IEEE Transactions on Quantum Engineering

Received 31 July 2023; revised 7 October 2023; accepted 12 October 2023; date of publication 16 October 2023; date of current version 16 November 2023.

Digital Object Identifier 10.1109/TQE.2023.3325167

Bayesian Optimization for QAOA

SIMONE TIBALDI[1,2], DAVIDE VODOLA[1,2], EDOARDO TIGNONE[3], AND ELISA ERCOLESSI[1,2]

[1]Dipartimento di Fisica e Astronomia dell'Università di Bologna, 40127 Bologna, Italy
[2]INFN, Sezione di Bologna, 40127 Bologna, Italy
[3]Leithà S.r.l. Unipol Group, 40128 Bologna, Italy

Corresponding author: Simone Tibaldi (e-mail: simone.tibaldi2@unibo.it).

This work was supported by the International Foundation Big Data and Artificial Intelligence for Human Development (IFAB) through the Project "Quantum Computing for Applications." The work of E. Ercolessi, S. Tibaldi, and D. Vodola was supported by Istituto Nazionale di Fisica Nucleare through the Project "QUANTUM" and the QuantERA 2020 Project "QuantHEP."

This work did not involve human subjects or animals in its research.

ABSTRACT The quantum approximate optimization algorithm (QAOA) adopts a hybrid quantum-classical approach to find approximate solutions to variational optimization problems. In fact, it relies on a classical subroutine to optimize the parameters of a quantum circuit. In this article, we present a Bayesian optimization procedure to fulfill this optimization task, and we investigate its performance in comparison with other global optimizers. We show that our approach allows for a significant reduction in the number of calls to the quantum circuit, which is typically the most expensive part of the QAOA. We demonstrate that our method works well also in the regime of slow circuit repetition rates and that a few measurements of the quantum ansatz would already suffice to achieve a good estimate of the energy. In addition, we study the performance of our method in the presence of noise at gate level, and we find that for low circuit depths, it is robust against noise. Our results suggest that the method proposed here is a promising framework to leverage the hybrid nature of QAOA on the noisy intermediate-scale quantum devices.

INDEX TERMS Bayesian optimization, quantum approximate optimization algorithm (QAOA), quantum optimization.

I. INTRODUCTION

Hybrid quantum-classical variational algorithms [1], [2], [3] play a central role in the current research on noisy intermediate-scale quantum (NISQ) devices [4]. In a hybrid variational setting, a classical computer is entrusted with the nontrivial task of optimizing the parameters of a quantum state. These algorithms implement a heuristic protocol to approximately solve variational problems including combinatorial optimization tasks, which are ubiquitous and have great practical importance [5], and are, indeed, one of the main drivers of the industry interest toward quantum computing applications. Unfortunately, the problems belonging to this class are hard to solve with classical methods [6]. In this article, we focus on the Max-Cut and the max independent set (MIS) problems defined on specific graph instances.

Among the hybrid variational algorithms, the quantum approximate optimization algorithm (QAOA) [7] is extensively studied [8] as a promising algorithm to investigate quantum speedups on NISQ devices and has been implemented on several experimental platforms, such as Rydberg atom arrays [9], superconducting processors [10], trapped-ions simulators [11], as well as simulated on classical devices [12].

Similarly to other hybrid variational algorithms, QAOA consists of a sequence of parametrized quantum gates applied to a wavefunction, on which an expectation value of some operator, typically the Hamiltonian, is reconstructed from measurements. The task of the classical subroutine is to optimize the gate parameters in order to minimize such expectation value. Every variational quantum algorithm, therefore, requires the estimation of the expectation value of a Hamiltonian [13]. Moreover, the interplay between the classical and quantum parts of the algorithm entails to run the quantum circuit a large number of times, thus being expensive in terms of resources. Finally, there is the notorious problem of barren plateaus (BPs), which are large portions of the optimization landscape in which the gradient becomes exponentially small with the number of qubits and layers [14]. This phenomenon was proven to be caused also by the presence of noise [15] or by the use of a cost function depending on global observables [16].

To overcome these issues, in fact, an efficient classical optimization routine is crucial. Different techniques have been proposed for optimizing variational quantum circuits, e.g., Nelder–Mead [17], machine learning [18], gradient descent [19], iterative schemes [8], [20], [21], Gaussian processes [22], [23], and Bayesian methods [15], [24], [25], [26], [27]. In particular, to tackle the problem of BPs, it might seem logical to avoid the calculation of the gradient. However, in [28], it was shown that gradient-free optimizers such as COBYLA, Powell, and Nelder–Mead suffer from BPs too. Here, we focus on a Bayesian optimization framework, which is suitable for gradient-free global optimization of black-box functions [29], [30]. We explore its behavior in comparison with other global optimizers and we show that the convergence rate to a local minimum is faster. We demonstrate that the Bayesian approach is efficient in terms of a number of circuit runs and is robust against noise sources.

The rest of the article is organized as follows. In Section II, we introduce the QAOA algorithm. In Section III, we give a detailed presentation of the Bayesian algorithm. In Section IV, we present the result of applying this method to QAOA, compare it to other global optimization methods, and evaluate its performance with a limited number of circuit runs and against simulated quantum noise. Finally, Section V concludes this article.

II. QAOA FOR COMBINATORIAL PROBLEMS

The QAOA is a variational quantum algorithm that performs hybrid quantum-classical optimization [7]. Given a cost function $C(z)$ with $z = (z_1, \ldots, z_i, \ldots, z_N)$ with $z_i \in \{0, 1\}$, QAOA aims at finding the bitstring $z^\star$ that minimizes the cost. In order to do so, the cost function is translated into a quantum operator H_C. This is done by replacing each binary variable z_i with a two-level quantum state $|z_i\rangle$ and each z_i term appearing in the cost function with a Pauli matrix Z_i. Since Z_i is diagonal on the qubit $|z_i\rangle$, H_C is diagonal in the computational basis $|z\rangle = |z_1 \ldots z_i \ldots z_N\rangle$ for N qubits. This means that applying H_C to $|z\rangle$ gives the classical cost $C(z)$ of such string, i.e.

$$H_C\,|z\rangle = C(z)\,|z\rangle\,. \tag{1}$$

The QAOA circuit consists of preparing an initial state of N qubits, usually $|+\rangle = \sum_z |z\rangle/\sqrt{2^N}$, and then applying two unitary operators alternatively: one generated by H_C and the other generated by $H_M = \sum_i X_i$, where X_i is the flip (NOT) operator acting on the ith qubit. The two unitaries together form one layer of the circuit and the operation is iterated for a number of layers p, which is called the depth of the circuit. The problems that we consider in this work (see Section IV) have a cost function at most quadratic in the binary variables. This means that H_C comprehends only Z_i and Z_iZ_j terms. For this reason, we can implement H_C by applying only rotations e^{-itZ} on the qubits and gates $e^{-itZ_iZ_j}$ on the pairs of qubits. Putting all together, the QAOA circuit prepares the state

$$|\boldsymbol{\theta}\rangle = \prod_{l=1}^{p} e^{-i\beta_l H_M} e^{-i\gamma_l H_C} |+\rangle \tag{2}$$

where $\boldsymbol{\theta} = (\boldsymbol{\gamma}, \boldsymbol{\beta})$ are $2p$ parameters. By measuring the state $|\boldsymbol{\theta}\rangle$ in the computational basis, we obtain the probability amplitudes of each bitstring. In this way, by using relation (1), an estimate of the energy $E(\boldsymbol{\theta}) = \langle\boldsymbol{\theta}|H_C|\boldsymbol{\theta}\rangle$ is obtained. This energy is then fed to a classical routine, which looks for the set of angles $\boldsymbol{\theta}^\star = (\boldsymbol{\gamma}^\star, \boldsymbol{\beta}^\star)$ that minimizes $E(\boldsymbol{\theta})$. Several strategies have been proposed for finding the optimal parameters $\boldsymbol{\theta}^\star$. In this work, we rely on Bayesian optimization.

III. BAYESIAN OPTIMIZATION

Bayesian optimization is a global optimization strategy, which allows us to find within relatively few evaluations the minimum of a noisy, black-box objective function $f(\boldsymbol{\theta})$ that is, in general, expensive to evaluate [31]. The algorithm can be summarized as follows.

1) It treats the objective function f as a random function by choosing a prior (also called surrogate model) for f. Several choices for the surrogate model are possible [29]; in this work, we adopt the so-called Gaussian process [32].
2) The prior is then updated through the likelihood function by gathering observations of f and, therefore, forming the posterior distribution.
3) The posterior distribution is finally used to construct an auxiliary function, called the acquisition function, that is, in general, cheap to evaluate.

