Experimental study of Shor's factoring algorithm using the IBM Q Experience

Mirko Amico,[1] Zain H. Saleem,[2] and Muir Kumph[3]
[1]*The Graduate School and University Center, The City University of New York, New York, New York 10016, USA*
[2]*Theoretical Research Institute of Pakistan Academy of Sciences, Islamabad 44000, Pakistan*
[3]*IBM T. J. Watson Research Center, Yorktown Heights, New York 10598, USA*

(Received 2 March 2019; published 8 July 2019)

We study the results of a compiled version of Shor's factoring algorithm on the *ibmqx5* superconducting chip, for the particular case of $N = 15$, 21, and 35. The semiclassical quantum Fourier transform is used to implement the algorithm with only a small number of physical qubits, and the circuits are designed to reduce the number of gates to the minimum. We use the square of the statistical overlap to give a quantitative measure of the similarity between the experimentally obtained distribution of phases and the predicted theoretical distribution of phases for different values of the period. This allows us to assign a period to the experimental data without the use of the continued fraction algorithm. A quantitative estimate of the error in our assignment of the period is then given by the overlap coefficient.

DOI: 10.1103/PhysRevA.100.012305

I. INTRODUCTION

Shor's factoring algorithm [1] is a well-known example of a quantum algorithm outperforming the best known classical algorithm. Experimental implementation of the algorithm with physical qubits, however, remains a challenge because of the errors introduced by the large number of qubits and gates required to execute the algorithm. In this paper we provide a proof-of-principle demonstration of a compiled version of Shor's factoring algorithm to factor the numbers $N = 15$, 21, and 35 using five, six, and seven superconducting qubits, respectively. Similar experiments have been done on setups like NMR [2], trapped ions [3], photons [4–6], photonic chips [7], and superconducting qubits [8,9]. However, with the exception of [3,5], all these realizations involve an oversimplified version of the algorithm which is equivalent to coin flipping [10] and no quantum hardware is needed to obtain the same results.

In our implementation, classical processing is used alongside quantum computation to overcome the lack of key functions of the device used. Furthermore, the number of physical qubits and the circuit depth are reduced to the minimum in order to minimize the effects of noise. The data are presented as estimates of the probability distribution of the values returned by the period register. While obtaining the probability distributions of the period register requires running the algorithm many times, as opposed to just once with the original continued fraction expansion, this allows the performance of quantum computers running this algorithm to be more directly evaluated. To measure the success of the experiments in different ways, the results are analyzed in both a qualitative way, with probability plots [11], and a quantitative way, with the square of statistical overlap (SSO) [12]. Probability plots are a useful tool to visualize differences between probability distributions, while the SSO provides a quantitative measure of their similarity. Using the overlap coefficient (see below), we can also use the SSO to assign a period to the experimental data, avoiding the continued fraction algorithm which does not work for such low number of qubits. Also, the overlap coefficient (OVL) gives an estimate of the probability that the experiment succeeded. The results of the experiments are in good agreement with the theory for $N = 15$ and 21. However, the experiment succeeded for $N = 35$ only about 14% of the time, where the cumulative errors coming from the high number of two-qubit gates became too large.

The paper is organized in the following way. A brief overview of Shor's factoring algorithm is given in Sec. II. Section III describes the hardware used for the experiment. In Sec. IV the implementation of the factoring experiment for $N = 15$, 21, and 35, respectively, is described. The results obtained from running the algorithm on the *ibmqx5* quantum processor are analyzed and discussed in Sec. V. Conclusions follow in Sec. VI.

II. OVERVIEW OF SHOR'S FACTORING ALGORITHM

The factoring algorithm invented by Shor [1] relies on the relation between the problem of factoring and the problem of order finding, for which a quantum speedup exists. In fact, finding the prime factors of a number N is equivalent to finding the exponent x for which the function $a^x \mathrm{mod} N = 1$, where a is an integer smaller than N picked at random. Such exponent is called the order, or period, of a. Let us briefly review the quantum part of the algorithm before diving into the details of the experiment. Two quantum registers are needed for the computation. One register is used to store the value of the period, called the period register, and the other is used to store the results of the computation, called the computational register. The size of both registers depends on the number N to be factored. In particular, the period register should have a number of qubits n_p in the interval $\log_2(N^2) \lesssim n_p \lesssim \log_2(2N^2)$ and the computational register should be large enough to be able to represent the number

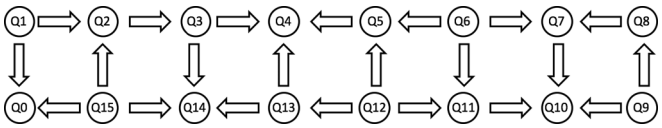


FIG. 1. Coupling map between the 16 qubits of the *ibmqx5* device. The arrows indicate the possible CNOT gates between pairs of qubits. In particular, the arrow starts from the control qubit and points to the target qubit.

$N-1$, resulting from the modular exponentiation function (MEF) $a^x \mathrm{mod} N$, thus requiring $n_q = \log_2 N$ qubits.

At the beginning of the quantum algorithm, the two registers are initialized to the state $|00...0\rangle_p |00...1\rangle_q$, where the subscripts p and q denote the period register and the computational register, respectively. The period register stores all the possible values of the exponent x, which will give an estimate of the period, by creating a uniform superposition of all possible bit strings through Hadamard gates on all qubits $\frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle_p$, where $Q = 2^{n_p}$, while the computational register stores the results of the MEF, $a^x \mathrm{mod} N$. After the first step, the two registers are in the state $\frac{1}{\sqrt{Q-1}} \sum_{x=0}^{Q-1} |x\rangle_p |a^x \mathrm{mod} N\rangle_q$. Then, the quantum Fourier transform (QFT) is applied to the period register so that $|x\rangle_p \rightarrow \frac{1}{\sqrt{Q}} \sum_{s=0}^{Q-1} e^{\frac{2\pi i s x}{Q}} |s\rangle_p$. As a result of the QFT, interference between all the possible states occurs. If the period register is then measured, a value of the phase s is measured with probability $P(s) = \frac{1}{Q} \sum_{x=0}^{Q-1} |e^{\frac{2\pi i s x}{Q}}|^2$. Rewriting x in terms of the period r as $x = x_0 + dr$, where x_0 and d are integers, the probability of an outcome s can be written as $P(s) = \frac{1}{Q} |e^{\frac{2\pi i x_0}{Q}}|^2 \sum_d |e^{\frac{2\pi i s d r}{Q}}|^2$. Clearly, a value of s such that $\frac{s}{Q} = \frac{c}{r}$, where c is an integer, will be observed with high probability.

