
Universiteit
Leiden
The Netherlands

Performance comparison of optimization methods on variational quantum algorithms

Bonet Monroig, X.; Wang, H.; Vermetten, D.L.; Senjean, B.; MOUSSA, C.; Bäck, T.H.W.; ... ; O'Brien, T.E.

Citation
Bonet Monroig, X., Wang, H., Vermetten, D. L., Senjean, B., MOUSSA, C., Bäck, T. H. W., ... O'Brien, T. E. (2023). Performance comparison of optimization methods on variational quantum algorithms. *Physical Review A*, *107*(3). doi:10.1103/PhysRevA.107.032407

Version:	Publisher's Version
License:	Leiden University Non-exclusive license
Downloaded from:	https://hdl.handle.net/1887/3570441

Note: To cite this publication please use the final published version (if applicable).

Performance comparison of optimization methods on variational quantum algorithms

Xavier Bonet-Monroig,[1] Hao Wang,[2] Diederick Vermetten,[2] Bruno Senjean,[3,1] Charles Moussa,[2] Thomas Bäck,[2] Vedran Dunjko,[2] and Thomas E. O'Brien[4,1]

[1]*Instituut-Lorentz, Universiteit Leiden, 2300 RA Leiden, Netherlands*
[2]*Leiden Institute of Advanced Computer Science, Universiteit Leiden, 2333 CA Leiden, Netherlands*
[3]*ICGM, Université Montpellier, CNRS, ENSCM, 34090 Montpellier, France*
[4]*Google Quantum AI, 80636 Munich, Germany*

(Received 25 January 2023; accepted 8 February 2023; published 7 March 2023)

Variational quantum algorithms (VQAs) offer a promising path toward using near-term quantum hardware for applications in academic and industrial research. These algorithms aim to find approximate solutions to quantum problems by optimizing a parametrized quantum circuit using a classical optimization algorithm. A successful VQA requires fast and reliable classical optimization algorithms. Understanding and optimizing how off-the-shelf optimization methods perform in this context is important for the future of the field. In this work, we study the performance of four commonly used gradient-free optimization methods [sequential least-squares quadratic programming, constrained optimization by linear approximations, the covariance matrix adaptation evolutionary strategy (CMA-ES), and the simultaneous perturbation stochastic approximation (SPSA)] to find ground-state energies of a range of small chemistry and material science problems. We test a telescoping sampling scheme (where the accuracy of the cost-function estimate provided to the optimizer is increased as the optimization converges) for all methods, demonstrating mixed results across our range of optimizers and problems chosen. We further hyperparameter tune two of the four optimizers (CMA-ES and SPSA) across a large range of models and demonstrate that with appropriate hyperparameter tuning, CMA-ES is competitive with and sometimes outperforms SPSA (which is not observed in the absence of hyperparameter tuning). Finally, we investigate the ability of an optimizer to beat the "sampling-noise floor" given by the sampling noise of each cost-function estimate provided to the optimizer. Our results demonstrate the necessity for tailoring and hyperparameter tuning known optimization techniques for inherently noisy variational quantum algorithms and that the variational landscape that one finds in a VQA is highly problem and system dependent. This provides guidance for future implementations of these algorithms in experiments.

DOI: 10.1103/PhysRevA.107.032407

I. INTRODUCTION

Recently, we have witnessed an explosion of quantum computer prototypes accessible to researchers in academic and industrial laboratories. Existing quantum hardware has already demonstrated the ability to outperform classical computers in specific mathematically contrived tasks [1,2]. However, it is still unclear whether noisy intermediate-scale quantum (NISQ) [3] hardware can outperform classical computers on practically useful tasks. Here, variational quantum algorithms (VQAs) [4–6] were introduced as a means of preparing classically hard quantum states by tuning the parameters of a quantum circuit to optimize a cost function by utilizing a classical optimizer.

The overall performance of a VQA depends heavily on the performance of the classical optimization algorithm. Delving into the limitations of these optimization methods for different VQA tasks is critical for future research and industrial applications. In light of this aim, researchers have proposed new classical optimization algorithms that exploit periodic properties of parametrized quantum circuits [7,8]. Due to the hardness of optimizing VQAs some researchers have focused on using machine learning techniques to help optimize VQAs, e.g., using the so-called surrogate models [9,10], or on simpler optimizers such as the simultaneous perturbation stochastic approximation (SPSA) [11]. The latter performs surprisingly well in a VQA setting, given that it has relatively weak performance for classical benchmarks [12] compared to more complex evolutionary strategies [13,14]. These articles benchmark new optimization techniques relative to standard classical optimizers on a wide variety of systems. However, in our view, a need exists for further in-depth analysis of the most common optimization algorithms for VQAs. One reason is that the existing VQA performance studies consider testing classical optimizers using only default hyperparameters (parameters controlling for the optimizer itself, such as step size and population size). Such a consideration is improper for studying the performance limitation of classical optimizers since, in classical communities (e.g., Automated Machine Learning), it is well known that the optimal hyperparameter setting of any optimizer varies drastically across different cost functions, and so does its performance.

In this work, we conduct a systematic hyperparameter tuning of various classical optimization algorithms and compare their empirical performance for a range of problem sizes and at different sampling-noise strengths. We begin by extending the three-stage sampling strategy used in Ref. [11] to split the total budget of function evaluations to a suite of four

 ©2023 American Physical Society

optimizers. We compare the one- and three-stage samplings on three quantum systems, i.e., H_4 square, H_4 chain, and the 2×2 Hubbard model, and find that switching to a three-stage strategy yields mixed results, sometimes improving but sometimes significantly decreasing performance. We suggest that whether or not multiple stages should be used in optimization should be treated as a hyperparameter for a variational optimizer in future work.

Next, we focus on comparing hyperparameter-tuned SPSA with the state-of-the-art gradient-free optimizer, the covariance matrix adaptation evolutionary strategy (CMA-ES) [14], for a range of seven problems with varying qubit size. We find that after tuning the hyperparameters of both optimizers, they show very similar performance. This is surprising, as previous studies of noisy classical optimization test beds [12,13] suggest that SPSA is already significantly outperformed by CMA-ES for midsize problems. We thus conjecture that the quantum cost landscape might be substantially different from that of the classical ones, which is worth investigating in the future.

Finally, we investigate the effect that a noisy evaluation of the cost function can return a value below the true one, breaking the variational principle. However, the parameters of this evaluation provide a much worse approximation to the exact solution. We see that the native "favorite" solution of CMA-ES—a robust statistical estimate of the optimal variational parameter—yields the best result in all cases considered, over the lowest energy observed during the optimization trace, and that it is not statistically sound to apply standard error estimates to the latter.

II. BACKGROUND

A variational quantum algorithm attempts to find approximate ground states of an N-qubit quantum system as the output of a circuit $U(\vec{\theta})$ with tunable parameters $\vec{\theta}$. This generates a variational *Ansatz*,

$$|\Psi(\vec{\theta})\rangle = U(\vec{\theta})\,|\Phi\rangle\,, \tag{1}$$

where the parameters $\vec{\theta} \in [0, 2\pi]^d$ control the rotations of single- and two-qubit gates in a quantum circuit implementation of U applied to an initial state $|\Phi\rangle$ [i.e., $U(\vec{\theta}) = U_k(\theta_k)U_{k-1}(\theta_{k-1})\cdots U_0(\theta_0)\,|\Phi\rangle$]. During a VQA run, these parameters are tuned to optimize a cost function $\mathcal{C}(\vec{\theta})$, which in our case is the expectation value of a Hermitian observable $\hat{O}$ relative to the state $|\Psi(\vec{\theta})\rangle$,

$$\mathcal{C}(\vec{\theta}) = \langle\hat{O}\rangle = \langle\Psi(\vec{\theta})|\hat{O}|\Psi(\vec{\theta})\rangle. \tag{2}$$

To measure the expectation value of $\hat{O}$ without additional quantum circuitry, it is typical to write $\hat{O}$ as a linear combination of easy-to-measure operators, i.e., Pauli operators $\hat{P}_i \in \{\mathbb{I}, X, Y, Z\}^{\otimes N}$,

$$\hat{O} = \sum_i c_i\hat{P}_i \rightarrow \mathcal{C}(\vec{\theta}) = \langle\hat{O}\rangle = \sum_i c_i\langle\hat{P}_i\rangle. \tag{3}$$

A VQA then passes the estimation of the cost function $\mathcal{C}(\vec{\theta})$ to some classical optimization routine to find the values of $\vec{\theta}$ minimizing $\mathcal{C}$. This optimization loop and the optimizer choices are the focus of this work.

