Surrogate-based optimization for variational quantum algorithms

Ryan Shaffer,[1,2,*] Lucas Kocia,[2] and Mohan Sarovar[2,†]
[1]*Department of Physics, University of California, Berkeley, California 94720, USA*
[2]*Quantum Algorithms and Applications Collaboratory, Sandia National Laboratories, Livermore, California 94550, USA*

(Received 26 April 2022; revised 11 July 2022; accepted 6 February 2023; published 16 March 2023)

Variational quantum algorithms are a class of techniques intended to be used on near-term quantum computers. The goal of these algorithms is to perform large quantum computations by breaking the problem down into a large number of shallow quantum circuits, complemented by classical optimization and feedback between each circuit execution. One path for improving the performance of these algorithms is to enhance the classical optimization technique. Given the relative ease and abundance of classical computing resources, there is ample opportunity to do so. In this work, we introduce the idea of learning surrogate models for variational circuits using a few experimental measurements, and then performing parameter optimization using these models as opposed to the original data. We demonstrate this idea using a surrogate model based on kernel approximations, through which we reconstruct local patches of variational cost functions using batches of noisy quantum circuit results. Through application to the quantum approximate optimization algorithm and preparation of ground states for molecules, we demonstrate the superiority of surrogate-based optimization over commonly used optimization techniques for variational algorithms.

DOI: 10.1103/PhysRevA.107.032415

I. INTRODUCTION

As the quality and scale of quantum information processors (QIPs) increase, the question of whether they can derive some advantage over conventional (classical) computers, even before reaching the fault-tolerant regime, is becoming increasingly important to the field. Hybrid algorithms that utilize quantum and classical computing are perhaps the most promising route to such an advantage, and variational algorithms (VQAs) where the QIP evaluates a parametrized cost function that is then optimized by a classical computer are the prime example of such hybrid algorithms [1,2].

Conventional implementations of variational algorithms evaluate a parametrized cost function $V(\boldsymbol{\theta})$, usually representing a parametrized quantum circuit, and then optimize over $\boldsymbol{\theta}$ using off-the-shelf multiparameter optimization routines like COBYLA, SPSA, and Nelder-Mead [3,4]. Such an approach only minimally exploits the structure of the underlying problem, and moreover, only minimally utilizes the computational power of the classical computing layer. While this approach has been used to demonstrate variational algorithms with a handful of parameters, $|\boldsymbol{\theta}| \equiv D \leqslant 10$, it is unclear how its effectiveness and the experimental resources it requires will scale to larger problems, where the number of variational parameters becomes hundreds or thousands.

Motivated by this, we introduce an approach to optimization in variational algorithms that utilizes modern statistical inference tools to reduce the experimental burden when running variational algorithms. This, in effect, moves more of the burden from the QIP to the classical computing layer. The core of our approach is the construction of a *surrogate model* for the variational cost function from QIP experimental data and performing optimization with this surrogate model instead of the original data. This is an established approach in optimization theory, and surrogate-based optimization (SBO) has found uses in applications where the optimization cost function is difficult to evaluate due to paucity of data or computational expense [5]. There are a variety of techniques for learning a surrogate model from data, including spline-based fitting, kriging, and neural network models [6]. In this work, we demonstrate SBO for VQAs using local kernel approximation techniques. Kernel approximation is particularly useful for building surrogate models for variational quantum circuits for several reasons: (i) the resulting models are explicitly smooth and smooth out unavoidable shot noise in quantum circuit measurements, (ii) the models can be learned with *batches* of circuit outputs, which has practical advantages for quantum computing platforms where circuit loading incurs latency, and (iii) the models allow numerically efficient computation of $V(\boldsymbol{\theta})$ and its derivatives, thus enabling optimization by scalable gradient-based algorithms. Intuitively, surrogate models based on a kernel approximation can be seen as explicitly taking advantage of the fact that the underlying variational cost function is smooth [$V(\boldsymbol{\theta}) \in C^{\infty}$], and thus its value at $\boldsymbol{\theta}$ indicates its value in its neighborhood. We couple this local surrogate model with an adaptive optimization procedure to efficiently find local optima of the variational cost function.

There have been several recent efforts to develop custom optimizers for VQAs, including: variations of stochastic gradient descent that adapt the number of experimental circuit

*Current address: AWS Quantum Technologies, Seattle, Washington 98170, USA. Work done prior to joining Amazon.
†mnsarov@sandia.gov

evaluations (shots) to manage the tradeoff between cost function and gradient estimation quality and experimental burden [7–9], techniques based on Bayesian optimization [10–12], and machine learning-based optimization approaches for specific VQAs [13]. Most relevant to this work is the study of Sung *et al.* [14], which in the framework of SBO, developed local quadratic models based on experimental data and coupled this with a trust-region optimization algorithm. Our work expands on this result by considering more general, nonparametric surrogate models that are designed to be valid over larger regions in parameter space, where the quadratic model might break down. We note that a related approach based on Gaussian process surrogate models has recently been proposed by Mueller *et al.* [15].

In the following, we introduce our optimization algorithm (Sec. II), analyze its theoretical properties and hyperparameter choices (Sec. III), and present several numerical illustrations of the approach, including comparisons to conventional variational optimization algorithms (Sec. IV). Finally, we conclude with a summary and discussion of possible extensions of our approach (Sec. V).

II. SURROGATE-BASED OPTIMIZATION

The goal of the classical computing layer in quantum variational algorithms is to compute

$$\min_{\boldsymbol{\theta}} V(\boldsymbol{\theta}) \tag{1}$$

and, often, also the argument that attains this minimum. Here, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_D) \in [0, 2\pi)^D$ are parameters that dictate the variational quantum circuit ansatz for the problem, $V(\boldsymbol{\theta}) : [0, 2\pi)^D \to \mathbb{R}$ is the variational cost function, which is related to the parametrized circuit, $\hat{U}(\boldsymbol{\theta})$, acting on n qubits: $V(\boldsymbol{\theta}) = \mathrm{tr}[\hat{O}\hat{U}(\boldsymbol{\theta})\hat{\rho}_0\hat{U}^\dagger(\boldsymbol{\theta})]$, for some initial n-qubit state $\hat{\rho}_0$ and observable $\hat{O}$. This quantum expectation must be estimated using many measurements on the circuit output. To do so, we first decompose the observable into a sum of noncommuting operators, $\hat{O} = \sum_{i=1}^{\nu} \alpha_i \hat{o}_i$, with $[\hat{o}_i, \hat{o}_j] \neq 0$ for $i \neq j$. For all practical VQAs, $\nu = O(\mathrm{poly}(n))$. Then, writing the circuit output as $\hat{\rho}(\boldsymbol{\theta}) \equiv \hat{U}(\boldsymbol{\theta})\hat{\rho}_0\hat{U}^\dagger(\boldsymbol{\theta})$,

$$V(\boldsymbol{\theta}) = \sum_{i=1}^{\nu} \alpha_i \mathrm{tr}(\hat{o}_i\hat{\rho}(\boldsymbol{\theta})) = \sum_{i=1}^{\nu} \alpha_i \mathbb{E}\{\mathbf{X}^i(\boldsymbol{\theta})\}, \tag{2}$$

where $\mathbf{X}^i(\boldsymbol{\theta})$ is a random variable distributed as $p^i(\boldsymbol{\theta})$ that represents the outcome of measuring $\hat{\rho}(\boldsymbol{\theta})$ in the eigenbasis of $\hat{o}_i$. In practice, the expectation in the final expression is estimated using a sample mean of a number of *shots* (executions of the circuit at $\boldsymbol{\theta}$ and measurements in one of the ν bases). That is, one takes K_i measurements of $\mathbf{X}^i(\boldsymbol{\theta}) : X_1^i(\boldsymbol{\theta}), \ldots, X_{K_i}^i(\boldsymbol{\theta})$, and approximates $\mathbb{E}\{\mathbf{X}^i(\boldsymbol{\theta})\} \approx \frac{1}{K_i}\sum_{j=1}^{K_i} X_j^i(\boldsymbol{\theta})$. The total number of shots, or circuit executions, necessary to form an estimate of the cost function at a given parameter value,

$$\tilde{V}(\boldsymbol{\theta}) = \sum_{i=1}^{\nu} \alpha_i \left(\frac{1}{K_i} \sum_{j=1}^{K_i} X_j^i(\boldsymbol{\theta}) \right) \tag{3}$$

is $\mathcal{K} = \sum_{i=1}^{\nu} K_i$.

