEB test. If we want to get a set of \(N\) bitstrings with \(F_{\text{XEB}} \geq f\), then we need to do the following steps:

1. Let \(b = \lceil \log_2(10N) \rceil\).
2. Choose the set \(F\) consisting of \(b\) free circuit outputs which gives the minimal contraction complexity.
3. Calculate the amplitudes for batch \(B\) of \(2^b\) bitstrings using partial slicing with target fidelity \(f\).
4. Select the \(N\) bitstrings from the batch \(B\) with maximal absolute values of amplitudes.

The computational cost of spoofing the XEB test in Google's quantum supremacy experiment for different number of cycles \(m\) is shown in Table IV. We also estimated the time to spoof the Linear XEB test in the recent experiment with 56-qubit circuit on Zuchongzhi quantum computer [7]. Our estimates show that in this case the spoofing can be done in 1 month on one Tesla V100 16GB GPU.

Assume that we have random circuit \(C\) on \(n\) qubits, random variables \(p_i = |\langle i | C | 0 \rangle|^2\) have Porter-Thomas distribution. Moreover, if circuit depth is enough, \(C | 0 \rangle\) is uniformly distributed on complex sphere \(S^2\), for given set of bitstrings \(B\), \(|B| = 2^n\) we can assume that random variables \(p_i, i \in B\) are almost independent. If we calculated approximate values \(p'_i\) with fidelity \(f\) and take set

\[S \text{ of } N = r|B|\text{ bitstrings with maximal values of } p'_i,\]

\[E F_{\text{XEB}}(S) = -f \ln r + O(N^{-1} \ln N),\]

(11)

| \(m\) | XEB slicing ratio | \(2^{25}\) amps batch complexity | spoofing time |
|------|-------------------|-------------------------------|--------------|
| 12   | 3.2%              | 7.5 \cdot 10^{14}            | 1 sec        |
| 14   | 1.2%              | 1.6 \cdot 10^{14}            | 2 sec        |
| 16   | 1.4%              | 1.7 \cdot 10^{17}            | 17 min       |
| 18   | 0.83%             | 1.1 \cdot 10^{18}            | 30 min       |
| 20   | 0.47%             | 6.9 \cdot 10^{18}            | 4 hours      |

TABLE IV. Linear XEB spoofing (the time is for one GPU Tesla V100 16GB)

see Appendix A for details. For real circuits assumption that \(p_i\) is independent is not always true especially when \(B\) is a batch of correlated bitstrings. In random subsets of a batch we can assume that all \(p_i\) are independent but distribution of \(p_i\) is exponential with expectation that depends on \(F_{\text{XEB}}(B)\). Taking into account this fact we have following heuristic equality

\[E(\mathcal{F}_{\text{XEB}}(S) - \mathcal{F}_{\text{XEB}}(B)) \approx -f \ln r.\]

(12)

**Experiment description.** For each number of cycles \(m = 12, 14, 16, 18, 20\) we calculate batch \(B_m(f)\) of \(2^{25}\) amplitudes with different ratio of slices \(f\) from 0.1% to 2% and obtain partially calculated probability distributions. From each batch of partially calculated amplitudes \(B_m(f)\) we take the set \(S_m(f,r)\) of \([r \cdot 2^{25}]\) samples with maximal absolute values where \(r \in \{0.1, 0.2, 0.36, 0.5\}\) (0.36 is approximate value of \(e^{-1}\)). Then we calculate precise amplitudes for the same batch and which we use to calculate \(\mathcal{F}_{\text{XEB}}\) for all sets of samples \(S_m(f,r)\). Calculated \(\mathcal{F}_{\text{XEB}}\) for all cases is shown on figure 4.

We can see that \(\mathcal{F}_{\text{XEB}}(S_m(f,0.36)) \approx \mathcal{F}_{\text{XEB}}(S_m(0,1)) + f\) where \(\mathcal{F}_{\text{XEB}}(S_m(0,1))\) corresponds to whole batch. In the case \(m = 14\) \(\mathcal{F}_{\text{XEB}}\) of full batch \(\approx -0.0094\) which means some weakness in entanglement of qubits in the output state of circuit with 14 cycles. For bigger \(m\) there are no such problems and \(\mathcal{F}_{\text{XEB}}\) of full batch is close to 0.
Appendix A: Theoretical XEB estimation for spoofing

Assume that all $p_i$ are independent and $p_i \sim \text{Exp}(\lambda)$ where $\lambda = 2^n$. If we consider $N = |B|$ random variables $x_k \sim \text{Exp}(\lambda)$, $k = 1, N$, then for order statistics $x(k)$ we have

\[
\mathbb{E} x_{(N-k+1)} = \frac{1}{\lambda} (\ln N - \ln k + O(1/k)),
\]

\[
\mathbb{E} \sum_{j=1}^{k} x_{(N-j+1)} = \frac{1}{\lambda} \left( k \left( \ln \frac{N}{k} + 1 \right) + O(\ln k) \right)
\]

