

Variational quantum eigensolver approach to prime factorization on IBM's noisy intermediate scale quantum computer

Mona Sobhani ^{1,2} Yahui Chai ^{1,*} Tobias Hartung ³ and Karl Jansen ^{1,4}

¹*Deutsches Elektronen-Synchrotron DESY, Platanenallee 6, 15738 Zeuthen, Germany*

²*Brandenburg University of Technology, Platz der Deutschen Einheit 1, 03046 Cottbus, Germany*

³*Northeastern University - London, Devon House, St Katharine Docks, London E1W 1LP, United Kingdom*

⁴*Computation-Based Science and Technology Research Center, The Cyprus Institute, 20 Kavafi Street, 2121 Nicosia, Cyprus*



(Received 18 October 2024; accepted 3 March 2025; published 8 April 2025)

This paper presents a hybrid quantum-classical approach to prime factorization. The proposed algorithm is based on the variational quantum eigensolver (VQE), which employs a classical optimizer to find the ground state of a given Hamiltonian. A numerical study is presented, evaluating the performance of the proposed method across various instances on both IBM's real quantum computer and its classical simulator. The results demonstrate that the method is capable of successfully factorizing numbers up to 253 on a real quantum computer and up to 1 048 561 on a classical simulator. These findings show the potential of the approach for practical applications on near-term quantum computers.

DOI: [10.1103/PhysRevA.111.042413](https://doi.org/10.1103/PhysRevA.111.042413)

I. INTRODUCTION

The factorization of large integers is a computationally challenging task for classical computers. In 1994, Shor demonstrated that quantum computers have the potential for exponential speedup in solving the factorization problem [1]. However, due to current quantum hardware constraints, the largest number factored today using Shor's algorithm on a real quantum computer is 21 [2].

While the large-scale implementation of Shor's algorithm in quantum computing remains a future challenge, near-term alternatives that harness the power of quantum computing to tackle optimization problems are being investigated. Through IBM's cloud-based quantum computer, noise intermediate scale quantum (NISQ) [3] devices have enabled algorithm testing. It is important to note that these devices are not error free and are vulnerable to hardware and mechanical quantum noise.

Variational quantum algorithms (VQAs) are particularly well suited for NISQ devices. VQAs operate by establishing a parameterized quantum circuit, where the parameters are optimized using a classical computer to minimize a cost function that is decided by the state prepared by the quantum circuit and the target problem. This hybrid quantum-classical approach allows VQAs to effectively tackle problems that are difficult for purely classical or quantum methods. By maintaining relatively small quantum circuits and utilizing classical resources for optimization, VQAs can lead to

significant results in finding the optimal solutions to realistic problems [4–6]. In this paper, we focus on the variational quantum eigensolver (VQE) [7], a specific type of VQA. The VQE is primarily designed to find the ground-state energy of a quantum system. We propose to encode the prime factorization problem, mapped to an optimization problem as per Burges [8], into a Hamiltonian. Our goal is to find the ground state, which corresponds to the solution of identifying the correct prime factors p and q for a given integer n , such that $n = p \times q$.

Investigating the potential of solving the prime factorization problem using a VQE contributes to our understanding of the practical applications of quantum computing and investigates the current limits of near-term quantum devices. The insights gained could lead to more efficient and effective solutions to complex computational problems in the future using a hybrid quantum-classical algorithm.

In this work, we aim to benchmark the performance of VQE combined with the conditional value at risk (CVaR) [9,10] of multiple different cost functions with respect to solving the prime factorization problem. However, in order to avoid any benchmarking effects of instance preprocessing, our benchmarks are performed without relying on any prior arithmetic simplifications. While this is not optimal for finding the largest possible values of n that can be successfully factorized, we believe that it offers the fairest comparison of different quantum circuits and cost functions to be used in the CVaR-VQE. Nonetheless, it is important to note that we have successfully factorized 1 048 561 using 27 qubits in an ideal simulation and the number 253 with nine qubits on a real quantum device.

The paper is structured as follows. Section II provides a review of the relevant literature and presents the current factorization records using a hybrid quantum-classical approach. In Sec. III, we introduce our proposed method and discuss the theoretical background. Section V details the

*Contact author: yahui.chai@desy.de

experimental setup, and Sec. VI explores the factors that impact the algorithms' performance. Finally, we conclude our findings in Sec. VII.

II. RELATED WORK

This section presents the results of prime factorization records achieved through the use of hybrid quantum-classical algorithms. In 2001, Burges [8] demonstrated that factoring a number n into its prime factors p and q can be mapped to an optimization problem as follows:

$$f(p, q) = (n - pq)^2. \quad (1)$$

The minimum value of this function is achieved when the correct primes p and q are used. Various approaches have been proposed to find this minimum by using a Hamiltonian to find the ground state. Most methods use arithmetic simplifications of equations resulting from the binary multiplication table when multiplying p and q as binary numbers. For a detailed explanation of those approaches, we refer to Ref. [11].

The adiabatic quantum computing model [12], based on the adiabatic theorem [13], has been used to factorize numbers using a gradually changing Hamiltonian. In 2008, the number 21 was factorized using three qubits on a nuclear magnetic resonance (NMR) machine [14], the record for factorization using this method is held by the number 291 311, requiring three qubits [15]. Furthermore, the number 35 was factorized using the adiabatic quantum computing model on a five-qubit IBM superconducting system [16].

D-wave systems utilize the annealing method [17] to solve the prime factorization problem, achieving a record by factorizing 376 289 with 94 qubits [18]. A comparison of quadratic unconstrained binary optimization (QUBO) and higher-order unconstrained binary optimization (HUBO) models led to the factorization of 102 454 763 and 1 000 070 001 221 using 26 and 12 qubits, respectively, on a quantum simulator [19].

The variational quantum factoring (VQF) algorithm, introduced in 2018 [20], uses the quantum approximate optimization algorithm (QAOA) [21] to find the ground state of the corresponding Hamiltonian. In 2021, this method was used to factorize the numbers 1 099 551 473 989, 3127, and 6557 using three, four, and five qubits, respectively [22]. The following year, a comparison of QAOA with the VQE led to the successful factorization of the numbers 25, 49, 91, and 247 using six, eight, ten, and 13 qubits, respectively, with VQE showing performance advantages [23].

Other prime factorization approaches include the quantum imaginary time evolution (QITE) method [24,25] to factorize numbers 1829 with nine qubits using a real quantum computer [26], Grover's algorithm to factorize 1 269 636 549 803 using only three qubits [27]. Additionally, several techniques combined allowed for the factorization of numbers using three, five, and ten qubits [28], and the digitized counteradiabatic quantum computing (DCQC) method was used to factorize 261 980 999 226 229 using ten qubits [29].

III. METHOD

The variational quantum eigensolver (VQE) is a promising hybrid quantum-classical algorithm for NISQ devices, as outlined in Ref. [7]. This algorithm utilizes a parametrized quantum circuit, which generates the quantum state $|\psi(\theta)\rangle$, where θ denotes the set of tunable parameters. These parameters are optimized through a classical optimizer to minimize the energy expectation value, expressed as $\langle\psi(\theta)|H|\psi(\theta)\rangle$. This enables the VQE to search for the ground state of a Hamiltonian, which has the minimal energy value and encodes the optimal solution to our problem.

Since the factoring problem is not directly associated with a Hamiltonian, there are many approaches one can pursue. Fundamentally, many possible choices come from the construction of a suitable Hamiltonian H and the choice of parametrized quantum circuit that generates the state $|\psi(\theta)\rangle$. We will discuss the impact of the quantum circuit in more detail in Sec. VI. The focus of this current section will be on the Hamiltonian and associated cost function.