To get an estimate of the expectation value $\langle\hat{P}_i\rangle$, one prepares and measures the state multiple times in the $\hat{P}_i$ basis and calculates the mean of the eigenvalues observed. This approximates the cost function $\mathcal{C}$ by an estimator $\bar{\mathcal{C}}$, whose distribution is dependent on the number of repetitions M used to calculate $\langle\hat{P}_i\rangle$,

$$\bar{\mathcal{C}}(\vec{\theta}, M) = \sum_i c_i[\langle\hat{P}_i\rangle + \varepsilon_i(M)]. \tag{4}$$

Here, ε_i is a random variable drawn from a binomial distribution with variance $\sigma_i^2 \sim 1/M$ that is used to simulate the experimental shot or sampling noise. Assuming that Pauli operators are measured independently, the variance of the estimator $\bar{\mathcal{C}}$ may be propagated directly,

$$\mathrm{Var}[\bar{\mathcal{C}}] = \sum_i c_i^2\sigma_i^2. \tag{5}$$

In general, the assumption of independence is violated. One may measure mutually commuting operators in parallel [15–20]. Then the resulting measurement has nonzero covariance [21,22], which should be accounted for. However, this introduces only a constant factor to the estimation cost and will not significantly impact the relative optimizer performance. Here, we will use M, defined in Eq. (4), as the overall cost for the quantum subroutine which takes $\vec{\theta}$ and M as inputs and outputs $\bar{\mathcal{C}}(\vec{\theta}, M)$.

To optimize within a VQA, access to $\mathcal{C}(\vec{\theta})$ is provided to a classical optimizer, which then minimizes the sampled cost function $\bar{\mathcal{C}}(\vec{\theta}, M)$ as a function of the classical parameters $\vec{\theta}$. One can additionally provide estimates of gradients $\nabla_\theta\mathcal{C}$ (or higher-order derivatives) in order to perform gradient-based (or Newton-like) optimization. To avoid a comparison of the runtime of gradient estimation to that of estimating the raw cost function $\bar{\mathcal{C}}(\vec{\theta}, M)$, we compare only four gradient-free optimization algorithms. Moreover, it has been shown that gradient-based optimization strategies suffer given noisy function evaluations with simple noise structures [23] (e.g., stationary and isotropic noisy covariance) in the sense that (1) the convergence rate to local optima is hampered [24] and (2) such simple noise does not help in the escape from local optima [25]. We select the following optimizers (see Appendix A for further details):

(1) Sequential least-squares quadratic programming (SLSQP) determines a local search direction by solving the second-order local approximation of the cost function that satisfies the constraints.

(2) Constrained optimization by linear approximations (COBYLA) uses linear approximations of the target and constraint function to optimize a simplex within a trust region of the parameter space.

(3) CMA-ES is a population-based optimization algorithm in which the points are drawn from a multivariate Gaussian distribution, whose parameters (covariance matrix and location) are adapted online.

(4) SPSA employs a stochastic perturbation vector to compute an approximate gradient of the cost function simultaneously.

We compare these algorithms across multiple systems with different sizes and numbers of parameters considered.

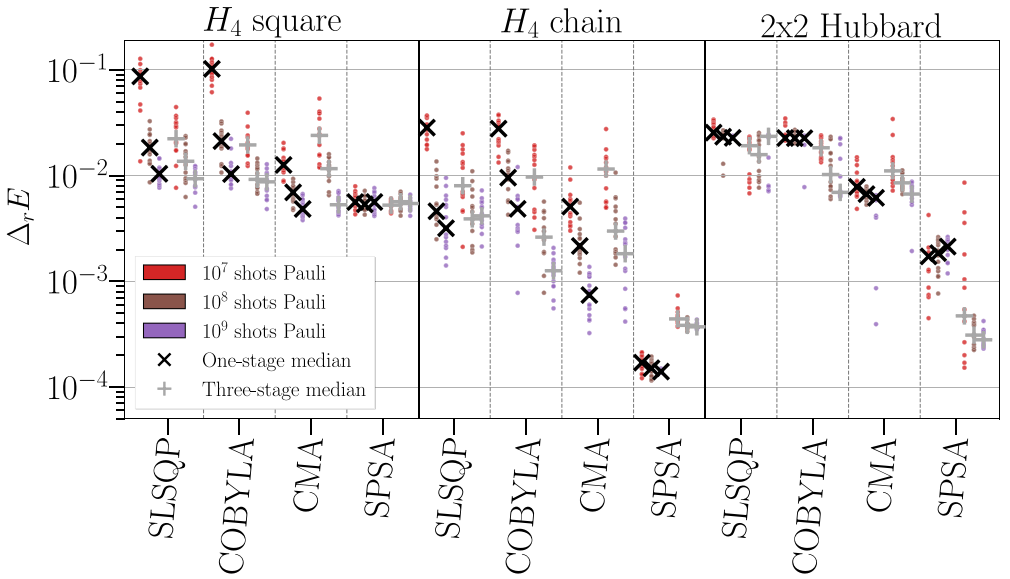

FIG. 1. Comparison of optimizers with default hyperparameters for the one-stage (left side) and three-stage (right side) methods. The final results of the 15 independent runs are shown by dots, with their spread giving an indication of the standard deviation of the sample. Red, brown, and purple indicate the number of total shots used for the optimization. Black crosses and gray pluses depict the median values of 15 independent runs for every problem for every number of shots used.

III. THREE-STAGE SAMPLING COMPARISON

Existing state-of-the-art quantum hardware is limited by the stability of the devices, which need to be tuned within timescales of hours up to a day. This imposes a hard limit on the total number of samples we can measure before the devices changes, of the order of $\sim 10^9$. This shot budget becomes the limiting factor in VQAs. One must carefully balance between exploring the parameter space and accurately measuring the cost function. In this spirit, Cade *et al.* [11] introduced a sampling procedure for SPSA that splits the total shot budget between three stages, resulting in improved performance of VQAs.

Naively, one can fix the total number of shots per Pauli operator and run the optimization until the budget is spent (i.e., using 1000 shots per Pauli operator per function call and allocating a total shot budget of 10^7 per Pauli operator would allow for a total of 10 000 evaluations). We refer to this approach as one-stage optimization. Alternatively, one could think of an optimization strategy in which the number of shots is increased as the optimization progresses toward better parameters, as introduced by Cade *et al.* We perform a three-stage optimization procedure in which the number of samples per Pauli operator increases per phase, reducing the number of total function calls. In our three-stage optimization, we fix the number of shots per Pauli operator for each stage (i.e., 100, 1000, and 10 000 shots for a total budget of 10^7). The number of function evaluations is then computed from a ratio of 10:3:1. For every 10 function calls in the first stage, we use three function calls in the second stage and one function call in the third stage (i.e., 7150, 2145, and 715 evaluations for a total budget of 10^7 shots per Pauli operator).

We compare the one- and three-stage protocols for the four optimization algorithms previously introduced, with the aimed to assess whether the three-stage protocol has any evident advantage over the standard sampling strategy. For this comparison we use the relative energy error,

$$\Delta_r E = \left| \frac{\mathcal{C}(\vec{\theta}_{\text{opt}}) - E_0}{E_0 - c_0} \right|, \tag{6}$$

where $\mathcal{C}(\vec{\theta}_{\text{opt}})$ is the noiseless cost function evaluated at the optimized parameters $\vec{\theta}_{\text{opt}}$ obtained from a noisy optimization. E_0 is the lowest eigenvalue of the problem. c_0 is the coefficient of the identity operator, which is the largest Hamiltonian term. This is the relevant figure of merit to capture the performance of the optimizer as it measures the relative error in estimating the traceless part of the Hamiltonian $H - c_0 I$, requiring a quantum computer. Our numerical experiments are performed under sampling noise with a total shot budget of 10^7 (red in Fig. 1), 10^8 (brown), and 10^9 (purple) per Pauli operator. In the one-stage method, we fix the total number of function evaluations to 10^4 and use 10^3, 10^4, and 10^5 shots per Pauli operator at each function call. In the three-stage procedure, the function calls are also fixed at 7150, 2145, and 715 for all budgets, and the shots per Pauli operator at every stage are 10^2, 10^3, and 10^4; 10^3, 10^4, and 10^5; and 10^4, 10^5, and 10^6, respectively. The optimization stops when the shot budget is reached. However, the SLSQP and COBYLA optimization algorithms have a termination criterion that, in most cases, results in exiting early and not utilizing their full shot budget.