Since $V(\boldsymbol{\theta})$ must be estimated from a finite number of measurement results, the resulting optimization landscape is noisy and becomes increasingly so as the number of available measurements, $\mathcal{K}$, decreases. This is the impact of so-called quantum *shot noise* (the irreducible uncertainty of quantum systems that results in indeterminate measurement outcomes in general) on the variational optimization problem. The poor performance of most optimization algorithms in such noisy landscapes places a burden on the QIP to produce as many measurements as possible to increase the accuracy of this expectation estimate, and therefore the smoothness of $\tilde{V}(\boldsymbol{\theta})$. In addition to this shot noise, in present and near-future generations of noisy intermediate scale quantum (NISQ) devices [16] there are other sources of noise coming from poor control, measurement, and isolation (decoherence) that produce distortions of the underlying probability distribution over measurement outcomes, i.e., $p^i(\boldsymbol{\theta}) \to \tilde{p}^i(\boldsymbol{\theta})$. We do not directly address this source of noise, although we note that several error mitigation techniques have been developed to address this problem, e.g., Refs. [17–20], and they can be used in tandem with our optimization approach to achieve some degree of robustness to both sources of noise (shot noise and decoherence).

We now introduce the concept of a local surrogate model to $V(\boldsymbol{\theta})$. This is a function $W : \Theta \to \mathbb{R}$ that is an approximation of $V(\boldsymbol{\theta})$ in a local *patch*, $\Theta \subset [0, 2\pi)^D$. We demand that this surrogate model must be (i) smooth and (ii) efficient to evaluate on a classical computer, requiring no additional measurements from a QIP than those required to construct it. In this work, we construct such a surrogate model using a kernel approximation, i.e.,

$$W_\Theta(\boldsymbol{\theta}) = \sum_{j=1}^{\tau} \tilde{V}(\boldsymbol{\theta}_j)\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j), \tag{4}$$

where $\tilde{V}(\boldsymbol{\theta}_j)$ are standard estimates of $V(\boldsymbol{\theta})$ (constructed using $\mathcal{K}$ shots) at τ distinct *sample points* $\boldsymbol{\theta}_j \in \Theta$, and $\kappa(\cdot, \cdot)$ is a *kernel function*. Note that the subscript on W_Θ serves to remind us that the surrogate model is valid in some local patch of parameter space, since it is formulated based on data from that local patch.

The choice of κ determines most of the properties of kernel-based surrogate models. In this work, we choose a Gaussian kernel, $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j) = \exp(-||\boldsymbol{\theta}-\boldsymbol{\theta}_j||^2/2\sigma)$, for two reasons. First, it is a simple kernel with only one free parameter, σ, that can be set in a data-driven manner, as we show below. And second, its form allows for easy analytic evaluation of derivatives of $W_\Theta(\boldsymbol{\theta})$, which is a useful property for gradient-based optimization of $W(\boldsymbol{\theta})$.

It is known that this kernel can result in a systematic bias [21]. In the context of VQAs, this is often manifest in an "offset" of the kernel-produced variational cost function values from the experimental values. However, given the prevalence of systematic noise in experimental measurements on current quantum hardware, there is frequently no gain to be made from expending computational resources to get the true experimental surface because it is offset already, and only relative magnitudes matter for optimization. Moreover, in applications where the goal is finding the parameter argument of the minimal objective function, the offset is irrelevant.

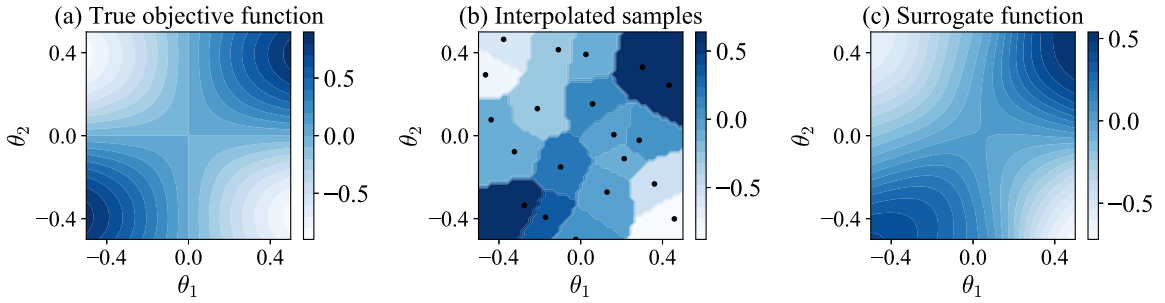

FIG. 1. An illustration of a local patch of (a) a true objective function $V(\boldsymbol{\theta})$ with dimension $D = 2$ where $\boldsymbol{\theta} = (\theta_1, \theta_2)$, (b) interpolated samples $\tilde{V}(\boldsymbol{\theta})$ using $\mathcal{K} = 100$ shots at each of the $\tau = 20$ sample points, and (c) surrogate function $W(\boldsymbol{\theta})$ constructed using a Gaussian kernel $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j) = \exp(^{-||\boldsymbol{\theta}-\boldsymbol{\theta}_j||^2}/_{2\sigma})$.

Finally, in applications where the minimal variational cost function value is desired, it is often possible to fix the offset, both from experimental noise and the kernel, by appealing to special cases when the parameter values simplify the objective function to known values, and shifting the offset of the full surface accordingly. Figure 1 provides an illustration of a true objective function $V(\boldsymbol{\theta})$, interpolated samples $\tilde{V}(\boldsymbol{\theta}_j)$, and surrogate function $W_\Theta(\boldsymbol{\theta})$ constructed using a Gaussian kernel.

A. Adaptive optimization

As described above, the kernel-based surrogate model is learned over a local patch Θ. In order to find a local optimum of $V(\boldsymbol{\theta})$, we couple this construction with an adaptive optimization procedure that we describe in this section.

We begin with an initial seed for the variational parameters, $\boldsymbol{\theta}^{(0)}$, and define a local patch around it as a D-dimensional hypercube of length ℓ: $\Theta^{(0)} = \cup_{m=1}^{D}[\theta_m^{(0)} - {}^{\ell}/_2, \theta_m^{(0)} + {}^{\ell}/_2]$. Then we randomly sample τ points in this patch, execute variational circuits defined by each of those sample points, and use the resulting data to form estimates $\tilde{V}(\boldsymbol{\theta}_1), \ldots, \tilde{V}(\boldsymbol{\theta}_\tau)$. We assume for simplicity that each of the estimates $\tilde{V}(\boldsymbol{\theta}_j)$ is formed using $\mathcal{K}$ shots, i.e., $\mathcal{K}$ does not depend on j, although this is not an essential assumption. The τ samples of $\boldsymbol{\theta}_j$ are sampled sparsely in $\Theta^{(0)}$; to achieve this in practice, we use Latin hypercube sampling over $\Theta^{(0)}$ to choose each $\boldsymbol{\theta}_j$. The number of samples τ and the patch "size" ℓ are important parameters; we develop heuristics for choosing their values and study their scaling with n and D in Sec. III (and also in Appendix A). The estimates $\tilde{V}(\boldsymbol{\theta}_j)$ are then used to formulate a surrogate model $W_{\Theta^{(0)}}$ for $V(\boldsymbol{\theta})$ on the patch $\Theta^{(0)}$, as defined in Eq. (4).

Given $W_{\Theta^{(0)}}(\boldsymbol{\theta})$, we perform optimization over this (explicitly smooth) function over the local domain $\Theta^{(0)}$. We do not specify the method to use for this optimization. However, given a smooth objective and easily computable gradients, gradient-based optimizers that incorporate parameter constraints [since the optimization should only be over $\Theta^{(0)}$] are well-suited for this task. In practice, it may be helpful to optimize over a slightly smaller domain to avoid edge effects in the kernel approximation, i.e.,

$$\min_{\boldsymbol{\theta}\in\Theta_\epsilon^{(0)}} W_{\Theta^{(0)}}(\boldsymbol{\theta}), \tag{5}$$

with $\Theta_\epsilon^{(0)} = \cup_{m=1}^{D}[\theta_m^{(0)} - {}^{(\ell-\epsilon)}/_2, \theta_m^{(0)} + {}^{(\ell-\epsilon)}/_2]$. The argument that achieves the above minimum defines the center of the next patch, $\boldsymbol{\theta}^{(1)}$, and this process is repeated.