The point where the acquisition function is maximized gives the next point where f will be evaluated [30]. See Appendix A for an overview of Bayesian terminology.

Since Bayesian optimization requires no previous knowledge on f, it appears to be a well-suited technique for optimizing the parameters of a variational circuit running on NISQ devices.

In the following sections, we describe the Gaussian process, the optimization routine, and the acquisition function in detail.

A. GAUSSIAN PROCESS

Since the function $f(\boldsymbol{\theta})$ ($\boldsymbol{\theta} \in A \subset \mathbb{R}^d$) to be optimized is unknown, we may think of it as belonging to a random process, i.e., an infinite collection of random variables defined for every point $\boldsymbol{\theta} \in A$. A random process is called Gaussian if the joint distribution of any finite collection of those random variables is a multivariate normal distribution defined by a mean function $\mu(\boldsymbol{\theta})$ and covariance (or kernel) function $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$ [32]. The mean function is the expected value of the function f while the kernel estimates the deviations of the mean function from the value of f

$$\mu(\boldsymbol{\theta}) = \mathbb{E}[f(\boldsymbol{\theta})] \tag{3}$$

Algorithm 1: Pseudocode For Bayesian Optimization.

Set the prior on f as a Gaussian process;
Evaluate f at N_W different points θ_i;
Define the initial training set $\mathscr{D} = \{(\theta_i, f(\theta_i))\}_{i=1}^{N_W}$;
Compute the hyperparameters σ^2, ℓ based on $\mathscr{D}$;
Set the guess for the minimum of f to $f_m = \min[\{f(\theta_i)\}_{i=1}^{N_W}]$;
while $n \leq N_{\text{BAYES}}$ **do**
  Update the posterior distribution on f using the training set $\mathscr{D}$;
  Compute the acquisition function with the updated posterior;
  Find $\tilde{\theta}$ that maximizes the acquisition function;
  Evaluate $f(\tilde{\theta})$;
  if $f(\tilde{\theta}) < f_m$ **then**
    Set the guess for the minimum of f to $f_m = f(\tilde{\theta})$;
  end
  Append $(\tilde{\theta}, f(\tilde{\theta}))$ to $\mathscr{D}$;
  Compute the new hyperparameters σ^2, ℓ;
  Increment n;
end
return f_m

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \mathbb{E}[(f(\boldsymbol{\theta}) - \mu(\boldsymbol{\theta}))(f(\boldsymbol{\theta}') - \mu(\boldsymbol{\theta}'))] \tag{4}$$

where $\mathbb{E}$ denotes the expectation w.r.t. the infinite collection of functions belonging to the random process. Conceptually, the mean encloses the knowledge of the function f to reconstruct while k represents the uncertainty we have on such reconstruction.

Since we assume f to be smooth, we choose for k the Matérn kernel, a stationary kernel [32] that depends on the distance between the points $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, defined as

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma^2 \left(1 + \frac{\sqrt{3}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2}{\ell}\right) e^{-\frac{\sqrt{3}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2}{\ell}} \tag{5}$$

where $\|\cdot\|_2$ is the 2-norm and σ^2 and ℓ are two hyperparameters characterizing the Gaussian process. The hyperparameter σ^2 defines the variance of the random variables whereas ℓ is a characteristic length-scale which regulates the decay of the correlation between points: in the limit of $\ell \to \infty$ all points are equally correlated, for $\ell \to 0$ all points are uncorrelated.

B. BAYESIAN OPTIMIZATION ALGORITHM

The main steps of the algorithm for Bayesian optimization can be summarized in the pseudocode in Algorithm 1 (see also Appendix B for details).

The optimization starts with a warmup phase where a number N_W of evaluations of the objective function f is performed. These evaluations take place at randomly chosen values of the points $\boldsymbol{\theta}_i$ and are collected in the training set $\mathcal{D} = \{(\boldsymbol{\theta}_i, y_i = f(\theta_i))\}_{i=1}^{N_W}$ of the optimization. Given the set $\mathcal{D}$, we define the design matrix $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{N_W})$ with the points and the vector $\boldsymbol{y} \in \mathbb{R}^{N_W}$ with the observations via $\boldsymbol{y} = (y_1, \ldots, y_{N_W})$. We form the covariance matrix $\boldsymbol{K} \in \mathbb{R}^{N_W \times N_W}$ by evaluating the covariance function in (4) for each pair of points $\boldsymbol{\theta}_i, \boldsymbol{\theta}_j \in \boldsymbol{\Theta}$ via

$$\boldsymbol{K}_{i,j} = k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \tag{6}$$

where $\boldsymbol{K}_{i,j}$ denotes the (i, j) element of the matrix $\boldsymbol{K}$. The hyperparameters entering the kernel function (5) are optimized at this step, as explained in Section III-D.

The training set will be used at each step of the optimization to incorporate the acquired knowledge in the Gaussian process. This happens in two steps. First, the Gaussian process prior is conditioned on the observations in $\mathcal{D}$ [32]. Conditioning is equivalent to a Bayesian step in which we multiply the prior with the likelihood, thus obtaining a posterior distribution (see Appendix A). Thanks to the properties of Gaussian distributions, the posterior is still described by a Gaussian process multinomial distribution but it is characterized by a posterior mean μ' and covariance k' given by

$$\mu' = \boldsymbol{\kappa}^T \cdot \boldsymbol{K}^{-1} \cdot \boldsymbol{y} \tag{7}$$

$$k' = k(\boldsymbol{\theta}, \boldsymbol{\theta}) - \boldsymbol{\kappa}^T \cdot \boldsymbol{K}^{-1} \cdot \boldsymbol{\kappa}. \tag{8}$$

Here, $\boldsymbol{\theta}$ is a generic point in A and $\boldsymbol{\kappa}$ is a column vector formed by evaluating the covariance function k between the generic point $\boldsymbol{\theta}$ and all the points in $\boldsymbol{\Theta}$, i.e., its jth element is $\kappa_j = k(\boldsymbol{\theta}, \boldsymbol{\theta}_j)$. Equation (7) shows that the new mean is a linear combination of the observations $\boldsymbol{y}$.

C. ACQUISITION FUNCTION

The next step in the Bayesian optimization involves computing the acquisition function, whose maximum gives the next point at which to evaluate the objective function. A common choice of an acquisition function is the expected improvement (EI): this function suggests which points, on average, improve on f_m the most [30]. This choice corresponds to defining the acquisition function $\text{EI}(\boldsymbol{\theta}) = \mathbb{E}[u(\boldsymbol{\theta})]$ as the average value of the utility function $u(\boldsymbol{\theta}) = \max[0, f_m - f(\boldsymbol{\theta})]$ such that the lower $f(\boldsymbol{\theta})$ is with respect to the current minimum, the larger the utility $u(\boldsymbol{\theta})$ will be.

By considering that $f(\boldsymbol{\theta})$ is a Gaussian process, we can obtain an analytical expression for $\text{EI}(\boldsymbol{\theta})$ as

$$\text{EI}(\boldsymbol{\theta}) = \Phi(z)(f_m - \mu') + \phi(z)k' \tag{9}$$

where μ' and k' are obtained for the point $\boldsymbol{\theta}$ by using (7) and (8); $\Phi(\cdot)$ and $\phi(\cdot)$ are, respectively, the cumulative distribution function and the probability density function of the standard normal distribution and the quantity z is defined as $z = (f_m - \mu')/k'$. The two terms in (9) represent the tradeoff between exploitation and exploration: The first term, being proportional to the difference between the current minimum and the mean value of the posterior, brings the optimization toward points with lower μ', whereas the second one promotes points with larger k', i.e., with higher uncertainty.

The point $\tilde{\boldsymbol{\theta}}$ that maximizes the acquisition function is then added to the training set $\mathcal{D}$ and the algorithm's loop is repeated (as written in Algorithm 1). Its value is found by using the differential evolution algorithm [33], a population-based metaheuristic search algorithm (see Appendix C for details).

