Benchmarking hybrid digitized-counterdiabatic quantum optimization

Ruoqian Xu,[1] Jialiang Tang,[1] Pranav Chandarana,[1,2] Koushik Paul,[1,2,*] Xusheng Xu,[3]
Manhong Yung,[4,5,6,7] and Xi Chen[1,2,†]

[1]*Department of Physical Chemistry, University of the Basque Country UPV/EHU, Apartado 644, 48080 Bilbao, Spain*
[2]*EHU Quantum Center, University of the Basque Country UPV/EHU, Barrio Sarriena, s/n, 48940 Leioa, Spain*
[3]*Department of Physics, State Key Laboratory of Low-Dimensional Quantum Physics, Tsinghua University, Beijing 100084, People's Republic of China*
[4]*Department of Physics, Southern University of Science and Technology, Shenzhen 518055, People's Republic of China*
[5]*Shenzhen Institute for Quantum Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, People's Republic of China*
[6]*Guangdong Provincial Key Laboratory of Quantum Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, People's Republic of China*
[7]*Shenzhen Key Laboratory of Quantum Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, People's Republic of China*

(Received 12 December 2023; accepted 17 January 2024; published 7 February 2024)

Hybrid digitized-counterdiabatic quantum computing (DCQC) is a promising approach for leveraging the capabilities of nearterm quantum computers, utilizing parameterized quantum circuits designed with counterdiabatic protocols. However, the classical aspect of this approach has received limited attention. In this study, we systematically analyze the convergence behavior and solution quality of various classical optimizers when used in conjunction with the digitized-counterdiabatic approach. We demonstrate the effectiveness of this hybrid algorithm by comparing its performance to the traditional QAOA on systems containing up to 28 qubits. Furthermore, we employ principal component analysis to investigate the cost landscape and explore the crucial influence of parametrization on the performance of the counterdiabatic ansatz. Our findings indicate that fewer iterations are required when local cost landscape minima are present, and the SPSA-based BFGS optimizer emerges as a standout choice for the hybrid DCQC paradigm.

DOI: 10.1103/PhysRevResearch.6.013147

I. INTRODUCTION

In the field of quantum computing, one crucial aspect of research involves benchmarking quantum algorithms. The purpose of benchmarking is to thoroughly evaluate the performance of a quantum algorithm using a well-defined set of metrics. This evaluation process is particularly important for the noisy intermediate-scale quantum (NISQ) era. Benchmarking quantum algorithms is essential for gaining insights into the capabilities and limitations of NISQ computers while executing quantum algorithms.

In recent years, the field of quantum computing has witnessed remarkable progress, opening up new frontiers for solving complex computational problems that were once considered intractable for classical computers. Among several promising quantum algorithms, variational quantum algorithms (VQAs) have emerged as a particularly intriguing and versatile approach [1]. VQAs utilize the unique properties of quantum systems to tackle optimization tasks by iteratively optimizing the parameters of a quantum circuit to minimize a given cost function. This flexibility and adaptability make VQAs well suited for a wide range of applications, including quantum chemistry simulations [2–4], machine learning [5,6], and combinatorial optimization [7,8].

One powerful example of VQA is the quantum approximate optimization algorithm (QAOA) [9]. QAOA employs p layers of unitary operations to iteratively explore the potential solutions using classical optimization and enhance the probability of obtaining the optimal solution. By adjusting the number of layers, QAOA offers efficient use of quantum hardware for solving a given optimization problem. One of the main challenges of QAOA is its sensitivity to the choice of hyperparameters. Finding the optimal values for these parameters can be a challenging and time-consuming task, especially for large-scale optimization problems, as it may require significant classical computational resources and extensive experimentation.

Over the past few years, numerous algorithms have been introduced to enhance the capabilities of original QAOA. One such example is multiangle QAOA (maQAOA), which incorporates additional parameters into the quantum circuit to achieve improved approximation ratios compared to the standard QAOA [10]. Similarly, adaptive derivative-assembled

[*] koushikpal09@gmail.com
[†] chenxi1979cn@gmail.com

problem-tailored QAOA (ADAPT-QAOA) employs an iterative process to select the ansatz from a predefined pool of operators [11]. This selection aims to maximize the gradient of the commutator between the pool operator and the cost Hamiltonian, leading to enhanced optimization outcomes. Several other techniques, such as recursive QAOA (RQAOA) [12,13], quantum alternating operator Ansatzes (QAOAnsatz) [14], spanning tree QAOA (ST-QAOA) [15], and adaptive bias QAOA (ab-QAOA) [16], have been proposed with the aim of enhancing various aspects of the QAOA. These methods claim varying degrees of improvements in circuit depth, parameter space, operator pool, and computational cost [17].

In this paper, we examine the performance and effectiveness of hybrid digitized-counterdiabatic quantum computing (DCQC) algorithms [18]. This method incorporates elements from shortcut to adiabaticity (STA) [19,20], particularly the use of counterdiabatic (CD) driving [21], to reduce the circuit depth and improve the optimization process of conventional QAOA. In prior studies, these algorithms have exhibited their efficacy in addressing diverse optimization problems, including MaxCut, portfolio optimization, protein folding, and several quantum many-body systems [18,22–25]. We examine the classical component of the hybrid DCQC in this study. Our investigation focuses on convergence while using different gradient-based and gradient-free optimizers. We illustrate the scalability of this approach concerning system size, highlighting its effectiveness in finding the ground state, even when $p = 1$ within systems containing up to 28 qubits. To consolidate these findings, we explore the Fourier landscape of the cost function for various parametrizations using principal component analysis (PCA).

This paper is structured as follows. Section II provides a brief discussion of hybrid DCQC followed by a detailed description of gradient-based and gradient classical optimizers and their application in hybrid DCQC. This is followed by respective findings regarding various parametrizations, cost landscape, and variance of gradients for different optimizers in Sec. III. We finally conclude in Sec. IV and provide direction for future work.

II. BACKGROUND

Hybrid DCQC algorithms fall under the umbrella of VQAs where the quantum part consists of a circuit ansatz with parameterized gates. These parameters are optimized by classical routines to minimize a given cost function. The selection of these parameterized gates can be done in two major ways: taking information from the problem or taking information from the hardware. The former is known as problem-inspired ansatz and the latter is known as hardware-efficient ansatz. One of the popular problem-inspired ansatzes is QAOA where two unitaries $U_b(\boldsymbol{\beta})$ and $U_c(\boldsymbol{\alpha})$ with tunable parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ are applied iteratively p times to an initial state $|\psi_0\rangle$. Here, $U_b(\boldsymbol{\beta}) = \exp(-i\boldsymbol{\beta} H_m)$ is the mixing unitary and $U_c(\boldsymbol{\alpha}) = \exp(-i\boldsymbol{\alpha} H_c)$ is the unitary corresponding to the cost Hamiltonian with $[H_m, H_c] \neq 0$. The aim is to minimize the expectation value $\langle\psi(\boldsymbol{\alpha}, \boldsymbol{\beta})|H_c|\psi(\boldsymbol{\alpha}, \boldsymbol{\beta})\rangle$ for achieving the ground state of the system that encodes the solution the chosen problem.