We refer to the process above as one *iteration* of the optimization run. Each iteration thus requires $\mathcal{K}\tau$ shots. We perform a fixed number of iterations M, giving a total of $\mathcal{K}\tau M$ shots in a full optimization run. To assist in the convergence of the optimization run, we linearly increase ϵ from some initial (small) value ϵ_i in the first iteration to a value near ℓ in the final iteration.

If the minimum $\boldsymbol{\theta}^{(i+1)}$ found after iteration i falls within the interior of the current patch $\Theta^{(i)}$, i.e., if $\boldsymbol{\theta}^{(i+1)} \in \Theta_{\epsilon_{\text{int}}}^{(i)}$ for some small $\epsilon_{\text{int}} \sim {}^{\ell}/_{20}$ which excludes the boundary of the patch, then we add the minimum $\boldsymbol{\theta}^{(i+1)}$ to a list of local minima Θ_{minima}. After completing M iterations, we calculate the final estimated optimum $\boldsymbol{\theta}_{\text{opt}}$ by taking the coordinate-wise mean of all of the elements of Θ_{minima} that fall within a distance $\ell - \epsilon_f$ (for $\epsilon_f \sim {}^{\ell}/_2$) of the minimum $\boldsymbol{\theta}^{(M)}$ found in the final iteration, i.e., for $\Theta_{\epsilon_f,\text{minima}}^{(M)} = \Theta_{\text{minima}} \cap \Theta_{\epsilon_f}^{(M)}$,

$$\boldsymbol{\theta}_{\text{opt}} = \frac{1}{\left|\Theta_{\epsilon_f,\text{minima}}^{(M)}\right|} \sum_{\boldsymbol{\theta}\in\Theta_{\epsilon_f,\text{minima}}^{(M)}} \boldsymbol{\theta}. \tag{6}$$

Figure 2 provides a graphical description of the surrogate-based adaptive optimization approach described above.

We note that the optimization approach is decoupled from the surrogate model. Although we have found that the adaptive optimization detailed above is effective, it is by no means unique or optimal. It is possible to modify it or even replace it with another approach while keeping the surrogate model idea intact. In particular, it is likely advantageous to incorporate a memory element that inclues information from previous patches into the decisions made at the current patch—this is a promising area for future study.

III. CONVERGENCE AND HYPERPARAMETER CHOICES

In this section, we discuss practical considerations for choosing various hyperparameters of SBO and the adaptive optimization technique described in Sec. II.

Optimization adjustments ϵ_i, ϵ_{int}, and ϵ_f. These parameters are used primarily to avoid boundary effects near the edges of each patch region, since during each iteration we sample

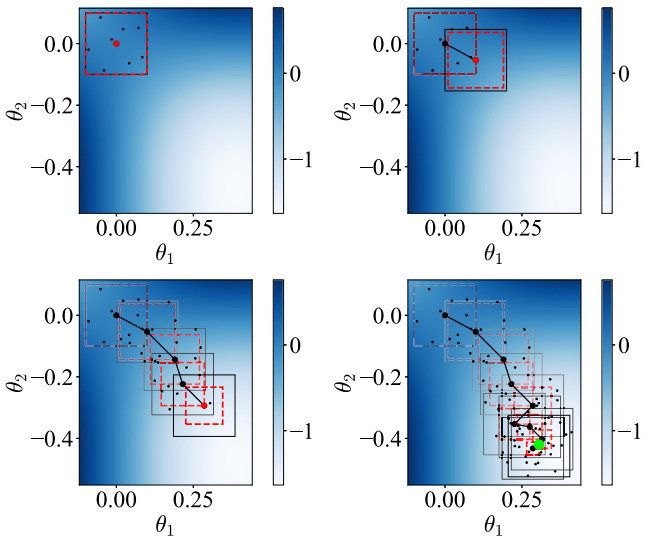

FIG. 2. A graphical description of the adaptive surrogate-based optimization procedure on a two-dimensional objective function surface parameterized by $\boldsymbol{\theta} = (\theta_1, \theta_2)$, showing snapshots of an optimization run during the first iteration (top left), after the first iteration (top right), after the fourth iteration (bottom left), and at the completion of the run after $M = 10$ iterations (bottom right). The larger, connected points mark the patch centers $\boldsymbol{\theta}^{(i)}$ of each iteration, with the red point indicating the center of the most recent patch. The smaller, unconnected points mark the locations of the $\tau = 8$ samples taken during each iteration. The solid black rectangles mark the boundaries of the sampling region $\Theta^{(i)}$ for each iteration, where each side has fixed length $\ell = 0.2$. The dashed red rectangles mark the boundaries of the optimization region $\Theta_\epsilon^{(i)}$ for each iteration, where ϵ is linearly increased from 0 to $\ell = 0.2$ over the course of the optimization run. The green point in the final plot (bottom right) marks the final estimated optimum $\boldsymbol{\theta}_{\text{opt}}$.

from only the interior of the patch. In this work, we have used $\epsilon_i = 0$, $\epsilon_{\text{int}} = \ell/20$, and $\epsilon_f = \ell/2$ with good results.

Measurement shots per measurement basis per sample point K. The choice of K will be primarily driven by experimental considerations. Larger K is always better since it will reduce shot noise and therefore improve the accuracy of the surrogate model, and in turn the performance of the optimization but at the cost of increased experimental demands (especially run time). As we shall demonstrate in the next section, one of the advantages of constructing a surrogate model is an increased robustness of optimization performance to shot noise, and thus SBO can alleviate the experimental burden without sacrificing optimization performance.

Patch size ℓ *and sample points per patch* τ. These parameters are intimately related. Intuitively, the larger the patch size, ℓ, the larger the number of sample points per patch, τ, will need to be in order for the surrogate model to be accurate to the true cost function $V(\boldsymbol{\theta})$. Since τ is closely tied to experimental resources, we find it most useful to think in terms of keeping τ fixed at a constant, and varying ℓ. In practice, especially in the near-term, experimental constraints such as device instability and access constraints will dictate how large τ can be, and therefore we think of it as a fixed parameter, independent of variational problem parameters such as n and D. In all of our numerical experiments, including the ones reported in the next section, we have kept $\tau \sim 20$.

Given a fixed, constant τ, the choice of ℓ is dictated by the need to accurately capture the shape of the objective function $V(\boldsymbol{\theta})$ over the patch in each of the D parameter dimensions. A conservative way to ensure that a fixed number of samples captures the objective function is to demand that this function varies minimally within the patch, i.e., to choose ℓ such that there is likely no more than one critical point of $V(\boldsymbol{\theta})$ in any ℓ^D volume in parameter space. In Appendix A, we study the number of critical points in a general variational cost function and based on a loose bound, derive the scaling $\ell = \Omega(1/\text{poly}(D,n))$. For the empirical studies reported in this paper, we have found that patch sizes in the range $0.1 \leqslant \ell \leqslant 0.2$ worked well for QAOA problems with $n \leqslant 12$ and $p \leqslant 7$ ($D \leqslant 14$), as well as VQE problems with $n \leqslant 8$ and $D \leqslant 8$, using $K \sim 100$.

Perhaps the most robust solution to determining ℓ is to employ an adaptive method that dynamically adjusts ℓ along the optimization path according to a quality of fit metric. This would be possible by moving to a trust-region framework for the optimization [22].

Surrogate model parameters. The procedure used to construct the surrogate model will typically have parameters to set. In this work, we only consider kernel-based surrogate models, and specifically an isotropic Gaussian kernel $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j)$, which has one parameter, the *Gaussian bandwidth* σ. Intuitively, this parameter describes the volume over which one sample data point influences the behavior of the surrogate model. There is a rich literature on Gaussian kernels and their use in approximation, regression, and smoothing, and as a result, many data-driven heuristics exist for choosing σ. In practice, we have observed good performance using the Silverman bandwidth heuristic [23] $\sigma = [4/\tau(D+2)]^{1/(D+4)}$, where τ is the number of sample points in the current patch.