The only fundamental constraint on the cost function and Hamiltonian comes from the fact that VQE attempts to minimize the value of the cost function. Hence, any Hamiltonian or cost function that encodes the solution to the factoring problem as a global minimum is viable. This freedom can be exploited through various means of preprocessing any given instance of the factoring problem in order to construct an optimized Hamiltonian that increases the chances of a successful factorization.

In contrast to many of the approaches presented in the previous section, we consider a generic approach that does not include simplifying equations arithmetically prior to the construction of the Hamiltonian. Instead, the cost function, as defined in Eq. (1), is encoded directly into a Hamiltonian in order to prevent any effects arising from the preprocessing of instances. A direct VQE application then minimizes the energy expectation value to resolve the ground state of the constructed Hamiltonian. Once the VQE has identified the ground state, it can be concluded that the solution to the factoring problem has been found, with the values of p and q for $n = pq$ having been determined.

This process is shown in Fig. 1. In Fig. 1, the cost function evaluation step refers to measurements of the energy $\langle\psi(\theta)|H|\psi(\theta)\rangle$ in the case of a direct VQE implementation. However, for combinatorial optimization problems such as the factoring problem, it is not necessarily advantageous to use the expected energy as a cost function. Since each individual shot result is a viable candidate for the ground state, the primary objective changes slightly from minimizing the energy to maximizing the probability of measuring the solution. While the two concepts are of course related, the latter offers more options. These include measuring functions of the Hamiltonian, such as $\lceil\log_2(\sqrt{H} + 1)\rceil$ or $-\frac{1}{\sqrt{H}+\epsilon}$, as well as using the conditional value at risk (CVaR), which only considers the relatively low-energy shots for the cost function.

Below, we will explain the various steps shown in Fig. 1. We begin by describing how to construct the Hamiltonian H from Eq. (1), the quantum circuit we employ, and the additional procedures used to evaluate and optimize the cost function.

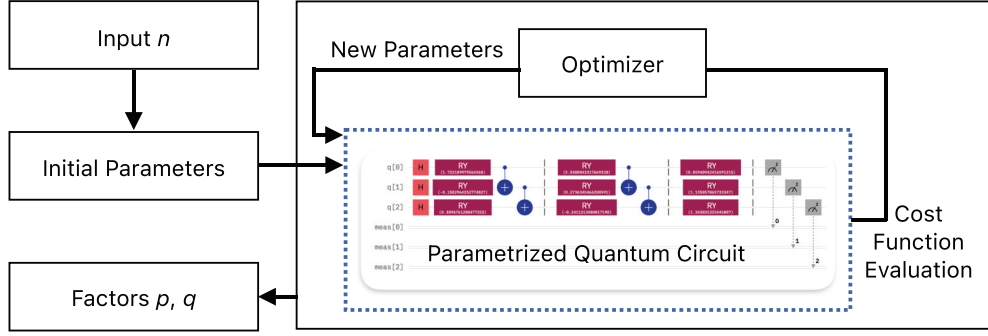


FIG. 1. Steps of the VQE for prime factorization.

Step 1: Construct Hamiltonian. We use the cost function defined in Eq. (1) where n is a B -bit binary number

$$n = \sum_{i=0}^{B-1} 2^i n_i. \quad (2)$$

We assume that n is odd as otherwise the factorization problem is trivial. Similarly, the prime factors p and q can be represented by N_p -bit and N_q -bit binary numbers

$$p = \sum_{j=0}^{N_p-1} p_j 2^j \quad \text{and} \quad q = \sum_{k=0}^{N_q-1} q_k 2^k. \quad (3)$$

The bit lengths of the integers p and q cannot be determined prior to the computation without further assumptions; however, the number of bits needed to represent p and q can be optimized. Assuming without loss of generality that $p \leq q$, it follows that $p \leq \sqrt{n}$. Consequently, the number of bits required to represent p is $N_p = \lceil \log_2(\sqrt{n}) \rceil$. Additionally, if n is not even, q must be less than $\frac{n}{2}$, thereby reducing the number of bits for q to $N_q = B - 1$. Since n is assumed to be an odd number, p and q are odd primes, which implies that the least significant bits, denoted as p_0 and q_0 , are always 1. These do not require encoding in qubits, leading to a total qubit requirement of $N = N_p + N_q - 2$.

Using this notation, the cost function can be written as:

$$(n - pq)^2 = n^2 - 2n \sum_{j=0}^{N_p-1} \sum_{k=0}^{N_q-1} p_j q_k 2^{j+k} + \sum_{j=0}^{N_p-1} \sum_{k=0}^{N_q-1} \sum_{l=0}^{N_p-1} \sum_{m=0}^{N_q-1} p_j q_k p_l q_m 2^{j+k+l+m}. \quad (4)$$

Now, we encode the cost function into a Hamiltonian by substituting the binary variables p_j, q_k (except p_0 and q_0) as defined in Eq. (3) with the Pauli operator $\frac{I-Z}{2}$:

$$p_j \rightarrow \frac{I - Z_{p,j}}{2}, \quad \text{with } j = 1, 2 \dots N_p, \\ q_k \rightarrow \frac{I - Z_{q,k}}{2}, \quad \text{with } k = 1, 2 \dots N_q, \quad (5)$$

where Z is the Pauli- Z operator. Thus we get the operator corresponding to factors p, q , and Hamiltonian

$$P = 1 + \sum_{j=1}^{N_p-1} 2^{j-1} (I - Z_{p,j}), \\ Q = 1 + \sum_{k=1}^{N_q-1} 2^{k-1} (I - Z_{q,k}), \\ H = (nI - PQ)^2. \quad (6)$$

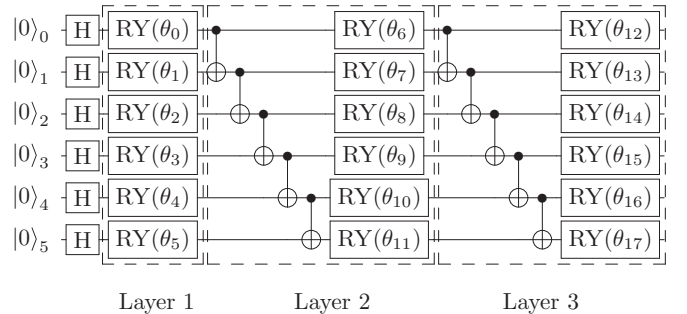
We will use this Hamiltonian in our cost function evaluation part, explained in step 3.

Step 2: Quantum circuit. We use a hardware-efficient Ansatz with linear-controlled-NOT (CNOT) entanglement layer for our method. For N qubits and L layers, we have $(N - 1) \cdot (L - 1)$ CNOT gates and $(N \cdot L)$ rotation (RY) gates. The initial parameters of each RY gate are randomly chosen in the range of $(-\pi, \pi)$. An example circuit is shown in Fig. 2.

Step 3: Cost function evaluation. For the cost function evaluation, we use the expectation value of the Hamiltonian $E = \langle \psi(\theta) | H | \psi(\theta) \rangle$ to determine how close the quantum state is to the solution. In more detail, we evaluate each quantum state $|p, q\rangle$ in the measurement output as follows:

$$H |p, q\rangle = \lambda_{p,q} |p, q\rangle = (n - pq)^2 |p, q\rangle. \quad (7)$$

Here, the quantum state is defined as $|p, q\rangle = |p_{N_p-1} \dots p_1 q_{N_q-1} \dots q_1\rangle$. For example, $|p, q\rangle = |010001\rangle$ with $N_q = 4, N_p = 4$ means $p = 0101$ and $q = 0011$, where $p_0 = q_0 = 1$ is added as the rightmost bits of p and q . Thus the bit string $|010001\rangle$ corresponds to factors $p = 5, q = 3$, and is the ground state of the corresponding Hamiltonian if $n = 15$.


 FIG. 2. Linear-CNOT Ansatz for $L = 3$.