We benchmark the algorithms for three different systems on eight qubits: H_4 in chain and square configurations and the 2×2 Hubbard model. Throughout this work, we use the unitary coupled-cluster (UCC) variational *Ansatz* for molecules and the variational Hamiltonian *Ansatz* (VHA) proposed in [11] for Hubbard models. Details of the *Ansatz* construction can be found in Appendix D. Our results are shown in Fig. 1: for each problem and each optimizer, we show the one-stage (left side, with its median indicated with a black cross) and

three-stage (right side, with its median indicated in a gray plus) variants side by side.

We find that whether a three-stage optimization is an improvement over a one-stage optimization is highly problem and optimizer dependent and no trends are statistically significant. For no single problem is there a consistent gain or loss in median performance across all optimizers. Moreover, we observe that only the three-stage COBYLA algorithm reports a consistent improvement over its one-stage counterpart across all problems. Given the small number of problems, this is more likely than not to be a fluke. However, we do see individual instances where going from one-stage to three-stage optimization can cause a significant (5 times) performance increase or decrease. As such, we suggest that although the multistage idea of Ref. [11] (originally proposed for SPSA and to solve different Hubbard models) can be extended to other problems, it should be treated as a problem-dependent hyperparameter that needs to be tuned for optimal performance.

Beyond the comparison of one-stage and three-stage optimization strategies, we can make an initial performance comparison between different optimizers in Fig. 1. We observe that in two out of three problems, SPSA obtains a significantly smaller error than all other optimizers, and in the third it converges faster (although CMA-ES at 10^9 shots reports a similar performance). This is at odds with the literature, where CMA-ES in particular is observed to significantly outperform SPSA (see, for example, a comparison between Refs. [12,13]). We conjecture that this is due to the default choice of hyperparameters in our experiments for Fig. 1; optimizing these hyperparameters is known to yield a significant increase in performance [26,27]. We hypothesize that, following proper hyperparameter tuning of both methods, the performance of CMA-ES should significantly improve over that of SPSA.

IV. HYPERPARAMETER TUNING

Most optimization algorithms require a choice of hyperparameters that detail the optimization. Default hyperparameters are either derived under idealized theoretical assumptions or evaluated from numerical experiments using standard benchmarks. It is common to tune the hyperparameter of the optimizers when it is used in a function that has not been previously studied [26,27]. Clearly, this procedure should be performed similarly for a VQA.

Finding optimal hyperparameters of an optimizer can be costly and generally problem dependent. A subfield of classical optimization has been devoted to automating such hyperparameter tuning. Here, we use the iterated racing for automatic algorithm configuration [28] (IRACE; see Appendix C for a description of the procedure) to tune the SPSA and CMA-ES settings for four molecular systems, H_4 square and chain and H_2O at equilibrium and stretched geometries (corresponding to weakly and strongly correlated regimes, respectively). Additionally, we perform hyperparameter tuning of CMA-ES for the Hubbard model for three different configurations: 1×6, 2×2, and 2×3. For SPSA, however, we take the results of Ref. [11], where the hyperparameters were tuned. The hyperparameters used for the numerical experiments can be found in Tables I and II in Appendix B.

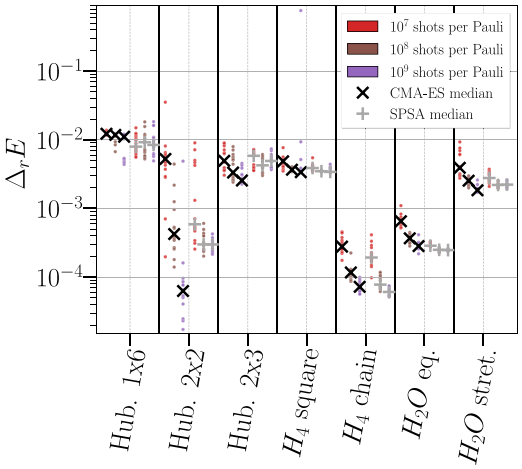

FIG. 2. Convergence of hyperparameter-tuned CMA-ES (black crosses) and SPSA (gray pluses) optimizers across a range of problems and accuracy regimes. Dots behind each marker show the 15 independent runs yielding the observed median. Each optimizer has its hyperparameters tuned separately for each problem (see Appendix C), treating the one- or three-stage strategy choice as a discrete hyperparameter.

With tuned hyperparameters, we repeat the previous experiments and study a set of additional problems: H_2O in equilibrium and stretched geometries (10 qubits) and Hubbard models on 1×6 and 2×3 lattices (12 qubits). We see (Fig. 2) that hyperparameter tuning yields significant improvement of the performance of CMA-ES across all problems. By contrast, SPSA is only moderately boosted after hyperparameter tuning. However, the resulting performances of SPSA and CMA-ES are comparable at 10^9 total shots. The only significant difference is a median fivefold performance gain from CMA-ES over SPSA for the 2×2 Hubbard model. We observe minor advantages from CMA-ES at 10^9 shots for the 2×3 Hubbard and H_2O in a stretched geometry and minor gains from SPSA for the 1×6 Hubbard model, H_4 chain, and H_2O in an equilibrium geometry, but here, the deviation lies well within the range of observed final convergence. This invalidates our hypothesis because, although CMA-ES no longer underperforms compared to SPSA, it does not significantly outperform it either, in contrast to what has been observed in other classical benchmarks [12,13]. We further do not observe any clear trend with the number of parameters or number of qubits in the system, although it would be expected to impact the performance of SPSA, as the gradient approximation of SPSA grows worse with the number of parameters. We suggest that this may be due to our problem sizes being insufficiently large. Nevertheless, we speculate that the slight overall performance increase for CMA-ES might be an indication that it might be a preferable choice for generic VQA optimization, assuming high accuracy is required and hyperparameter tuning is affordable.

In addition to the above, we observe that CMA-ES both converges slower than SPSA and typically has a wider spread in final energy error. The increased spread matches previously

reported empirical studies on classical benchmarks [29] and can be explained as CMA-ES sampling points from a larger region than SPSA. This makes it possible to escape local minima, but this escape cannot be guaranteed, which explains the spread in performance. The convergence of CMA-ES is also hindered by the need to build a large covariance matrix, which can be observed in call-by-call optimization traces (see Appendix F). This implies that it is not immediately possible to gain from "warm-starting" CMA using a good guess of initial parameters (from, e.g., a low-cost SPSA run), as we would still need to build the covariance matrix at this point. We suggest that when sufficiently low accuracy is needed, SPSA may always be preferable to more complex methods in the VQA setting.

V. THE SAMPLING-NOISE FLOOR

A successful VQA requires the optimization algorithm to return the optimal parameters of $\bar{\mathcal{C}}$. It is common to take the best measured $\bar{\mathcal{C}}$ as the optimal candidate for the optimization task.

However, in VQAs, the optimization is performed using a proxy cost function $\bar{\mathcal{C}}$, a sampled version of the real objective $\mathcal{C}$ with variance $\mathrm{Var}[\bar{\mathcal{C}}]$. This may cause us to misidentify a global minimum: given two points $\vec{\theta}_1$ and $\vec{\theta}_2$ such that $\mathcal{C}(\vec{\theta}_2) > \mathcal{C}(\vec{\theta}_1) > \mathcal{C}(\vec{\theta}_2) - a\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}$, we have a chance $\sim \exp(-a)$ to observe $\bar{\mathcal{C}}(\vec{\theta}_1) > \bar{\mathcal{C}}(\vec{\theta}_2)$. If we sample repeatedly around the global minimum as we converge and take the best result, this causes two problems. First, by taking the minimum over so many data points, we sample from a very skewed distribution and can report results with errors far beyond what one might expect from standard 1σ or 2σ confidence intervals. Second, the corresponding value of $\vec{\theta}$ might not be a good choice for approximating the ground state of the problem. We have a high probability of declaring $\vec{\theta}$ the "best result" even when given access to a noisy estimation $\mathcal{C}(\vec{\theta}_g)$ of the cost function at its true minimum as long as $\mathcal{C}(\vec{\theta}) - \mathcal{C}(\vec{\theta}_g) \lesssim \sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}$. This leads us to hypothesize that the values $\vec{\theta}_{\mathrm{best}}$ that give the best observed estimates of $\bar{\mathcal{C}}$ will have $\mathcal{C}(\vec{\theta}_{\mathrm{best}}) \gtrsim \mathcal{C}(\vec{\theta}_{\mathrm{min}}) + \sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}$. We call $\mathcal{C}(\vec{\theta}_{\mathrm{min}}) + \sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}$ the "sampling-noise floor."

CMA-ES has been designed not to rely on the best function evaluation. In particular, CMA-ES returns two different candidates: the best measured and a so-called favorite. The favorite candidate is a statistical estimator of $\vec{\theta}_g$ constructed at the end of the optimization process using all previous measurements. In principle, such an estimator can average out the sampling noise over the optimization landscape and beat the sampling-noise floor. We note that $\bar{\mathcal{C}}(\vec{\theta}_g)$ is never computed during optimization, so it makes no sense to compare noisy cost-function estimates here.

In Fig. 3, we compare the noisy and noiseless evaluations of our cost function at θ_{best} to a noiseless evaluation of our cost function at θ_{fav} for the optimization traces taken using 10^7 shots in Fig. 2. For each estimated energy we compute the relative energy error $\Delta E^{(\mathrm{var})} = \frac{\mathcal{C}(\vec{\theta}_{\mathrm{opt}}) - \mathcal{C}(\vec{\theta}_{\mathrm{min}})}{|\mathcal{C}(\vec{\theta}_{\mathrm{min}}) - c_0|}$, where $\vec{\theta}_{\mathrm{opt}} = \vec{\theta}_{\mathrm{best}}, \vec{\theta}_{\mathrm{fav}}$ and $\mathcal{C}(\vec{\theta}_{\mathrm{min}})$ is the best parameters found within the variational *Ansatz* from 15 traces using 10^9 shots. (We take this as a proxy for the true variational minimum.) We observe that the best function evaluation (orange points) is typically below the variational minimum (or our proxy thereof) and often below 0 (solid horizontal line), breaking the variational principle. This error is much larger than the standard deviation $\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}$ (gray shaded region) would suggest, which implies this is no longer a good error bar for $\bar{\mathcal{C}}(\vec{\theta}_{\mathrm{best}})$, as predicted. We further see that, as expected, the true cost-function estimates at θ_{best} lie around the sampling-noise floor. There are some notable exceptions, namely, for the H_4 square, H_4 chain, and Hubbard 2×3 square. In all of these cases, the median error is no less than $\sqrt{\mathrm{Var}[\bar{\mathcal{C}}]}/2$. We can explain this by inspecting Fig. 2: in these cases, the optimizer converges quickly (evidenced by the lack of gain beyond 10^7 samples), and most function calls are made very close to the true minimum. We finally observe that in all cases, the median value of $\mathcal{C}(\vec{\theta}_{\mathrm{fav}})$ lies below the median value of $\mathcal{C}(\vec{\theta}_{\mathrm{best}})$. Indeed, for certain problems, the difference is significant, and in some cases (H_4 chain), the gain over the sampling-noise floor can be up to tenfold. This suggests that generating a statistical estimator for θ_{fav} instead of taking the best value observed is a necessity for an accurate result of any VQAs.

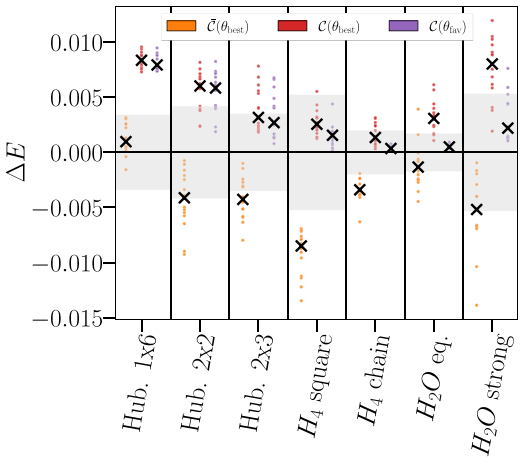

FIG. 3. Best versus favorite candidate from CMA-ES under noisy optimization. The optimization uses 10^7 shots per Pauli operator over the course of the experiment with individual estimations of $\bar{\mathcal{C}}(\vec{\theta})$ made using only 10^4 shots per Pauli operator. From left to right, the best measured function evaluation (orange), the best candidate evaluated without noise (red), and the favorite candidate evaluated without noise (purple). The gray area shows the sampling-noise floor for each problem, computed from the best measured optimization candidate of the 10^9-shot scenario. The black crosses indicate the sample median of the metric value ΔE of 15 independent runs of each pair of algorithm and scenario.

VI. CONCLUSION

Variational quantum algorithms have recently prompted significant interest as candidates amenable to near-term hardware. However, the performance of these quantum algorithms relies on a classical optimization of a difficult cost function. This task is, in general, intractable to solve optimally. It is, therefore, important to benchmark the available optimizers for

this purpose. We studied the performance of four optimization algorithms under the effect of sampling noise for the task of finding ground-state energies for a range of quantum simulation problems. We implemented the three-stage sampling method of Ref. [11] in all four optimizers and found that the resulting performance improvement is highly problem and optimizer dependent. This suggests such an adaptation should be treated as a discrete hyperparameter for a VQA optimizer going forward. We next performed a detailed hyperparameter tuning exercise for the SPSA and CMA-ES optimizers for a larger range of seven quantum simulation problems. Hyperparameter tuning was found to significantly improve CMA-ES, but only to the point where we observed a slight performance improvement over hyperparameter-tuned SPSA. This suggests CMA-ES would be a preferable optimizer in a high-cost, high-accuracy setting, and SPSA may be preferable otherwise. We finally demonstrated that using the parameter set that produces the best cost estimate during an optimization trace is a statistically unsound procedure for a VQA with sampling noise. The error in such an estimate can be further below the VQA minimum than error bars would suggest, and a statistical parameter estimate provided by optimizers such as CMA-ES performs better in terms of the evaluation of the noiseless cost function. Indeed, the energy error from this statistical parameter estimate can be up to 10 times lower than the energy error from any single function call (which we term the sampling-noise floor). Extending this study to determine how the performance of CMA-ES relative to SPSA varies as one shifts to larger system sizes is a clear target for future work.

The code and data to reproduce the figures in this work can be found on Zenodo [30].

ACKNOWLEDGMENTS

The authors would like to thank C. Beenakker for the support on this project and J. Tura, Y. Herasymenko, S. Polla, A. Dutkiewicz, S. Yalouz, and E. Scerri for fruitful discussions. C.M., T.B., and V.D. acknowledge support from TotalEnergies. V.D. acknowledges the support of SURF through the QC4QC project. This work was in part supported by the Dutch Research Council (NWO/OCW) as part of the Quantum Software Consortium program (Project No. 024.003.037).

APPENDIX A: DETAILS OF OPTIMIZATION ALGORITHMS

In this Appendix, we provide a more detailed description of the optimization algorithms used in this work.

(1) Simultaneous perturbation stochastic approximation (SPSA) algorithm [31,32] is designed for noisy evaluations of a cost function, where a stochastic perturbation vector (for instance, a vector whose components are independently sampled from the Rademacher distribution) is used to simultaneously estimate all partial derivatives at given a point. Compared to the well-known finite-difference method to estimate the gradient, which requires $2d$ evaluations of the cost function defined over $\mathbb{R}^d$, the stochastic approximation always consumes two evaluations, hence saving many function evaluations when the search dimension is high. However, this algorithm does not follow exactly the gradient direction due to the use of stochastic perturbation.

(2) Constrained optimization by linear approximations (COBYLA) [33] is designed for constrained derivative-free optimization. It employs linear approximations to the objective and constraint functions via a linear interpolation given $M+1$ points (or simplex). These approximations are then optimized within a trust region at each step.

(3) Sequential least-squares quadratic programming (SLSQP) [34,35] is an implementation of the more general sequential quadratic programming (SQP) approach [36] for solving constrained optimization problems.[1] Loosely speaking, in each iteration, SQP proposes a local search direction by solving a subproblem defined at the current search point in which the nonlinear cost function is replaced by its local second-order approximation, and the constraints are approximated by their affine approximation. When there is no constraint, this method degenerates to Newton's method.

(4) The covariance matrix adaptation evolutionary strategy (CMA-ES) [14] is the state-of-the-art direct search algorithm for the continuous black-box optimization problem, which distinguishes itself from other algorithms in the self-adaptation of its internal variables to the energy landscape. Briefly, this algorithm iteratively draws a number of candidate solutions from a multivariate Gaussian distribution, in which the shape of this distribution (e.g., covariance matrix and location) is adapted online based on the evaluated points in its trajectory.

APPENDIX B: NUMERICAL EXPERIMENTS

In this Appendix, we describe the numerical experiments used to generate the data for the figures.

To generate the target problems, we use the open-source electronic structure package OPENFERMION [37]. In addition, we generate the molecular systems with the computational chemistry software PSI4 through the OPENFERMION plug-in. The classical numerical simulations are performed using the open-source quantum circuit simulator package CIRQ [38]. Regarding the optimization methods, we use the SCIPY [39] software for COBYLA and SLSQP, PYCMA [40] for CMA-ES, and an in-house version of SPSA based on the code in [41].

As described in the main text, we focus on the performance of the optimization methods for VQAs under sampling noise. In order to include the sampling noise, in our experiments, we compute a noisy expectation value for every Pauli operator in the Hamiltonian with a fixed number of shots as follows:

(1) Prepare the ideal quantum state and measure $\langle P_i\rangle$ and $p=\frac{1-\langle P_i\rangle}{2}$.

(2) Sample $\tilde{p}=\mathcal{B}(p,M)$ from a binomial distribution with M shots.

(3) Compute the noise expectation value $\langle\tilde{P}_i\rangle=1-2\tilde{p}$.

(4) Calculate the noisy Hamiltonian expectation value as $\langle\tilde{H}\rangle=\sum_i c_i\langle\tilde{P}_i\rangle$.

[1] We took the implementation from the SCIPY package, which is based on the original software as described in [34].

TABLE I. The configurations of the atoms for the two water molecule problems used in this work.

H_2O	Equilibrium	Stretched
O	(0.0, 0.0, 0.1173)	(0.0, 0.0, 0.0)
H	(0.0, 0.7572, −0.4692)	(0.0, 1.8186, 1.4081)
H	(0.0, −0.7572, −0.4692)	(0.0, −1.8186, 1.4081)

This is a good approximation to the sampling noise generated by measuring the expectation values of Pauli operators in real hardware when the number of shots is large enough. Moreover, we avoid the bottleneck of preparing and measuring the same state multiple times.

In the Fermi-Hubbard model experiments (see Appendix E for further details), we set the parameters of the Hamiltonian to $t = 1.0$ and $U = 2.0$. The *Ansatz* circuit for these problems is constructed using the VHA with five, two, and four layers for the 1×6, 2×2, and 2×3 Hubbard models, respectively. These are the minimum numbers of layers needed to achieve a ground-state fidelity of 0.99 in Ref. [11].

For H_4 in the chain configuration, the first hydrogen atom is located at 0.0 in all coordinates; then every atom is separated in the x direction by 1.5 Å. In the square configuration, we fix the hydrogen atoms in two dimensions. The positions of the atoms are parametrized by their polar coordinates with $R = 1.5$ Å and $\theta = \frac{\pi}{4}$, and we locate them at $(x, y, 0)$, $(x, -y, 0)$, $(-x, y, 0)$, $(-x, -y, 0)$, with $x = R\cos(\theta)$ and $y = R\sin(\theta)$. For the water molecule problems, the (x, y, z) coordinates of the atoms are given in Table I. Additionally, in both of the problems, we reduce the active space by freezing the lowest two lowest orbitals, thus reducing the problem from 14 qubits to 10 qubits (or from seven to five spin orbitals). As a trial state to approximate the ground state of the molecular systems, we use the so-called unitary coupled-cluster *Ansätze*. A detailed description of how we construct the UCC *Ansätze* can be found in Appendix D.

Finally, the total number of parameters to be optimized for each target problem is given in Table II.

APPENDIX C: OPTIMIZATION ALGORITHMS HYPERPARAMETERS

Prior to applying the aforementioned optimizers on VQAs, we also optimize the hyperparameters of those optimizers.

TABLE II. Number of parameters of the *Ansätze* for each target problem.

System	No. of parameters
H_4 chain	14
H_4 square	10
H_2O equilibrium	26
H_2O stretched	26
Hubbard 1×6	15
Hubbard 2×2	6
Hubbard 2×3	16

TABLE III. List of values for SPSA hyperparameters used in Fig. 1 (default) and Fig. 2 after tuning using IRACE. We perform hyperparameter optimization with only H_4 chain and H_2O equilibrium and use the same values for the respective square and stretched configurations. For the Hubbard models, we take the default values, as Ref. [11] suggests their optimality.

	SPSA hyperparameter			
	a	α	c	γ
Default	0.15	0.602	0.2	0.101
H_4 chain	1.556	0.809	0.106	0.097
H_4 square	0.867	0.593	0.133	0.113
H_2O equilibrium	0.103	0.878	0.149	0.131
H_2O stretched	0.660	0.743	0.253	0.108
Hubbard	0.15	0.602	0.2	0.101

Such an extra tuning task aims at bringing up the performance of each optimizer to the maximum, hence facilitating a fair comparison of each problem. To achieve this task efficiently, we utilize the well-known IRACE algorithm for the hyperparameter tuning. IRACE has been extensively applied in automated machine learning research for configuring machine learning models and optimizers [42,43].

Built upon a so-called iterated racing procedure, this algorithm employs a statistical test (usually the Wilcoxon ranked-sum test) to obtain a robust (with respect to the sampling noise in measured energy values) ranking of hyperparameter settings, thereby serving as a suitable choice for our task. The tuning process with IRACE starts with fixing the subset of optimizer hyperparameters to be modified, including bounds and potential constraints. Then, a "race" between randomly sampled values begins. These configurations are evaluated a fixed number of times, and the less favorable configurations are disregarded based on a statistical test. The configurations that survived are then raced again until the evaluation budget is depleted or the number of configurations is below a threshold. Next, IRACE updates the candidate generation model based on the survival configurations and generates a set of new configurations to race against the elites. The racing procedure is repeated until the total budget is depleted. The surviving configurations are returned as the optimal configurations of the algorithm. The final hyperparameters used for the experiments and the average of the survivors are shown in Tables III and IV.

In detail, the hyperparameters we tuned are as follows:

(1) For SPSA, we use the following ranges for each hyperparameter: $a \in [0.01, 2]$, $\alpha \in [0, 1]$, $c \in [0.01, 2]$, and $\gamma \in [0, 1/6]$.

(2) For CMA-ES, we use the following ones: population size $\in [30, 130]$, c mean $\in [0, 1]$, $\mu \in [0, 0.5]$, damping factor $\in [0, 1]$, and $\sigma_0 \in [0.25, 1.1]$.

To run the IRACE algorithm, we allocated 500 evaluations of the hyperparameters as the total budget, as well as a maximum total running time of 7 days, and used two evaluations of each hyperparameter at the beginning of each race. Also, we used the F test to eliminate worse configurations in the racing procedure. The finally suggested configurations in Tables III

TABLE IV. List of values for CMA-ES hyperparameters used in Fig. 1 (default) and Fig. 2 after tuning using IRACE. Here, m indicates the number of free parameters of the *Ansatz* in each problem.

	CMA-ES hyperparameter[a]				
	σ_0	Population	μ	c mean	Damping factor
Default	0.15	$\lceil 4+3\log(m)\rceil$	0.5	1.0	1.0
H_4 chain	0.20	149	0.383	0.293	0.665
H_4 square	0.309	99	0.409	0.561	0.852
H_2O equilibrium	0.344	99	0.460	0.192	0.770
H_2O stretched	0.310	104	0.380	0.802	0.819
Hubbard 1×6	0.9131	51	0.3814	0.3614	0.6006
Hubbard 2×2	0.8561	113	0.2741	0.6317	0.6771
Hubbard 2×3	0.897	128	0.1898	0.988	0.8391

[a]The rest of the hyperparameters are set to their default values.

and IV are the best elites from four independent runs of IRACE.

As for COBYLA and SLSQP, we took their default hyperparameter settings, i.e., $\rho_{\text{initial}} = 0.1$ and tolerance $= 10^{-8}$ for COBYLA and $\varepsilon = 0.055$ and tolerance $= 10^{-8}$.

APPENDIX D: UNITARY COUPLED-CLUSTER *ANSATZ* BASED ON COUPLED-CLUSTER AMPLITUDES

Several classes of systems remain challenging to solve, even for coupled-cluster methods, which are considered the gold standard in quantum chemistry. Those systems are usually plagued by "quasidegeneracy," meaning that the wave function cannot be decomposed into a single leading component. This leads to an important deterioration of methods relying on the single determinant assumption (also said to be monoreference) [44]. This issue can be partially solved by developing multireference coupled-cluster approaches (see Refs. [45,46] for a review). Owing to the recent developments of quantum algorithms in the NISQ era, there has been renewed interest in the unitary formulation of the coupled-cluster (UCC), which is naturally suited for quantum computation and naturally extendable to generating multireference wave functions [4,47], while being intractable on classical computers [21]. Several formulations of UCC have been investigated to go beyond the standard Unitary Coupled-Cluster Singles and Doubles (UCCSD) method in which only fermionic excitations from occupied to virtual orbitals (with respect to the reference determinant, usually the Hartree-Fock one) are considered [21,48–52]. However, the number of operators (and thus the number of parameters) can rapidly become problematic if implemented naively. A powerful approach is provided by the adaptive derivative-assembled pseudo-Trotter (ADAPT) types of *Ansätze* [52–58], which allows us to adaptively increase the number of operators in the *Ansatz* one by one until reaching a given accuracy. In this work, we employ a different strategy by taking advantage of the amplitudes extracted from the traditional coupled-cluster method performed on a classical computer. In the coupled cluster (CC), the exponential *Ansatz* reads as follows:

$$|\Psi(\vec{t})\rangle = e^{\hat{T}}\,|\Phi_0\rangle\,, \tag{D1}$$

where $|\Phi_0\rangle$ denotes the reference determinant (like the Hartree-Fock wave function) and

$$\hat{T} = \sum_{i=1}^{\eta} \hat{T}_i = \sum_{\mu} t_\mu \hat{\tau}_\mu \tag{D2}$$

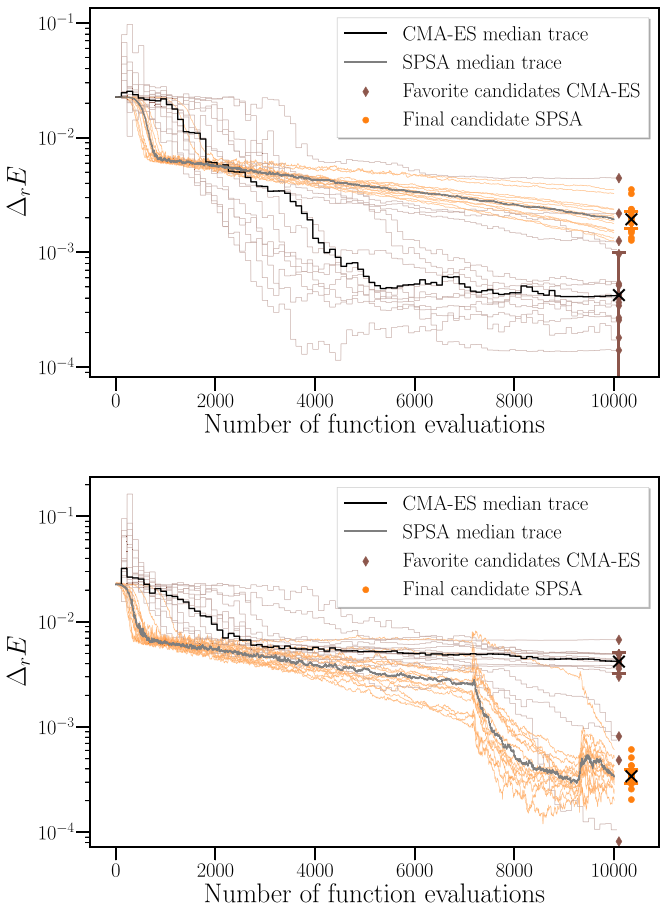

FIG. 4. Optimization traces of SPSA (orange and gray) and CMA-ES (blue and black) for the 2×2 Hubbard model with 10^8 total shots. The thin lines depict an independent optimization trace, with the thick line on top being the median of the 15 independent runs. Top and bottom panels show the one- and three-stage samplings, respectively.

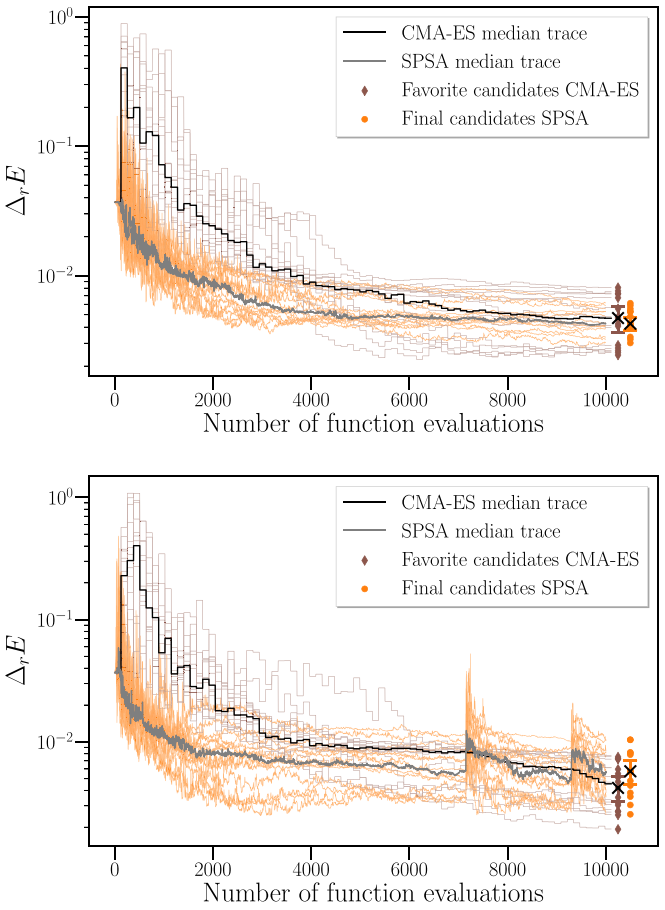

FIG. 5. Optimization traces of SPSA (orange and gray) and CMA-ES (blue and black) for the 2×3 Hubbard model with 10^8 total shots. The thin lines depict an independent optimization trace, with the thick line on top being the median of the 15 independent runs. Top and bottom panels show the one- and three-stage samplings, respectively.

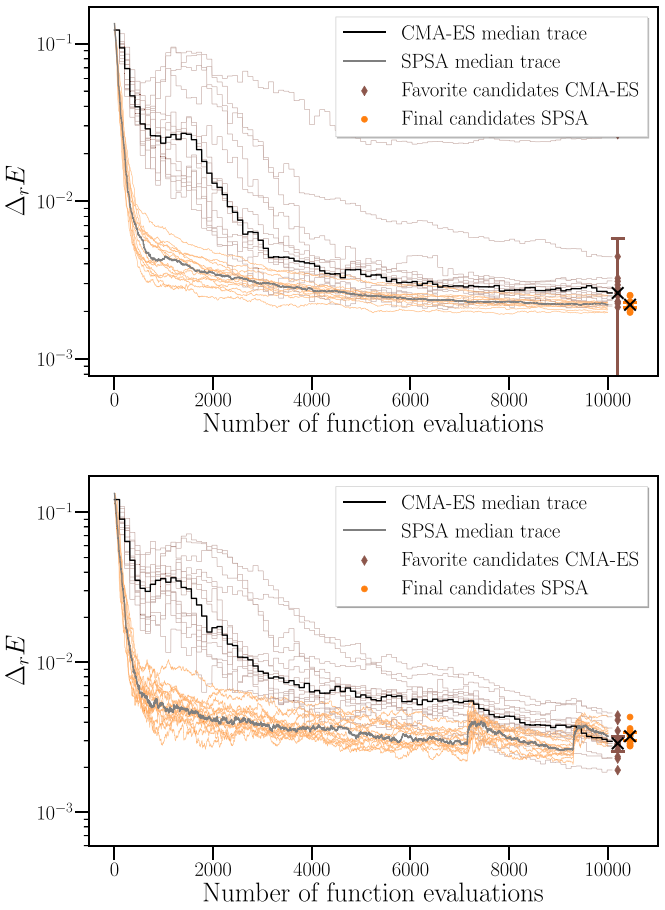

FIG. 6. Optimization traces of SPSA (orange and gray) and CMA-ES (blue and black) for the water molecule in the stretched configuration with 10^8 total shots. The thin lines depict an independent optimization trace, with the thick line on top being the median of the 15 independent runs. Top and bottom panels show the one- and three-stage samplings, respectively.

(η denotes the total number of electrons) is usually truncated to only singles and doubles:

$$\hat{\mathcal{T}}_1 = \sum_{\substack{i\in\text{occ}\\ a\in\text{virt}}} t_a^i \hat{a}_a^\dagger a_i,$$
$$\hat{\mathcal{T}}_2 = \sum_{\substack{i>j\in\text{occ}\\ a>b\in\text{virt}}} t_{ab}^{ij} \hat{a}_a^\dagger \hat{a}_b^\dagger a_i a_j. \tag{D3}$$

One could think of determining the CC amplitudes $\mathbf{t}$ variationally, but this is not convenient in practice because the Baker-Campbell-Hausdorff (BCH) expansion cannot be used (because $\hat{\mathcal{T}}^\dagger \neq -\hat{\mathcal{T}}$). Tractable implementations rely on a nonvariational optimization using the "linked" formulation:

$$e^{-\hat{\mathcal{T}}} \hat{H} e^{\hat{\mathcal{T}}} |\Phi_0\rangle = E(\vec{t}) |\Phi_0\rangle . \tag{D4}$$

The amplitudes are then determined by solving a set of nonlinear equations defined by projecting Eq. (D4) against a set of excited configurations $\{|\mu\rangle\}$ (configurations obtained from the excitation operators in $\hat{\mathcal{T}}$):

$$\langle\mu| e^{-\hat{\mathcal{T}}} \hat{H} e^{\hat{\mathcal{T}}} |\Phi_0\rangle = 0, \tag{D5}$$

for which the BCH expansion can be used, as it can be naturally truncated to the fourth order.

In this work, we computed the coupled-cluster amplitudes of all our molecular systems (H_4 chain, H_4 square, and H_2O) and defined our UCC *Ansatz* according to these amplitudes. Instead of implementing UCC naively by considering all possible excitations, we keep only the excitation operators for which the corresponding CC amplitude is nonzero. This already reduces the total number of operators (and thus the total number of parameters) significantly. In practice, we use the Trotterized-UCC *Ansatz*,

$$|\Psi(\vec{\theta})\rangle = \prod_\mu e^{\theta_\mu(\hat{\tau}_\mu - \hat{\tau}_\mu^\dagger)} |\Phi_0\rangle . \tag{D6}$$

This Trotterized form is an approximation (although it may be mitigated by the classical optimization [59]) which depends on the ordering of the operators. We decided to order the operators with respect to the value of the CC amplitudes in descending order, meaning that the first operator to be applied to the reference state within the UCC *Ansatz* will be the operator with the highest associated CC amplitude. We figured out that the operators in our *Ansatz* were also the ones picked by the

ADAPT-VQE *Ansatz* [52], although the ordering might differ. However, ADAPT-VQE can add new operators (or select and repeat an already present operator) to reach a higher accuracy. To avoid performing the (somewhat costly) first ADAPT-VQE steps, one could think about using our strategy first and then applying ADAPT-VQE for a few more steps to increase the pool of operators slightly. Note that a stochastic classical UCC can also be employed as a preprocessing step to determine the important excitation operators of the UCC *Ansatz*, as shown in the recent work of Filip and Thom [60].

In our numerical experiments, the initial state is always the Hartree-Fock state corresponding to the number of electrons in the system. The parameters of the circuit are initialized at 0.0.

APPENDIX E: VARIATIONAL HAMILTONIAN *ANSATZ* FOR THE HUBBARD MODEL

In this Appendix, we provide the details of the VHA used for the Hubbard model problems. The Fermi-Hubbard Hamiltonian describes the behavior of fermions on a lattice of $n_x \times n_y$ sites. Fermions can hop to the nearest-neighbor sites with some strength t and observe a repulsion or Coulomb term of strength U to move to the same site with the same spin,

$$\begin{aligned} H_{\text{Hubbard}} &= H_{\text{t}} + H_{\text{U}} \\ &= -t \sum_{(i,j),\sigma} (a^\dagger_{i\sigma} a_{j\sigma} + a^\dagger_{j\sigma} a_{i\sigma}) + U \sum_i n_{i\uparrow} n_{i\downarrow}. \end{aligned}$$

One can further split the hopping term with respect to the vertical and horizontal hopping terms $H_{\text{t}} = H_v + H_h$.

The VHA was introduced in Ref. [61] as a means of constructing parametrized quantum states motivated by time evolution by Troterrization for the Hubbard model. However, in our numerical experiments, we use the VHA introduced by Cade *et al.* [11] in which the horizontal and vertical terms can be implemented in parallel [see Eq. (2) in that study]. The parametrized quantum state is constructed as

$$|\Psi(\vec{\theta})\rangle = U(\vec{\theta})\,|\Phi\rangle = \Pi_{l=1}^{L} e^{i\theta_{v_2,l} H_{v_2}} e^{i\theta_{h_2,l} H_{h_2}} \tag{E1}$$

$$\times\, e^{i\theta_{v_1,l} H_{v_1}} e^{i\theta_{h_1,l} H_{h_1}} e^{i\theta_{U,l} H_U}. \tag{E2}$$

The initial state $|\Phi\rangle$ is the Gaussian state of the noninteracting part of the Hamiltonian, and the parameters of the circuit are set to 0.0.

The Fermi-Hubbard Hamiltonian is generated with the open-source package OPENFERMION [37].

APPENDIX F: OPTIMIZATION CONVERGENCE TRACES

In this Appendix, we provide examples of the convergence traces of SPSA and CMA-ES with their hyperparameter tuned. We fix the total number of shots to 10^8 and separate the one- and three-stage sampling procedures. In Fig. 4 we show the traces of the 2×2 Hubbard model. In Fig. 5 the traces of the 2×3 Hubbard model are depicted. Finally, we show in Fig. 6 the water molecule in stretched-configuration optimization traces. In Figs. 4–6, the top panel shows the traces for the one-stage sampling procedure. The bottom panel shows the same traces, but for the three-stage sampling method.