IV. ILLUSTRATIONS

In this section, we demonstrate our SBO procedure on some model VQAs through numerical simulation. We also compare optimization performance with one of the most commonly used and recommended optimization methods for VQAs, simultaneous perturbation stochastic approximation (SPSA) [24]. SPSA is designed to find optima in the presence of noise in the objective function. Key to its popularity in the resource-constrained setting of VQAs is the fact that it estimates gradients using only two evaluations of the (multi-parameter) objective function.

A. Quantum approximate optimization algorithm

The quantum approximate optimization algorithm (QAOA) is a variational circuit approach to combinatorial optimization [25], where the optimization problem is encoded in a *problem Hamiltonian*, $\hat{H}_p$, whose ground state encodes the solution to the problem. A commonly studied example is the MaxCut problem, which aims to partition an n-node graph into two sets of nodes, such that the weight of the edges going between the partitions is maximized. A MaxCut problem instance is encoded in an n-qubit Ising Hamiltonian of the form $\hat{H}_p = \sum_{(i,j)\in\mathcal{E}} w_{ij}\hat{Z}_i\hat{Z}_j$, where $\hat{Z}_i$ is a Pauli Z matrix on

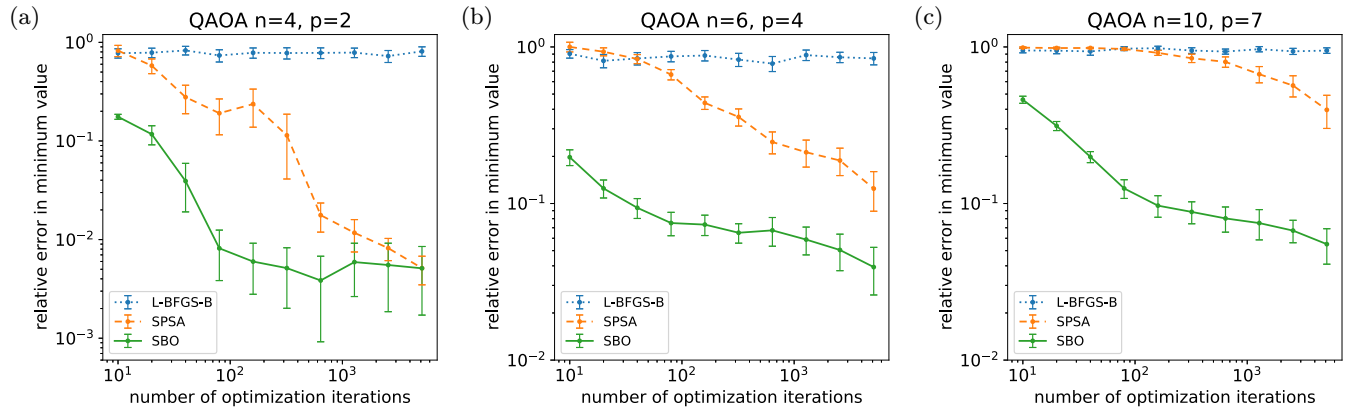

FIG. 3. Performance of L-BFGS-B, SPSA, and Gaussian kernel-based SBO optimization runs on QAOA applied to unweighted MaxCut problems of various sizes using an ideal simulator which has only shot noise (and no other errors). Each plot displays the results of running p-layer QAOA on a single randomly generated connected graph with n vertices. The x axis represents the number of optimization iterations M. The y axis represents the relative absolute error achieved by the optimization run, i.e., $|1 - V_{\mathrm{QAOA}}(\boldsymbol{\gamma}_{\mathrm{opt}}, \boldsymbol{\beta}_{\mathrm{opt}})/V_{\mathrm{QAOA,min}}|$, where $\boldsymbol{\gamma}_{\mathrm{opt}}$ and $\boldsymbol{\beta}_{\mathrm{opt}}$ are the optimal coordinates obtained by the optimization run and $V_{\mathrm{QAOA,min}} = \min_{\boldsymbol{\gamma},\boldsymbol{\beta}} V_{\mathrm{QAOA}}(\boldsymbol{\gamma}, \boldsymbol{\beta})$ is the true optimum of the objective function. Each run uses $K\tau = 5000$ total shots per iteration, where τ is the number of sample points per iteration and K is the number of shots taken per sample point. Each data point represents the mean of 50 independent optimization runs on one representative problem instance, represented by an Erdős-Rényi random unweighted graph. The initial parameter choice, $\boldsymbol{\theta}^{(0)}$ is the same for all runs; however, the sample points on each patch obtained via Latin hypercube sampling are chosen independently for each run. Error bars indicate standard error of the mean. Additional details on hyperparameter choices and implementation notes can be found in Appendix B.

qubit i tensored with the identity on all other qubits, and $\mathcal{E}$ is the set of edges in the graph, each with weight $w_{ij} \in \mathbb{R}$.

QAOA approaches the goal of preparing low energy eigenstates of $\hat{H}_p$ by p iterated applications of a two-layer ansatz to a product input state to produce the output state:

$$|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \prod_{l=1}^{p} e^{-i\beta_p \hat{H}_d} e^{-i\gamma_p \hat{H}_p} |+\rangle^{\otimes n}, \tag{7}$$

where $|+\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle)$, and $\hat{H}_d = \sum_{i=1}^{n} \hat{X}_i$. The variational parameters $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ are optimized such that the energy of the output state is minimized, i.e., the objective function is $V_{\mathrm{QAOA}}(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})|\hat{H}_p|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle$. The *approximation ratio*, which quantifies how close to the true ground state any $|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle$ is, is defined by $r = V_{\mathrm{QAOA}}/E_0$, where E_0 is the true ground state energy of $\hat{H}_p$.

If global optima to V_{QAOA} can be found, in the $p \to \infty$ limit QAOA prepares the ground state of $\hat{H}_p$, which encodes the solution to the original combinatorial optimization problem [25]. Moreover, in this case, r increases monotonically with p, although the question of what p is required for r to surpass approximation ratios achievable by classical approximation methods is an open one. It is clear that the variational optimization, and even finding good quality local minima of V_{QAOA}, becomes challenging with increasing p.

In terms of the parameters defined in the general description of VQAs in Sec. II, it is important to note that the QAOA objective is defined through an observable, H_p, that only consists of commuting terms. Therefore, one only needs to measure in the computational basis for QAOA, meaning that we have $\nu = 1$ and we require $\mathcal{K} = K$ total shots per sample point.

Figure 3 shows the results of simulated L-BFGS-B, SPSA, and SBO optimization runs of QAOA applied to MaxCut instances on (Erdős-Rényi) random unweighted graphs of (a) $n = 4$ vertices using $p = 2$ layers, (b) $n = 6$ vertices using $p = 4$ layers, and (c) $n = 10$ vertices using $p = 7$ layers. We plot the results of each run using $K\tau = 5000$ shots per iteration. We repeated these tests with various values of $K\tau$ ranging from 10^3 to 10^5 and observed qualitatively similar results.

We choose L-BFGS-B here as an example of a gradient-free optimizer. We use a gradient-free optimizer because the gradient of our noisy objective function is not directly available and therefore traditional gradient descent cannot be used. We found that L-BFGS-B, although it still performs very poorly due to the noisy objective function, significantly outperformed Nelder-Mead, another widely used gradient-free optimizer.

At the smallest problem size, SBO achieves a lower error than SPSA for up to $M \sim 10^3$ iterations, indicating that it converges on a good approximation of the local minimum more efficiently. At the larger problem sizes, SBO achieves a lower error than SPSA for even larger numbers of iterations. For perspective, we note that an optimization run with $K\tau = 5000$ shots per iteration and $M = 10^3$ iterations would require a total of $K\tau M = 5\times10^6$ experimental shots. This would require an experimental duration on the order of several minutes using a typical superconducting QIP, or on the order of several days using a typical trapped-ion QIP. Because our results indicate that SBO significantly outperforms SPSA in this regime, it appears likely that SBO will achieve lower error than SPSA for many QAOA experiments that can be realistically implemented on current and near-future devices.