The main drawback of QAOA is the need for high circuit depth (large p) as it takes inspiration from the adiabatic process, which is inherently slow. To circumvent this, DCQC makes use of counterdiabatic protocols to decrease the circuit depth of the ansatz. In DCQC, the circuit ansatz is chosen by finding suitable operators using the nested commutator (NC) method [26]. The resultant parameterized unitary is of the form [25]

$$U_d(\boldsymbol{\theta}) = \exp(-i\boldsymbol{\theta}\mathcal{A}), \tag{1}$$

where $\mathcal{A} \in A_\lambda^{(l)}$, and $A_\lambda^{(l)}$ is an operator pool obtained from the NC method [25]. $\boldsymbol{\theta}$ is the set of parameters to be optimized.

The classical optimizers, employed to optimize these parameters, also play a crucial role in deciding the performance of an ansatz. Classical optimizers can be broadly classified into two categories: gradient based and gradient free. In gradient-based optimization, the iteration step is decided based on the gradient direction. The gradient values can be evaluated by various methods such as finite difference or parameter shift [27]. There are also newer methods that take fewer circuit evaluations to compute the gradient [28]. After evaluating the gradient, classical optimizers like Adam [29] and Adagrad [30] are implemented to find the next iteration steps.

On the other hand, gradient-free optimizers do not rely on the gradient information to determine the next iteration. These include optimizers like COBYLA [31], Nelder-Mead [32], etc. There also exist other optimizers like the simultaneous perturbation stochastic approximation (SPSA) [33] that do not directly utilize gradient information. In SPSA, the gradient is approximated along a randomly chosen direction by a single partial derivative computed by finite difference. This has shown a faster convergence rate than the finite difference method [33].

Before starting to explore gradient-based optimizers, we commence with describing a few methods to calculate gradients, which will be used later.

Parameter shift (PS). Let us consider a cost function $J(\boldsymbol{\theta})$, which is parameterized by some variables $\boldsymbol{\theta} \in \mathbb{R}$. PS enables us to calculate the exact gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta} = \{\theta_1, \theta_2, \cdots, \theta_m\}$ using the following relation [27]:

$$\nabla_{\theta_i} J(\boldsymbol{\theta}) = r\left[J\left(\theta_i + \frac{\pi}{4r}\right) - J\left(\theta_i - \frac{\pi}{4r}\right)\right], \tag{2}$$

where $\nabla_{\theta_i} J(\boldsymbol{\theta})$ represents the gradient with respect to element $\theta_i \in \boldsymbol{\theta}$. r is the shift constant, determined by the ansatz chosen. For instance, it is proved that $r = \frac{1}{2}$ for all single qubit gates [34]. This method evaluates the same ansatz twice based on different shifted parameters to find the gradient making the runtime proportional to twice the total number of parameters.

SPSA. SPSA approaches the gradient by introducing a m-dimensional perturbation vector $\boldsymbol{\delta} = (\delta_1, \delta_2, \cdots, \delta_m)^T$, whose elements δ_i are all randomly picked from $\{-1, 1\}$. The gradient estimator can be written as [35]

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{J(\boldsymbol{\theta} + c\boldsymbol{\delta}) - J(\boldsymbol{\theta} - c\boldsymbol{\delta})}{2c\boldsymbol{\delta}}, \tag{3}$$

where c is a small positive scaling factor. Due to this perturbation, SPSA is robust under noisy environments.

TABLE I. Update rules of parameters and gradients for hybrid gradient-based optimizers using parameter shift rules (PS) or SPSA rules and gradient-free optimizers.

Optimizers	Gradient calculation	Update rules
SPSA-Adam	$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{J(\boldsymbol{\theta}+c\boldsymbol{\delta})-J(\boldsymbol{\theta}-c\boldsymbol{\delta})}{2c\boldsymbol{\delta}}$	$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\eta}{\sqrt{\hat{v}_k}+\epsilon}\hat{m}_k$
SPSA-SGD		$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - a_k \cdot \nabla_k J(\boldsymbol{\theta})$
SPSA-BFGS		$\nabla_k J(\boldsymbol{\theta}) = \nabla_{k+1} J(\boldsymbol{\theta}^{k+1}) + B_{k+1}(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+1})$
PS-Adam	$\nabla_{\theta_i} J(\boldsymbol{\theta}) = r\big[J\big(\theta_i + \frac{\pi}{4r}\big) - J\big(\theta_i - \frac{\pi}{4r}\big)\big]$	$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\eta}{\sqrt{\hat{v}_k}+\epsilon}\hat{m}_k$
PS-SGD		$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - a_k \cdot \nabla_k J(\boldsymbol{\theta})$
PS-BFGS		$\nabla_k J(\boldsymbol{\theta}) = \nabla_{k+1} J(\boldsymbol{\theta}^{k+1}) + B_{k+1}(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+1})$
COBYLA	Terminates when the target radius is satisfied.	
Nelder-Mead	Terminates when the cost function is minimized according to the criteria in Eq. (B4).	

Adjoint differentiation. Consider the circuit ansatz defined by the unitary evolution operator $U = U_m(\theta_m)\cdots U_2(\theta_2)U_1(\theta_1)$ and any observable $\hat{O}$ for which we compute the trial state as $|\varphi_i\rangle = U_{i+1}^\dagger(\theta_{i+1})\cdots U_m^\dagger(\theta_m)\hat{O}U_m(\theta_m)\cdots U_2(\theta_2)U_1(\theta_1)|0\rangle$ and define $|\psi_i\rangle = U_i(\theta_i)\cdots U_2(\theta_2)U_1(\theta_1)|0\rangle$, such that the gradient with respect to $\theta_i \in \boldsymbol{\theta}$ can be formulated as

$$\nabla_{\theta_i} J(\boldsymbol{\theta}) = \frac{\partial\langle 0|U^\dagger \hat{O} U|0\rangle}{\partial\theta_i} = 2\mathbf{Re}\langle\varphi_{i-1}|\frac{\partial U_i(\theta_i)}{\partial\theta_i}|\psi_{i-1}\rangle. \quad (4)$$

Using the above-mentioned methods of computing gradients, we implement a total of six gradient-based optimizers, which are PS-SGD, PS-BFGS, PS-Adam, SPSA-SGD, SPSA-BFGS, SPSA-Adam, to compare their performance with different ansatzes. Further details on the optimizers can be found in Appendix A. In addition, we also used two gradient-free optimizers, namely, COBYLA, and Nelder-Mead (see Appendix B). Table I provides a brief summary of the update rules for every optimizer. Due to the high computational costs required for parameter shift, we use the adjoint method in simulations. This is because we have found a strong similarity between these two methods of computing gradients, which is explained explicitly in Appendix C.