D. HYPERPARAMETERS

We are now only left with the task of picking the best hyperparameters σ, ℓ for the Matérn kernel. This is typically done by considering the marginal likelihood [32] (and Appendix A)

$$p(\boldsymbol{y}|\boldsymbol{\Theta}) = \int p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\Theta}) p(\boldsymbol{f}|\boldsymbol{\Theta}) d\boldsymbol{f} \tag{10}$$

where the prior $p(\boldsymbol{f}|\boldsymbol{\Theta})$ and the likelihood $p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\Theta})$ are Gaussian and the marginalization is done over the function values $\boldsymbol{f}$. Given the Gaussian nature of the likelihood and the prior, a closed form of the log marginal likelihood can be obtained (for the standard derivation of this formula see, for example, [32])

$$\log p(\boldsymbol{y}|\boldsymbol{\Theta}) = -\frac{1}{2}\boldsymbol{y}^T \cdot \boldsymbol{K}^{-1} \cdot \boldsymbol{y} - \frac{1}{2}\log\det \boldsymbol{K} - \frac{N}{2}\log 2\pi \tag{11}$$

where N is the number of observations in the design matrix $\boldsymbol{\Theta}$. In (11), the first term specifies how well the process fits the data and the second term instead acts as a regularization factor on the elements of the kernel matrix. When fitting the Gaussian process to a new set of points, the best hyperparameters $(\tilde{\sigma}^2, \tilde{\ell})$ can be found by maximizing the log marginal likelihood in (11). For the optimization of $\log p(\boldsymbol{y}|\boldsymbol{\Theta})$, we use the quasi-Newton method `L-BFGS` [34] with multiple restarting points, which proved to be efficient on the flat landscape of the likelihood (see Appendix B for details).

IV. RESULTS

In this section, we apply the Bayesian optimization to the QAOA parameters. We consider two well-known combinatorial problems defined on graphs: 1) the Max-Cut and 2) the MIS.

Max-Cut: Given a graph $G = (V, E)$ where V is the set of nodes and E the set of edges, the Max-Cut problem consists of finding a partition of the graph's vertices V, $P = \{V_0, V_1\}$, such that the number of edges between V_0 and its complement V_1 is as large as possible. It is known to be a NP-hard problem [35]. We can define the assignment of the nodes to the sets V_0 and V_1 by labelling with the label "0" the nodes $v \in V_0$ and with the label "1" the nodes $v \in V_1$. In these terms, the Max-Cut consists of finding the largest number of edges connecting the bits labeled with "0" to the bits labeled with "1." On a quantum computer, the labels 0 and 1 are replaced by the computational basis states $|0\rangle$ and $|1\rangle$, and the cost Hamiltonian can be written as

$$H_C^{\mathrm{MC}} = -\sum_{(i,j)\in E} (1 - Z_i Z_j)/2. \tag{12}$$

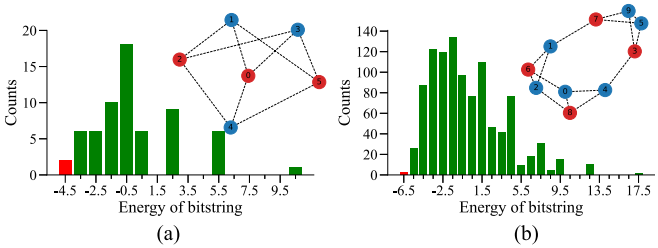

FIG. 1. Energy distributions of graphs. (a) MIS: Distribution of the energies of the possible bitstrings for the graph of 6 nodes (shown in the inset, nodes in red correspond to one solution). The red bar to the left highlights the two solution bitstrings of the MIS problem on such a graph. (b) Max-Cut: Distribution of the energies of the possible bitstrings for the graph of 10 nodes (shown in the inset, nodes in red correspond to one solution). The red bar highlights the two solution bitstrings of the Max-Cut problem on such a graph.

The states with minimum energy then represent the bitstrings that maximize the number of edges with two opposite values on their vertices.

MIS: The MIS problem consists of finding the largest number of graph nodes, which are not adjacent. The corresponding cost Hamiltonian in its classical formulation [36] is

$$C(x) = -\sum_{i\in V} x_i + \omega \sum_{(i,j)\in E} x_i x_j \tag{13}$$

where $x_i = 0, 1$, and ω is a parameter that balances the effect of the first term (which maximizes the number of bits in $|1\rangle$) and the second one (which prevents neighbor bits to be activated at the same time). In order to translate the problem into its quantum version, we make the variable substitution $x_i = (1 - z_i)/2$ so that $z_i = +1, -1$. Then, we replace each z_i with Z_i and obtain the quantum Hamiltonian (discarding constant terms)

$$H_C^{\mathrm{MIS}} = \sum_i \frac{Z_i}{2} + \omega \sum_{(i,j)\in E} \frac{Z_i Z_j - Z_i - Z_j}{4}. \tag{14}$$

During the optimization process, we monitor the approximation ratio $R = E(\boldsymbol{\theta})/E_{GS}$ [7] where E_{GS} is the energy of the solution bitstring. Since $E_{GS} < 0$ [due to our definition of the problems Hamiltonians (12), (14)], $|E(\boldsymbol{\theta})| \leq |E_{GS}|$, and thus, R is upper bounded by 1. We also look at the fidelity defined as $F = |\langle \boldsymbol{\theta}|z^\star\rangle|^2$ where $|z^\star\rangle$ is the state that encodes the solution. The following results are obtained on two 3-regular graphs of 6 and 10 nodes, which are plotted in the insets of Fig. 1(a) and (b).

QAOA at Low Versus Large Depth: We start by looking at the QAOA at depth $p = 1$ for the 6 nodes graph. It corresponds to a shallow circuit that depends only on two parameters $\boldsymbol{\theta_1} = (\gamma_1, \beta_1)$. We consider the MIS problem, and we plot the landscapes of both the energy $E(\boldsymbol{\theta}_1)$ and the fidelity $F(\boldsymbol{\theta}_1)$ [see Fig. 2(a) and (b), respectively] for values of $\gamma_1, \beta_1 \in [0, \pi]$ due to the symmetry of the problem. We see that the landscape of the energy, which is the function to minimize, is rather flat with two global maxima and minima, corresponding to the best solutions possible at $p = 1$.

IEEE Transactions on Quantum Engineering

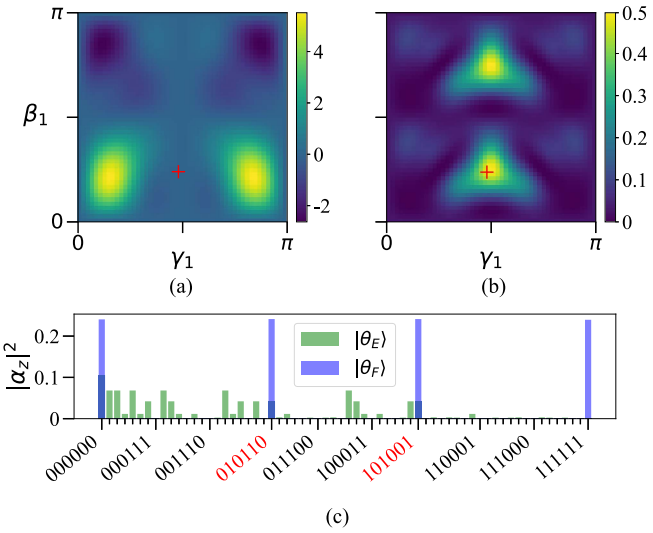

FIG. 2. QAOA at $p = 1$. (a) Landscape of the energy $E(\theta_1)$ obtained on the 6 nodes graph solving the MIS problem. The red cross indicates the angles corresponding to the final state $|\theta_F\rangle$ with the largest fidelity. (b) Landscape of the fidelity $F(\theta_1)$. (c) Squared amplitudes of the two states $|\theta_E\rangle$, $|\theta_F\rangle$. The solution bitstrings are highlighted in red. Values of the qubits are given in the order shown in the inset of Fig. 1(a).

Interestingly, we find that the QAOA state $|\boldsymbol{\theta_E}\rangle = \sum_z \alpha_{z,E}\,|z\rangle$, corresponding to the parameters that minimize the energy, is not the state $|\boldsymbol{\theta_F}\rangle = \sum_z \alpha_{z,F}\,|z\rangle$ with the largest fidelity. To see how they differ, we plot the squared amplitudes $|\alpha_{z,E}|^2$ and $|\alpha_{z,F}|^2$ of both states in Fig. 2(c) as histograms. The fidelity of $|\boldsymbol{\theta_F}\rangle$ w.r.t. the solution $|z^\star\rangle$ is, as expected, much larger than that of $|\boldsymbol{\theta_E}\rangle$, yet the latter has a lower energy because it has many nonzero amplitudes along excited states with low energy. This unravels the problem of optimizing the QAOA parameters by only looking at the energy $E(\boldsymbol{\theta})$. There is a large concentration of excited states with energy comparable to the energy of the ground state, as shown in the histograms of panels (a) and (b) of Fig. 1. It is difficult to increase the amplitude corresponding to the solution when many other states can contribute with low values of the energy.