B. Variational quantum eigensolver

The first VQA was the so-called variational quantum eigensolver (VQE) [26], which aims to prepare the ground

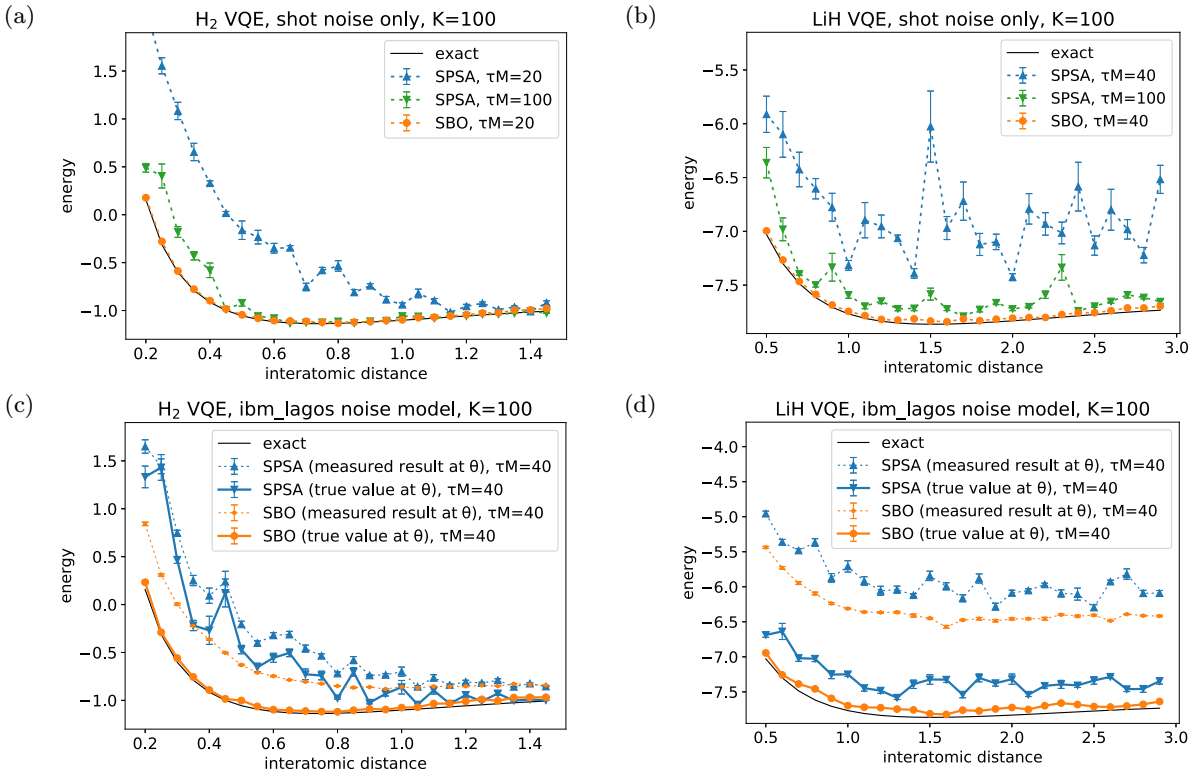

FIG. 4. Performance of SPSA and Gaussian kernel-based SBO optimization runs on common small-scale VQE problems using simulators with and without realistic hardware noise. Each plot displays the minimum energy obtained for each bond length under the specified optimization conditions. The x axis represents the interatomic distance (in angstroms) used for the energy calculation. The y axis represents the energy value (in hartrees) obtained at the conclusion of each optimization run. τM is the total number of energy measurements performed in the optimization run, where τ is the number of sample points per iteration and M is the number of optimization iterations. To measure the energy, $K = 100$ shots are taken per measurement basis per sample point. Each data point represents the mean of five independent optimization runs at the given setting. Error bars indicate standard error of the mean. On each plot, a solid black curve indicates the exact minimum energy value for the given setting. In (a) and (b), we use an ideal simulator which has only shot noise and no other errors. The data points connected by dashed curves represent the energy value obtained by evaluating the ansatz on the ideal simulator at the found optimal parameter values $\boldsymbol{\theta}$. In (c) and (d), we use a noisy simulator implementing a typical noise model and coupling map obtained from the seven-qubit IBM Q Lagos device. The data points connected by dashed curves represent the energy value obtained by evaluating the ansatz on the noisy simulator at the found optimal parameter values $\boldsymbol{\theta}$. The data points connected by solid curves represent the energy value obtained by evaluating the ansatz at $\boldsymbol{\theta}$ using an ideal simulator. Additional details on hyperparameter choices and implementation notes can be found in Appendix B.

state of an n-qubit Hamiltonian, $\hat{H}_E$, that encodes the energy of a molecule. The variational circuits and parameters, $\boldsymbol{\theta}$, that prepare candidate states vary according to the wave-function *ansatz* that is used [27]. In all cases, the objective function is defined as $V_{VQE}(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|\hat{H}_E|\psi(\boldsymbol{\theta})\rangle$. In general, $\nu > 1$ for nontrivial $\hat{H}_E$ and hence measurements in multiple bases are necessary.

Figure 4 shows the results of simulated SPSA and SBO optimization runs of VQE for estimating the ground state energy of H_2 and LiH molecules at various interatomic bond lengths using the unitary coupled cluster ansatz with Hartree-Fock initial state. The H_2 ansatz uses four qubits and $|\boldsymbol{\theta}| = 3$ variational parameters, while the LiH ansatz uses four qubits and $|\boldsymbol{\theta}| = 8$ variational parameters. In Figs. 4(a) and 4(b), under shot noise only, we observe that SBO produces a much more accurate estimate of the ground state energy than SPSA using the same number of energy measurements τM, and it remains more accurate even than using SPSA with τM increased by a factor of 2.5 to 5. In Figs. 4(c) and 4(d), using a simulator with a realistic hardware noise model, we observe that SBO achieves consistently lower estimates of the ground state energy than SPSA. In addition, by taking the parameters $\boldsymbol{\theta}$ found by the noisy optimization runs and evaluating the ansatz with those values on an ideal simulator, we observe that the parameter values obtained by SBO correspond to energy values which are much closer to the exact ground state than those obtained by SPSA.

V. DISCUSSION

From both the QAOA and VQE illustrations in Sec. IV, we observe that SBO often achieves a lower error than SPSA for an equivalent number of iterations or experimental shots. From the QAOA results in Fig. 3, we note that this advantage tends to become more pronounced as the problem complexity increases. We believe these results are a good indication that, for many near-term applications, SBO will achieve better variational parameter estimates with fewer experimental

repetitions than existing techniques such as SPSA. Additionally, because the surrogate function smooths out shot noise, SBO often requires fewer shots per sample point than SPSA to produce a result that is equivalent or better.

One unique feature of SBO is that each iteration requires taking samples for many different parameter settings, as opposed to a technique like SPSA which uses only two sample points per iteration. This may provide a particular advantage for experimental platforms that suffer a high latency cost from loading new circuits between each optimization iteration. If the system can program and execute an entire batch of circuits without paying this latency cost between each circuit, this may provide an additional speed advantage, as well as increased robustness against drift in experimental parameters.

Finding global optima of parametrized quantum circuits often suffers from the problem of "barren plateaus" [27], wherein the objective function, $V(\boldsymbol{\theta})$, exhibits exponentially vanishing gradients, both in the absence and presence of hardware noise, making optimization exceeding challenging. Some techniques around this problem are to formulate *local cost functions* [28] and to utilize variational circuit forms that do not exhibit barren plateaus [29]. We emphasize that SBO is not a technique to address the problem of barren plateaus. Instead, it is an approach to increase the performance of classical optimization loops and to reduce the experimental burden in the VQA setting. These issues are orthogonal to the barren plateaus issue—strategies to construct variational circuits that do not possess barren plateaus *and* the use of more advanced classical optimization techniques like SBO will be critical for scaling VQAs.

A promising avenue for future work is the application of more powerful surrogate models to the VQA setting, e.g., neural network-based methods for approximation [30] might have better rates of convergence with limited experimental data. In addition, building surrogate models to not only perform smoothing and approximation, as we have done here, but also physics-informed error mitigation to counter decoherence is a potentially fruitful direction.

A freely available Python implementation of the Gaussian kernel-based SBO optimizer, including examples of integration with IBM's Qiskit library, is available in Ref. [31].

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under the Quantum Computing Application Teams program. R.S. was also supported by NSF Award No. DMR-1747426. M.S. was also supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers. Sandia National Laboratories is a multimission laboratory managed and operated by NTESS, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. DOE's NNSA under Contract No. DE-NA-0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

APPENDIX A: BOUND ON NUMBER OF CRITICAL POINTS IN A VARIATIONAL COST FUNCTION

As discussed in the main text, a conservative heuristic for choosing the SBO patch size, ℓ, is to choose it such that there are $O(1)$ critical points in the variational cost function within a ℓ^D hypercube. In the following, we will formulate a bound on the total number of critical points in general variational cost functions N_{crit}, in terms of the key parameters in a variational circuit: n, the number of qubits, and D, the number of variational parameters. If we then assume that these critical points are distributed uniformly in parameter space, we require

$$\left(\frac{\ell}{2\pi}\right)^D N_{\text{crit}} \sim 1 \quad \Rightarrow \quad \ell \sim N_{\text{crit}}^{-1/D}. \tag{A1}$$

First, we write a general variational cost function as

$$V(\boldsymbol{\theta}) = \text{tr}\left[\hat{O}\left(\prod_{j=1}^{D} e^{-i\theta_j \hat{H}_j}\right)\rho_0\left(\prod_{j=1}^{D} e^{i\theta_j \hat{H}_j}\right)\right], \tag{A2}$$

where $\hat{H}_j$ are the n-qubit Hamiltonians representing the variational ansatz. The H_j are multiqubit Hamiltonians in general, and to derive an informative bound on N_{crit} we should take into account the complexity in decomposing $e^{-i\theta\hat{H}_j}$ into implementable unitaries. To make things concrete, we will work with a decomposition of the form:

$$e^{-i\theta_j \hat{H}_j} = \hat{U}^{(j)}_{\lambda_j+1}\hat{Z}_{\iota(\lambda_j)}(\theta_j)\hat{U}^{(j)}_{\lambda_j-1}\ldots\hat{Z}_{\iota(1)}(\theta_j)\hat{U}^{(j)}_{1}, \tag{A3}$$

where the $\hat{U}^{(j)}_i$ are θ_j-independent n-qubit unitaries, and $\hat{Z}_{\iota(i)}(\theta_j)$ is a Z rotation on qubit $\iota(i)$. This decomposition implements $e^{-i\theta\hat{H}_j}$ with λ_j rotations by the variational parameter θ_j, and we think of λ_j as parametrizing the complexity of $\hat{H}_j$. Note that λ_j can have a dependence on n since it is the complexity of decomposing an n-qubit unitary. For practical quantum computations, $\lambda_j = O(\text{poly}(n))$. The decomposition above is not unique, but we note that it is experimentally relevant since in many modern quantum computing architectures the only variable angle gates are single qubit $\hat{Z}(\theta)$ rotations.

As an example, consider the decompositions of the two terms in a layer of the QAOA ansatz:

$$e^{-i\beta\sum_{i=1}^{n}\hat{X}_i} = \hat{H}^{\otimes n}\hat{Z}(\beta)^{\otimes n}\hat{H}^{\otimes n}, \tag{A4}$$

$$e^{-i\gamma\sum_{(i,j)\in\mathcal{E}} w_{ij}\hat{Z}_i\hat{Z}_j} = \prod_{k=1}^{|\mathcal{E}|}\text{CNOT}_{i_k,j_k}\hat{Z}_{j_k}(w_{i_kj_k}\gamma)\text{CNOT}_{i_k,j_k}, \tag{A5}$$

where $\hat{H}$ in the first line is a Hadamard gate, and in the second line i_k and j_k index the nodes that edge k connects. $\lambda = 1$ in the first line, and in the second line, $\lambda \leqslant |\mathcal{E}|$. The exact λ for a decomposition of the $\hat{Z}\hat{Z}$ interactions will depend on the QAOA problem graph and which CNOT gates can be executed in parallel – since a CNOT_{ij} and a CNOT_{jm} cannot be simultaneously applied, a node j that has edges to both node i and node m will need its $\hat{Z}\hat{Z}$ interactions implemented sequentially (even assuming full connectivity in the hardware). In general, for a κ-regular problem graph, $\lambda = \kappa$ if the device is fully connected (i.e., a $\hat{Z}\hat{Z}$ gate can be implemented between all qubits connected by an edge in $\mathcal{E}$).

Returning to the general variational cost function in Eq. (A2) and substituting the compiled form of each unitary, we get

$$V(\boldsymbol{\theta}) = \mathrm{tr}\left[\hat{O}\left(\prod_{j=1}^{D}\hat{U}_{\lambda_j+1}^{(j)}\prod_{k=1}^{\lambda_j}\hat{Z}_{\iota(k)}(\theta_j)\hat{U}_k^{(j)}\right)\rho_0\left(\prod_{j=1}^{D}\prod_{k=1}^{\lambda_j}\hat{U}_k^{(j)\dagger}\hat{Z}_{\iota(k)}(-\theta_j)\hat{U}_{\lambda_j+1}^{(j)\dagger}\right)\right]. \tag{A6}$$

Finally, without loss of generality taking $\rho_0 = |0\rangle\langle 0|$ (where $|0\rangle$ is shorthand for the n-qubit state $|0\rangle^{\otimes n}$), we write

$$V(\boldsymbol{\theta}) = \langle 0|\prod_{j=1}^{D}\prod_{k=1}^{\lambda_j}\hat{U}_k^{(j)\dagger}\hat{Z}_{\iota(k)}(-\theta_j)\hat{U}_{\lambda_j+1}^{(j)\dagger}\,\hat{O}\prod_{j=1}^{D}\hat{U}_{\lambda_j+1}^{(j)}\prod_{k=1}^{\lambda_j}\hat{Z}_{\iota(k)}(\theta_j)\hat{U}_k^{(j)}|0\rangle = \langle 0|\prod_{j=1}^{D}\mathcal{U}_j^\dagger\,\hat{O}\prod_{j=1}^{D}\mathcal{U}_j|0\rangle, \tag{A7}$$

where $\mathcal{U}_j \equiv \hat{U}_{\lambda_j+1}^{(j)}\prod_{k=1}^{\lambda_j}\hat{Z}_{\iota(k)}(\theta_j)\hat{U}_k^{(j)}$ are the decompositions. Taking the derivative with respect to one of the angles, we get

$$\begin{aligned}\frac{\partial V(\boldsymbol{\theta})}{\partial\theta_t} = \sum_{r=1}^{\lambda_t}\langle 0|\left[\prod_{j=1}^{t-1}\mathcal{U}_j^\dagger\right]\left(\prod_{k=1}^{r-1}\hat{U}_k^{(t)\dagger}\hat{Z}_{\iota(k)}(-\theta_t)\right)\hat{U}_r^{(t)\dagger}\hat{Z}_{\iota(r)}(-(\pi/2+\theta_t))\left(\prod_{k=r+1}^{\lambda_t}\hat{U}_k^{(t)\dagger}\hat{Z}_{\iota(k)}(-\theta_t)\right)\hat{U}_{\lambda_t+1}^{(t)\dagger}\left[\prod_{j=t+1}^{D}\mathcal{U}_j^\dagger\right]\\ \times\hat{O}\left[\prod_{j=1}^{D}\mathcal{U}_j\right]|0\rangle + \text{c.c.},\end{aligned} \tag{A8}$$

since $\partial/\partial\theta\,\hat{Z}(-\theta) = e^{i(\pi/2+\theta)\hat{Z}}$.

Since all the dependence of this expression on the angles $\boldsymbol{\theta}$ are within the $\hat{Z}$ rotations, and $\hat{Z}(\theta) = \cos(\theta)\hat{I} + i\sin(\theta)\hat{Z}$, we conclude that $\partial V(\boldsymbol{\theta})/\partial\theta_t$, and $V(\boldsymbol{\theta})$ for that matter, are trigonometric polynomials in the angles. Since taking the derivative of $V(\boldsymbol{\theta})$ does not introduce any new $\hat{Z}$ rotations and only shifts the angle of some of the rotations in $V(\boldsymbol{\theta})$, the maximum degree of this trigonometric polynomial is the same for $\partial V(\boldsymbol{\theta})/\partial\theta_t$ and $V(\boldsymbol{\theta})$.

Now we write each decomposition more explicitly as a trigonometric polynomial, using the fact that $\hat{U}\hat{Z}(\theta)\hat{U}^\dagger = \cos(\theta)\hat{I} - i\sin(\theta)\hat{U}\hat{Z}\hat{U}^\dagger$:

$$\mathcal{U}_j = \hat{\Lambda}_{\lambda_j+1}^{(j)}\prod_{k=1}^{\lambda_j}\left(\cos(\theta_j)\hat{I} - i\sin(\theta_j)\hat{\Lambda}_k^{(j)}\right) = \sum_{\{\alpha,\beta\geqslant 1:\alpha+\beta=\lambda_j\}}\cos^\alpha(\theta_j)\sin^\beta(\theta_j)\hat{\Gamma}_{\alpha,\beta}^{(j)}, \tag{A9}$$

where $\hat{\Lambda}_k^{(j)} = \hat{U}_1^{(j)\dagger}\ldots\hat{U}_k^{(j)\dagger}\hat{Z}_{\iota(k)}\hat{U}_k^{(j)}\ldots\hat{U}_1^{(j)}$ for $k\leqslant\lambda_j$, $\hat{\Lambda}_{\lambda_j+1}^{(j)} = \prod_{s=1}^{\lambda_j+1}\hat{U}_s^{(j)}$, and $\hat{\Gamma}_{\alpha,\beta}^{(j)}$ is an operator that is a multiple of some of the $\hat{\Lambda}_k^{(j)}$ that we do not need to specify. Thus $\mathcal{U}_j$ is a trigonometric polynomial with maximum degree λ_j and operator coefficients. Using the same argument for all the $\mathcal{U}_j$, we can write $V(\boldsymbol{\theta})$ as

$$\begin{aligned}V(\boldsymbol{\theta}) &= \langle 0|\prod_{j=1}^{D}\left(\sum_{\{\alpha_j,\beta_j\geqslant 1:\alpha_j+\beta_j=\lambda_j\}}\cos^{\alpha_j}(\theta_j)\sin^{\beta_j}(\theta_j)\hat{\Gamma}_{\alpha_j,\beta_j}^{(j)\dagger}\right)\hat{O}\prod_{j=1}^{D}\left(\sum_{\{\alpha_j,\beta_j\geqslant 1:\alpha_j+\beta_j=\lambda_j\}}\cos^{\alpha_j}(\theta_j)\sin^{\beta_j}(\theta_j)\hat{\Gamma}_{\alpha_j,\beta_j}^{(j)}\right)|0\rangle\\ &= \sum_{\{\alpha_j,\beta_j\geqslant 1:\alpha_j+\beta_j=2\lambda_j\}}\prod_{j=1}^{D}\cos^{\alpha_j}(\theta_j)\sin^{\beta_j}(\theta_j)g_{\alpha_j,\beta_j}^{(j)},\end{aligned} \tag{A10}$$

where in the final line $g_{\alpha_j,\beta_j}^{(j)}\in\mathbb{R}$. Not all of the terms in this sum will be present since $g_{\alpha_j,\beta_j}^{(j)}$ could be zero. However, without further assumptions about the problem, we must assume they are all present. In that case, this is a trigonometric polynomial with maximum degree $d = \sum_{j=1}^{D}2\lambda_j$. And as argued above, all derivatives of $V(\boldsymbol{\theta})$ are trigonometric polynomials with the same maximum degree. Therefore, critical points of the variational cost function are defined by a set of D trigonometric polynomial equations in D angle variables, i.e., $\partial V(\boldsymbol{\theta})/\partial\theta_t = 0$ for all t.

To count the number of critical points, we wish to count the number of solutions to this system of equations. We are unaware of any applicable bounds on the number of solutions of such trigonometric polynomial systems, and therefore proceed by transforming this into a system of standard polynomials. To do so, we introduce new variables, $s_j = \sin(\theta_j)$, and $c_j = \cos(\theta_j)$, $1\leqslant j\leqslant D$, which transforms Eq. (A10) into a degree d polynomial equation in $2D$ variables. Hence in these two variables, the system $\partial V(\boldsymbol{\theta})/\partial\theta_t = 0$ for all t, is a system of D polynomial equations in $2D$ variables, with each equation being degree d. We supplement these with the equations encoding the constraint between c_j and s_j, namely $c_j^2 + s_j^2 - 1 = 0$, for $1\leqslant j\leqslant D$, to arrive at a system of $2D$ polynomial equations in $2D$ variables. D of these equations have degree d, and D of them (the constraints equations) have degree 2.

Now that we have a system of polynomial equations we can formulate a bound on the number of solutions to this system. For this purpose, we use Bézout's theorem, which states that in general, the number of common zeros for a set of n polynomials in n variables in $\mathbb{C}^n$ is given by the product of

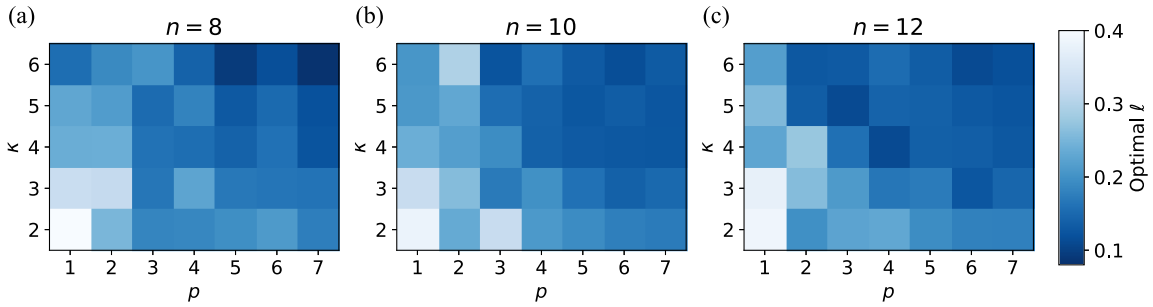

FIG. 5. Numerical estimation of the optimal SBO patch size ℓ for various instances of n-qubit, p-layer MaxCut QAOA on randomly generated κ-regular graphs. Each data point is obtained by averaging the final error of 10 independent SBO runs for each $\ell \in \{0.02, 0.04, \ldots, 0.40\}$, and then using cubic splines to fit the results and find the value of ℓ which minimizes the average error. Each run uses $\tau = 30$ sample points per patch, $K = 60$ measurement shots per sample point, and $M = 100$ optimization iterations.

the degrees of the polynomials [32]. Applying this, we arrive at our bound for the number of critical points in $V(\boldsymbol{\theta})$:

$$N_{\text{crit}} \leqslant (2d)^D = \left(4\sum_{j=1}^{D}\lambda_j\right)^D. \tag{A11}$$

In cases where $\lambda_j = \lambda$ for all j, $N_{\text{crit}} \leqslant (4\lambda D)^D$.

We pause to emphasize that this is a particularly loose bound. Firstly, it is well-known that the bound provided by Bézout's theorem is loose. Compounding this, Bézout's theorem counts the number of zeros over $\mathbb{C}^{2D}$, whereas we are concerned with zeros over the domain $[-1, 1]^{2D}$. Thus, although we do not expect this bound to be tight, it does highlight some useful parameter dependencies: (i) there is an exponential dependence on the number of parameters D and (ii) the only dependence on n is through the ansatz complexity λ_j.

Returning to the scaling of the patch size parameters, ℓ, Eq. (A1), we arrive at

$$\ell \gtrsim \left(4\sum_{j=1}^{D}\lambda_j\right)^{-1}, \tag{A12}$$

which, taking into account $\lambda_j = \mathcal{O}(\text{poly}(n))$, results in the scaling $\ell = \Omega(1/\text{poly}(D,n))$.