One of the most important factors that determines the performance of an ansatz is its expressibility, which is defined as how uniformly the ansatz is capable of searching the solution space of the problem Hamiltonian. When the solution is not known, a highly expressible ansatz is required to explore larger spaces such that the probability of finding the solution is higher. On the other hand, even when the ansatz includes the solution, it should be trainable, such that the optimization can reach the solution effectively. So, the requirement for a "good" ansatz is that it should include the solution and be sufficiently trainable. Therefore, the choice of classical optimization can have a drastic impact on the trainability of the circuit ansatz. High expressibility is not always as good as one may face the problem of barren plateaus where the gradient vanishes exponentially with increasing system size and parameter space. Random circuits have already shown the presence of barren plateaus with increasing system size [36].

Another crucial element related to expressibility and trainability is the parametrization of the ansatz. Several studies have been done to investigate the effect of parametrization on the performance of the ansatz chosen [37]. Thus, making the right choice of parametrization and classical optimizers is crucial for an optimization algorithm to work. For example, if we have many local minima in the energy landscape, even with a good classical optimizer, we can get stuck to solutions that are not close to what we expect. Therefore, analysis of the energy landscape becomes important. If parameters scale with the system size, the energy landscape becomes high dimensional, which makes the analysis nontrivial.

PCA is a commonly utilized technique in data analysis, aimed at reducing the dimensionality of a dataset while preserving as much information as possible. Generally, this is done by transforming the dataset to a new space determined by principal components, through which a lower-dimensional hyperplane can be obtained such that the visualization of multidimensional data becomes easier. The computation of the principal components is done using the eigenvectors of the covariance matrix generated by the dataset.

Here, we employ PCA to reduce the dimensionality of the m-dimensional parameter space generated by DCQC ansatz, $\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_m)^T$. Using PCA, we identify the n $(n \leqslant m)$ number of principal components, which represent the highest variance among the parameters during the optimization process. The principal components are determined by the orthogonal eigenvectors $\vec{\mathcal{E}}_i$ of the covariance matrix $\mathcal{E}$ where the eigenvector corresponding to the largest eigenvalue gives the direction of maximum variance. The original dataset and the principal component space are related by this unitary projection matrix $\mathcal{E}$ such that

$$\boldsymbol{\gamma} = \mathcal{E} \cdot \boldsymbol{\theta}, \quad (5)$$

where $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \cdots, \gamma_m)$ represents the principal components and $\mathcal{E} = (\vec{\mathcal{E}}_1, \vec{\mathcal{E}}_2, \cdots, \vec{\mathcal{E}}_m)^T$ defines its direction. Each γ_i contains the information about the variance of ith principal component. A reduced n-dimensional principal component space can thus be achieved by truncating at n position. To obtain a more profound comprehension of optimization process using different optimizers for hybrid DCQC, we implement the first two principal components to demonstrate the cost landscape. Corresponding original parameters $\boldsymbol{\theta}$ can then be obtained using Eq. (5) and utilized to visualize the optimization trajectory on the cost landscape.

III. RESULTS

In this section, we benchmark hybrid DCQC ansatz with different parametrizations as well as different optimizers to

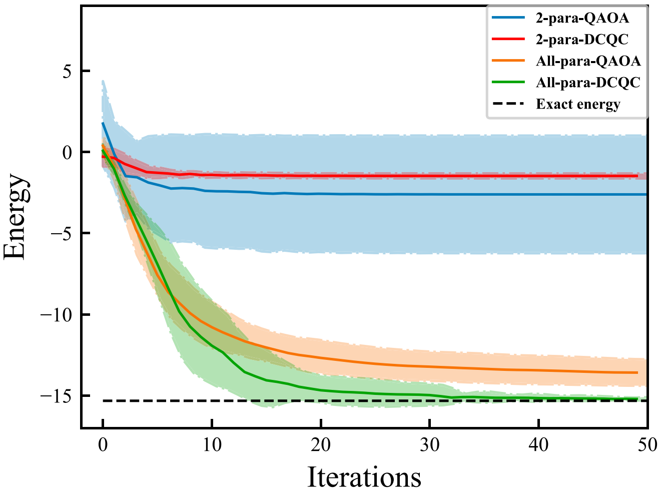

FIG. 1. Energy as a function of iterations for the $p = 1$ layer, $N = 10$ qubits. Results show simulator data for 50 iterations using the PS-Adam optimizer comparing QAOA and DCQC ansatz. Ten initial parameters are randomly chosen for each ansatz. The shaded area shows the standard deviation.

find the ground state of the Sherrington-Kirkpatrick (SK) model. SK model is a classical spin model that possesses all-to-all connectivity [38,39]. This model is particularly interesting because several optimization problems can be encoded into spin glass systems [40]. The Hamiltonian of the SK model is given by

$$H_c = -\sum_{i,j} J_{ij}\sigma_z^i\sigma_z^j, \tag{6}$$

where J_{ij} are coupling coefficients associated to spin i and spin j. We study the SK model with qubits ranging from $N = 14$ to $N = 28$ qubits with $J_{ij} \in \{-1, 1\}$ with mean 0 and variance 1. For this model, the favourable CD ansatz is $\mathcal{A} = \sum_i \sigma_y^i + \sum_{i,j} \sigma_y^i\sigma_z^j$.

In our investigation of the parametrization, we consider two types of ansatz: A two-parameter ansatz, where one parameter is associated with all the single-qubit gates, and the other with the two-qubit gates. The second one is a fully parameterized ansatz, wherein each gate contains its own independent free parameter, which amounts to a total of $N(N-1)/2 + N$ number of free parameters. In Fig. 1, we study the impact of this parametrization in both conventional QAOA and hybrid DCQC for $p = 1$, utilizing the PS-ADAM optimizer for $N = 10$ qubits. The two-parameter ansatz fails to reach the ground state where QAOA shows better minimization compared to the hybrid DCQC ansatz. However, DCQC shows a lower standard deviation (the shaded region), which means it quickly converges to particular minima. On the other hand, QAOA shows high standard deviation showing random behavior in the convergence process. Upon fully parametrizing the ansatz, DCQC shows superior performance in efficiently reaching the ground state. The standard deviation is also small for the final solution in DCQC proving its efficacy against gradient-based optimizers. Note that this fully parameterized version of QAOA is known as maQAOA [10], which has been shown to have a superior performance than QAOA. That being said, increasing the number of parameters per layer will result in an increase in expressibility. From this, we can conclude that the fully parameterized DCQC ansatz performs the best. Therefore, we select this ansatz and perform further analysis with various optimizers.

To compare the impact of different classical optimizers, we restrict ourselves to a system size of $N = 10$ qubits, initialize with ten random instances. Figure 2 illustrates the performance of all the optimizers in 1000 function evaluations. The number of function evaluations is defined by the total number of measurements performed to evaluate the cost function during the optimization process. This metric is most suitable for comparing the performances of both gradient-based and gradient-free optimizers. In the case of gradient-based optimizers, this also includes the number of measurements required to compute the gradient. For instance, PS requires two function evaluations to calculate gradients with respect to each parameter resulting in $(2m + 1)$ function evaluations for each iteration. Note that gradient calculation using PS is costly in our study as the number of function evaluations increases linearly with the number of parameters. From Fig. 2(a), we show the comparison of three optimizers where the gradients are computed using the PS rule. We observe that, among these PS-based optimizers, PS-Adam can approach ground-state energy faster than PS-SGD and PS-BFGS.