Suppose we select the set $S$ of $rN$ samples with the maximal probabilities from a batch of size $N$, then

\[
\mathbb{E} F_{\text{XEB}}(S) = \mathbb{E} \left( \frac{2^n}{rN} \sum_{j=1}^{rN} x_{(N-j+1)} - 1 \right) = \frac{2^n}{\lambda} \left( \ln N - \ln rN + 1 + O((rN)^{-1} \ln rN) \right) - 1
\]

\[
= -\ln r + O(N^{-1} \ln N).
\]

For partial slicing we have orthogonal projection $v_f$ of state vector $v$ on some subspace with the norm $\|v_f\|^2 = f$. We assume that $v_f$ has the uniform distribution on the sphere of radius $\sqrt{f}$, so the components $p_{f,i} = |v_{f,i}|^2$ have exponential distribution with the parameter $2^n/f$. We have $v = v_f + v_f^\perp$, $\langle v_f | v_f^\perp \rangle = 0$. Moreover, the random vectors $v_f$ and $v_f^\perp$ are independent. It is not hard to see that

\[
\mathbb{E}(p_i | p_{f,i}) = \mathbb{E} p_{f,i}^2 + p_{f,i} = p_{f,i} + \frac{1 - f}{2^n}.
\]
Let $S = \{i_j \mid j = 1, \ldots, rN\}$ be the set of $rN$ indices $i$ with the maximal $p_{f,i}$, then

$$\mathbb{E}_{\mathcal{XEB}}(S) = \mathbb{E} \left( \frac{2^n}{rN} \sum_{j=1}^{rN} p_{i_j} - 1 \mid p_{f,i_j} = x_{(N-j+1)} \right)$$

$$= \frac{2^n}{rN} \sum_{j=1}^{rN} \mathbb{E} \left( p_{i_j} \mid p_{f,i_j} = x_{(N-j+1)} \right) - 1$$

$$= \frac{2^n}{rN} \left( rN \frac{1 - f}{2^n} + \sum_{j=1}^{rN} x_{(N-j+1)} \right) - 1$$

$$= 1 - f + f \left( \ln \frac{N}{rN} + 1 + O \left( \frac{\ln N}{N} \right) \right) - 1$$

$$= -f \ln r + O(N^{-1} \ln N).$$

**Appendix B: Formal slice selection algorithm**

Let $C$ be quantum circuit, then by $G(C)$ we denote the set of all its gates. Each gate inside the circuit is represented by a tuple $(\ell, A, q)$, where $\ell$ is the index of a gate in the circuit, $A$ is a unitary matrix, $q$ is a tuple of qubit indices. If $X$ is a set of gates, then by $C(X)$ we denote the circuit composed from these gates. By vertices of the circuit we understand the tensor legs in the corresponding tensor network. The lightcone $L_C(v)$ of a vertex $v \in V(C)$ contains all the gates the vertex $v$ depends on. Denote by $L_C(v)$ the set of all vertices that are the inputs of the gates from $L_C(v)$. For a set $S \subset V(C)$ we can define $L_C(S) = \cup_{v \in S} L_C(v)$, $L_C^*(S) = \cup_{v \in S} L_C^*(v)$. Algorithm 1 selects the subset $S$ of partially sliced vertices from the set $I$ of all sliced vertices.

**Algorithm 1: Slice selection**

- $k_0 := \lfloor 3 - \log_2 f \rfloor$;
- $S := \text{SlicedVertexSelect}(C, I, k_0)$;
- $k := |S|$;
- $C_1 := C(L_C(S))$;

  // Find tensor $R$, $R[i] = \| \psi_i \|^2$ for $i \in \{0, 1\}^k$

  $R := (\langle 0 | C_1^* \delta_{S=i} C_1 | 0 \rangle)_{i \in \{0, 1\}^k}$;

- Sort $i \in \{0, 1\}^k$ such that $R[i(1)] \geq \ldots \geq R[i(2^k)]$;
- $j := \min \{ t \in \{1, \ldots, 2^k \} : \sum_{j=1}^{t} R[i(j)] \geq f \}$;
- return $(S, \{i(1), \ldots, i(j)\})$;

The operator $\delta_{S=i}$ is a projector on the subspace where the qubit corresponding to the vertex $s_j$ is $|i_j \rangle$. In the tensor network it corresponds to fixing indices $s_1 = i_1, \ldots, s_k = i_k$.

The vertex subset $S$ is selected by the following algorithm that optimizes the gate number in the lightcone $L(S)$ in a greedy fashion.

**Algorithm 2: SlicedVertexSelect(C, I, k)**

- $S := \emptyset$;

  while $|S| < k$ and $I \setminus L_C^*(S) \neq \emptyset$ do

    $v := \arg \min_{u \in I \setminus L_C^*(S)} |L_C^*(S \cup \{v\})|$

    $S := (S \setminus L_C^*(v)) \cup \{v\}$;

  return $S$;