[1] F. Arute *et al.*, Nature (London) **574**, 505 (2019).
[2] Q. Zhu *et al.*, Sci. Bull. **67**, 240 (2022).
[3] J. Preskill, Quantum **2**, 79 (2018).
[4] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, Nat. Commun. **5**, 4213 (2014).
[5] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, New J. Phys. **18**, 023023 (2016).
[6] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cinci, and P. J. Coles, Nat. Rev. Phys. **3**, 625 (2021).
[7] K. M. Nakanishi, K. Fujii, and S. Todo, Phys. Rev. Res. **2**, 043158 (2020).
[8] M. Ostaszewski, E. Grant, and M. Benedetti, Quantum **5**, 391 (2021).
[9] M. Wilson, R. Stromswold, F. Wudarski, S. Hadfield, N. M. Tubman, and E. G. Rieffel, Quantum Mach. Intell. **3**, 13 (2021).
[10] K. J. Sung, J. Yao, M. P. Harrigan, N. C. Rubin, Z. Jiang, L. Lin, R. Babbush, and J. R. McClean, Quantum Sci. Technol. **5**, 044008 (2020).
[11] C. Cade, L. Mineh, A. Montanaro, and S. Stanisic, Phys. Rev. B **102**, 235122 (2020).
[12] S. Finck and H.-G. Beyer, in *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7–11, 2010, Companion Material*, edited by M. Pelikan and J. Branke (Association for Computing Machinery, New York, 2010), pp. 1665–1672.
[13] A. Auger, D. Brockhoff, and N. Hansen, in *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7–11, 2010, Companion Material*, edited by M. Pelikan and J. Branke (Association for Computing Machinery, New York, 2010), pp. 1611–1616.
[14] N. Hansen and A. Ostermeier, Evol. Comput. **9**, 159 (2001).
[15] V. Verteletskyi, T.-C. Yen, and A. F. Izmaylov, J. Chem. Phys. **152**, 124114 (2020).
[16] W. J. Huggins, J. R. McClean, N. C. Rubin, Z. Jiang, N. Wiebe, K. B. Whaley, and R. Babbush, npj Quantum Inf. **7**, 23 (2021).
[17] X. Bonet-Monroig, R. Babbush, and T. E. O'Brien, Phys. Rev. X **10**, 031064 (2020).
[18] A. Zhao, A. Tranter, W. M. Kirby, S. F. Ung, A. Miyake, and P. J. Love, Phys. Rev. A **101**, 062322 (2020).
[19] O. Crawford, B. v. Straaten, D. Wang, T. Parks, E. Campbell, and S. Brierley, Quantum **5**, 385 (2021).
[20] J. Cotler and F. Wilczek, Phys. Rev. Lett. **124**, 100401 (2020).
[21] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik, Quantum Sci. Technol. **4**, 014008 (2018).
[22] N. Rubin, R. Babbush, and J. McClean, New J. Phys. **20**, 053020 (2018).

[23] J. Wu, W. Hu, H. Xiong, J. Huan, V. Braverman, and Z. Zhu, in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event*, Proceedings of Machine Learning Research Vol. 119 (PMLR, 2020), pp. 10367–10376.

[24] L. Yu, K. Balasubramanian, S. Volgushev, and M. A. Erdogdu, arXiv:2006.07904.

[25] Z. Zhu, J. Wu, B. Yu, L. Wu, and J. Ma, in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA*, edited by K. Chaudhuri and R. Salakhutdinov, Proceedings of Machine Learning Research Vol. 97 (PMLR, 2019), pp. 7654–7663.

[26] C. Doerr, F. Ye, N. Horesh, H. Wang, O. M. Shir, and T. Bäck, Appl. Soft Comput. **88**, 106027 (2020).

[27] N. Hansen, D. Brockhoff, O. Mersmann, T. Tusar, D. Tusar, O. A. ElHara, P. R. Sampaio, A. Atamna, K. Varelas, U. Batu, D. M. Nguyen, F. Matzner, and A. Auger, Zenodo, doi: 10.5281/zenodo.2594848.

[28] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, Oper. Res. Perspect. **3**, 43 (2016).

[29] D. Vermetten, H. Wang, M. López-Ibáñez, C. Doerr, and T. Bäck, in *GECCO '22: Genetic and Evolutionary Computation Conference, Boston, Massachusetts, USA, July 9–13, 2022*, edited by J. E. Fieldsend and M. Wagner (Association for Computing Machinery, New York, 2022), pp. 867–875.

[30] X. Bonet-Monroig, H. Wang, D. Vermetten, B. Senjean, C. Moussa, T. Back, V. Dunjko, and T. E. O'Brien, Zenodo, doi: 10.5281/zenodo.5721349.

[31] J. C. Spall, IEEE Trans. Autom. Control **37**, 332 (1992).

[32] J. C. Spall, Johns Hopkins APL Tech. Dig. **19**, 482 (1998).

[33] M. J. D. Powell, Acta Numerica **7**, 287 (1998).

[34] D. Kraft, *A Software Package for Sequential Quadratic Programming*, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht (DFVLR, Cologne, 1988).

[35] D. Kraft, ACM Trans. Math. Software **20**, 262 (1994).

[36] J. Nocedal and S. J. Wright, in *Numerical Optimization* (Springer, New York, 2006), pp. 529–562.

[37] J. R. McClean *et al.*, Quantum Sci. Technol. **5**, 034014 (2020).

[38] Cirq Developers, CIRQ, https://github.com/quantumlib/Cirq/graphs/contributors.

[39] P. Virtanen *et al.*, Nat. Methods **17**, 261 (2020).

[40] N. Hansen, Y. Akimoto, and P. Baudis, Zenodo, doi: 10.5281/zenodo.2559634.

[41] A. Mayer and S. O. Aas, ANDIM/NOISYOPT, Zenodo, doi: 10.5281/zenodo.580120.

[42] J. de Nobel, D. Vermetten, H. Wang, C. Doerr, and T. Bäck, in *GECCO '21: Genetic and Evolutionary Computation Conference, Companion Volume, Lille, France, July 10–14, 2021*, edited by K. Krawiec (Association for Computing Machinery, New York, 2021), pp. 1375–1384.

[43] D. Vermetten, H. Wang, T. Bäck, and C. Doerr, in *GECCO '20: Genetic and Evolutionary Computation Conference, Cancún Mexico, July 8–12, 2020*, edited by C. A. C. Coello (Association for Computing Machinery, New York, 2020), pp. 654–662.

[44] I. W. Bulik, T. M. Henderson, and G. E. Scuseria, J. Chem. Theory Comput. **11**, 3171 (2015).

[45] R. J. Bartlett and M. Musiał, Rev. Mod. Phys. **79**, 291 (2007).

[46] D. I. Lyakh, M. Musiał, V. F. Lotrich, and R. J. Bartlett, Chem. Rev. **112**, 182 (2012).

[47] F. A. Evangelista, G. K.-L. Chan, and G. E. Scuseria, J. Chem. Phys. **151**, 244112 (2019).

[48] G. Greene-Diniz and D. Muñoz Ramo, Int. J. Quantum Chem. **121**, e26352 (2021).

[49] J. Lee, W. J. Huggins, M. Head-Gordon, and K. B. Whaley, J. Chem. Theory Comput. **15**, 311 (2018).

[50] W. Mizukami, K. Mitarai, Y. O. Nakagawa, T. Yamamoto, T. Yan, and Y.-Y. Ohnishi, Phys. Rev. Res. **2**, 033421 (2020).

[51] I. O. Sokolov, P. K. Barkoutsos, P. J. Ollitrault, D. Greenberg, J. Rice, M. Pistoia, and I. Tavernelli, J. Chem. Theory Comput. **152**, 124107 (2020).

[52] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, Nat. Commun. **10**, 3007 (2019).

[53] D. Claudino, J. Wright, A. J. McCaskey, and T. S. Humble, Front. Chem. **8**, 606863 (2020).

[54] Y. S. Yordanov, V. Armaos, C. H. W. Barnes, and D. R. M. Arvidsson-Shukur, Commun. Phys. **4**, 228 (2021).

[55] N. Gomes, A. Mukherjee, F. Zhang, T. Iadecola, C.-Z. Wang, K.-M. Ho, P. P. Orth, and Y.-X. Yao, Adv. Quantum Technol. **4**, 2100114 (2021).

[56] J. Liu, Z. Li, and J. Yang, J. Chem. Phys. **154**, 244112 (2021).

[57] H. L. Tang, V. O. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou, PRX Quantum **2**, 020310 (2021).

[58] Z.-J. Zhang, T. H. Kyaw, J. Kottmann, M. Degroote, and A. Aspuru-Guzik, Quantum Sci. Technol. **6**, 035001 (2021).

[59] H. R. Grimsley, D. Claudino, S. E. Economou, E. Barnes, and N. J. Mayhall, J. Chem. Theory Comput. **16**, 1 (2019).

[60] M.-A. Filip and A. J. Thom, J. Chem. Phys. **153**, 214106 (2020).

[61] D. Wecker, M. B. Hastings, and M. Troyer, Phys. Rev. A **92**, 042303 (2015).