In general, SPSA-based optimization tends to perform better compared to PS as it requires only three function evaluations for each iteration regardless of the size of the parameter space. For gradient-based optimizers using SPSA, we observe that BFGS outperforms the other two with significant advantages, with fewer function evaluations. To make a thorough analysis, we have also compared two gradient-free optimizers in Fig. 2(c). Like SPSA, in COBYLA we also need three function evaluations for each iteration whereas Nelder-Mead requires one evaluation only. However, COBYLA converges more efficiently compared to Nelder-Mead. In Fig. 2(d), we compare the best optimizers from the three above-discussed groups where SPSA-BFGS comes out to be the best optimizer among all the optimizers studied in this paper. It is also important to note that this benchmarking heavily depends on the system size. Nonetheless, for smaller N, SPSA-BFGS demonstrates the best performance for DCQC. In addition, to comprehensively understand the process of reaching the minima, we count the number of iterations and the function evaluations needed for each optimizer to achieve 90% of the exact ground state energy. Figure 3 clearly depicts the SPSA-BFGS outperforms other optimizers in terms of both the number of iterations and the function evaluations.

In order to numerically analyze the scalability of DCQC in the NISQ regime, we conducted evaluations based on the mean of 10 randomly initialized parameters for a specific instance of the model, employing $p = 1$ layers of the ansatz and the SPSA-BFGS optimizer from $N = 10$ to $N = 28$ qubits. In Fig. 4, we plot the approximation ratio with respect to the number of qubits where the approximation ratio is defined as the $\langle Hc\rangle/E_g$. We observe that the performance of DCQC ansatz is better than QAOA considering the fact that the size of the parameter space is the same for both the ansatz. Figure 4 also shows that the performance of both ansatzes highly depends on the initial parameters chosen. This statement is

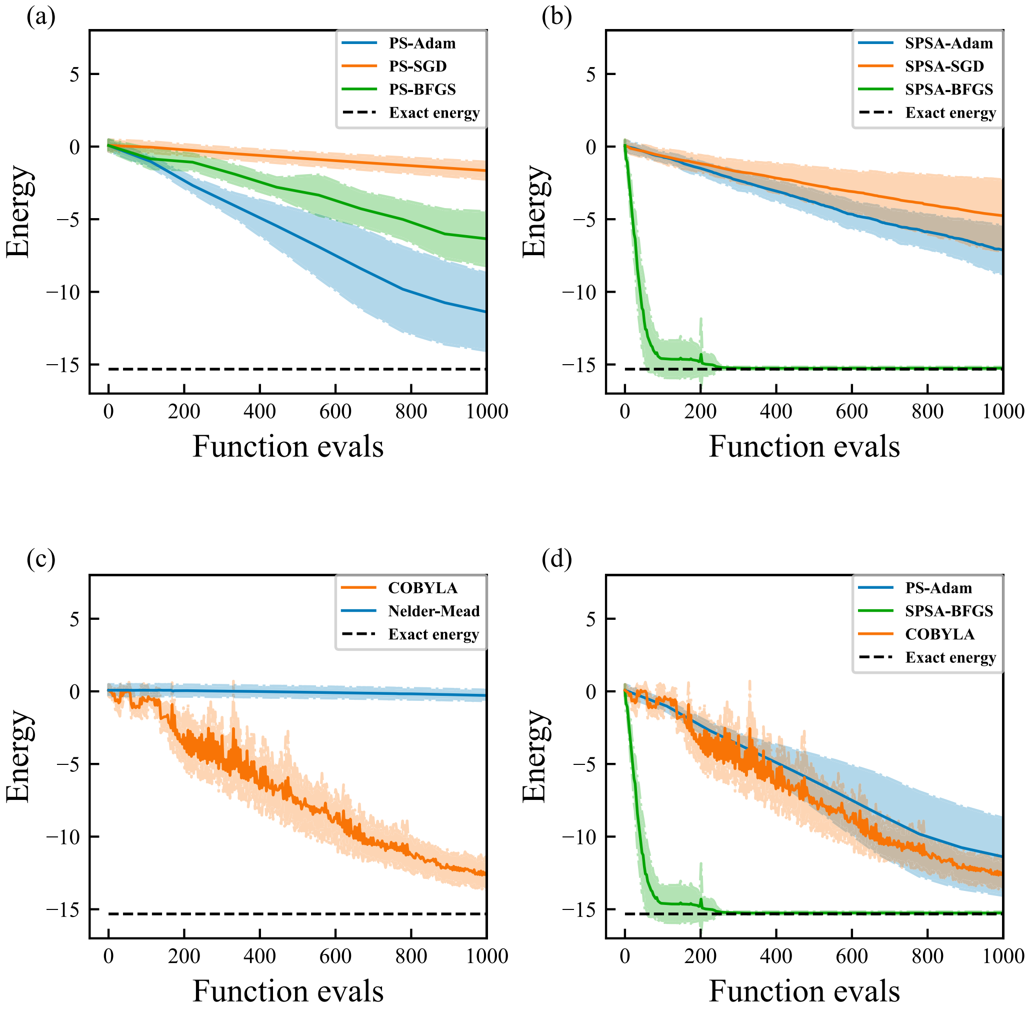

FIG. 2. Energy as a function of a number of function evaluations for (a) PS-based optimizers, (b) SPS-based optimizers, (c) gradient-free optimizers, and (d) the best ones from the previous group. Comparisons are made between different optimizers on $p = 1, N = 10$ qubits system using fully parameterized DCQC ansatz. Ten random initial parameters are chosen to be the same for each optimizer.

evident from the high error bars, signifying that when the initialization is not good, the optimization lands into local minima. Nevertheless, with appropriate initial parameters, we can find the exact ground state for systems up to $N = 28$. Note that even for an ansatz with $m = 406$, DCQC can reach the exact ground state proving the absence of issues like barren plateau in the cost landscape.

Next, to visualise the cost landscape and the optimization trajectory, we employ PCA, which is designed to reduce the dimensionality of the parameter space by identifying a hyperplane that effectively captures the properties of all the samples. As mentioned in the previous section, PCA enables us to obtain the variance for each dimension of the hyperplane, which serves as a measure of the sparsity in the samples. By leveraging PCA, we gain valuable insights into the high-dimensional data, allowing us to better understand and navigate through the complex cost landscape.

The first two principal components (x and y axes in Fig. 5) account for at least 70% of the total variance in our data, which permits us to effectively visualize and understand the primary trends and patterns in the cost function. This enables us to make informed decisions to refine our optimization strategies accordingly [41].