The difference between the lowest energy and highest fidelity points is guaranteed to disappear theoretically for $p \to \infty$. For this reason, we apply Bayesian optimization to the problem and we show in Fig. 3 that the approximation ratio and fidelity both tend to 1 for $p \sim 12$. Yet, we already see a good performance at $p = 4$ where $R \sim 0.7$ and we have $F \sim 0.5$ meaning about a 50% chance of measuring the solution on the state obtained with QAOA.

Comparing Resources: Increasing the depth of a variational circuit increases the number of parameters that must be optimized. In turn, this is expected to increase the number of calls to the quantum circuit needed to reach a good approximate solution, which is a problem in the current NISQ era, since running a quantum circuit can be costly due to both state preparation routines and recalibrations of the device.

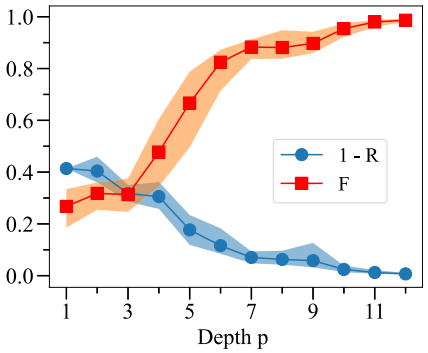

FIG. 3. Results increasing depth. Average approximation ratio (plotted as $1 - R$) and fidelity F for increasing values of circuit depth from 1 to 12 over 50 runs. Shaded areas correspond to one standard deviation. Results were obtained on the 6 nodes graph of Fig. 1(a).

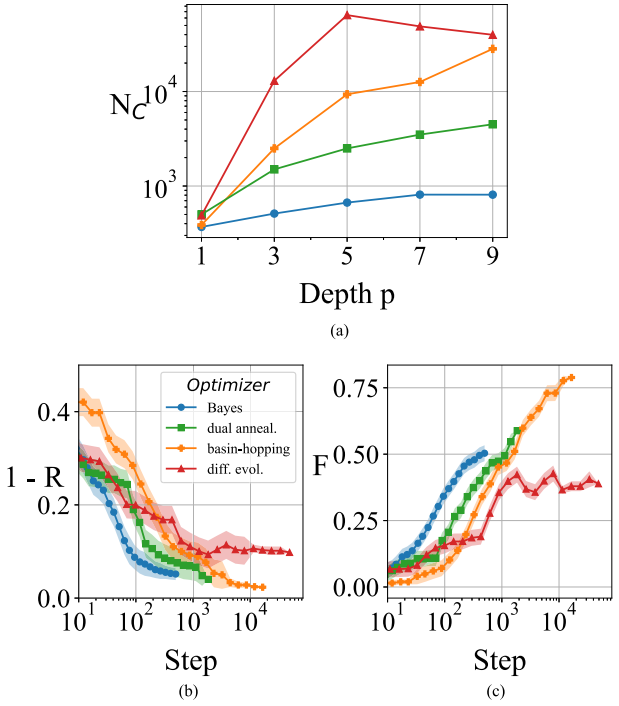

FIG. 4. Comparison among optimizers. (a) Plot shows the average number of calls N_C to the quantum circuit of each optimizer in order to obtain the same approximation ratio as the Bayesian optimization. (b) and (c) Average approximation ratio (plotted as $1 - R$) and fidelity during the optimization with the different methods at $p = 7$ over 30 runs. Shaded areas correspond to one standard deviation. Results are obtained on the 10-node graph of Fig. 1(b).

Bayesian optimization mitigates such problems and allows us to achieve a good approximate solution within relatively fewer calls to the circuit compared to other global optimization methods. To show this, we ran differential evolution, basin-hopping, and dual annealing (see Appendixes C and D for details) on the 10-node graph of Fig. 1(b) for the Max-Cut problem at different depths. Fig. 4(a) shows the average number of calls to the circuit that the other optimizers need in order to reach the same approximation ratio of Bayesian optimization. To gain more insight, we plot in panels (b) and (c) of Fig. 4, the average approximation ratio and fidelity

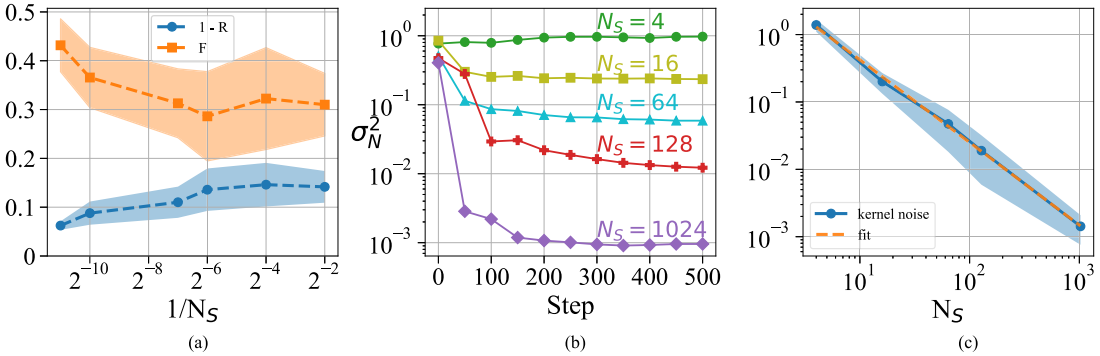

FIG. 5. **Slow measurements. (a) Average approximation ratio (plotted as $1 - R$) and fidelity (F) as a function of the number of shots N_S. The $1/N_S = 0$ points indicates the exact evaluation of the energy $E(\theta)$. Shaded areas correspond to one standard deviation. (b) Kernel noise σ_N^2 learned by fitting the data at each step of the optimization for different numbers of shots N_S. (c) Average kernel noise learned by the Gaussian process as a function of N_S (blue circles). The plot also shows a linear fit ($\sim 1/N^{1.1}$, orange line) of the logarithm of the data.**

during the run of the algorithm for each method at $p = 7$. We see that Bayesian optimization stops at a lower R than basin-hopping and dual annealing, but it reaches a value of $R = 95\%$ with ~ 500 runs of the circuit compared to the other two methods which take, in order, 1400 and 10 800.

For the noiseless circuit, it is very clear (see Fig. 4) that the Bayesian approach can mitigate this problem better than any other tested techniques, from different points of view (such as the number of calls, number of steps, and number of measurements). Let us stress that this approach gives a fidelity that is always higher than the other methods at a fixed number of steps [and fixed p, see Fig. 4(c)]. All the simulations were run on Python. The quantum circuit was simulated using the `qutip` package [37], the Bayesian optimization part was built by expanding the class GaussianProcessRegressor of `scikit-learn` [38], and the other global optimizers were implemented using the standard `scipy.optimize` class [39].

Slow Measurements: The energy $E(\boldsymbol{\theta})$ is obtained by measuring the QAOA state after running the circuit: We refer to these two operations combined together as a "shot." By measuring on the Z basis at each shot, we get a bitstring, and we calculate its classical energy associated with the combinatorial problem. The precision in the reconstruction of $E(\boldsymbol{\theta})$ depends on the number of shots N_S. Since we consider this as a multinomial sampling problem, we expect the variance of the reconstructed energy to depend on N_S^{-1}. In many scenarios of NISQ devices, it is necessary to balance N_S with the desired standard deviation. For this reason, we compare the average approximation ratio obtained with the exact energy (simulated) with the energy reconstructed with a limited number of shots.

We show in Fig. 5 such a comparison with a number of shots N_S equal to 1024, 128, 64, 16, and 4. Looking at the approximation ratio R [see Fig. 5(a)], we see that taking $N_S = 128$ shots reduces R by 5% w.r.t. $N_S = 1024$ and going to $N_S = 64$ reduces it by a further 5%. This behavior then stops and even reverses its trend. In fact, we even see an average increase going from $N_S = 16$ to 4. This is understandable since the reconstruction of the energy with as little as 4 shots is not indicative of the real energy of the state. Specifically, from a final QAOA state, we might sample the solution bitstring 2, 3, or even 4 times out of 4 and the expectation of the energy on these three samplings would be very different. This behavior is, indeed, confirmed also by the fidelity in Fig. 5(a), which follows the same trend as the approximation ratio.

To have a better understanding of how the algorithm adapts to the sampling noise, we look at the kernel noise parameter σ_N^2, which is learned by the Gaussian process during the fitting at each step of the optimization (see Appendix B for details on the noise hyperparameter). The plot in Fig. 5(b) shows that, after an initial phase, the kernel noise sets at a specific value at around 400 steps. In addition to that, the lower the number of shots, the larger the noise parameter learned. In fact, by fitting the average kernel noise found at the end of the training [see Fig. 5(c)], we obtain that σ_N^2 follows a power law with $N_S^{-1.1}$. This trend is comparable to the expected trend for the variance N_S^{-1} of the reconstruction of the energy. This shows that the Gaussian process adapts to sampling noise.