The eigenvalue $\lambda_{p,q}$ of the state $|p, q\rangle$ is equivalent to our cost function value $(n - pq)^2$, with corresponding n , p , and q . This approach can be directly extended to expressions $f(\sqrt{H})$ of the Hamiltonian provided that $f(|n - pq|)$ can be efficiently evaluated. This will be the focus of Sec. VI C. Prior to Sec. VI C, we will only consider energy values $e_k = (n - pq)^2$ as obtained from single shots measuring H .

Instead of using the energy $E = \langle \psi(\theta) | H | \psi(\theta) \rangle$ directly, the explicit cost function we evaluate is the conditional value at risk (CVaR) [9,10]. The procedure involves using S shots to obtain S measurement outcomes, resulting eigenvalues denoted as $E_S = \{e_1, e_2, \dots, e_S\}$. These eigenvalues are organized in ascending order and we consider the expectation of the lowest α percentile of the sorted list E_S as the cost function evaluation

$$\text{CVaR}(\alpha, S) = \frac{1}{\lceil \alpha S \rceil} \sum_{k=1}^{\lceil \alpha S \rceil} e_k. \quad (8)$$

Thus, for $\alpha = 1$, we recover the energy $E = \langle \psi(\theta) | H | \psi(\theta) \rangle$, and in the limit $\lceil \alpha S \rceil = 1$ we only use the shot with the lowest eigenvalue. The explicit choice of α therefore determines how much of the low-energy spectrum is used for the classical optimization. Intuitively, a smaller value of α is better since we want to maximize the chance of finding the solution to the factoring problem at least once while the high-energy contributions matter less. For example, if we wish to factorize 9, the state $|00\rangle$ corresponds to $p = q = 1$, $|01\rangle$ corresponds to $p = 1$ and $q = 3$, $|10\rangle$ corresponds to $p = 3$ and $q = 1$, and $|11\rangle$ corresponds to the solution $p = q = 3$. We can consider two states of the quantum device, $|\psi_1\rangle = \sqrt{0.9}|00\rangle + \sqrt{0.1}|11\rangle$ which has energy $E = 57.6$ and $|\psi_2\rangle = \sqrt{0.5}|01\rangle + \sqrt{0.5}|10\rangle$, which has energy $E = 36$. Classical VQE would consider $|\psi_2\rangle$ better, while CVaR with $\alpha = 0.1$ would prefer $|\psi_1\rangle$ with a score of 0.0 over $|\psi_2\rangle$ with a score of 36. For the factoring problem, $|\psi_1\rangle$ is the preferred state as it has a 90% chance of observing the solution while $|\psi_2\rangle$ has a 0% chance. This highlights how CVaR inherently improves the probability of observing low-energy states over simply optimizing the average energy. Since we only need to observe the correct factors p and q once, CVaR is the more natural choice of cost function. Choosing α as small as possible therefore seems to make sense, however, it should be noted that the cost function landscape becomes discontinuous in the $\lceil \alpha S \rceil = 1$ limit, which hinders the classical optimization. Furthermore, the standard error of the estimated CVaR is unbounded in the $\alpha \rightarrow 0$ limit for any given number of shots S . Thus, a compromise value of α needs to be chosen.

Step 4: Classical optimization. The CVaR value will then be used by the classical optimizer COBYLA [30] to optimize the parameters θ for the RY gates in the quantum circuit. We refer to Ref. [31] for an in-depth explanation of the COBYLA optimizer.

The COBYLA optimizer proves to be effective for solving optimization problems [32,33]. It is particularly useful when dealing with cost functions that are computationally expensive to evaluate. The classical optimizer optimizes the parameters θ until it has reached convergence or until it has reached the maximum number of iterations $50 \cdot N \cdot L$ set for the algorithm.

TABLE I. The largest numbers n for N qubits. p and q are the prime factors of n , and N is the number of qubits required to find p and q .

p	q	n	N
3	5	15	3
3	7	21	5
3	19	57	6
3	41	123	8
11	23	253	9
7	73	511	11
3	337	1011	12
23	89	2047	14
61	67	4087	15
19	431	8189	17
11	1489	16379	18
137	239	32743	20
109	601	65509	21
53	2473	131069	23
349	751	262099	24
269	1949	524281	26
911	1151	1048561	27

IV. RESULT OF IDEAL SIMULATION

This section shows the result simulated by IBM's classical simulator for quantum computers, the Aer simulator, which utilizes a state vector representation for the quantum state within the QISKIT framework [34]. Table I displays the values of the integer n examined in this study. Each n represents the largest number within a set of integers requiring the same number of qubits N .

In our analysis of the algorithm, we focus on two key performance metrics: the success rate and the runtime. We define the success rate as the fraction of instances that have a fidelity, i.e., probability of the optimal solution, larger than the predefined fidelity threshold t during the VQE process. Specifically, we monitor the fidelity during the VQE iteration. If the maximal fidelity throughout the VQE iterations is higher than the threshold t , we say that this VQE run is successful. For an instance achieving a fidelity threshold t , the probability of getting the solution at least once on a real quantum computer for S shots is $P = 1 - (1 - t)^S$. For example, when using 1000 shots and $t = 0.01$, the probability of finding the solution at least once is $P = 1 - (1 - 0.01)^{1000} = 99\%$.

Figure 3 depicts the success rate for varying numbers of qubits, layers L , CVaR percentages α , and fidelity threshold t set equal to α . For each instance, the success rate is evaluated from 100 runs with random initial parameters. The result indicates $\alpha = 0.01$ achieves a higher success rate than $\alpha = 0.1$ in the ideal simulation. Additionally, increasing the number of layers enhances the performance for both CVaR values. Notably, the success rate almost remains 100% across all problem sizes in the plot when $\alpha = 0.01$ and layer $L = 4$.

The solution for the highest number $n = 1\,048\,561$ factorized was obtained only with $t = 0.01$, $L = 3$, and $\alpha = 0.01$. Here, we ran 20 different initial parameters, but due to the computationally difficult calculations for 2^{27} numbers, we stopped the calculations after we found the solution.

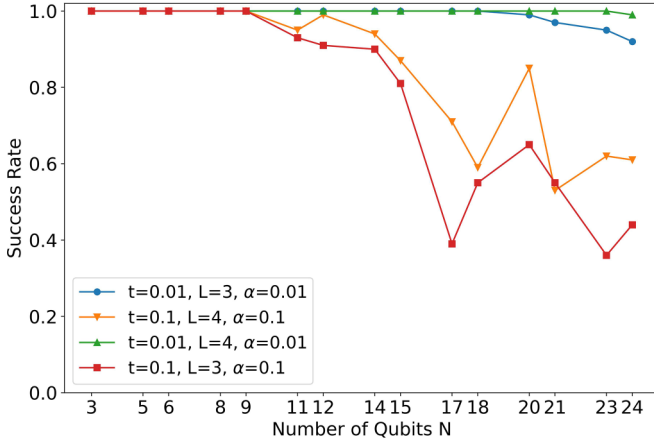


FIG. 3. Success rate for largest numbers n requiring N qubits as outlined in Table I.

To assess the runtime of the VQE, we analyze the number of iterations required for convergence to the solution, which correlates with the number of function evaluations used by the COBYLA optimizer. We define the first iteration that is higher than the fidelity threshold as the minimal number of iterations required and compute the average of this metric across all successful runs. The scaling of these minimal necessary iterations is displayed in Fig. 4. Notice that the CVaR coefficient $\alpha = 0.01$ not only results in a higher success rate but also achieves faster convergence.