The final part of the algorithm involves classical processing of the measurement obtained in the quantum part. The value of the period r can be found from the fraction $\frac{s}{Q}$ by using the continued fraction algorithm or, as done in this paper, by running the algorithm many times to get a direct estimate of the probability distribution of the values for the period register. A comparison between the measured probability distribution and the theoretically predicted distribution for the period r can be made using the SSO and the best fit gives the most likely period. If the period r calculated in this way is odd or $r = 0$, the algorithm fails and one restarts by picking a different base a. If r is even, $(a^r - 1) \mathrm{mod} N$ can be factored into $(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \mathrm{mod} N$. The final step is to check if $(a^{\frac{r}{2}} + 1) \mathrm{mod} N$ has a common divisor with N by checking that $\gcd(a^{\frac{r}{2}} + 1, N) \neq 1$. If that is true, then the two factors of N are $\gcd(a^{\frac{r}{2}} + 1, N)$ and $\gcd(a^{\frac{r}{2}} - 1, N)$.

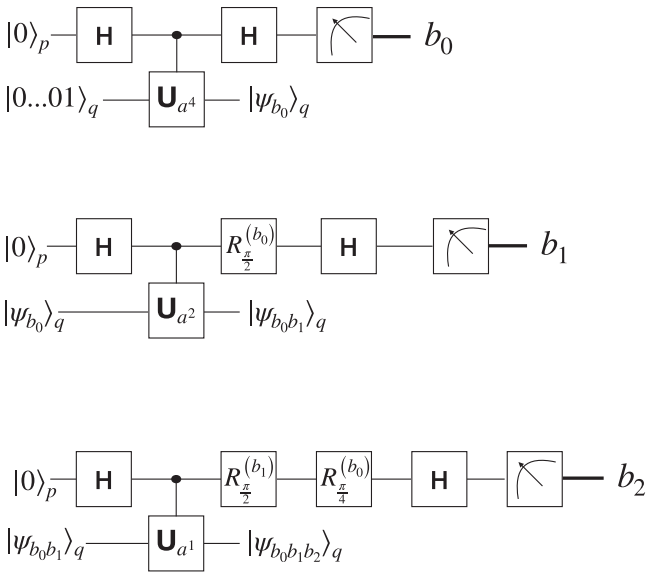


FIG. 3. Circuits used in the experimental implementation of Shor's algorithm on *ibmqx5*. The circuit of Fig. 2 is divided in three separate parts. Each circuit contains a stage of modular exponentiation and a measurement of the period register. The different circuits are joined using a classical algorithm which computes the quantum state of the computational register at the end of the previous circuit and feeds it as input to the next circuit. The classical algorithm also adds the right rotation gates on the period qubit in each successive circuit, based on the results of previous measurements.

As mentioned earlier, the execution of this version of the algorithm requires $n_q = \log_2(N)$ qubits in the computational register to perform the modular exponentiation and at least another $n_p = 2\log_2(N)$ qubits in the period register to perform the QFT. Thus the complete algorithm requires a total number of $3\log_2(N)$ qubits. Even the factoring of a number as small as $N = 15$ needs 12 qubits in the input register to execute this algorithm, which is still a challenge for today's physical realizations of quantum computers. However, Kitaev [13] observed that for the purpose of algorithms like Shor's, where one does not need the information on the relative phase of the output states but only their measured probability amplitudes, one can replace the fully coherent quantum Fourier transform with the semiclassical quantum Fourier transform (sc-QFT). In the sc-QFT, one of the qubits of the period register is measured each time. The result of the measurement of the qubit is then used to determine the type of measurement on the next one. This enables the replacement of the $2\log_2(N)$ qubits of the period register with a single qubit measured

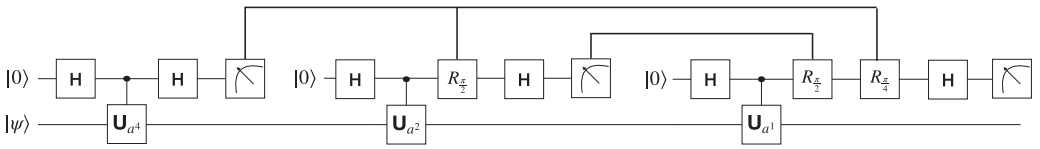
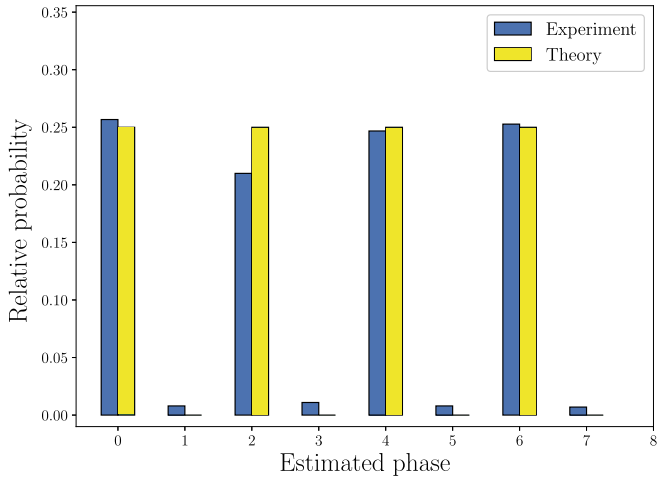
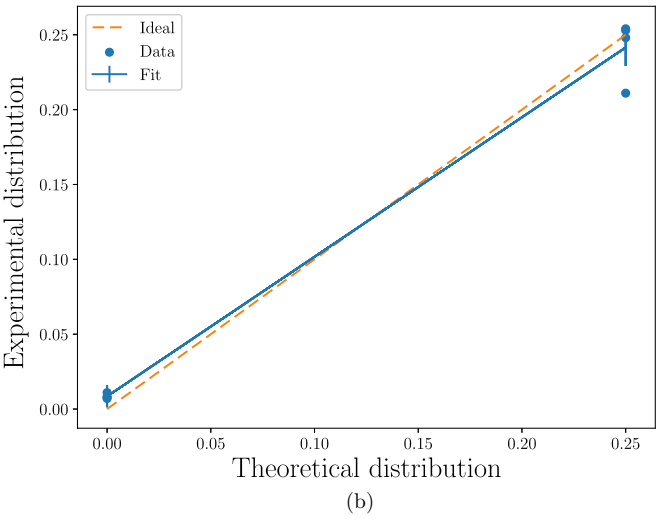


FIG. 2. Circuit for factoring $N = 15$, 21, and 35 implemented using the scheme shown in [3]. The first register (top), the period register, stores the estimation of the period, which can be done by using only one qubit through the sc-QFT. The second register (bottom), the computational register, stores the outcomes of the modular exponentiation function $a^x \mathrm{mod} N$ computed through the controlled-U_{a^x} gate. The specific circuits for U_{a^x} used in factoring $N = 15$, 21, and 35 for a specific base a are shown in the Appendix.