Simulation of Noise: Another relevant issue in the state-of-the-art NISQ devices is the sources of quantum noise, which can interfere with the quantum circuit. Every device has different sources of noise depending on the underlying technology. In order to simulate it without specifying the device technology, we add random noise on every QAOA parameter. In this way, (2) for the Max-Cut problem is modified as

$$|\boldsymbol{\theta}\rangle = \prod_{l=1}^{p} e^{-i\sum_i \beta_l^i X_i}\, e^{i\sum_{\langle i,j\rangle} \gamma_l^{(i,j)} Z_i Z_j} |+\rangle \tag{15}$$

where β_l^i and $\gamma_l^{(i,j)}$ act differently on every qubit/edge of the graph at every layer because they are affected by Gaussian random noise with mean zero and standard deviation σ_{QN}. The noise model we are considering, although very simplified, can be considered as an example of coherent control errors that can be caused, e.g., by gate overrotations due to

IEEE Transactions on Quantum Engineering

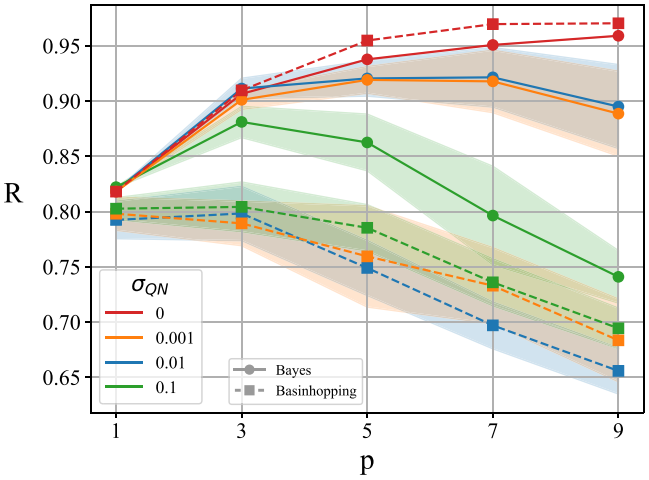

FIG. 6. Approximation ratio R for different values of the quantum noise σ_{QN}. The noise is simulated by adding random Gaussian noise with mean zero and standard deviation σ_{QN} to the variational parameters (γ, β). The plot shows the effects of the noise σ_{QN} on the final obtained approximation ratio R (fidelity F) as a function of the QAOA depth p for different σ_{QN}. Shaded areas correspond to one standard deviation. The results are obtained on the graph with 10 nodes in Fig. 1(b).

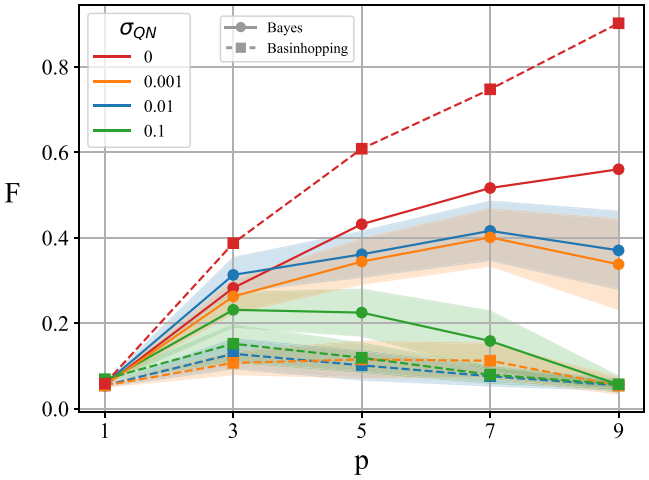

FIG. 7. Fidelity F for different values of the quantum noise σ_{QN}. The plot shows the effects of the noise σ_{QN} on the final obtained fidelity F as a function of the QAOA depth p for different σ_{QN}. Shaded areas correspond to one standard deviation. The results are obtained on the graph with 10 nodes in Fig. 1(b).

gate-time miscalibrations. In Fig. 6, we plot R and F as a function of depth at different values of σ_{QN} on the 10-node graph for the Max-Cut problem.

By increasing σ_{QN} and p, we expect to obtain a worse approximation ratio R because the error accumulates along the circuit as the number of parameters grows. Indeed, as we can see in the figure, from $p \geq 5$ the obtained R decreases w.r.t. the noiseless case, decreasing even by 20% for $p = 9$ with $\sigma_{QN} = 0.1$. Considering $\sigma_{QN} = 0.001, 0.01$, both R and F grow/remain stable up to $p = 7$, which indicates that, for shallow circuits, Bayesian optimization is robust against noise.

We care to stress that the case $\sigma_{QN} = 0.1$ was considered in order to show the effect of an exponential growth of machine noise. Realistically, a Gaussian white noise with variance 0.1 affecting each of the parameters (in range $[0, \pi]$, see Appendix B) would completely destroy the state preparation. In fact at $p = 9$, the fidelity F is the same as $p = 1$ (see Fig. 6).

We compare the results of our algorithm with the second best-performing algorithm of Fig. 4, basin-hopping. It is clear that when subjected to noise, this algorithm performs poorly: considering the approximation ratio, basin-hopping shows barely any improvement with respect to the depth of the circuit with the results starting to plummet from $p = 3$ (see Fig. 6). Most importantly, the fidelity peaks at $p = 3$ with $F \simeq 0.15$ (see Fig. 7) and then remains contained under this value. About the seemingly increase in fidelity with the noise that can be seen at $p = 3$, we also notice that the behavior inverts going up to $p = 7$ so we do not consider it relevant and assume that this means that the results are too randomic and basin-hopping is, thus, nonreliable. The poor performance in terms of fidelity confirms that basin-hopping while being an effective algorithm in the noise-free scenario—visible thanks to the high, yet costly, performance at $\sigma_{QN} = 0$ (cf. Fig. 4)—is not apt for optimization in the presence of noise. This is probably due to the fact that basin-hopping is a global optimizer that exploits a local gradient-based optimization routine (see Appendix D). Calculating gradient in the presence of noise is in fact nonoptimal since even a small variation of the parameters can impact greatly the evaluation of the function, returning a gradient that does not represent the local structure of the landscape.

V. CONCLUSION

In this article, we have presented the Bayesian optimization algorithm as a subroutine to optimize the variational parameters of the QAOA. We have applied it to find the solutions to two combinatorial optimization problems, the Max-Cut and the MIS on two graph instances.

After introducing the QAOA and the details of the Bayesian optimization algorithm, we have focused on its capability to adapt to the data and to predict new possible optimal points by exploiting both the accumulated knowledge from the previous observations and the uncertainty with respect to the optimization landscape.

We have analyzed some details of the QAOA at low circuit depth with the purpose of presenting some of the issues related to the optimization of a variational quantum algorithm. These include the flatness of the energy landscape and the limited information that we can retain from the energy of the QAOA state compared to its overlap with the ground state. After that, we have compared the Bayesian optimization with other global optimization methods, and we have shown that they require more calls, in the order of tens or hundreds, to the quantum circuit with respect to the Bayesian optimizer. This is the first sign that this method responds

more efficiently to the requirements from the quantum part of the QAOA. With this analysis, we explored the scenarios of low-depth circuits and did not test our algorithm at higher depths for mainly two reasons: it is well known that Bayesian optimization presents particular challenges for higher dimensions [30]. Moreover, considering the NISQ devices where those algorithms could, in principle, be executed, large depths constitute yet an obstacle for running QAOA on real devices.

We have also considered the effects of a finite number of measurements for the reconstruction of the energy landscape. We have shown that the results are slightly altered by a 5% decrease in the approximation ratio by using 1024 measurements compared to the optimization with the exact energy. A lower number of measurements will result in a decreasing approximation ratio. We have also shown that the Gaussian process learns to add a noise hyperparameter, which is proportional to the variance expected from the reconstruction of the energy. This can be seen as a further example of adaptation of the Bayesian algorithm to the data.

Finally, we have simulated a noisy algorithm and we have shown that for shallow circuits, with depth $p \in [1, 3, 5, 7]$, approximation ratio, and fidelity are improved even for reasonable values of the noise. For deeper circuits, up to $p \geq 9$, the intensity of noise sensibly affects the final approximation ratio, as expected. Eventually, we compared it to basin-hopping, which uses a local gradient-based optimizer, and shown that it performs very poorly, suggesting that a gradient-free optimizer is, indeed, the better choice.