In Fig. 5, we start the optimization with the same initial points for all the optimizers (shown by the blue dots) and show the optimization trajectory that eventually reaches to the minimum (the red dots). The number of points represents the number of iterations required to obtain the global minimum. The observations of the cost landscape validate the results we obtained in Fig. 2. We can observe that the optimization requires less number of iterations when local minima exist in the cost landscape of the principal components. As depicted in Figs. 5(a) and 5(f), ADAM works best for the PS-based optimization whereas, among SPSA based optimizers, BFGS reaches the minima in the fewest possible steps. Furthermore, cost landscapes in Figs. 5(d) and 5(h) indicate the inability of SPSA-SGD and Nelder-Mead to reach a global optimum.

Considering the results presented earlier, we can confidently assert that SPSA based BFGS optimizer emerges as the most promising candidate for hybrid DCQC. Its effectiveness, combined with its ability to efficiently navigate through the cost landscape, makes it most suitable for the optimization

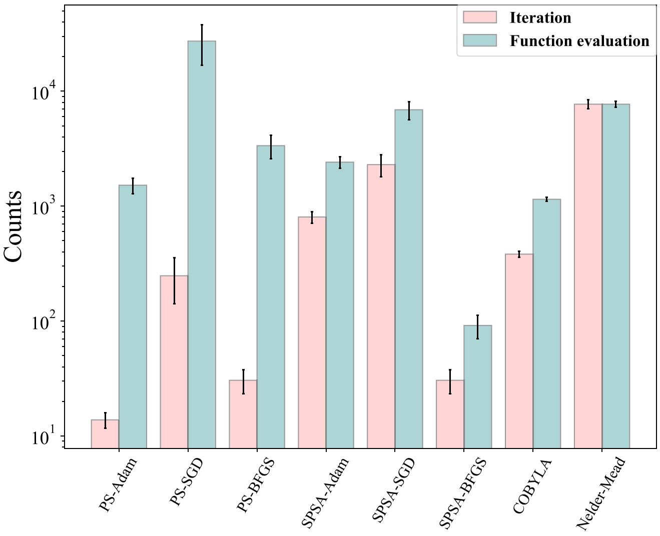

FIG. 3. Number of operations as a function of optimizer type for 10 instances of SK model when $p = 1$ layer, $N = 10$ qubits. The figure shows the counts (iterations, function evaluations) needed to reach the fidelity of 90% using SPSA-BFGS with fully parameterized DC ansatz. The gray lines at the tips of the bars represent the width of each standard error.

process. This conclusion underscores the importance of selecting appropriate classical optimizers to enhance the performance and practicality of the DCQC algorithm for various quantum computing applications.

IV. CONCLUSIONS

In this article, we have conducted an extensive study of a hybrid DCQC algorithm, benchmarking its performance and efficacy with respect to various classical optimizers. Our results suggest that gradient-based optimizers like ADAM and BFGS typically exhibit faster convergence in comparison to gradient-free optimizers such as COBYLA and Nelder-Mead as they need less function evaluations. Nonetheless, the method used for gradient calculation plays a pivotal role in distinguishing the most efficient optimizer. When the number of circuit evaluations is considered as the performance metric, SPSA-BFGS emerges as the best, demonstrating a distinct advantage over other optimizers, whether they are gradient based or gradient free. For a fully parameterized ansatz, DCQC with SPSA-BFGS provides a better approximation ratio compared to maQAOA even when it is scaled up to a system size $N = 28$, avoiding the occurrence of any trainability issues such as barren plateaus. Moreover, the study of the cost landscape yields valuable insights into the behavior of the optimization processes, particularly in relation to variations in the first two principal components obtained through PCA. However, one issue requires addressing. The assumption of equality between parameter shift and adjoint derivative is founded solely on analytic, and further validation is necessary as the library continues to be developed.

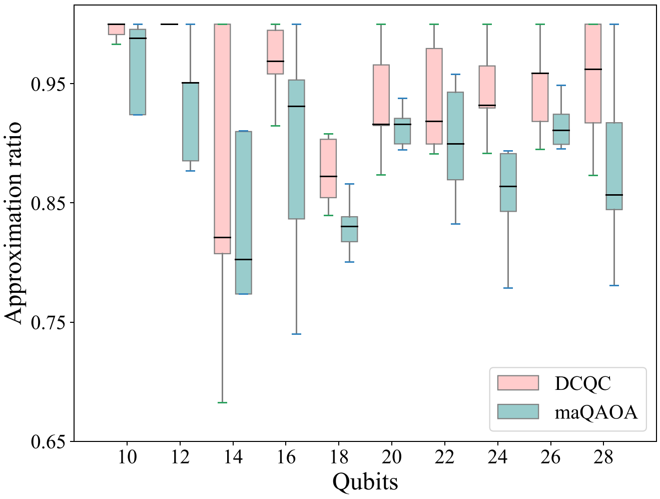

FIG. 4. Approximation ratio as a function of qubit number. The result represents $p = 1$ layered, $N = 10$ to $N = 28$ qubits ansatz. SPSA-BFGS optimizer is considered for both maQAOA and DCQC ansatz.

In summary, this study serves as a comprehensive benchmark for hybrid DCQC algorithms, specifically examining their performance in optimizing the parameter space and its correlation with the cost landscape. Through an analysis of various gradient-based and gradient-free optimizers, valuable insights were gained into their convergence rates, efficiency, and sensitivity to system size and initial parameters. Additionally, employing PCA to study the cost landscape shed light on the optimization trajectory, deepening our understanding of the process itself. Notably, our emphasis on scalability sets this research apart. By evaluating DCQC in a system with up to 28 qubits, we underscore the substantial potential and relevance of this approach in addressing larger-scale quantum computing challenges. This research establishes a basis for selecting classical optimizers in hybrid DCQC techniques, enhancing their potential applications across diverse domains to address complex problems.

ACKNOWLEDGMENTS

This work is supported by the Basque Government through Grant No. IT1470-22, the Project Grants No. PID2021-126273NB-I00, No. PID2021-123131NA-I00 funded by MCIN/AEI/10.13039/501100011033 and by “ERDF A way of making Europe” and “ERDF Invest in your Future”, NSFC (12075145), and STCSM (2019SHZDZX01-ZX04), the China Scholarship Council (CSC) under Grants No. 202206890003 and No. 202306890004. This project has also received support from the Spanish Ministry of Economic Affairs and Digital Transformation through the QUANTUM ENIA project call-Quantum Spain, and by the EU through the Recovery, Transformation and Resilience Plan-NextGenerationEU within the framework of the Digital Spain 2026 Agenda. X.C. acknowledges ayudas para contratos Ramón y Cajal-2015-2020 (RYC-2017-22482).

APPENDIX A: GRADIENT-BASED OPTIMIZERS

The optimizer processes the training by adjusting the model’s parameters iteratively based on the gradients of the cost function with respect to these parameters. The primary

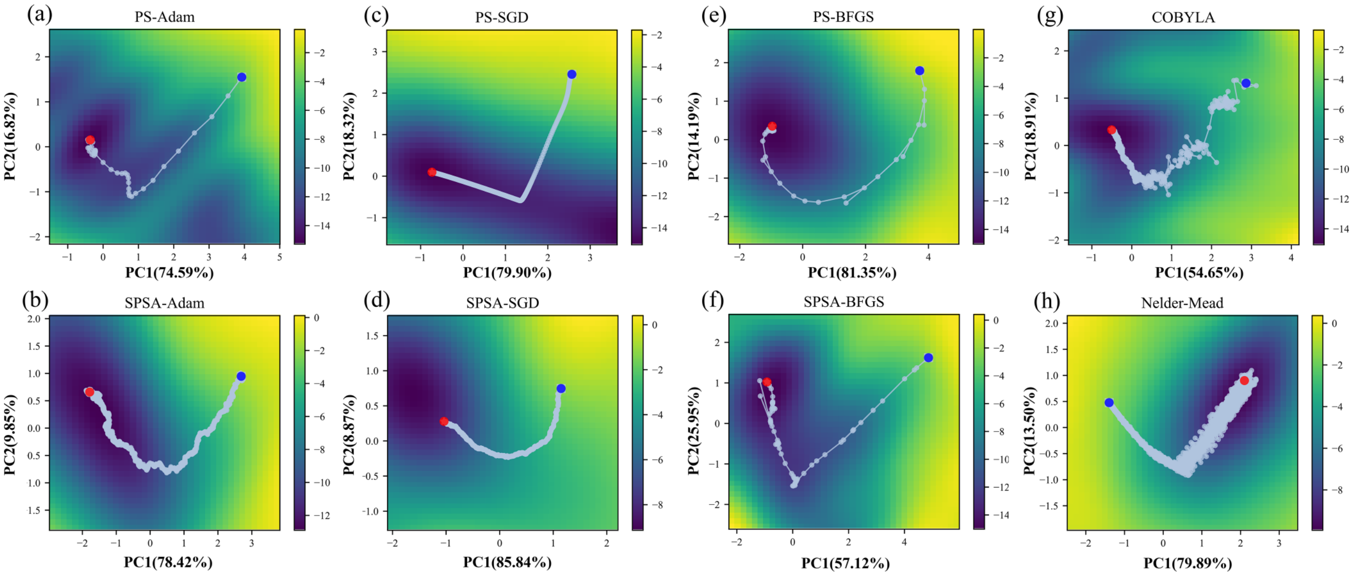

FIG. 5. Principal component (PC) analysis result on the cost landscape of $p = 1$, $N = 10$ qubits system. The figure shows how the optimization trajectory goes with different hybrid optimizers on the cost landscape. The x, y axes denote how much the first and second PC will affect the variance of the data. The red and blue dots represent the final and initial points respectively.

goal is to find the optimal set of parameters that lead to the best performance of the model on the given task [42]. Three gradient-based optimizers will be introduced in this section.

Stochastic gradient descent (SGD). Normally, its advantage lies in using small batches in each iteration rather than entire data sets. In our case, the parameter is fed one by one and updated according to the rule

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - a_k \cdot \nabla_k J(\boldsymbol{\theta}), \tag{A1}$$

where $a_k = \frac{a}{(A+k)^b}$, b, a, A are all scaling parameters to control the learning rate and k is the iteration number.

Broyden-Fletcher-Goldfarb-Shanno (BFGS). The BFGS algorithm belongs to a class of quasi-Newton optimization methods that extend Newton's method [43],

$$\nabla_k J(\boldsymbol{\theta}) = \nabla_{k+1} J(\boldsymbol{\theta}^{k+1}) + B_{k+1}(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+1}), \tag{A2}$$

where $B_{k+1}\Delta\theta^k = y^k$ is defined. The algorithm's update formula is given by imposing constraints on the inverse Hessian B, we can produce the update following the rule

$$B_{k+1} = B_k + \frac{y^k (y^k)^T}{(y^k)^T \Delta\boldsymbol{\theta}^k} - \frac{B_k \Delta\boldsymbol{\theta}^k (\Delta\boldsymbol{\theta}^k)^T B_k}{(\Delta\boldsymbol{\theta}^k)^T B_k \Delta\boldsymbol{\theta}^k}. \tag{A3}$$

The only requirement for updating the value is to know the previous gradient information.

Adaptive moment estimation (Adam). To work with the Adam optimizer, we define the first moment (the mean) m and the second moment (the variance) v of each gradient respectively [44],

$$m_k = a_1 m_{k-1} + (1 - a_1) g_k, \tag{A4}$$

$$v_k = a_2 v_{k-1} + (1 - a_2) g_k^2, \tag{A5}$$

with a_1, a_2 being the decay rates. Therefore, we have the momentums for each iteration

$$\hat{m_k} = \frac{m_k}{1 - a_1^k}, \tag{A6}$$

$$\hat{v_k} = \frac{v_k}{1 - a_2^k}. \tag{A7}$$

With these, we can write the updated formula for Adam,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\eta}{\sqrt{\hat{v_k}} + \epsilon} \hat{m_k}, \tag{A8}$$

where default values are $a_1 = 0.9$, $a_2 = 0.999$, $\epsilon = 10^{-8}$ in the neural network module of Mindquantum [45].

APPENDIX B: GRADIENT-FREE OPTIMIZERS

In the following sections, we will briefly introduce two methods that do not require calculating gradients.

Constrained optimization by linear approximation (COBYLA). This is a model-based local optimization method that builds a linear approximation of the objective function over a simplex. We use it through SciPy, which follows the principles below [46]:

(1) Start from $\rho = \rho_i, \boldsymbol{\theta_0} = \boldsymbol{\theta_i}$, the optimizer evaluates $J(\boldsymbol{\theta})$ then from an $\boldsymbol{\theta}$ such that $\boldsymbol{\theta} - \boldsymbol{\theta}_i \leqslant \rho$, calculates $J(\boldsymbol{\theta}_i)$.

(2) Compares its value with $J(\boldsymbol{\theta})$. If the new value $\boldsymbol{\theta}_i$ gives $J(\boldsymbol{\theta}_i) \leqslant J(\boldsymbol{\theta})$ then $\boldsymbol{\theta} = \boldsymbol{\theta}_i$ is the new starting point for the next iteration.

(3) The control distance ρ is decreased during iterations in order to evaluate the minimum of the function as accurately as possible.

(4) The optimization algorithm finishes when the control radius ρ becomes less than a predefined value ρ_{end} (fixed to 10^{-5}).

In other words, the initial input of the algorithm is a starting point θ_0, a control radius initial ρ_i, and a target radius ρ_{end} [47].