V. REAL HARDWARE RESULTS

For the results obtained from the real quantum computer, a successful factorization is defined as the occurrence of the correct solution bit string containing p and q in the measurement output. The integers 15, 21, 57, 123, and 253 were successfully factorized. Classical simulation with the noise model was employed using 1000 shots to identify the optimal initial parameters out of 350 distinct random initial parameters for the hardware run. Figure 5 shows the cost function during the VQE process of number 253 from the hardware run, which converges to the optimal solution around 85 iterations.

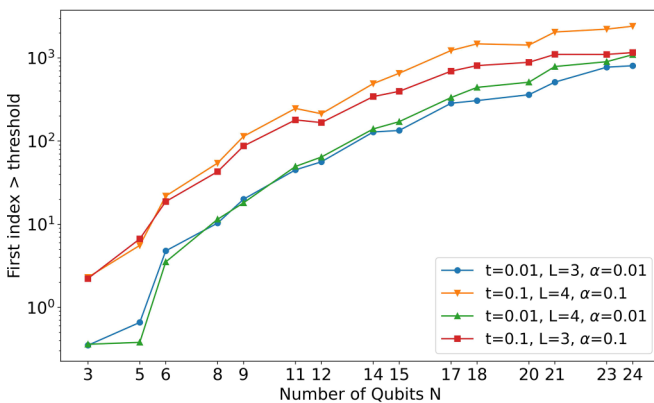


FIG. 4. Scaling of minimal required iterations for different numbers n requiring N qubits.

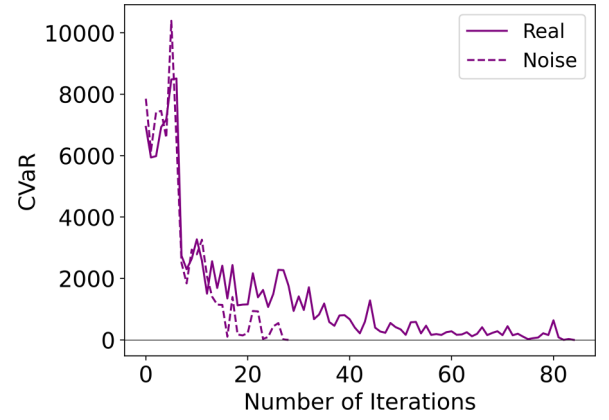


FIG. 5. Hardware run for $n = 253$, three layers, $\alpha = 0.25$, 1000 shots requiring nine qubits. The solid line indicates the hardware run result, and the dashed line represents the result from the noisy simulation with the same initial parameters.

The state vector simulation corresponding to infinite measurement shots in the last section indicates that the smaller alpha value $\alpha = 0.01$ has a better performance. While considering the practical hardware run, only finite shots can be executed. As outlined in Ref. [10], the standard error of the estimated $\text{CVaR}(\alpha, S)$ using S shots scale as $O[1/(S\alpha)]$, implying that smaller α may introduce a larger uncertainty in estimating the cost function CVaR, which could mislead the classical optimizer during parameter updates. Therefore, it is crucial to strike a balance between the ideal performance and the statistical error when choosing the α value for the case of finite shots. In our hardware, we start from $\alpha = 0.5$ and decrease it to $\alpha = 0.25$ if $\alpha = 0.5$ fails to find the solution. Table II shows the first iteration index to converge to the minimal cost function value 0. Note that for larger numbers $n = 123, 253$, a smaller α value is used because $\alpha = 0.5$ was found too large to lead to convergence. This highlights the importance of selecting an appropriate α value in hardware run, prompting us to explore the impact of the α value in VQE more systematically in Sec. VI B.

VI. EXPLORING FACTORS AFFECTING VQE'S PERFORMANCE

A. Difference between p and q

In this section, we investigate whether the difference between p and q has an impact on the success rate. The motivation behind this question comes from classical

TABLE II. VQE iterations required to converge to the solution for a real hardware run.

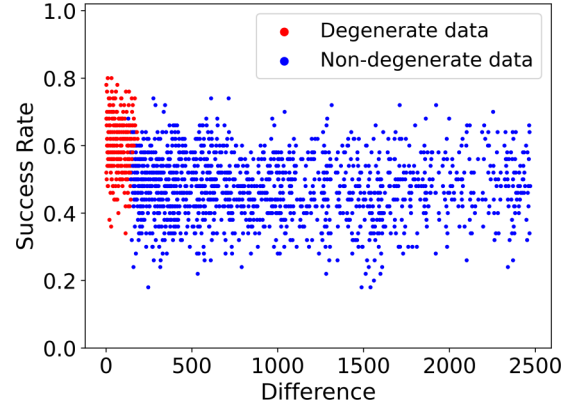
Number n	N qubits	Req. iteration	α
15	3	4	0.5
21	5	9	0.5
57	6	22	0.5
123	8	76	0.25
253	9	85	0.25

application cases. For example, in RSA applications, it is common to choose p and q to have the same bit-length. Thus, seeing a pattern in this range would be interesting. Similarly, if $p \approx q \approx \sqrt{n}$ or if one of the prime factors is very small, then brute-forcing a solution is very easy. Hence, it is interesting to see whether classically easy cases are easier for the quantum algorithm.

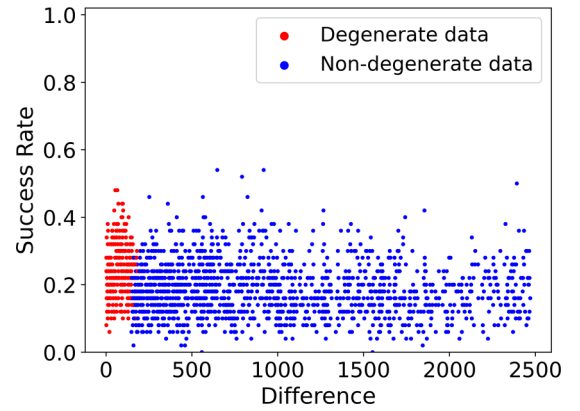
We conducted a comprehensive performance analysis for all numbers n that require 20 qubits. For each of these numbers, we executed 50 random initial parameters and calculated the average success rate. It is important to note that we maintained consistency by using the same 50 initial parameters for each number n across four different *Ansätze*, as depicted in Appendix A in Fig. 11. It is important to emphasize that we choose the CVaR coefficient $\alpha = 0.1$ here rather than the $\alpha = 0.01$. The choice of $\alpha = 0.01$ is able to achieve a success rate around 100% for 20 qubits as shown in Fig. 3, it would obscure the variance in success rates, which is the focus of this study. Hence, $\alpha = 0.1$ was chosen to provide a more informative analysis by allowing us to observe variations in success rates more clearly, particularly with the difference between p and q .

The results for two distinct *Ansätze* are presented in Fig. 6(a) for linear-CNOT and in Fig. 6(b) for parallel-CNOT. The details of the circuit and results for two more *Ansätze* circular-CNOT and parallel-controlled-Z (parallel-CZ) can be found in the Appendix A. The term degenerate data refers to the presence of two degenerate ground states in the associated Hamiltonian. This degeneracy typically occurs when the differences between the two prime factors are small, because the larger factor q can also be accommodated by N_p qubits in cases where $q \approx p \approx \sqrt{n}$. As a result, the success rate for the degenerate data is higher due to the doubled probability of identifying a solution, given the two potential bit strings for the solution, either $|p, q\rangle$ or $|q, p\rangle$. Besides, Fig. 6 shows linear-CNOT has a much higher success rate than the parallel-CNOT across all differences values, which indicates that the selection of the *Ansätze* plays a critical role in the performance of the VQE in prime factorization problem. It is currently unknown what the primary factors influencing *Ansatz* effectiveness are and thus how to find the optimal *Ansätze* for the prime factorization is still an open question, which needs to be further investigated.

To ascertain whether larger or smaller differences between p and q influence the success rate, we employed the linear least-squares regression, as described in Ref. [35]. Our findings indicate that, for all *Ansätze*, there is a slightly negative slope, suggesting that larger differences result in lower success rates than those with smaller differences between p and q . For example, fitting a linear regression to the success rate of the linear-CNOT *Ansatz* shown in Fig. 6(a) yields that the success rate is approximately $0.5 - 3.7 \times 10^{-5}|p - q|$. While this slope of -3.7×10^{-5} is very small, its standard deviation is 3.5×10^{-6} , i.e., there is a 10-standard deviation separation of the slope from 0, making it highly significant. Similarly, for the parallel-CNOT case shown in Fig. 6(b), the slope is negative with a 7.3-standard deviation separation from 0. However, when examining the linear regression of the degenerate and nondegenerate data separately, the slopes are generally less than three standard deviations away from 0. Furthermore, we



(a) Linear-CNOT ansätze



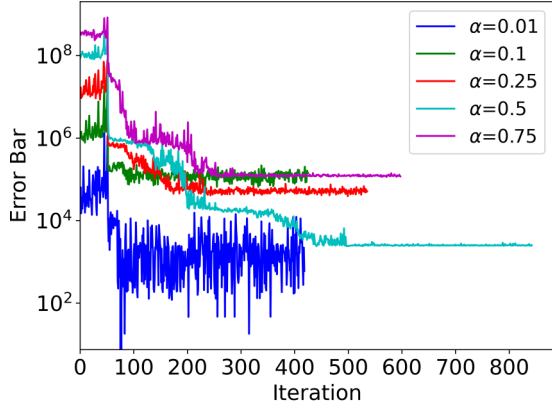
(b) Parallel-CNOT ansätze

FIG. 6. Average success rate for 50 random initial parameters for the ansatz (a) linear-CNOT and (b) parallel-CNOT for all possible numbers n requiring 20 qubits, using $\alpha = 0.1$, fidelity threshold $t = 0.1$, 3 layers, and a maximum iteration number of 3000. The x axis represents the absolute value of the difference between the prime factors of a specific n , i.e., $|p - q|$ for $p \cdot q = n$.

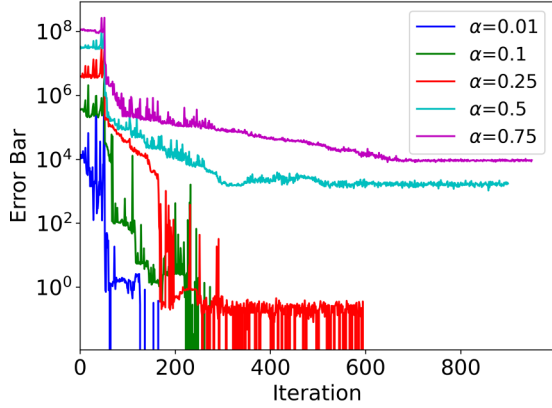
observe that the slope for the *Ansatz* parallel-CNOT is slightly positive (1.4 standard deviations positive), indicating that larger differences between p and q have a higher success rate. A notable exception is the case of a parallel-CZ *Ansatz*, which is identical to parallel-CNOT except each CNOT-gate is replaced with the corresponding CZ-gate, for which the degenerate data shows a five-standard deviation positive slope. Consequently, at this time, it appears as if degeneracy slightly improves the success rate but no definitive conclusion can be made about the success rate as a function of $|p - q|$ beyond any potential degeneracy effects. Instead it seems more prudent to expect that the small variations in perceived difficulty are instance and *Ansatz* dependent.

B. CVaR coefficient

Analyzing our initial hardware runs, it seems that smaller values of α are required as the system size increases. Furthermore, based on the ideal simulation results with infinite shots, as shown in Fig. 3, it is evident that a smaller α leads to a higher success rate. However, when using a finite number of shots S , the native intuition is that the standard error of



(a) 1000 shots



(b) 10000 shots

 FIG. 7. Error bar in VQE process for different α values for $n = 8189$, requiring 17 qubits.

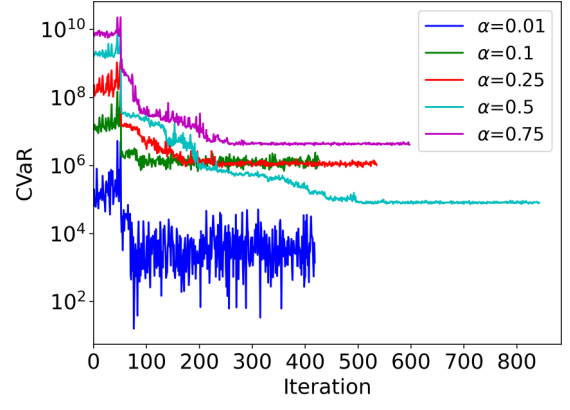
estimated CVaR scales as $O[1/(\alpha S)]$, indicating a smaller α might result in fewer effective measurements and larger standard error, which may impact the optimization process of VQE. Therefore, it is crucial to balance the improved performance of a smaller α with the associated statistical error. In this section, We explore the behavior of the standard error for different α with finite shots.

The standard error with a given number of shots S is defined as:

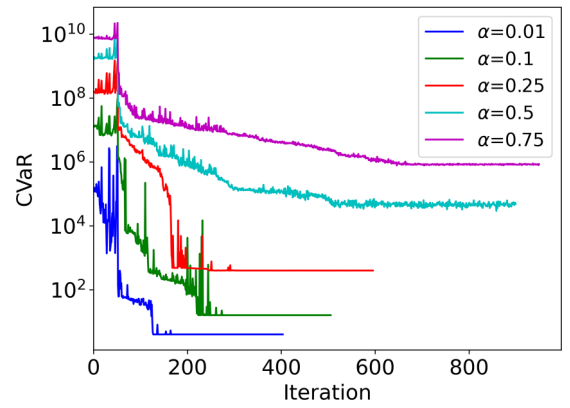
$$\sigma = \sqrt{\frac{\sum_{k=1}^{\lceil \alpha \cdot S \rceil} [e_k - \text{CVaR}(\alpha, S)]^2}{\lceil \alpha \cdot S \rceil (\lceil \alpha \cdot S \rceil - 1)}}, \quad (9)$$

where the measured eigenvalues are sorted in ascending order $e_1 \leq e_2 \leq \dots \leq e_S$, and $\text{CVaR}(\alpha, S)$ is detailed in Eq. (8). As shown in the above equation, there are two factors that impact the value of the standard error: first, a smaller α leads to less effective measurements $\lceil \alpha \cdot S \rceil$, which will lead to a smaller denominator in Eq. (9); on the other hand, smaller α means selecting a smaller portion of the left tail of the measured results, which will lead to a smaller numerator in Eq. (9), these two factors will compete with each other.

Figure 7 displays the standard error of CVaR for different α values in the VQE process used to factorize $n = 8189$, requiring 17 qubits. For consistency, VQE for all the α values starts at the same initial state $|+\rangle^{\otimes N}$. At the beginning of the



(a) 1000 shots



(b) 10000 shots

 FIG. 8. CVaR in VQE process for different α values for $n = 8189$, requiring 17 qubits.

VQE process, smaller α results in a smaller standard error for both 1000 shots and 10000 shots, which indicates the numerator in Eq. (9) dominate. As optimization progresses, the VQE with different α might converge to different local minima, potentially leading to a smaller standard error for larger α , as depicted in Fig. 7(a). While Fig. 7(b) with a larger shot number $S = 10000$ indicates a clear trend that smaller α is more likely to have a smaller standard error during the VQE process. Nonetheless, the $\alpha = 0.01$ always keeps the smallest standard error during the whole VQE process for both shots $S = 1000$ and $S = 10000$.