These findings show that Bayesian optimization is a robust method that can account for both quantum and sampling noise. For this reason, it represents a valid tool for solving optimization problems via hybrid algorithms to be run on an NISQ device.

Code and Data Availability: Code and data are available from the corresponding author, upon reasonable request.

APPENDIX

A. DETAILS ON THE BAYESIAN OPTIMIZATION

In this appendix, we give a general overview of the Bayesian terms used in the article and a detailed review of the algorithm, we used in this work to perform the Bayesian optimization for the QAOA.

1) OVERVIEW OF BAYESIAN TERMINOLOGY

The Gaussian process is a surrogate model, which aims at reconstructing the landscape of optimization of an unknown function $f(\boldsymbol{\theta})$. It is one of the two main ingredients of the Bayesian optimization algorithm, along with the acquisition function. We can now define the Bayesian terms used throughout the article.

1) *Prior $p(\boldsymbol{f})$:* This distribution encapsulates the previous knowledge we have about the target function f. Typically, it consists of a multivariate normal distribution centered around 0 in which the covariance between points is assigned by a kernel function like (5)

$$p(\boldsymbol{f}) = \mathcal{N}(\mathbf{0}, k(x, x')). \tag{16}$$

To make an initial guess on our function, we can sample a function $\boldsymbol{f}^*$ from this distribution. To do so, we choose a set of points $\boldsymbol{\Theta}$ and evaluate

$$\boldsymbol{f}^* \sim \mathcal{N}(\mathbf{0}, k(\boldsymbol{\Theta}, \boldsymbol{\Theta})). \tag{17}$$

2) *Likelihood $p(\boldsymbol{y}|\boldsymbol{f})$:* This distribution represents the compatibility between the prior $p(\boldsymbol{f})$ and the observations $\boldsymbol{y}$. In the context of Gaussian processes, it is defined by another Gaussian distribution.
3) *Posterior $p(\boldsymbol{f}|\boldsymbol{y})$:* This describes the knowledge we have about f after having collected some observations $\boldsymbol{y} = f(\boldsymbol{\Theta})$. The aim of the Gaussian process is to make the posterior generate functions as similar as possible to f. It is related to the prior and likelihood through the Bayesian theorem

$$p(\boldsymbol{f}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})}{p(\boldsymbol{y})} \tag{18}$$

which states that a posterior distribution is proportional to their product. In the context of Gaussian processes, the posterior is calculated from the prior by an operation, which is called *conditioning*, resulting in

$$p(\boldsymbol{f}|\boldsymbol{y}) \sim \mathcal{N}(\mu', k') \tag{19}$$

where the new mean and covariance are defined in the text (7), (8). From (19), we see that the posterior is itself a Gaussian multivariate distribution. Therefore, we can sample functions $\boldsymbol{f}^*$ as we do with the prior (17) but now their values will coincide with $\boldsymbol{y}$ at every point $\boldsymbol{\Theta}$ where we sampled f.
4) *Marginal likelihood $p(\boldsymbol{y})$:* Also called *evidence*, it is the normalization term in (18). In Bayesian inference, it represents the total probability of generating the observed samples $\boldsymbol{f}$ from the prior. It is, indeed, obtained integrating over all possible function values $\boldsymbol{f}$

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})d\boldsymbol{f}. \tag{20}$$

Like the posterior distribution, it has a closed form in term of a multivariate normal distribution. Thus, the maximization of its logarithm is used in Gaussian processes to pick the best hyperparameters (as done in Section III-D).

2) BAYESIAN OPTIMIZATION ALGORITHM

This algorithm is made out of three phases, i.e., 1) warmup, 2) kernel optimization, and 3) acquisition function maximization, and is summarized in Algorithm 1.

Warmup: In the warmup phase, we start with a set of $N_W = 10$ points $X = \{\boldsymbol{\theta}_j\}_{j=1}^{N_W}$ with $\boldsymbol{\theta}_j \in \mathbb{R}^d$ where $d = 2p$ and p is the depth of the QAOA circuit. Each point $\boldsymbol{\theta}_j$ is a set of angles $(\boldsymbol{\gamma}, \boldsymbol{\beta})$. These points are sampled from the

Latin hypercube of bounds $[0, \pi]^d$. For each set of angles, we estimate the energy of QAOA y_j and we create the design matrix $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{N_W})$ and the observation vector $\boldsymbol{y}$ with the energies of the points $\boldsymbol{\theta}_j$. We also store the point with the lowest energy as $(\boldsymbol{\theta}_m, f_m)$. At this step, there is no calculation involving the Gaussian process which is currently set to its prior: a multinomial Gaussian distribution centered around zero.

Kernel Optimization: In this part, we look for the hyperparameters $(\tilde{\sigma}, \tilde{\ell})$, which maximize the marginal likelihood function $p(\boldsymbol{y}|\boldsymbol{\Theta})$ (11). This optimization is performed by repeating the `L-BFGS` [34] minimization on $-p(\boldsymbol{y}|\boldsymbol{\Theta})$ for 10 times and selecting the best parameters found. The parameters found at every step of the optimization are plotted in panels (c) and (d) of Fig. 9.

Acquisition Function Maximization: Once the hyperparameters are set the algorithm exploits its knowledge and uncertainty of the data to propose a new point $\boldsymbol{\theta}'$ with the new parameters where we evaluate the QAOA circuit. This is done by maximizing the expected improvement in (9).

The new point $\boldsymbol{\theta}'$ maximizing the expected improvement is then added to the dataset $\mathcal{D}$ and the algorithm is repeated. The procedure stops after N_{BAYES} iterations. Fig. 8 provides an illustrative example (with $p = 1, N_W = 5$, and $N_{\text{BAYES}} = 20$) of how the Bayesian method operates and moves through the landscapes.

B. BAYESIAN OPTIMIZATION WITH NOISE

In this article, we consider two scenarios in which the energy $E(\boldsymbol{\theta})$ is affected by a source of noise: 1) the finite number of samplings and 2) the quantum noise at the gate level. In the context of Bayesian optimization, we can account for noise modifying the kernel function by adding a term $\sigma_N^2 \mathbb{I}$ like so

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma^2 \left(1 + \frac{\sqrt{3}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2}{\ell}\right) e^{-\frac{\sqrt{3}\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|_2}{\ell}} + \sigma_N^2 \mathbb{I}. \quad (21)$$

This is usually called a white kernel and it is a parameter added to the diagonal to account for random fluctuations around the true value of $f(\boldsymbol{\theta})$. In this way, the new predicted mean and covariance for a set of data $\boldsymbol{\Theta}$ can be easily calculated to be

$$\mu' = \boldsymbol{\kappa}^T \cdot (\boldsymbol{K} + \sigma_N^2 \mathbb{I})^{-1} \cdot \boldsymbol{y} \quad (22)$$

$$k' = k(\boldsymbol{\theta}, \boldsymbol{\theta}) - \boldsymbol{\kappa}^T \cdot (\boldsymbol{K} + \sigma_N^2 \mathbb{I})^{-1} \cdot \boldsymbol{\kappa}. \quad (23)$$

The constant σ_N^2 belongs to the list of hyperparameters (along with σ^2 and ℓ) that are optimized with the log marginal likelihood, which now takes the form

$$\log p(\boldsymbol{y}|\boldsymbol{\Theta}) = -\frac{1}{2}\boldsymbol{y}^T \cdot (\boldsymbol{K} + \sigma_N^2 \mathbb{I})^{-1} \cdot \boldsymbol{y} - \frac{1}{2}\log\det(\boldsymbol{K} + \sigma_N^2 \mathbb{I}) - \frac{N}{2}\log 2\pi.$$

We show how the hyperparameter σ_N^2 is learned during training in the text in Fig. 5(b).