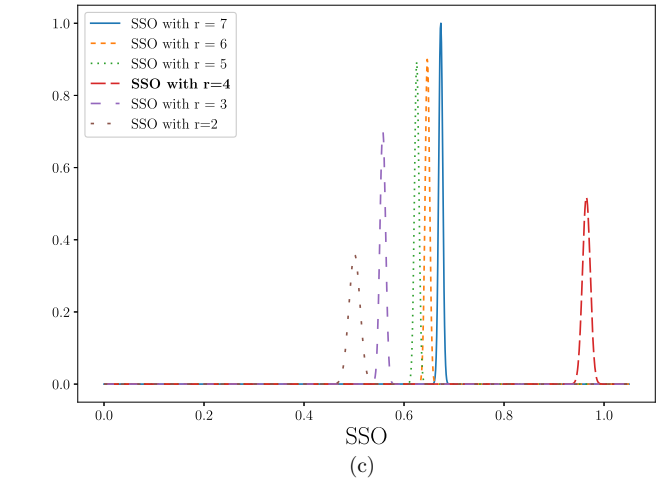


FIG. 4. (a) Probability of finding a given phase for $N = 15$ with base $a = 2$ and (b) probability plot of the theoretical distribution and the experimental distribution for $r = 4$. The experimental distribution is depicted through the collection of data and a fit of the data. (c) SSO of the experimental data with the theoretical probability distribution corresponding to all possible values of the period r.

multiple times. For the case of factoring $N = 15$, Kitaev's approach reduces the total number of qubits required to $n = 5$ and for the case of $N = 21$ and 35 to $n = 6$ and 7, respectively, which are small enough numbers for the presently available hardware to handle. This decrease in the system size, however, comes with the drawback of requiring in-sequence single-qubit readout and state reinitialization together with feed-forward of gate settings based on previous measurement results. The implementation of the sc-QFT has been described in [12,14] and realized in [3]. At present the IBM[1] quantum computer does not perform in-sequence single-qubit readout and qubit reinitialization. Below, we provide a procedure for going around this hurdle to implement the sc-QFT on the IBM Q device.

III. HARDWARE

We use the IBM *ibmqx5* chip with 16 superconducting qubits to implement our experiments for factoring the numbers $N = 15$, 21, and 35. The qubits are distributed on the plane, as two adjacent arrays of eight qubits each with couplings shown in Fig. 1.

The qubits' relaxation time T_1 ranges from 25 to 60 μs and their dephasing time T_2 ranges from 20 to 100 μs. The single-qubit gates have a high fidelity, measured at ~99.8% at the time of the experiment. The multiqubit gate fidelity was measured around 95–98% depending on the pairs of qubits considered. All gate errors are measured using simultaneous randomized benchmarking. Another source of error comes from the readout of the states of the qubits, which amounts to roughly an error of 5%. Using these parameters, the effects of noise can be incorporated in the simulation, obtaining a more accurate prediction for the output of the device.

IV. EXPERIMENT

Following the example given in [3], we implement the quantum part of Shor's factoring algorithm using the circuit depicted in Fig. 2. As can be seen in the circuit diagram in Fig. 2, rotations of the control qubit depend on the outcome of each of its measurements in the previous steps. Since the *ibmqx5* chip does not allow for qubit reset and conditional operation based on measurements, which are required to implement the sc-QFT suggested by Kitaev, we implement the algorithm as three separate quantum circuits as shown in Fig. 3.

In the first circuit, the system is initialized in the state $|0\rangle_p|0...01\rangle_q$ and the first bit, b_0, encoding the value of the period, is measured at the end. In the second circuit, the initial state $|0\rangle_p|\psi_{b_0}\rangle_q$ is prepared. Different states $|\psi_{b_0}$ are prepared depending on the value of b_0 measured in the previous circuit. Rotation gates on the period register are also inserted conditional on the value of b_0 before measuring the second bit encoding the value of the period, b_1. In the third circuit, depending on the values of b_0 and b_1, the qubit registers are initialized to $|0\rangle_p|\psi_{b_0b_1}\rangle_q$ and rotation gates are inserted before the measurement of b_2. The possible quantum states of the computational register can be computed classically for

[1]The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.