As an example, consider QAOA on κ-regular graphs. For a QAOA variational ansatz with p layers, $D = 2p$, and as discussed above, λ_j alternates between 1 and κ. Therefore, for this example we get $\ell \gtrsim (4p(\kappa+1))^{-1}$. Note that since the λ_j have no dependence on n in this case, we get n-independent scaling.

In Fig. 5, we present numerically determined optimal ℓ for QAOA on κ-regular graphs as we vary the relevant parameters: the number of QAOA layers p (where $D = 2p$), the number of qubits n, and the graph regularity κ. The independence of ℓ from n is supported by this data, as the variation of the surfaces is negligible as n is varied. In order to test the scaling prediction above, we fit the data in Fig. 5(a) to a functional form

$$\ell = \beta(\kappa p + \kappa)^{-\alpha}, \tag{A13}$$

where the parameters α, β allow for numerical factors in the relations (A1) and (A11), and accommodate for the looseness of the bound in Eq. (A11). The fit of this form to the $n = 8$ data, along with the error of the fit, is shown in Fig. 6.

APPENDIX B: IMPLEMENTATION NOTES FOR SIMULATIONS IN SEC. IV

Here we collect the implementation notes and hyperparameter choices for the simulations presented in the main text.

1. Quantum approximate optimization algorithm

The QAOA circuit simulations were implemented using the PYQAOA package [33]. SPSA was implemented using the noisyopt package [34]. L-BFGS-B was implemented using the `optimize.minimize` function in scipy.

Simulation hyperparameters. For Fig. 3, we chose hyperparameters by manual scans to optimize the performance of both SBO and SPSA on these problems. We use $\tau = 20$ for SBO, while $\tau = 2$ for SPSA by definition. In Figs. 3(a)–3(c), we use SBO parameter $\ell = (0.2, 0.2, 0.1)$ and SPSA parameter $a = (0.2, 0.2, 0.1)$, respectively. We use SPSA parameters $c = 0.2$, $\alpha = 0.602$, and $\gamma = 0.101$ for all simulations presented in this figure.

2. Variational quantum eigensolver

VQE simulations were implemented via qiskit using unitary coupled cluster (UCC) ansatz with Hartree-Fock initial state, following the procedure described in Ref. [35]. For SPSA, we used the implementation provided by qiskit in Ref. [36], which includes automatic hyperparameter calibration.

Simulation hyperparameters. In Figs. 4(a)–4(d), we used $\tau = (4, 5, 8, 10)$ and $\ell = (0.15, 0.1, 0.15, 0.1)$ for SBO, respectively, while $\tau = 2$ for SPSA by definition.

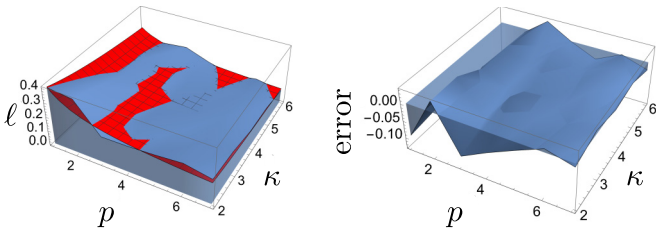

FIG. 6. Fit (left) and error in fit (right) of data in Fig. 5(a) to the functional form shown in Eq. (A13) with fitting parameters $\alpha = 0.5$ and $\beta = 0.7$.

[1] A. B. Magann, C. Arenz, M. D. Grace, T.-S. Ho, R. L. Kosut, J. R. McClean, H. A. Rabitz, and M. Sarovar, From pulses to circuits and back again: A quantum optimal control perspective on variational quantum algorithms, PRX Quantum **2**, 010101 (2021).

[2] 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).

[3] W. Lavrijsen, A. Tudor, J. Müller, C. Iancu, and W. de Jong, Classical optimizers for noisy intermediate-scale quantum devices, in *Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, Piscataway, NJ, 2020), pp. 267–277.

[4] X. Bonet-Monroig, H. Wang, D. Vermetten, B. Senjean, C. Moussa, T. Back, V. Dunjko, and T. E. O'Brien, Performance comparison of optimization methods on variational quantum algorithms, Phys. Rev. A **107**, 032407 (2023).

[5] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker, Surrogate-based analysis and optimization, Prog. Aerosp. Sci. **41**, 1 (2005).

[6] A. Forrester, A. Sobester, and A. Keane, *Engineering Design Via Surrogate Modelling* (Wiley, Hoboken, NJ, 2008).

[7] J. M. Kübler, A. Arrasmith, L. Cincio, and P. J. Coles, An adaptive optimizer for measurement-frugal variational algorithms, Quantum **4**, 263 (2020).

[8] R. Sweke, F. Wilde, J. Meyer, M. Schuld, P. K. Faehrmann, B. Meynard-Piganeau, and J. Eisert, Stochastic gradient descent for hybrid quantum-classical optimization, Quantum **4**, 314 (2020).

[9] A. Gu, A. Lowe, P. A. Dub, P. J. Coles, and A. Arrasmith, Adaptive shot allocation for fast convergence in variational quantum algorithms, arXiv:2108.10434.

[10] C. N. Self, K. E. Khosla, A. W. R. Smith, F. Sauvage, P. D. Haynes, J. Knolle, F. Mintert, and M. Kim, Variational quantum algorithm with information sharing, npj Quantum Inf. **7**, 116 (2021).

[11] S. Tamiya and H. Yamasaki, Stochastic gradient line bayesian optimization: Reducing measurement shots in optimizing parameterized quantum circuits, arXiv:2111.07952.

[12] G. Iannelli and K. Jansen, Noisy bayesian optimization for variational quantum eigensolvers, arXiv:2112.00426.

[13] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, Learning to optimize variational quantum circuits to solve combinatorial problems, Proc. AAAI Conf. Artif. Intellig. **34**, 2367 (2020).

[14] K. J. Sung, J. Yao, M. P. Harrigan, N. C. Rubin, Z. Jiang, L. Lin, R. Babbush, and J. R. McClean, Using models to improve optimizers for variational quantum algorithms, Quantum Sci. Technol. **5**, 044008 (2020).

[15] J. Mueller, W. Lavrijsen, C. Iancu, and W. de jong, Accelerating noisy VQE optimization with gaussian processes, arXiv:2204.07331.

[16] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[17] K. Temme, S. Bravyi, and J. M. Gambetta, Error Mitigation for Short-Depth Quantum Circuits, Phys. Rev. Lett. **119**, 180509 (2017).

[18] S. Endo, S. C. Benjamin, and Y. Li, Practical Quantum Error Mitigation for Near-Future Applications, Phys. Rev. X **8**, 031027 (2018).

[19] A. Kandala, K. Temme, A. D. Corcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, Error mitigation extends the computational reach of a noisy quantum processor, Nature **567**, 491 (2019).

[20] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, Error mitigation with clifford quantum-circuit data, Quantum **5**, 592 (2021).

[21] H. Wendland, *Scattered Data Approximation*, Cambridge Monographs on Applied and Computational Mathematics (Cambridge University Press, Cambridge, UK, 2004).

[22] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research (Springer, Berlin, 2006).

[23] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Monographs on Statistics and Applied Probability (Chapman and Hall, London, 1986).

[24] J. C. Spall, An overview of the simultaneous perturbation method for efficient optimization, Johns Hopkins APL Tech. Dig. **19**, 4 (1998).

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

[26] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[27] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

[28] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, Nat. Commun. **12**, 1791 (2021).

[29] H. R. Grimsley, G. S. Barron, E. Barnes, S. E. Economou, and N. J. Mayhall, ADAPT-VQE is insensitive to rough parameter landscapes and barren plateaus, arXiv:2204.07179.

[30] K. Lee, N. A. Trask, R. G. Patel, M. A. Gulian, and E. C. Cyr, Partition of unity networks: deep hp-approximation, arXiv:2101.11256.

[31] https://github.com/sandialabs/sbovqaopt.

[32] E. Penchévre, Etienne bézout on elimination theory, arXiv:1606.03711.

[33] https://github.com/gregvw/pyQAOA.

[34] https://github.com/andim/noisyopt.

[35] https://qiskit.org/documentation/nature/tutorials/03_ground_state_solvers.html.

[36] https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.SPSA.html.