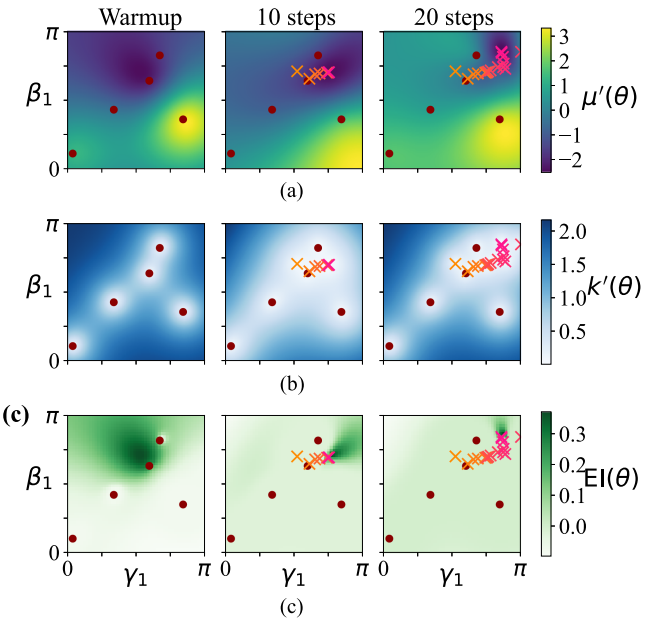

FIG. 8. Posterior and acquisition function during optimization. Here, we plot the posterior and the acquisition function at three different steps of the optimization: (From left to right) At warmup, after 10 steps and after 20 steps. In rows (a) and (b), we plot, respectively, the mean $\mu'(\theta)$ and the variance $k'(\theta)$ of the posterior while, in row (c), the acquisition function EI(θ) [see (7)–(9) in the text]. These data, obtained running the Bayesian optimization on QAOA with $p = 1$ on the graph of Fig. 1(a), are shown as a function of the two variational parameters $\theta = (\gamma_1, \beta_1)$. The red dots indicate the warmup points ($N_W = 5$, in this case) while the crosses are the points that have been selected by subsequents Bayesian optimization steps as new candidate solutions [with a color scale from first (orange) to last (pink)]. At every Bayesian optimization step, $\mu'(\theta)$ encodes the knowledge of the landscape of the function to optimize while $k'(\theta)$ contains the uncertainty. The acquisition function EI(θ) combines the information from μ' and k' and the position of its maximum proposes the next possible optimal point. From these considerations, we see that after 20 steps, the mean $\mu'(\theta)$ [row (a)] recreates the landscape (our knowledge of $E(\theta_1)$) with more precision than at warmup [compare with the real energy landscape in Fig. 2(a)]. In addition to that, EI(θ) becomes more and more flat in all the landscape except for a small area in the top right corner on the right panel of (c). This means that the Gaussian process has acquired enough knowledge from the data to converge to the energy minimum.

C. DIFFERENTIAL EVOLUTION

Finding the point $\tilde{\boldsymbol{\theta}}$ of the parameter space that maximizes the expected improvement EI($\boldsymbol{\theta}$) (9) is not an easy task since EI($\boldsymbol{\theta}$) can show a fairly flat landscape [31], in particular after many optimization steps [for example, see Fig. 8(c)].

To compute the maximum of EI($\boldsymbol{\theta}$) in this work, we use the differential evolution algorithm [33]. This is an evolutionary method in which populations of points $\{\boldsymbol{\theta}\}$, called generations, are iteratively obtained from the previous ones until convergence. The algorithm starts by initializing (a) a generation, and then, the population is updated following three main steps: (b) mutation, (c) cross-over, and (d) selection.

a) We choose as starting population $N_P = 15 \cdot 2p$ points $\{\boldsymbol{\theta}_{i,1}\}$ where the index $i \in \{1, \ldots, N_P\}$ uniquely identifies the point within the belonging population while the index 1 indicates that the point belongs to the first

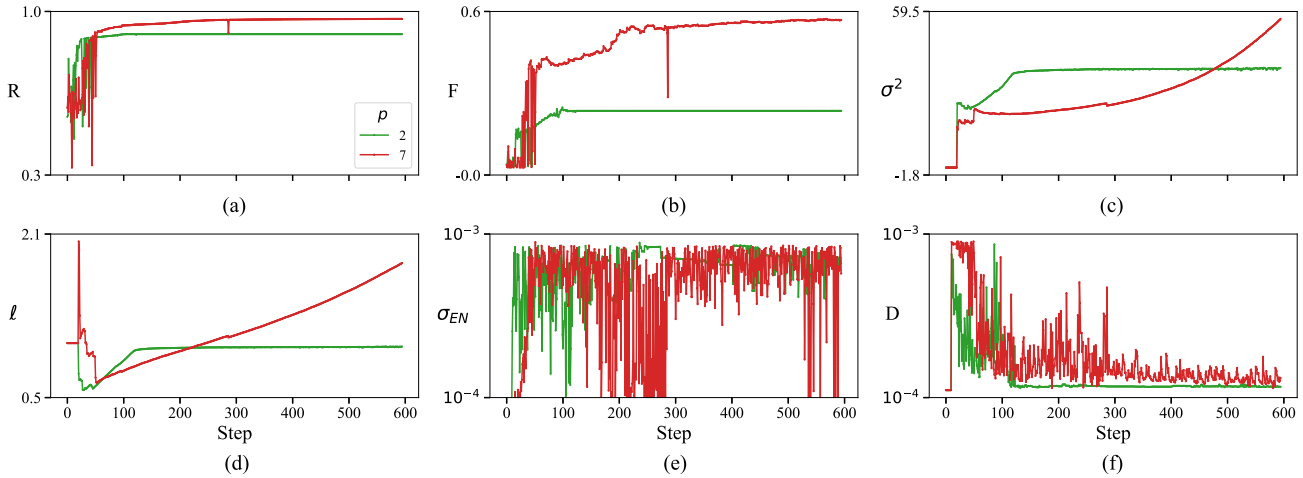

FIG. 9. Parameters of Bayesian optimization at p = 2, 7. Plots of the parameters changing during Bayesian optimization for two runs with $N_{BAYES} = 600$ steps. (a) Approximation ratio *R*. (b) Fidelity *F*. (c) Kernel constant σ^2. (d) Kernel correlation length ℓ. (e) Standard deviation of the expected improvement. (f) Average distance of the points of differential evolution at the last generation N_T.

generation $g = 1$. These points are randomly generated on the Latin $2p$-cube of bounds $[0, \pi]^{2p}$, and to each point, there is an associated expected improvement $\mathrm{EI}(\boldsymbol{\theta}_{i,1})$.

b) For each $\boldsymbol{\theta}_{i,g}$ (called parent point) in the population, the differential evolution picks three random points, different from $\boldsymbol{\theta}_{i,g}$, labeled by r_0, r_1, r_2 within the corresponding population, and creates a new point as

$$\boldsymbol{v}_{i,g} = \boldsymbol{\theta}_{r_0,g} + M(\boldsymbol{\theta}_{r_1,g} - \boldsymbol{\theta}_{r_2,g}) \tag{24}$$

where $M \in (0.5, 1)$ is a hyperparameter that is selected randomly at every generation. Through (24), the differential evolution mutates and recombines the population to create another set of parent points $\boldsymbol{v}_{i,g}$.

c) A new point $\boldsymbol{u}_{i,g}$ (offspring point) is created from $\boldsymbol{\theta}_{i,g} = (\theta^1_{i,g}, \ldots, \theta^{2p}_{i,g})$ and $\boldsymbol{v}_{i,g} = (v^1_{i,g}, \ldots, v^{2p}_{i,g})$ choosing randomly between their coordinates $\theta^j_{i,g}$ and $v^j_{i,g}$ for every $j = 1, \ldots, 2p$.

d) Finally, if $\mathrm{EI}(\boldsymbol{u}_{i,g}) \geq \mathrm{EI}(\boldsymbol{\theta}_{i,g})$, the algorithm replaces $\boldsymbol{\theta}_{i,g}$ with $\boldsymbol{u}_{i,g}$ in the next generation; otherwise, $\boldsymbol{\theta}_{i,g}$ is kept.

Steps (b)–(d) are repeated for N_T generations g. The algorithm stops when two convergence criteria are fulfilled: 1) the standard deviation σ_{EN} of the population's expected improvement [see Fig. 9(e)] and 2) the average distance D among the population points [see Fig. 9(f)] is below a certain threshold (we set $\sigma_{EN} = D = 10^{-3}$). When this happens, the point in the population with the maximum expected improvement is selected as $\tilde{\boldsymbol{\theta}}$. We notice that the criterion on the distance guarantees that in a flat landscape like the one of $\mathrm{EI}(\boldsymbol{\theta})$, the points do not get stuck on a plateau and concentrate closer to a unique candidate maximum. Although the algorithm requires many evaluations of $\mathrm{EI}(\boldsymbol{u}_{i,g})$ (as shown also in the main text), it is a valid algorithm for finding the maximum within the flat landscape of the acquisition function.

D. OTHER OPTIMIZERS

Basin-Hopping: Basin-hopping is a global stochastic optimization algorithm [40]. It combines two steps: 1) a local optimization that proposes a candidate solution and 2) a perturbation of such candidate in order to make it hop to other basins, which might contain a global optimal point. The new point is accepted or rejected according to a probability, which depends on a "temperature" parameter. The "temperature" parameter decreases with the iteration number so that, at the beginning, new proposals are easily accepted while, at larger iterations, the algorithms become more and more selective. The algorithm runs for a fixed number of iterations and the local optimizer used in this context is the gradient-based *BFGS* algorithm.