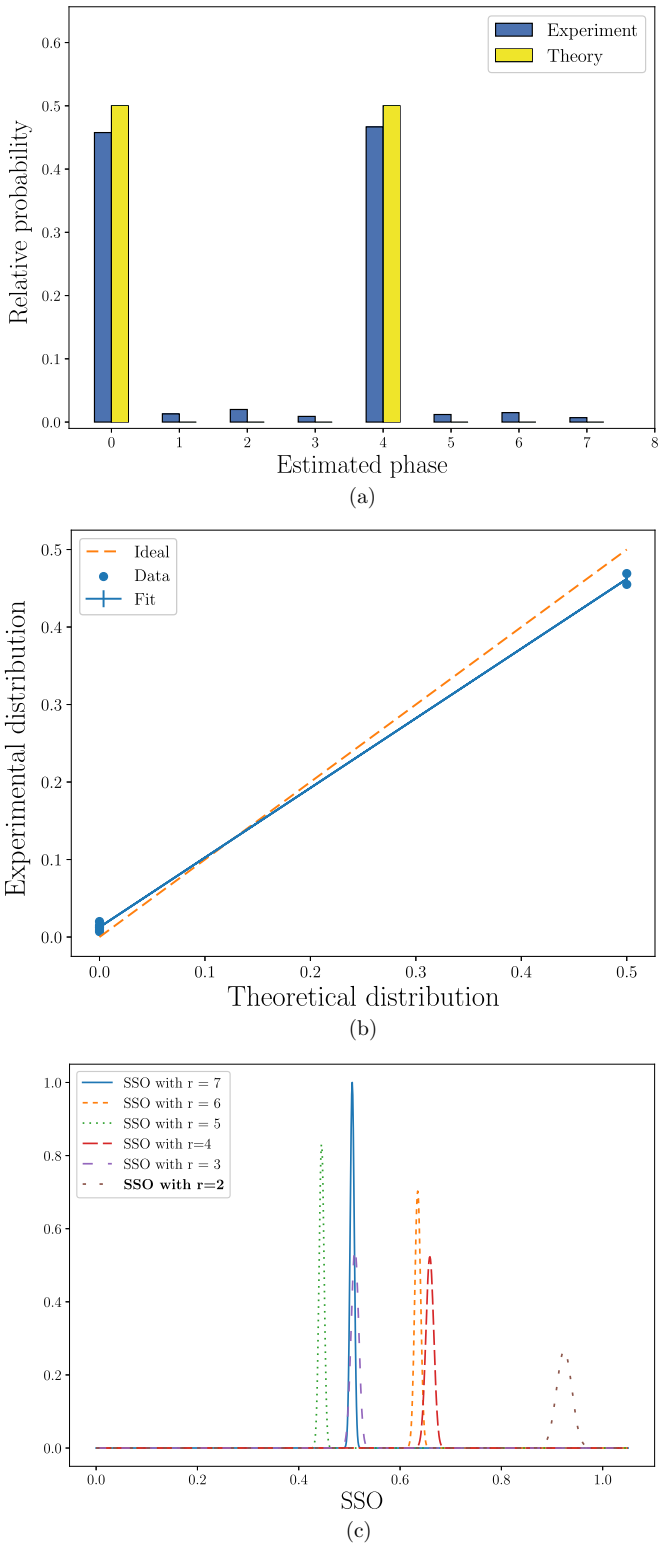


FIG. 5. (a) Probability of finding a given phase for $N = 15$ with base $a = 11$ and (b) probability plot of the theoretical distribution and the experimental distribution. (c) SSO of the experimental data with the theoretical probability distributions corresponding to all possible periods.

the full algorithm, conditional on the measurement results of the period register. This is just the result of successive modular exponentiation. At the beginning of each circuit, except the first one, there are two possible states of the computational register that have to be prepared depending on the value of the period register measured in the previous stage. If the measurement of the period register gives zero, then the computational register is prepared to the state $|\psi_0\rangle_q = |\psi\rangle_q + ZU_{a^x}|\psi\rangle_q$. If the period register gives 1, the computational register is initialized to $|\psi_1\rangle_q = |\psi\rangle_q - ZU_{a^x}|\psi\rangle_q$. This means that for an implementation with m stages a superposition of 2^m product states has to be prepared. However, the state at the mth stage is, at worst, the result of $m - 1$ modular exponentiations. Thus, breaking the circuit in this way only adds an extra number of gates which is polynomial in the number of stages m: $1 + 2 + 3 + \cdots + m = \frac{m(m+1)}{2}$, due to the gates needed for the state initialization at each stage. This retains the scalability of the implementation given in [3].

In the following experiments, we limit ourselves to the choice of one, or two, bases a to avoid redundancy. We specifically choose a nontrivial base (in the sense of [10]) for which a working quantum processor is needed to find the results. One could adopt the same approach to treat any such nontrivial bases. To understand what happens in the case of a trivial base, consider factoring $N = 15$. The possible periods r for any of the bases a are all powers of 2. This means that any even value of the phase s measured from the period register will give a fraction $\frac{s}{Q}$ proportional to $\frac{1}{r}$ which always allows one to find the period. In fact, by analyzing the state of the quantum registers along the circuit, it is possible to see that no quantum interference happens between the states in the computational register. Therefore, in this case the correct results can be obtained regardless of the quality of the entangling gates of the device, as long as one can entangle the period register with the computational register. To show that the quantum processor *ibmqx5* is giving us the correct answer by exploiting quantum interference it is sufficient to run the experiment for one of the possible bases. This in turn is related to the quality of the entangling gates and the noise of the device. Thus, the ability to factor higher and higher N using a nontrivial base (one which has a period that is not a power of 2) gives a benchmark of the performance of the device.

In the experiment for the $N = 15$ case, five input qubits are required: one qubit initialized to $|0\rangle_p$ for the period register, acting as a control qubit, and all other qubits initialized to the state $|\psi\rangle = |0001\rangle_q$ belonging to the computational register. Alongside the quantum registers, we also need a three-bit classical register to store the results of the measurement of the control qubit, which encodes the value of the period.

The case $N = 15$ is the simplest possible case and it does not provide an example where quantum interference between the states of the computational register brings an advantage to the computation. For this reason, we attempt to factor the second smallest number which is a product of two primes, $N = 21$. In this case, there are bases a for which the period is not a power of 2, thus constructive quantum interference between states in the computational register is needed to increase the likelihood of finding the correct result. An example of such case was first demonstrated in [5].

We implement an algorithm for factoring $N = 21$ with base $a = 2$ using three bits of precision for the estimation of the

phase which encodes the period. In this case the quantum register is composed of five qubits in the computational register and one qubit in the period register. We adopt the same methodology used previously, breaking each stage of the modular exponentiation and manually feeding the output of each section as input to the next. This means that the circuit will have three stages of modular exponentiation, where a single bit of the phase which encodes the period is estimated at each stage (details in the Appendix). Therefore, the circuit looks like the one in Fig. 3. The modular exponentiation circuit is specifically designed to calculate $a^x \bmod 21$, where we choose the base $a = 2$. This base has periods $r = 6$, thus $1/r$ cannot easily be represented in binary. Therefore, the accuracy of the estimation of the period depends on the number of bits used for the phase estimation.

The same method is applied to factor $N = 35$ with base $a = 4$. In this case we need six qubits in the computational register and one qubit in the period register. As in the case of $N = 21$, the period of $4^x \bmod 35$ is $r = 6$, therefore $1/r$ cannot be easily represented in binary. As a result of running the quantum algorithm we obtain a probability distribution for the estimated phase s which is peaked around the multiples of $1/r$. We use a three-bit register for the estimation of the phase which encodes the period. Again, the circuit for running the algorithm is realized as shown in Fig. 2; each stage, estimating one bit of the phase, is implemented separately and then joined through a classical algorithm. The individual circuits which compute the MEF at the different stages can be found in the Appendix.

V. RESULTS AND DATA ANALYSIS

Figures 4(a), 5(a), 6(a), and 7(a) show the results obtained running the quantum part of the factoring algorithm on the *ibmqx5* superconducting device. Depicted are the experimental relative probabilities found (in blue or dark gray) side by side with the expectation values which can be computed theoretically (in yellow or light gray) for each value of the estimated phase s for the bases a used. The algorithm was run 1000 times for each base.