Furthermore, the corresponding cost function value using different α in the VQE process is shown in Fig. 8. With shots $S = 1000$, the smaller α can not certainly result in a better result, which might be due to the optimizer being misled by the large standard error; when increasing the shots number to reduce the standard error, smaller α can keep smaller standard error and result in better performance.

In summary, smaller α does not necessarily lead to larger standard error in CVaR when using finite shots. Although only α portion of sampled result is selected to do the average, which reduce the dominator in Eq. (9), a more concentrated distribution with a smaller α also results in a smaller numerator, ultimately leading to a smaller standard error of CVaR. Additionally, although smaller α achieves smaller

standard error in the beginning, a relatively large statistic error might still mislead the optimization and result into a worse local minimum, while with a sufficiently large number of shots, smaller α can keep a small standard error and good stable performance. Our result indicate the most stable choice of CVaR value is $\alpha = 0.01$ among our testing values $\alpha \in \{0.01, 0.1, 0.25, 0.5, 0.75\}$. This is in agreement with our initial observation that smaller α is better. However, it should be noted that this can only be expected to be true within reasonably large numbers of shots used. More precisely, if $\alpha \leq \frac{1}{S}$, then the CVaR will only use the smallest observed value, which leads to discontinuous optimization landscapes. Thus, α should be kept significantly larger than $\frac{1}{S}$, but within that regime a smaller α seems to be better. Finding the critical point at which α becomes too small is likely highly problem and instance specific, so we recommend users to check for their individual applications.

C. Cost functions

In Sec. IV, we present the success rate of the CVaR-VQE in the ideal simulation with infinite measurement shots, the success rate is almost 100% if using CVaR coefficient $\alpha = 0.01$, as shown in Fig. 3. This indicates the ability of CVaR-VQE to find the optimal solution when the expectation of cost function can be estimated with perfect accuracy. While in practical scenarios, only finite shots can be used when estimating the expectation of cost function, the statistic error as described in Eq. (9) may mislead the optimizer and disrupt the optimization process to the optimal solution. In this section, we investigate the effect of the finite shots on success rate. Furthermore, we explore different cost function types, trying to improve the success rate in the case of finite shots.

Using $n = 32743$ as an example, which requires 20 qubits, we explore the success rate with finite shots among 500 runs with random initial parameters, the number of measurements for estimating the CVaR value ranges from 1000–200000. The *Ansatz* layer $L = 3$ and CVaR coefficient is selected as $\alpha = 0.01$, which also has the best performance with finite shots as indicated in the last section. Besides the optimizer COBYLA we used in the previous section, we also test the performance of NFT [36] in the case of finite shots. As shown in Fig. 9(a), using the Hamiltonian cost function Eq. (6), the success rate is almost zero when the number of shots is smaller than 10000 when using COBYLA, NFT performs better in this scenario. However, for both optimizers, COBYLA and NFT, the success rates are only around 20% for even 200000 shots.

The Hamiltonian cost function can produce extremely large eigenvalues. For instance, in the case $p \sim q \sim 1$, the function $(n - pq)^2 \sim n^2 \sim 10^9$, which means optimizing the low-energy part needs very high accuracy when estimating the expectation value. For example, the change from the suboptimal solution $(n - pq)^2 = 1$ to the optimal solution $(n - pq)^2 = 0$ might only result in a relative change in the CVaR value of approximately $\frac{1}{n^2} \sim 10^{-9}$, which might be even smaller than the standard error of CVaR in the case of finite shots. Therefore, we consider additional cost functions. One option is a logarithmic cost function, which compresses

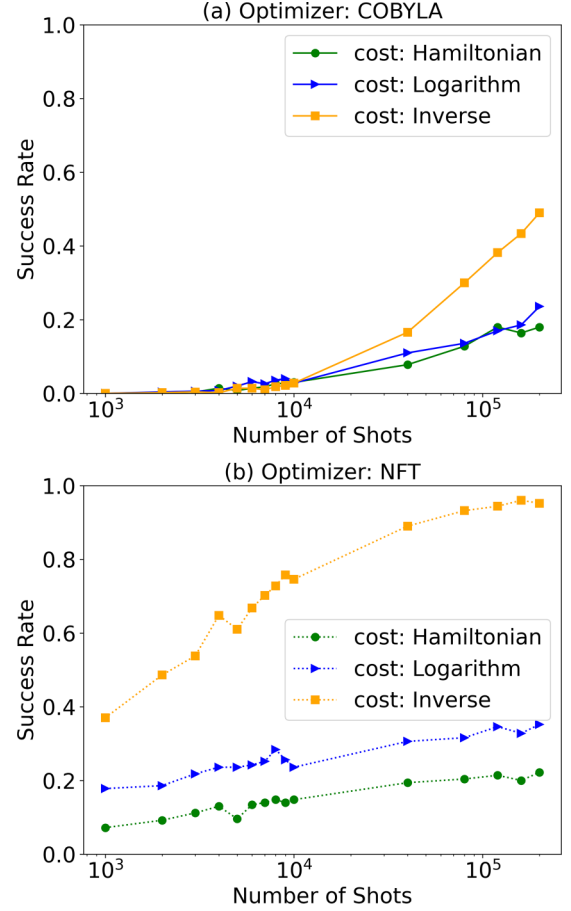


FIG. 9. Success rates a function of the number of measurement shots for different types of cost functions using (a) the COBYLA optimizer and (b) NFT.

the high-energy domain, making it less sensitive to large eigenvalues. Another option is an inverse type of cost function, which not only compresses the high-energy domain but also introduces a large energy gap between the solution and suboptimal solutions. The following are the cost functions used:

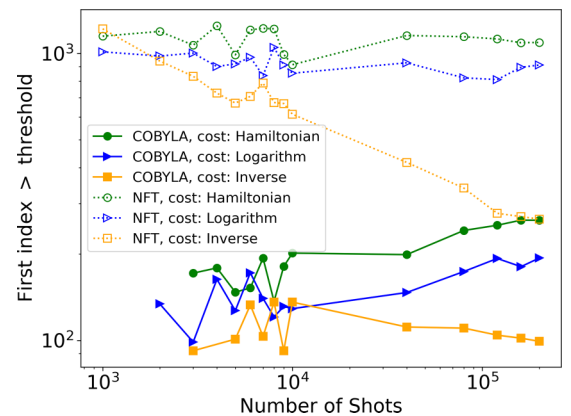


FIG. 10. Number of iterations required to reach the fidelity threshold for the successful runs, with various types of cost functions and using COBYLA or NFT optimizers.

- (1) Hamiltonian: $(n - pq)^2$,
- (2) Logarithm: $\lceil \log(|n - pq| + 1) \rceil$,
- (3) Inverse: $-\frac{1}{|n - pq| + 0.001}$.

Since each observed bit string encodes the binary representation of a candidate solution for p and q , we can directly compute the cost function values from each shot. As shown in Fig. 9, the logarithmic cost function has comparable performance with the Hamiltonian cost function, while the inverse type of cost function has much better performance, especially with the NFT optimizer. However, it is important to note that the NFT optimizer needs a larger number of function evaluations to achieve the fidelity 1% as shown in Fig. 10. It is not clear at the moment whether the logarithmic and inverse type of cost function are related to global operators, which lead to the problem of barren plateau [37]. To address this, future work will focus on identifying suitable initial parameters for the inverse cost function to overcome the potential barren plateau issue and reduce optimization time. This will be a critical area of exploration moving forward.

VII. CONCLUSION

In this work, we benchmark the performance of CVaR-VQE in solving the prime factorization problem without relying on prior arithmetic simplifications. The number 1 048 561 requiring 27 qubits is factorized in the ideal simulation, and the number 253 requiring nine qubits is factorized by the IBM hardware.