Nelder-Mead. For the Nelder-Mead optimizer, it compares the values of $J(\theta_i)$ at the n iteration and a simplex, trying to approximate the object function with the simplex. A simplex with size a is initialized at θ_0 and we have [48]

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_0 + p\vec{e}_i + \sum_{k=1,k\neq i}^{n} q\vec{e}_k, \tag{B1}$$

where $\vec{e}_i$, $\vec{e}_k$ are unit base vectors and are defined by

$$p = \frac{a}{n\sqrt{2}}(\sqrt{n+1} + n - 1), \tag{B2}$$

$$q = \frac{a}{n\sqrt{2}}(\sqrt{n+1} - 1). \tag{B3}$$

The algorithm carries out reflection, expansion, and contraction operations to update the cost function, with the aim of similizing the function value [43]. If the algorithm can terminate depends on the following criteria:

$$\sqrt{\sum_{i=1}^{n+1} \frac{(J(\boldsymbol{\theta}_i) - \overline{J(\boldsymbol{\theta})})^2}{n}} < \epsilon, \overline{J(\boldsymbol{\theta})} = \frac{1}{n+1}\sum_{i=1}^{n+1} J(\boldsymbol{\theta}_i), \tag{B4}$$

where ϵ is a infinitesimal positive scalar.

APPENDIX C: PARAMETER SHIFT AND ADJOINT DIFFERENTIATION

Given a quantum circuit with m unitary operators as $\{U_0(\theta_0), ..., U_i(\theta_i), ..., U_m(\theta_m)\}$, we calculate the expectation of an observable $\hat{O}$,

$$J(\theta) = \langle 0|\prod_{i=0}^{m} U_i^{\dagger}(\theta_i)\hat{O}\prod_{j=m}^{0} U_j(\theta_j)|0\rangle. \tag{C1}$$

We set the notions like

$$|\psi_i\rangle = \prod_{j=i}^{0} U_j(\theta_j)|0\rangle, \tag{C2}$$

$$B_i = \prod_{j=i}^{m} U_j^{\dagger}(\theta_j)\hat{O}\prod_{k=m}^{i} U_k(\theta_k), \tag{C3}$$

so that we can calculate the gradient of $J(\boldsymbol{\theta})$ with respect to parameter θ_i,

$$\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_i} = \frac{\partial}{\partial\theta_i}\langle\psi_{i-1}|U_i^{\dagger}(\theta_i)B_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle. \tag{C4}$$

Using adjoint differentiation method to calculate the gradient,

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_i} &= \langle\psi_{i-1}|\frac{\partial U_i^{\dagger}(\theta_i)}{\partial\theta_i}B_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle \\ &\quad + \langle\psi_{i-1}|U_i^{\dagger}(\theta_i)B_{i+1}\frac{\partial U_i(\theta_i)}{\partial\theta_i}|\psi_{i-1}\rangle \\ &= 2\mathrm{Re}\left(\langle\psi_{i-1}|U_i^{\dagger}(\theta_i)B_{i+1}\frac{\partial U_i(\theta_i)}{\partial\theta_i}|\psi_{i-1}\rangle\right) \\ &= 2\mathrm{Re}\left(\langle\varphi_{i-1}|\frac{\partial U_i(\theta_i)}{\partial\theta_i}|\psi_{i-1}\rangle\right),\end{aligned} \tag{C5}$$

where $\langle\varphi_{i-1}| = \langle\psi_{i-1}|U_i^{\dagger}(\theta_i)B_{i+1}$ is set for simplification, so that it is easy to calculate $\partial J(\boldsymbol{\theta})/\partial\theta_{i-1}$.

Using parameter shift method to calculate gradient, now we support $U(\boldsymbol{\theta}) = e^{-i\theta\hat{P}/2}$, where $\hat{P} = \otimes_{i=0}^{n}\sigma_i$, $\sigma_i \in \{I, X, Y, Z\}$, and the gradient will be

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_i} &= \langle\psi_{i-1}|\frac{\partial U_i^{\dagger}(\theta_i)}{\partial\theta_i}B_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle \\ &\quad + \langle\psi_{i-1}|U_i^{\dagger}(\theta_i)B_{i+1}\frac{\partial U_i(\theta_i)}{\partial\theta_i}|\psi_{i-1}\rangle \\ &= \frac{i}{2}[\langle\psi_{i-1}|U_i^{\dagger}(\theta_i)(\hat{P}_iB_{i+1} - B_{i+1}\hat{P}_i)U_i(\theta_i)|\psi_{i-1}\rangle] \\ &= \frac{i}{2}(\langle\psi_{i-1}|U_i^{\dagger}(\theta_i)[\hat{P}_i, B_{i+1}]U_i(\theta_i)|\psi_{i-1}\rangle).\end{aligned} \tag{C6}$$

According to [34],

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_i} &= \frac{1}{2}\Big[\langle\psi_{i-1}|U_i^{\dagger}(\theta_i)\Big(U_i^{\dagger}\Big(\frac{\pi}{2}\Big)B_{i+1}U_i\Big(\frac{\pi}{2}\Big) \\ &\quad - U_i^{\dagger}\Big(-\frac{\pi}{2}\Big)B_{i+1}U_i\Big(-\frac{\pi}{2}\Big)\Big)U_i(\theta_i)|\psi_{i-1}\rangle\Big] \\ &= \frac{1}{2}\Big[\langle\psi_{i-1}|U_i^{\dagger}\Big(\theta_i + \frac{\pi}{2}\Big)B_{i+1}U_i\Big(\theta_i + \frac{\pi}{2}\Big)|\psi_{i-1}\rangle \\ &\quad - \langle\psi_{i-1}|U_i^{\dagger}\Big(\theta_i - \frac{\pi}{2}\Big)B_{i+1}U_i\Big(\theta_i - \frac{\pi}{2}\Big)|\psi_{i-1}\rangle\Big] \\ &= \frac{1}{2}\Big[J\Big(\theta_i + \frac{\pi}{2}\Big) - J\Big(\theta_i - \frac{\pi}{2}\Big)\Big].\end{aligned} \tag{C7}$$

From the given evidence, we can infer that the gradient computed via both the adjoint differentiation and the parameter shift method coincides when no noise exists in the quantum circuit. These two methods employ distinct mathematical approaches to obtain the exact value of the gradient.

[1] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[2] J. I. Colless, V. V. Ramasesh, D. Dahlen, M. S. Blok, M. E. Kimchi-Schwartz, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi, Computation of molecular spectra on a quantum processor with an error-resilient algorithm, Phys. Rev. X **8**, 011021 (2018).

[3] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, An adaptive variational algorithm for exact molecular simulations on a quantum computer, Nat. Commun. **10**, 3007 (2019).

[4] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature (London) **549**, 242 (2017).

[5] J. Yao, L. Lin, and M. Bukov, Reinforcement learning for many-body ground-state preparation inspired by counterdiabatic driving, Phys. Rev. X **11**, 031070 (2021).

[6] H. Zhang, X. Xu, C. Zhang, M.-H. Yung, T. Huang, and Y. Liu, Variational quantum circuit learning of entanglement purification in multiple degrees of freedom, Phys. Rev. A **108**, 042611 (2023).

[7] E. Anschuetz, J. Olson, A. Aspuru-Guzik, and Y. Cao, Variational quantum factoring, in *Quantum Technology and Optimization Problems* (Springer, New York, 2019), pp. 74–85.

[8] 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).

[9] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028.

[10] R. Herrman, P. C. Lotshaw, J. Ostrowski, T. S. Humble, and G. Siopsis, Multi-angle quantum approximate optimization algorithm, Sci. Rep. **12**, 6781 (2022).

[11] L. Zhu, H. L. Tang, G. S. Barron, F. A. Calderon-Vargas, N. J. Mayhall, E. Barnes, and S. E. Economou, Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer, Phys. Rev. Res. **4**, 033029 (2022).

[12] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, Obstacles to variational quantum optimization from symmetry protection, Phys. Rev. Lett. **125**, 260505 (2020).

[13] E. Bae and S. Lee, Recursive QAOA outperforms the original QAOA for the MAX-CUT problem on complete graphs, arXiv:2211.15832.

[14] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, Algorithms **12**, 34 (2019).

[15] J. Wurtz and P. J. Love, Classically optimal variational quantum algorithms, IEEE Trans. Quantum Eng. **2**, 1 (2021).

[16] Y. Yu, C. Cao, C. Dewey, X.-B. Wang, N. Shannon, and R. Joynt, Quantum approximate optimization algorithm with adaptive bias fields, Phys. Rev. Res. **4**, 023249 (2022).

[17] O. Lockwood, An empirical review of optimization techniques for quantum variational circuits, arXiv:2202.01389.

[18] P. Chandarana, N. N. Hegade, K. Paul, F. Albarrán-Arriagada, E. Solano, A. del Campo, and X. Chen, Digitized-counterdiabatic quantum approximate optimization algorithm, Phys. Rev. Res. **4**, 013141 (2022).

[19] X. Chen, E. Torrontegui, and J. G. Muga, Lewis-Riesenfeld invariants and transitionless quantum driving, Phys. Rev. A **83**, 062116 (2011).

[20] E. Torrontegui, S. Ibáñez, S. Martínez-Garaot, M. Modugno, A. del Campo, D. Guéry-Odelin, A. Ruschhaupt, X. Chen, and J. G. Muga, Shortcuts to adiabaticity, Adv. At. Mol. Opt. Phys. **62**, 117 (2013).

[21] A. del Campo, Shortcuts to adiabaticity by counterdiabatic driving, Phys. Rev. Lett. **111**, 100502 (2013).

[22] N. N. Hegade, K. Paul, Y. Ding, M. Sanz, F. Albarrán-Arriagada, E. Solano, and X. Chen, Shortcuts to adiabaticity in digitized adiabatic quantum computing, Phys. Rev. Appl. **15**, 024038 (2021).

[23] N. N. Hegade, K. Paul, F. Albarrán-Arriagada, X. Chen, and E. Solano, Digitized adiabatic quantum factorization, Phys. Rev. A **104**, L050403 (2021).

[24] N. N. Hegade, P. Chandarana, K. Paul, X. Chen, F. Albarrán-Arriagada, and E. Solano, Portfolio optimization with digitized counterdiabatic quantum algorithms, Phys. Rev. Res. **4**, 043204 (2022).

[25] P. Chandarana, N. N. Hegade, I. Montalban, E. Solano, and X. Chen, Digitized counterdiabatic quantum algorithm for protein folding, Phys. Rev. Appl. **20**, 014024 (2023).

[26] P. W. Claeys, M. Pandey, D. Sels, and A. Polkovnikov, Floquet-engineering counterdiabatic protocols in quantum many-body systems, Phys. Rev. Lett. **123**, 090602 (2019).

[27] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).

[28] T. Hoffmann and D. Brown, Gradient estimation with constant scaling for hybrid quantum machine learning, arXiv:2211.13981.

[29] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[30] J. Duchi, E. Hazan, and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. **12**, 2121 (2011).

[31] M. J. D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in *Advances in Optimization and Numerical Analysis*, edited by S. Gomez and J. P. Hennart (Springer, New York, 1994), p. 51.

[32] J. A. Nelder and R. Mead, A simplex method for function minimization, Comput. J. **7**, 308 (1965).

[33] J. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Trans. Autom. Control **37**, 332 (1992).

[34] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, Phys. Rev. A **98**, 032309 (2018).

[35] J. Spall, Implementation of the simultaneous perturbation algorithm for stochastic optimization, IEEE Trans. Aerosp. Electron. Syst. **34**, 817 (1998).

[36] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. **9**, 4812 (2018).

[37] M. Larocca, N. Ju, D. García-Martín, P. J. Coles, and M. Cerezo, Theory of overparametrization in quantum neural networks, Nat. Comp. Sci. **3**, 542 (2023).

[38] K. Binder and A. P. Young, Spin glasses: Experimental facts, theoretical concepts, and open questions, Rev. Mod. Phys. **58**, 801 (1986).

[39] D. Sherrington and S. Kirkpatrick, Solvable model of a spin-glass, Phys. Rev. Lett. **35**, 1792 (1975).

[40] A. Lucas, Ising formulations of many NP problems, Front. Phys. **2**, 5 (2014).

[41] M. S. Rudolph, S. Sim, A. Raza, M. Stechly, J. R. McClean, E. R. Anschuetz, L. Serrano, and A. Perdomo-Ortiz, ORQVIZ: Visualizing high-dimensional landscapes in variational quantum algorithms, arXiv:2111.04695.

[42] M. Wiedmann, M. Hölle, M. Periyasamy, N. Meyer, C. Ufrecht, D. D. Scherer, A. Plinge, and C. Mutschler, An empirical comparison of optimizers for quantum machine learning with

SPSA-based gradients, in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE), Bellevue, WA* (Springer, New York, 2023), pp. 450–456.

[43] J. Nocedal and S. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering (Springer, New York, 2006).

[44] P. Chandarana, P. S. Vieites, N. N. Hegade, E. Solano, Y. Ban, and X. Chen, Meta-learning digitized-counterdiabatic quantum optimization, Quantum Sci. Technol. **8**, 045007 (2023).

[45] MindQuantum Developer, MindQuantum, version 0.6.0 (2021), https://gitee.com/mindspore/mindquantum.

[46] M. J. D. Powell, Direct search algorithms for optimization calculations, Acta Numerica **7**, 287 (1998).

[47] F. Taccone, C. Goeury, F. Zaoui, and A. Petralia, Pumping station design based on shape optimization process, in *2020 TELEMAC-MASCARET User Conference October 2020*, edited by W. A. and Breugem, L. Frederickx, T. Koutrouveli, K. Chu, R. Kulkarni, and B. Decrop (International Marine & Dredging Consultants (IMDC), Antwerp, 2020), pp. 91–98.

[48] M. A. Luersen and R. Le Riche, Globalized Nelder-Mead method for engineering optimization, Computers & Structures **82**, 2251 (2004).