The success of the experiment is evaluated in two different ways. We use probability plots to give a qualitative estimation of the correctness of the results, while the SSO is used as a quantitative measure. Probability plots [11] are a useful tool to visually compare two distributions. In a probability plot, one distribution is plotted against the other. If the two distributions are identical, the plot will show a straight line ($y = x$). The amount of deviation from the straight $y = x$ line is an indication of the difference between the two probability distributions plotted. For the case at hand, this means plotting on the (x, y) plane a point for each value of the phase, where the value of the x coordinate is given by the theoretical value of the probability distribution for that phase and the value of the y coordinate is given by the corresponding experimental value found. The data are then fitted with a straight line for comparison with the ideal $y = x$ case. Error bars on the fit are given as a range of y values compatible with the error on the fit coming from both slope and offset of the fitted line at a fixed x value. Thus, all straight lines contained within these error bars are compatible with the experimental data within

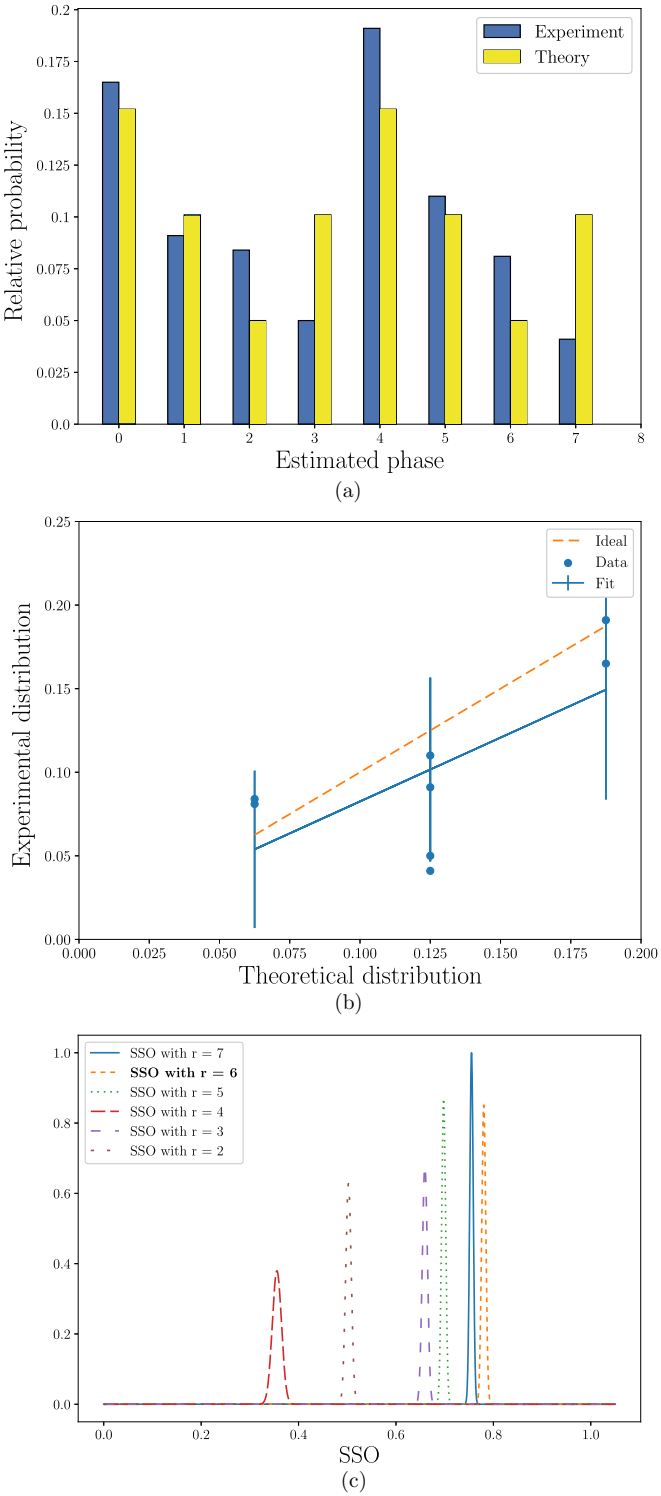
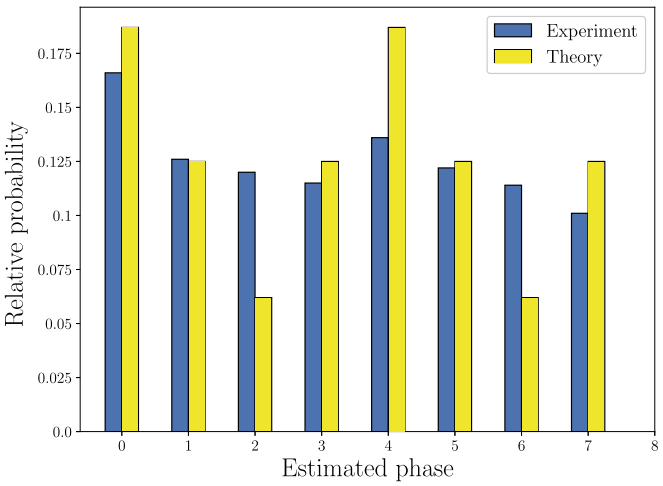
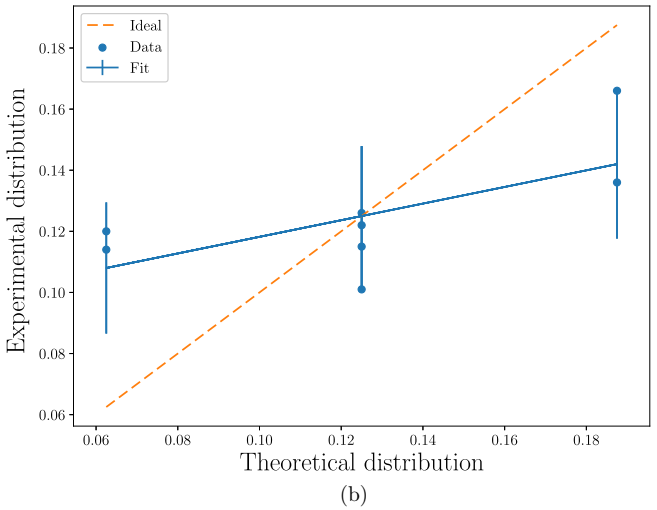


FIG. 6. (a) Probability of finding a given phase for $N = 21$ with base $a = 2$ and (b) probability plot of the theoretical distribution and the experimental distribution. (c) SSO of the experimental data with the theoretical probability distributions corresponding to different periods.

the estimated error for the fit. The probability plots between the experimental distribution and the expected theoretical one for each case are shown in Figs. 4(b), 5(b) 6(b), and 7(b). In the case of $N = 15$, the data in Figs. 4(b) and 5(b) are on a





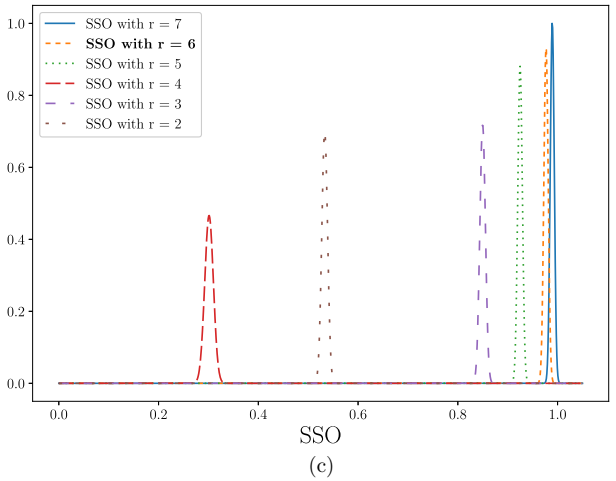


FIG. 7. (a) Probability of finding a given phase for $N = 35$ with base $a = 4$ and (b) probability plot of the theoretical distribution and the experimental distribution. (c) Plot of the SSO between the experimental data and all the possible theoretical distributions for the different values of r.

straight line very close to the $y = x$ line (tagged as "Ideal" on the plots). For the $N = 21$ case, the data lie on a straight line parallel to the $y = x$ ideal line as can be seen from Fig. 6(b). This means that there is an offset in the relative frequency of each phase in our experimental distribution. However, the overall shape coincides with the theoretical one, indicating that the difference in relative frequencies between phases is preserved. Finally, for $N = 35$, the data in Fig. 7(b) lie on a straight line which is very far from the $y = x$ line, indicating an important deviation of the experimental results from the theoretically expected ones. In fact, looking at the histogram in Fig. 7(a) shows that the experimental results were affected by noise, which tends to make all phases equally probable. In summary, for $N = 15$ and 21 the fit is close to the ideal line (within the error bars) but for $N = 35$ it is not. Therefore, we believe that the probability plots provide a good qualitative measure of the similarity between probability distributions, as they correctly describe the similarity which is apparent by the comparison of the histograms of the distributions.

Next we give a quantitative measure of the correctness of the results. In particular, we want to answer the following question: given the experimental data obtained, what is the likelihood that these data come from a given probability distribution? The answer to this question will reveal two aspects of our experiment. First, it will allow us to assign a period to the results without the need for the continued fraction algorithm. Second, it will give us a measure of the error we make in the assignment. Our method of assigning the period to the experimental data relies on the following observation: the probability of obtaining a certain phase s is

$$P(s) = \frac{1}{Q}\left|e^{\frac{2\pi i s x_0}{Q}}\right|^2 \sum_{d\in\mathbb{Z}} \left|e^{\frac{2\pi i s d r}{Q}}\right|^2. \tag{1}$$

As a function of the estimated phase s, the probability distribution $P(s)$ is completely characterized by the values of the parameters r and Q: the period and the number of bits $\log_2 Q$ used to encode the value of the period, respectively. Therefore, there is a fixed probability distribution for each value of r and Q. To determine the period to assign to the experimental data, we compare the probability distribution $P_{\text{exp}}(s)$ obtained experimentally, with all the possible probability distributions $P^r_{\text{th}}(s)$ given by values of r from 2 to $Q - 1$, for fixed number of bits $\log_2 Q$ encoding the period. The period of the theoretical distribution which is most similar to the experimental data is then assigned to the experiment.

Following [3], we use the SSO introduced in [12] as a measure of similarity between probability distributions. The SSO is defined as

$$\text{SSO} = \left(\sum_{j=0}^{7} m_j^{1/2} e_j^{1/2}\right)^2, \tag{2}$$

where m_j and e_j are the measured and expected output-state probabilities of state $|j\rangle$, respectively.

One can calculate the error on the SSO from the Poissonian counting error of the data, assuming Gaussian propagation of errors:

$$\Delta\text{SSO} = \sqrt{\sum_{j=0}^{7} \frac{\partial}{\partial m_j}\left(m_j^{1/2} e_j^{1/2}\right)^2 \Delta m_j^2}. \tag{3}$$

For each base used in the experiments, we calculate the SSO of $P_{\text{exp}}(s)$ with all possible $P^r_{\text{th}}(s)$. To better visualize which $P^r_{\text{th}}(s)$ most resembles the data, we plot unit area normalized Gaussian distributions with the SSO as the mean and ΔSSO as the standard deviation. The Gaussian whose value of the mean is closest to 1 comes from the $P^r_{\text{th}}(s)$ most similar to $P_{\text{exp}}(s)$. Therefore, we assign period r to our data. While the spread of each Gaussian gives an indication of the error in the calculation of the SSO. To quantitatively determine the error in the assignment of the period, we calculate the area of overlap between the Gaussian distribution with the highest SSO and the second closest one. This is done through the overlap coefficient [15] between the normal distributions. The OVL is defined as

$$\text{OVL}[f(x_1), f(x_2)] = \sum_x \min(f(x_1), f(x_2)), \tag{4}$$

where $f(x_1)$ is the normal distribution with the highest SSO and $f(x_2)$ is the normal distribution with second highest SSO. The OVL tells us what is the probability that the assignment is done incorrectly, i.e., the highest SSO for our experimental data comes from a different theoretical probability distribution than the assigned one. Thus, we quantify the error on our assignment as $\epsilon_{ij} \equiv \text{OVL}[f(x_i), f(x_j)]$ where i denotes the period of the distribution with the highest SSO and j denotes the period of the distribution with the second highest SSO.

The results of the comparison for all experiments are presented in Figs. 4(c), 5(c) 6(c), and 7(c). Figures 4(c) and 5(c) show the SSO of the experimental distributions and their deviations obtained for $N = 15$, $a = 2$, and $a = 11$, respectively. For $a = 2$, the highest SSO is 0.97 for the theoretical distribution corresponding to the period $r = 4$. Thus, we assign the period $r = 4$ to the experimental distribution obtained. The error we make in assigning the period $r = 4$ instead of period $r = 7$, which is the closest match, is $\epsilon_{47} = 3.8\times10^{-134}$. For $a = 11$, the highest SSO is 0.92, which corresponds to $r = 2$. The error in the assignment of $r = 2$ with respect to $r = 4$, which has the second highest SSO, is $\epsilon_{24} = 4.1\times10^{-31}$. The results obtained for $N = 21$ with $a = 2$ are shown in Fig. 6(c). Here, it is more difficult to determine the period with certainty. The highest SSO is 0.78, which corresponds to the theoretical distribution with $r = 6$. The error in assigning $r = 6$ to the experimental data is $\epsilon_{67} = 1.2\times10^{-3}$. Therefore, there is a $\sim$0.1% chance that we assigned the period incorrectly and the true period was $r = 7$ instead. For the case of $N = 35$ and $a = 4$, the results presented in Fig. 7(c) show that the highest SSO between the experimental data and the theoretical distribution corresponding to all possible periods is 0.99 for $r = 7$, although this is not the expected period. There is another close match with an SSO of 0.98 for $r = 6$, which is the correct one. The error in assigning period $r = 7$ to the experimental data instead of $r = 6$ is $\epsilon_{76} = 0.14$. Thus, in this case it is quite difficult to discern the correct period.

VI. CONCLUSIONS

Although the results are obtained with a compiled and simplified version of Shor's factoring algorithm, our purpose is to show a way to proceed with the implementation of generic algorithms on the approximate quantum computers available now. In practice, the non-negligible noise and the lack of key functions of the device force us to rethink how to design algorithms that can work on these machines. As it is evident from this paper, one needs to supplement the deficiencies of the hardware with a more detailed theoretical analysis and classical processing. By doing so, one can reduce the length of the circuit needed to implement the algorithm, mitigating the effects of noise and overcoming the lack of particular functions assumed for a general-purpose quantum computer. We emphasize that the simplification by inspection done here was possible only due to the small size of the circuit. Larger circuits would require a more sophisticated optimization. We used different methods to evaluate the success of the experiment. The first one is the probability plot, which gives a qualitative measure of the similarity between the distribution of the experimental data and the expected theoretical distributions. The second one is the SSO, which gives a quantitative measure of the similarity between probability distributions. By using the SSO, we introduced an alternative way to assign a certain period to the probability distribution obtained from the experimental data. In this way, we avoid using the continued fraction algorithm, which fails when the number of bits used to encode the value of the period is particularly low, as in our situation. To correctly quantify the error which can be made in this assignment, the OVL between different candidates for the period is calculated. Overall, the experimental results obtained from running the algorithm on the *ibmqx5* device are in agreement with the theoretical expectation values. Excellent agreement is found for $N = 15$, while deviations from the theoretical results become more noticeable for $N = 21$. Eventually, the algorithm fails to factor $N = 35$. This is due to the cumulative errors coming from the increasing number of two-qubit gates necessary to implement the more complex MEF needed for this case.

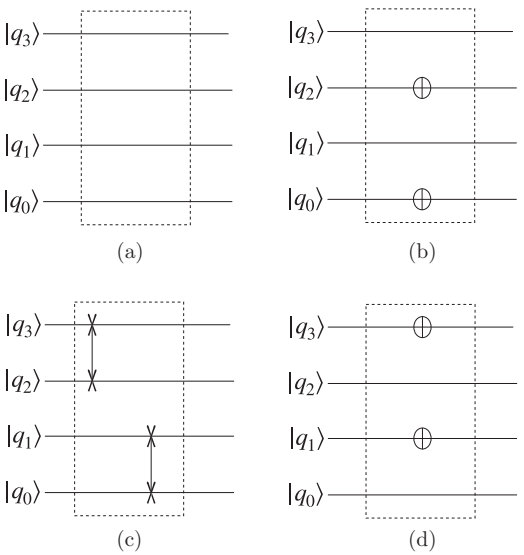


FIG. 8. Modular exponentiation circuits for $N = 15$. (a) $a^4 \bmod 15$ for any a and $a^2 \bmod 15$ for $a = 11$, (b) $a^2 \bmod 15$ for $a = \{2, 7, 8, 13\}$, (c) $2^1 \bmod 15$ for $a = 2$, and (d) $11^1 \bmod 15$ for $a = 11$.

ACKNOWLEDGMENTS

We acknowledge use of the IBM Q for this work. The authors are grateful to N. T. Bronn and R. Ya. Kezerashvili for valuable and stimulating discussions.

APPENDIX: CIRCUITS FOR THE MEF

Here we present the procedure used to implement the MEF in the experiments for factoring $N = 15,\ 21$, and 35. These were specifically designed to reduce the number of gates to the minimum and mitigate the effects of noise. To make the approach scalable, one would need an automatic way to generate the modular exponentiation circuits as proposed in [3].

The circuits used for the MEF in the experiment for factoring $N = 15$ are shown in Fig. 8. The MEF in the first circuit of Fig. 3 shown in Fig. 8(a) is the identity operation for any base a, making it a deterministic step. The output of the first circuit is then fed into the second one. As shown in [3], the MEF here reduces to a very simple circuit depending on the base a selected for factoring. If the base a is any one of the elements of the set {4, 11, 14}, the modular exponentiation function is again the identity shown in Fig. 8(a) and this step turns again into a deterministic step. If the base is one of the elements of the set {2, 7, 8, 13}, the MEF has the same simple circuit for any of these a, which can be seen from Fig. 8(b). The MEFs for the two bases $a = 2$ and 11 for the third circuit are given in Figs. 8(c) and 8(d), respectively.

The circuits of the MEF used in the experiment of factoring $N = 21$ are presented in Fig. 9. The experiment was conducted only with the base $a = 2$, therefore all circuits have been designed only for this base. The MEF for the first circuit is shown in Fig. 9(a). For the second circuit, the MEF in Fig. 9(b) was used. In the third circuit, depending on the values of the bits of the period register measured in the previous stages, different states are prepared as input. For this reason, different modular exponentiation circuits are designed according to the results of the measurements of the period register. The various possibilities are shown in Figs. 9(c), 9(d) 9(e), and 9(f) corresponding to the four possible outcomes 00, 01, 10, and 11, respectively.

The MEFs implemented in the experiment of factoring $N = 35$ are depicted in Fig. 10. The circuits are designed for the algorithm with base $a = 4$. The MEFs for first, second, and third circuits are shown in Figs. 10(a)–10(c), respectively. In this case, one circuit which works for any input was designed for the MEF at each stage.

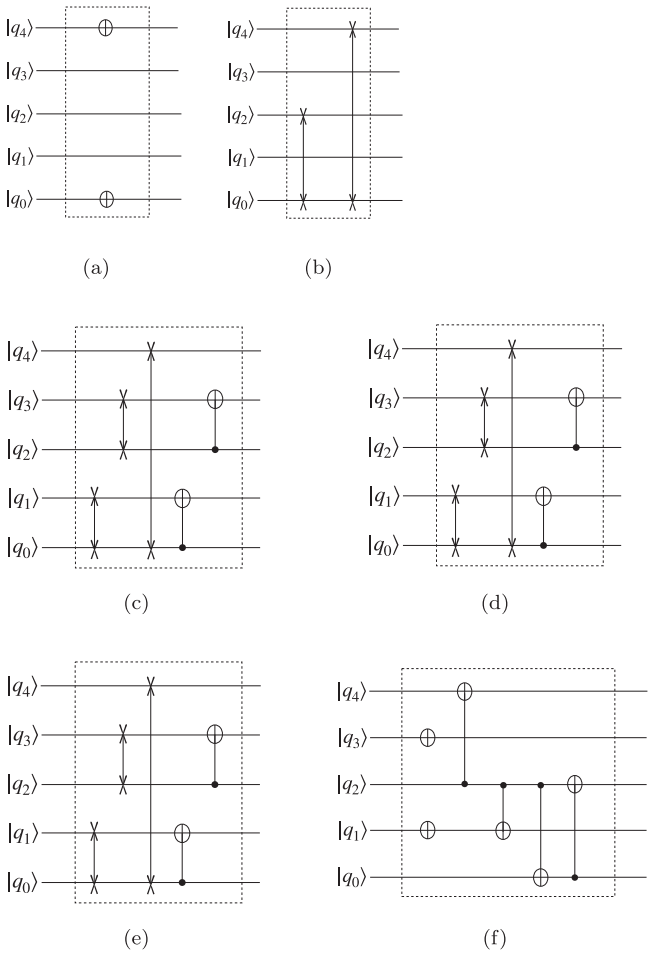


FIG. 9. Modular exponentiation circuits for $N = 21$ with base $a = 2$. (a) $2^4 \bmod 21$. (b) $2^2 \bmod 21$. Depending on the results of the measurement of the period register in the previous circuits we have (c) $2^1 \bmod 21$ for $\text{bit}^{(0)} = 0$ and $\text{bit}^{(1)} = 0$, (d) $2^1 \bmod 21$ for $\text{bit}^{(0)} = 1$ and $\text{bit}^{(1)} = 0$, (e) $2^1 \bmod 21$ for $\text{bit}^{(0)} = 0$ and $\text{bit}^{(1)} = 1$, and (f) $2^1 \bmod 21$ for $\text{bit}^{(0)} = 1$ and $\text{bit}^{(1)} = 1$.

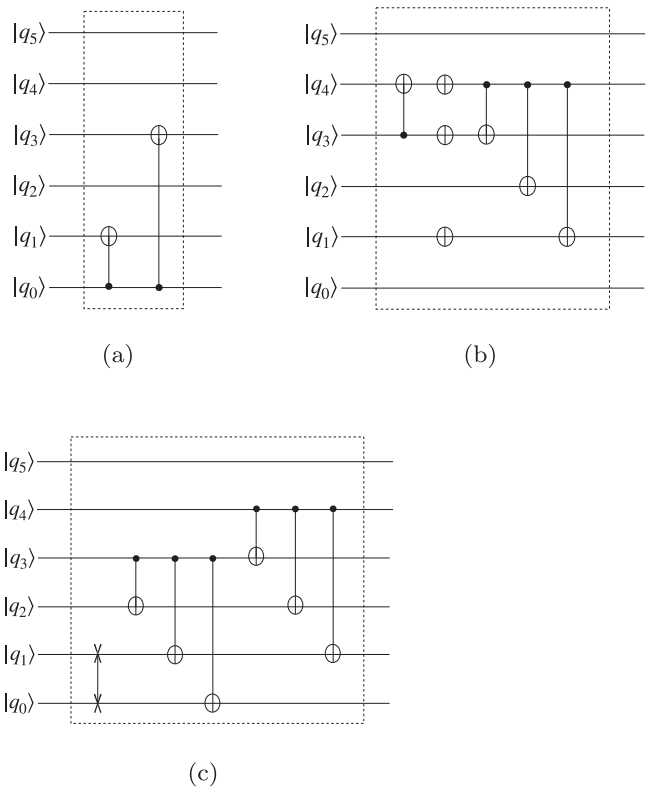


FIG. 10. Modular exponentiation circuits for $N = 35$. (a) $4^4 \bmod 35$. (b) $4^2 \bmod 35$. (c) $4^1 \bmod 35$.

[1] P. Shor, in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE, New York, 1994), pp. 124–134.
[2] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, Nature (London) **414**, 883 (2001).
[3] T. Monz, D. Nigg, E. A. Martinez, M. F. Brandl, P. Schindler, R. Rines, S. X. Wang, I. L. Chuang, and R. Blatt, Science **351**, 1068 (2016).
[4] C. Y. Lu, D. E. Browne, T. Yang, and J. W. Pan, Phys. Rev. Lett. **99**, 250504 (2007).
[5] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien, Nat. Photon. **6**, 773 (2012).
[6] B. P. Lanyon, T. J. Weinhold, N. K. Langford, M. Barbieri, D. F. V. James, A. Gilchrist, and A. G. White, Phys. Rev. Lett. **99**, 250505 (2007).
[7] A. Politi, J. C. F. Matthews, and J. L. O' Brien, Science **325**, 1221 (2009).
[8] E. Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland, and John M. Martinis, Nat. Phys. **8**, 719 (2012).
[9] P. J. Coles *et al.*, arXiv:1804.03719 (2018).
[10] J. A. Smolin, G. Smith, and A. Vargo, Nature (London) **499**, 163 (2013).
[11] B. Kleiner, J. M. Chambers, W. S. Cleveland, and P. A. Tukey, *Graphical Methods for Data Analysis* (Duxbury Press, Boston, MA, 1983).
[12] J. Chiaverini, J. Britton, D. Leibfried, E. Knill, M. D. Barrett, R. B. Blakestad, W. M. Itano, J. D. Jost, C. Langer, R. Ozeri *et al.*, Science **308**, 997 (2005).
[13] A. Y. Kitaev, arXiv:quant-ph/9511026 (1995).
[14] R. B. Griffiths and C. S. Niu, Phys. Rev. Lett. **76**, 3228 (1996).
[15] H. F. Inman and E. L. Bradley, Jr., Commun. Stat.: Theory Methods **18**, 3851 (1989).