Furthermore, we explore various factors influencing VQE's performance on the prime factorization problem. First, we observe the difference between two factors $|p - q|$, which is crucial in the difficulty of classical algorithms, does not impact the success rate of VQE a lot, while the choice of

Ansatz impacts VQE's performance on the factoring problem obviously. Second, we explore how the statistic error behaves across different CVaR coefficients α . In contrast to naive intuition, the smaller α achieves a smaller standard error and better performance with a sufficient number of measurement shots. Especially, $\alpha = 0.01$ performs best in our problem with finite shots.

Finally, we use $\alpha = 0.01$ to examine the scaling of success rate with shots number, the result indicates the Hamiltonian cost function needs a huge number of shots to improve the success rate, prompting us to explore a better cost function for VQE. We find an inverse-type cost function with the NFT optimizer can achieve a much higher success rate. However, the inverse-type cost function needs more iterations to converge, and it is an open question whether this cost function corresponds to a global observable and might have the problem of barren plateau. A possible way to overcome the above problems is to find a good initial parameter of the VQE for this inverse cost function, which shows significant improvement in reducing iteration number and mitigating the barren plateau problem in the QUBO problem [38]. While it is nontrivial to extend this method to the inverse-type cost function, we leave it to future work.

ACKNOWLEDGMENTS

This work is funded by the European Union's Horizon Europe Framework Programme (HORIZON) under the ERA Chair scheme with grant Agreement No. 101087126 and under the European Research Council (ERC) with Grant Agreement No. 787331. This work is supported with funds from the Ministry of Science, Research and Culture of the State of Brandenburg within the Centre for Quantum Technologies and Applications (CQTA).

APPENDIX A: PERFORMANCE OF DIFFERENT ANSÄTZE WITH DIFFERENCE BETWEEN p AND q

Figure 11 shows all four different *Ansätze* that we tested and Fig. 12 shows the performance of parallel-CZ and circular-CNOT *Ansätze* regarding the difference between two factors p and q .

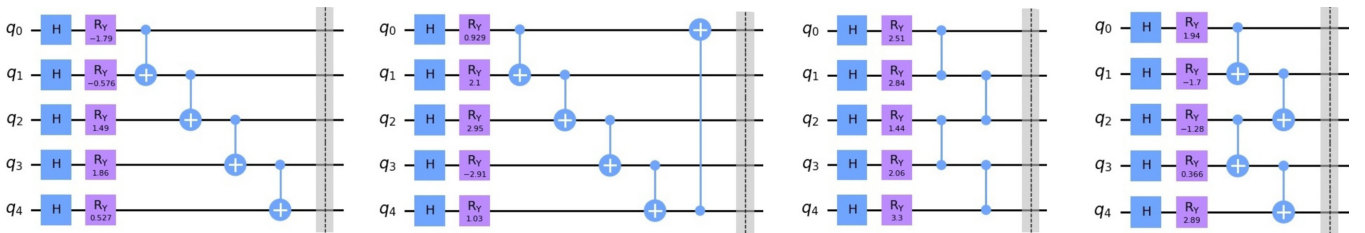


FIG. 11. *Ansätze* from left to right: linear-CNOT, circular-CNOT, CZ-parallel, CNOT-parallel. Circuits illustrated with QISKIT library [34].

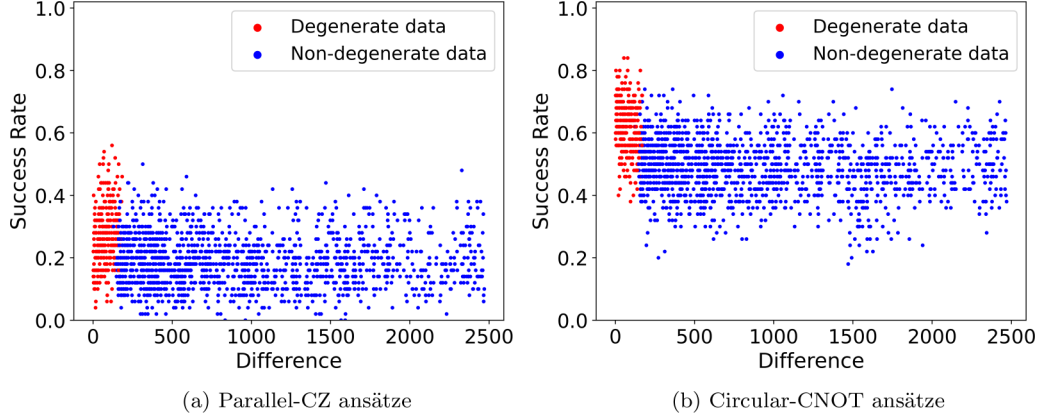


FIG. 12. Average success rate for 50 random initial parameters for the *Ansatz* (a) parallel-CZ and (b) circular-CNOT *Ansätze* for all possible numbers n requiring 20 qubits, using $\alpha = 0.1$, fidelity threshold $t = 0.1$, three layers, and a maximum iteration number of 3000. The x axis represents the absolute value of the difference between the prime factors of a specific n , i.e., $|p - q|$ for $p \cdot q = n$.

APPENDIX B: FACTORIZATION RECORDS OF DIFFERENT METHODS

Table III presents all factorization records we found for hybrid quantum-classical approaches. We do not claim completeness here.

TABLE III. Factorization records using hybrid quantum-classical approaches.

Number n	N qubits	Algorithm	Hardware/Sim.	Year
15	7	Shor	NMR	2001 [39]
15	4	Shor	Photonic Qubits	2007 [40]
15	4	Shor	Photonic Qubits	2007 [41]
21	3	Adiabatic	NMR	2008 [14]
15	3	Shor	Photonic	2009 [42]
143	4	Adiabatic	NMR	2011 [43]
15	3	Shor	Superconducting	2012 [44]
21	2	Shor	Photonic	2012 [45]
11663	4	Adiabatic	NMR	2014 [46]
291 311	3	Adiabatic	NMR	2017 [15]
4 088 459	4	Grover	IBM	2018 [47]
291 311	6	VQF	IBM (SIM)	2018 [20]
376 289	94	Annealing	D-Wave (SIM)	2018 [18]
1 005 973	89	Annealing	D-Wave (SIM)	2019 [48]
1 028 171	88	Annealing	D-Wave (SIM)	2020 [11]
1 099 551 473 989	5	VQF	IBM (SIM)	2021 [22]
21	5	Shor	IBM	2021 [2]
1829	9	QITE	IBM	2021 [26]
35	5	Adiabatic	IBM	2021 [16]
247	13	VQF	IBM	2022 [23]
261 980 999 226 229	10	Babai's, Lattice, QAOA	IBM	2022 [28]
261 980 999 226 229	10	DCQF	Trapped-ion	2023 [29]
102 454 763	26	Annealing	D-Wave (SIM)	2023 [19]
1 269 636 549 803	4	Grover	IBM	2023 [27]
1 000 070 001 221	12	Annealing	D-Wave (SIM)	2023 [19]

- [1] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (IEEE, Santa Fe, NM, USA, 1994), pp. 124–134.
- [2] U. Skosana and M. Tame, Demonstration of Shor’s factoring algorithm for $N = 21$ on IBM quantum processors, *Sci. Rep.* **11**, 16599 (2021).
- [3] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
- [4] G. Nannicini, Performance of hybrid quantum-classical variational heuristics for combinatorial optimization, *Phys. Rev. E* **99**, 013304 (2019).
- [5] Y. Chai, L. Funcke, T. Hartung, K. Jansen, S. Kuehn, P. Stornati, and T. Stollenwerk, Optimal flight-gate assignment on a digital quantum computer, *Phys. Rev. Appl.* **20**, 064025 (2023).
- [6] D. Amaro, M. Rosenkranz, N. Fitzpatrick, K. Hirano, and M. Fiorentini, A case study of variational quantum algorithms for a job shop scheduling problem, *EPJ Quant. Technol.* **9**, 5 (2022).
- [7] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O’Brien, A variational eigenvalue solver on a photonic quantum processor, *Nat. Commun.* **5**, 4213 (2014).
- [8] C. J. C. Burges, Factoring as optimization. microsoft research Technical Report TR-2002-83, 2002.
- [9] R. T. Rockafellar and S. Uryasev, Conditional value-at-risk for general loss distributions, *J. Banking Finance* **26**, 1443 (2002).
- [10] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, Improving variational quantum optimization using CVaR, *Quantum* **4**, 256 (2020).
- [11] B. Wang, F. Hu, H. Yao, and C. Wang, Prime factorization algorithm based on parameter optimization of Ising model, *Sci. Rep.* **10**, 7106 (2020).
- [12] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem, *Science* **292**, 472 (2001).
- [13] M. Born and V. Fock, Proof of the adiabatic theorem, *Z. Phys.* **51**, 165 (1928).
- [14] X. Peng, Z. Liao, N. Xu, G. Qin, X. Zhou, D. Suter, and J. Du, Quantum adiabatic algorithm for factorization and its experimental implementation, *Phys. Rev. Lett.* **101**, 220405 (2008).
- [15] Z. Li, N. S. Dattani, X. Chen, X. Liu, H. Wang, R. Tanburn, H. Chen, X. Peng, and J. Du, High-fidelity adiabatic quantum computation using the intrinsic Hamiltonian of a spin system: Application to the experimental factorization of 291311, *arXiv:1706.08061*.
- [16] A. Saxena, A. Shukla, and A. Pathak, A hybrid scheme for prime factorization and its experimental implementation using IBM quantum processor, *Quantum Info. Proc.* **20**, 112 (2021).
- [17] D-Wave Systems, *What is Quantum Annealing* (D-Wave Systems Inc., 2024).
- [18] S. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, and S. Kais, Quantum annealing for prime factorization, *Sci. Rep.* **8**, 17667 (2018).
- [19] K. Jun and H. Lee, Hubo and QUBO models for prime factorization, *Sci. Rep.* **13**, 10080 (2023).
- [20] E. R. Anschuetz, J. P. Olson, A. Aspuru-Guzik, and Y. Cao, Variational quantum factoring, *arXiv:1808.08927*.
- [21] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, *arXiv:1411.4028*.
- [22] A. H. Karamlou, W. A. Simon, A. Katabarwa, T. L. Scholten, B. Peropadre, and Y. Cao, Analyzing the performance of variational quantum factoring on a superconducting quantum processor, *npj Quantum Inf.* **7**, 156 (2021).
- [23] V. Phan, A. Pönni, M. Raasakka, and I. Tittonen, On quantum factoring using noisy intermediate scale quantum computers, *arXiv:2208.07085*.
- [24] M. Motta, C. Sun, A. T. K. Tan, M. J. O’Rourke, E. Ye, A. J. Minnich, F. G. S. L. Brandão, and G. K.-L. Chan, Determining eigenstates and thermal states on a quantum computer using quantum imaginary time evolution, *Nat. Phys.* **16**, 205 (2020).
- [25] S. McArdle, T. Jones, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan, Variational ansatz-based quantum simulation of imaginary time evolution, *npj Quantum Inf.* **5**, 75 (2019).
- [26] R. Selvarajan, V. Dixit, X. Cui, T. S. Humble, and S. Kais, Prime factorization using quantum variational imaginary time evolution, *Sci. Rep.* **11**, 20835 (2021).
- [27] R. Dhaulakhandi, B. K. Behera, and F. J. Seo, Factorization of large tetra and penta prime numbers on IBM quantum processor, *APL Quantum* **1**, 026113 (2024).
- [28] B. Yan, Z. Tan, S. Wei, H. Jiang, W. Wang, H. Wang, L. Luo, Q. Duan, Y. Liu, W. Shi, Y. Fei, X. Meng, Y. Han, Z. Shan, J. Chen, X. Zhu, C. Zhang, F. Jin, H. Li, C. Song, Z. Wang, Z. Ma, H. Wang, and G.-L. Long, Factoring integers with sublinear resources on a superconducting quantum processor, *arXiv:2212.12372*.
- [29] N. N. Hegade and E. Solano, Digitized-counterdiabatic quantum factorization, *Phys. Rev. Res.* **4**, L042030 (2022).
- [30] M. J. D. Powell, *A Direct Search Optimization Method that Models the Objective and Constraint Functions By Linear Interpolation*, (Springer, Dordrecht, 1994), pp. 51–67.
- [31] T. M. Ragonneau and Z. Zhang, PDFO: A cross-platform package for Powell’s derivative-free optimization solvers, *arXiv:2302.13246*.
- [32] P. Diez-Valle and J. J. Garcia-Ripoll, Quantum variational optimization: The role of entanglement and problem hardness, *Phys. Rev. A* **104**, 062426 (2021).
- [33] T. M. Ragonneau, Model-based derivative-free optimization methods and software, *arXiv:2210.12018*.
- [34] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, J. M. Gambetta, Quantum computing with Qiskit, *arXiv:2405.08810*.
- [35] S. Seabold and J. Perktold, Statsmodels: Econometric and statistical modeling with Python, *Proceedings of the 9th Python in Science Conference* (SciPy, Austin, Texas, USA, 2010).
- [36] K. M. Nakanishi, K. Fujii, and S. Todo, Sequential minimal optimization for quantum-classical hybrid algorithms, *Phys. Rev. Res.* **2**, 043158 (2020).
- [37] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, *Nat. Commun.* **12**, 1791 (2021).
- [38] Y. Chai, K. Jansen, S. Kühn, T. Schwägerl, and T. Stollenwerk, Structure-inspired ansatz and warm start of variational quantum algorithms for quadratic unconstrained binary optimization problems, *arXiv:2407.02569*.
- [39] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, Experimental realization of Shor’s quantum factoring algorithm using nuclear magnetic resonance, *Nature (London)* **414**, 883 (2001).

- [40] B. P. Lanyon, T. J. Weinhold, N. K. Langford, M. Barbieri, D. F. V. James, A. Gilchrist, and A. G. White, Experimental demonstration of a compiled version of shor's algorithm with quantum entanglement, *Phys. Rev. Lett.* **99**, 250505 (2007).
- [41] C. Y. Lu, D. E. Browne, T. Yang, and J. W. Pan, Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits, *Phys. Rev. Lett.* **99**, 250504 (2007).
- [42] A. Politi, C. F. J. Matthews, and J. L. O'Brien, Shor's quantum factoring algorithm on a photonic chip, *Science* **325**, 1221 (2009).
- [43] N. Xu, J. Zhu, D. Lu, X. Zhou, X. Peng, and J. Du, Quantum factorization of 143 on a dipolar-coupling nuclear magnetic resonance system, *Phys. Rev. Lett.* **109**, 269902 (2012).
- [44] 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 J. M. Martinis, Computing prime factors with a Josephson phase qubit quantum processor, *Nat. Phys.* **8**, 719 (2012).
- [45] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien, Experimental realization of Shor's quantum factoring algorithm using qubit recycling, *Nat. Photon.* **6**, 773 (2012).
- [46] N. S. Dattani and N. Bryans, Quantum factorization of 56153 with only 4 qubits, *arXiv:1411.6758*.
- [47] A. Dash, D. Sarmah, B. K. Behera, and P. K. Panigrahi, Exact search algorithm to factorize large biprimes and a triprime on IBM quantum computer, *arXiv:1805.10478*.
- [48] W. Peng, B. Wang, F. Hu, Y. Wang, X. Fang, X. Chen, and C. Wang, Factoring larger integers with fewer qubits via quantum annealing with optimized parameters, *Sci. China Phys. Mech. Astron.* **62**, 60311 (2019).