Dual Annealing: This global optimization algorithm is the generalized form of the simulated annealing and it is paired with a local optimization, which is performed at the end of the annealing to refine the solution [41]. It is a variation of a hill climbing algorithm in which a solution is randomly perturbed and the new proposed point is accepted with a probability that depends on the difference in energy between the two points. This probability also depends on a "temperature" parameter that, like in the basin-hopping case, decreases with the number of iterations in order to converge to a candidate solution.

ACKNOWLEDGMENT

The authors thank C. Sanavio and F. Dell'Anna for useful discussions.

REFERENCES

[1] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New J. Phys.*, vol. 18, no. 2, Feb. 2016, Art. no. 023023, doi: 10.1088/1367-2630/18/2/023023.

[2] A. Peruzzo et al., "A variational eigenvalue solver on a photonic quantum processor," *Nature Commun.*, vol. 5, no. 1, 2014, Art. no. 4213, doi: 10.1038/ncomms5213.

[3] N. Moll et al., "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Sci. Technol.*, vol. 3, no. 3, Jun. 2018, Art. no. 030503, doi: 10.1088/2058-9565/aab822.

[4] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, Aug. 2018, Art. no. 79, doi: 10.22331/q-2018-08-06-79.

[5] D. Du et al., *Handbook of Combinatorial Optimization.* Berlin, Germany: Springer, 1999, doi: 10.1007/978-1-4757-3023-4.

[6] M. R. Garey and D. S. Johnson, *Comput. and Intractability: A Guide to the Theory of NP-Completeness.* New York City, NY, USA: Freeman, 1982, doi: 10.1137/1024022.

[7] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*, doi: 10.48550/arXiv.1411.4028.

[8] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Phys. Rev. X*, vol. 10, Jun. 2020, Art. no. 021067, doi: 10.1103/PhysRevX.10.021067.

[9] S. Ebadi et al., "Quantum optimization of maximum independent set using Rydberg atom arrays," *Science*, vol. 376, no. 6598, 2022, Art. no. 1209, doi: 10.1126/science.abo6587.

[10] M. P. Harrigan et al., "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," *Nature Phys.*, vol. 17, no. 3, pp. 332–336, Feb. 2021, doi: 10.1038/s41567-020-01105-y.

[11] G. Pagano et al., "Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 41, pp. 25396–25401, 2020, doi: 10.1073/pnas.2006373117.

[12] M. Medvidović and G. Carleo, "Classical variational simulation of the quantum approximate optimization algorithm," *npj Quantum Inf.*, vol. 7, no. 1, 2021, Art. no. 101, doi: 10.1038/s41534-021-00440-z.

[13] H.-Y. Huang, R. Kueng, and J. Preskill, "Predicting many properties of a quantum system from very few measurements," *Nature Phys.*, vol. 16, no. 10, pp. 1050–1057, 2020, doi: 10.1038/s41567-020-0932-7.

[14] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Commun.*, vol. 9, no. 1, 2018, Art. no. 4812, doi: 10.1038/s41467-018-07090-4.

[15] G. Wang, D. E. Koh, P. D. Johnson, and Y. Cao, "Minimizing estimation runtime on noisy quantum computers," *PRX Quantum*, vol. 2, Mar. 2021, Art. no. 010346, doi: 10.1103/PRXQuantum.2.010346.

[16] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, "Cost function dependent barren plateaus in shallow parametrized quantum circuits," *Nature Commun.*, vol. 12, no. 1, 2021, Art. no. 1791, doi: 10.1038/s41467-021-21728-w.

[17] G. G. Guerreschi and M. Smelyanskiy, "Practical optimization for hybrid quantum-classical algorithms," 2017, *arXiv:1701.01450*, doi: 10.48550/arXiv.1701.01450.

[18] D. Wecker, M. B. Hastings, and M. Troyer, "Training a quantum optimizer," *Phys. Rev. A*, vol. 94, Aug. 2016, doi: 10.1103/PhysRevA.94.022309.

[19] Z. Wang, S. Hadfield, Z. Jiang, and E. G. Rieffel, "Quantum approximate optimization algorithm for MaxCut: A fermionic view," *Phys. Rev. A*, vol. 97, Feb. 2018, Art. no. 022304, doi: 10.1103/PhysRevA.97.022304.

[20] A. A. Mele, G. B. Mbeng, G. E. Santoro, M. Collura, and P. Torta, "Avoiding barren plateaus via transferability of smooth solutions in Hamiltonian variational Ansatz," *Phys. Rev. A*, vol. 106, no. 6, 2022, Art. no. L060401, doi: 10.1103/PhysRevA.106.L060401.

[21] G. B. Mbeng, R. Fazio, and G. E. Santoro, "Quantum annealing: A journey through digitalization, control, and hybrid quantum variational schemes," 2019, *arXiv:1906.08948*, doi: 10.48550/arXiv.1906.08948.

[22] R. Shaffer, L. Kocia, and M. Sarovar, "Surrogate-based optimization for variational quantum algorithms," *Phys. Rev. A*, vol. 107, Mar. 2023, Art. no. 032415, doi: 10.1103/PhysRevA.107.032415.

[23] J. Mueller, W. Lavrijsen, C. Iancu, and W. de Jong, "Accelerating noisy VQE optimization with Gaussian processes," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, 2022, pp. 215–225, doi: 10.1109/QCE53715.2022.00041.

[24] J. S. Otterbach et al., "Unsupervised machine learning on a hybrid quantum computer," 2017, *arXiv:1712.05771*, doi: 10.48550/arXiv.1712.05771.

[25] D. Zhu et al., "Training of quantum circuits on a hybrid quantum computer," *Sci. Adv.*, vol. 5, no. 10, 2019, Art. no. eaaw9918, doi: 10.1126/sciadv.aaw9918.

[26] C. N. Self et al., "Variational quantum algorithm with information sharing," *NPJ Quantum Inf.*, vol. 7, no. 1, 2021, Art. no. 116, doi: 10.1038/s41534-021-00452-9.

[27] S. Tamiya and H. Yamasaki, "Stochastic gradient line Bayesian optimization for efficient noise-robust optimization of parameterized quantum circuits," *npj Quantum Inf.*, vol. 8, no. 1, 2022, Art. no. 90, doi: 10.1038/s41534-022-00592-6.

[28] A. Arrasmith, M. Cerezo, P. Czarnik, L. Cincio, and P. J. Coles, "Effect of barren plateaus on gradient-free optimization," *Quantum*, vol. 5, Oct. 2021, Art. no. 558, doi: 10.22331/q-2021-10-05-558.

[29] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jun. 2016, doi: 10.1109/JPROC.2015.2494218.

[30] P. I. Frazier, "A tutorial on Bayesian optimization," 2018, *arXiv:1807.02811*, doi 10.48550/arXiv.1807.02811.

[31] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 2951–2959, doi: 10.5555/2999325.2999464.

[32] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* Cambridge, MA, USA: MIT Press, 2005, doi: 10.7551/mitpress/3206.001.0001

[33] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization.* Berlin, Germany: Springer, 2006, doi: 10.1007/3-540-31306-0.

[34] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, no. 1, pp. 503–528, 1989, doi: 10.1007/BF01589116.

[35] C. S. Edwards, "Some extremal properties of bipartite subgraphs," *Can. J. Math.*, vol. 25, no. 3, pp. 475–485, 1973, doi: 10.4153/CJM-1973-048-x.

[36] A. Lucas, "Ising formulations of many NP problems," *Front. Phys.*, vol. 2, pp. 1–15, 2014, doi: 10.3389/fphy.2014.00005.

[37] J. Johansson, P. Nation, and F. Nori, "QuTiP 2: A Python framework for the dynamics of open quantum systems," *Comput. Phys. Commun.*, vol. 184, no. 4, pp. 1234–1240, 2013, doi: 10.1016/j.cpc.2012.11.019.

[38] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[39] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[40] D. J. Wales and J. P. K. Doye, "Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms," *J. Phys. Chem. A*, vol. 101, no. 28, pp. 5111–5116, Jul. 1997, doi: 10.1021/jp970984n.

[41] Y. Xiang, D. Sun, W. Fan, and X. Gong, "Generalized simulated annealing algorithm and its application to the Thomson model," *Phys. Lett. A*, vol. 233, no. 3, pp. 216–220, 1997, doi: 10.1016/S0375-9601(97)00474-X.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement