Avoiding local minima in Variational Quantum Algorithms with Neural Networks

Javier Rivera-Dean,1,∗ Patrick Huembeli,1 Antonio Acín,1,2 and Joseph Bowles1

1ICFO – Institut de Ciencies Fotoniques, The Barcelona Institute of Science and Technology, 08860 Castelldefels (Barcelona), Spain
2ICREA – Institució Catalana de Recerca i Estudis Avançats, 08011 Barcelona, Spain

(Dated: April 8, 2021)

Variational Quantum Algorithms have emerged as a leading paradigm for near-term quantum computation. In such algorithms, a parameterized quantum circuit is controlled via a classical optimization method that seeks to minimize a problem-dependent cost function. Although such algorithms are powerful in principle, the non-convexity of the associated cost landscapes and the prevalence of local minima means that local optimization methods such as gradient descent typically fail to reach good solutions. In this work we suggest a method to improve gradient-based approaches to variational quantum circuit optimization, which involves coupling the output of the quantum circuit to a classical neural network. The effect of this neural network is to perturb the cost landscape as a function of its parameters, so that local minima can be escaped or avoided via a modification to the cost landscape itself. We present two algorithms within this framework and numerically benchmark them on small instances of the Max-Cut optimization problem. We show that the method is able to reach deeper minima and lower cost values than standard gradient descent based approaches. Moreover, our algorithms require essentially the same number of quantum circuit evaluations per optimization step as the standard approach since, unlike the gradient with respect to the circuit, the neural network updates can be estimated in parallel via the backpropagation method. More generally, our approach suggests that relaxing the cost landscape is a fruitful path to improving near-term quantum computing algorithms.

I. INTRODUCTION

The quantum computing community has now entered the so-called Noisy Intermediate-Scale Quantum (NISQ) computing era [1], having access to programmable quantum devices which have on the order of hundreds of non-error-corrected qubits. Although scientists have shown strong evidence of quantum computational supremacy [2, 3] for some class of problems within this paradigm, most of the well-known quantum algorithms [4] for problems of practical interest have necessities that go well beyond the capabilities of NISQ devices. A huge effort has therefore been made in recent years to develop algorithms that can suit the capabilities of NISQ devices.

The majority of such NISQ-ready algorithms fall into the class of variational quantum algorithms [5–9]. In this approach, one defines a problem-dependent cost function that is evaluated on the output of a classically parameterized quantum circuit. These parameters are then updated iteratively via a classical optimization method that interfaces with the quantum circuit and seeks to minimize the cost. Although the method is expected to be powerful in principle, one typically faces the problem of non-convexity in the cost function landscape, whereby local optimization methods such as gradient descent, commonly encounter sub-optimal local minima. If such local optimization methods are to succeed, it is therefore crucial to develop techniques that tackle the problem of local minima in the optimization landscape.

The majority of approaches to this problem keep the cost landscape fixed and incorporate elements such as momentum, adaptive step sizes, and randomization to overcome or escape local minima. In this work we introduce a conceptually different method that involves modifying the cost landscape itself in a controlled manner. This is achieved by coupling a classical neural network to the output of the quantum circuit, which can continuously deform the cost landscape as a function of its weights. The addition of the neural network weights to the problem results in a higher level of parameterization which—as is known to be the case in classical machine learning [10]—can be expected to help in avoiding local minima.

We define two algorithms that take advantage of our architecture. The first acts as a means to escape a local minimum in which a standard variational quantum algorithm has become stuck. The second iteratively optimizes the cost landscape throughout the entire optimization process as a means of evading local minima. In each step of the algorithm, the neural network parameters are updated in parallel by applying the backpropagation method on batches of bit strings that are sampled from a single quantum circuit. This means that the method requires essentially the same number of quantum circuit evaluations as the standard approach per optimization step, since the gradient estimation itself already requires sampling from a number of circuits that scales with the number of quantum circuit parameters [11].

We benchmark our algorithms on instances of the Max-Cut optimization problem [12] via numerical simulations, considering two classes of circuit ansätze. Interestingly, we show that for a simple problem, the cost landscape is modified in such a way that connects local minima of the original landscape to the global minimum of the problem, thus leading to much better solutions. For more complex

∗ mail to: javier.rivera@icfo.eu
problems, we show that our algorithms improve over the standard approach by finding both deeper minima and lower cost values on average.

The article is structured as follows: in Sec. II we briefly review the basics of Variational Quantum Algorithms and Artificial Neural Networks, and then outline our approach and algorithms. In Sec. III we present the results of our benchmarks and the effect of hyperparameter selection, and discuss possible lines for further research in Sec. IV.

II. METHODS

Our architecture combines variational quantum algorithms and artificial neural networks. In this section we present these two ingredients separately and then discuss how to use them together to solve optimization problems in NISQ devices.

A. Variational quantum circuit optimization

The variational method [13, 14] is one of the most widely used techniques for finding the ground state of a system with a Hamiltonian $H$. The method defines a parameter-dependent quantum state, $|\psi(\theta)\rangle$, and seeks the optimal $\theta^*$ that minimizes the energy of $H$:

$$C(\theta^*) = \min_{\theta} C(\theta) = \min_{\theta} \langle \psi(\theta) | H | \psi(\theta) \rangle. \quad (1)$$

Here $\theta = (\theta_1, \ldots, \theta_P)$ is a list of $P$ real parameters and the function $C(\theta)$ will be referred to as the cost function. From the Rayleigh-Ritz variational principle, a lower bound to Eq. (1) is given by the ground state energy of the system, as the ground state may not be written in terms of the parameter-dependent state $|\psi(\theta)\rangle$.

Variational Quantum Algorithms (VQAs) [15] attempt to solve the optimization problem of Eq. (1) using a quantum-classical hybrid approach. They prepare and measure states $|\psi(\theta)\rangle$ via a parametrized quantum circuit, usually referred to as a variational quantum circuit (VQC). This circuit is described by a unitary transformation $U(\theta)$ that acts on an easy-to-prepapre initial state $|\psi_0\rangle$, where $U(\theta)$ is typically composed as a sequence of unitaries from a predefined gate set that each depend on a single parameter $\theta_i$:

$$|\psi(\theta)\rangle = U(\theta) |\psi_0\rangle = \prod_{i=1}^{P} U_i(\theta_i) |\psi_0\rangle, \quad (2)$$

with $P$ the total number of parameters. In order to find good parameters that minimize the cost, the system is interfaced with a classical optimization method, which controls the preparation of states $|\psi(\theta)\rangle$ and iteratively updates a set of candidate solution parameters for Eq. (1), see Fig. 1 (a).

There exist a large variety of classical optimization methods that could be used in a VQA. In this work, we consider gradient based methods [16, 17], which are currently the most common method in VQAs, although derivative-free methods [6, 18, 19] can also be applied. The standard first order gradient descent method involves stepping the circuit parameters in the opposite direction of the gradient of the cost:

$$\theta_j^{(i+1)} = \theta_j^{(i)} - \eta \frac{\partial C(\theta)}{\partial \theta_j}; \quad (3)$$

where $\theta_j^{(i)}$ is the value of parameter $\theta_j$ at the $i$th iteration of the algorithm, and $\eta$ is a predefined step size. One is thus required to estimate the partial derivatives of the cost function with respect to the circuit parameters. The best known method that achieves this for a wide range of circuit architectures and is suitable to near-term devices is the parameter-shift rule [11, 20]. Here one considers gates of the form $U(\theta_j) = e^{-i \theta_j \gamma}$, where $O$ is an Hermitian operator with at most two unique eigenvalues $\pm r$. The derivative $\partial C(\theta)/\partial \theta_j$ is then found to be equal to

$$\frac{\partial C(\theta)}{\partial \theta_j} = r \left[ \langle \psi_0 | U^\dagger (\theta + e_j) H U (\theta + e_j) | \psi_0 \rangle - \langle \psi_0 | U^\dagger (\theta - e_j) H U (\theta - e_j) | \psi_0 \rangle \right], \quad (4)$$

where $\theta_j \pm e_j = (\theta_1, \ldots, \theta_j \pm \frac{\pi}{2r}, \ldots, \theta_P)$. Thus, the gradient with respect to each parameter is given by the difference between the cost of two parameter-shifted circuits. Note that the cost must be estimated by sampling the quantum circuit $n$ times, where $n$ defines the precision of the estimation. Since each partial derivative requires shifting the parameters differently, the number of circuit evaluations needed to estimate the full gradient to this precision is $2nP$. We note quantum backpropagation algorithms that allow for parallel gradient estimation have been introduced [21], although are less suitable for near term devices since the optimization parameters need to be stored coherently in additional ancilla systems.

B. Artificial neural networks

Artificial Neural Networks are a computational model inspired by the architecture of biological neural connections, where each neuron is connected to other multiple neurons and transmits information in the form of electric pulses. They have been applied successfully to many different problems such as computer vision [22], reinforcement learning [23] and natural language processing [24]. Furthermore, they have been employed to study phenomena in many-body physics, being useful for the study of phase transitions [25], ground state energy minimization [26], and quantum circuit simulation [27]. With respect to VQAs, neural networks have been used for meta-learning the optimization method [28, 29] and searching for solutions to combinatorial optimization problems [30].
A feedforward neural network is a simple and widely used neural network architecture. Such a neural network returns an output vector \( y \) that is the result of a sequence of matrix multiplications and element-wise nonlinear transformations, called activation functions, applied to an input vector \( x \):

\[
y = a^{(L)}(a^{(L-1)}(\cdots (a^{(1)}(W^{(1)}(a^{(0)}(x)))) \cdots)),
\]

(5)

see Fig. 1 (b) for a schematic representation. Here the \( W^{(i)} \) denote the matrices and the \( a^{(i)}(\cdot) \) the activation functions. Common choices of activation function are the Heaviside function (resulting in a perceptron [31, 32] neural network), the sigmoid function or the linear rectifier function (ReLU). In some models one also adds an additional bias vector after each matrix multiplication, but we do not consider this in our work. From hereon we focus on the simplest model of a single layer feedforward neural network, i.e. the output is given by

\[
y(x) = a(W(x)) \equiv f_W(x)
\]

(6)

and we refer to the elements of the matrix \( W \) as weights.

The output \( y \) is used to evaluate a cost function \( C(y) = C(f_W(x)) \). In this work we are interested in a scenario in which \( x \) is not a fixed vector but a classical random variable \( X \sim p(x) \) over a discrete set of vectors \( \{x\} \). Sampling this \( x \) and passing it through the neural network results in a new random variable \( f_W(X) \) with an associated mean cost

\[
C(W) = \sum_x p(x)C(f_W(x))
\]

(7)

The above cost can be minimized via a gradient descent procedure that iteratively updates the weights. The gradient vector of the above with respect to the weights is

\[
\nabla_W C(W) = \sum_x p(x) \nabla_W C(f_W(x)).
\]

(8)

For fixed \( x \) the gradient vector \( \nabla_W C(f_W(x)) \) can be calculated at a similar computational cost to that of evaluating \( C(f_W(x)) \) via the celebrated backpropagation algorithm [33]. Thus, from Eq. (8), the gradient of \( C \) can be estimated by sampling a batch of vectors \( x \) and summing the corresponding gradient vectors for each \( x \). This gradient can then be used in a gradient descent procedure to minimize the cost in Eq. (7).

C. Neural network assisted variational quantum optimization

We now present our neural network assisted VQA algorithms. We consider a variational problem as in Eq. (1)

\[
\min_{\theta} \bra{\psi(\theta)} H_D \ket{\psi(\theta)},
\]

(9)

where \( H_D \) is diagonal in the computational basis:

\[
H_D = \sum_{x \in \{\pm 1\}^N} C(x) \ket{x} \bra{x}.
\]

(10)

An extension of the algorithm to the case where \( H \) is not diagonal in the computational basis but given by a sum of local Hamiltonians is straightforward and is discussed in Sec. IV. Mappings of classical optimization and classical machine learning problems to VQAs are typically expressed in the form of Eq. (9). As mentioned, many works have studied finding solutions to such problems using a standard VQA architecture. Here we describe how to combine such architectures with classical feedforward neural networks.

The algorithms we present are based on the architecture shown in Fig. 1 (c). First, a state \( \ket{\psi(\theta)} \) is prepared by a generic VQC and an \( N \) qubit computational basis measurement is made, producing the vector \( x \in \{-1,+1\}^N \). This vector is fed into a single layer feedforward neural network, leading to the output vector \( f_W(x) = \tanh(Wx) \in \mathbb{R}^N \). Here we have used the hyperbolic tangent function as an activation function, which serves as a continuous relaxation to the sign function. Finally, the cost \( C \) is evaluated on the output vector. Sampling from the quantum circuit, one obtains the mean cost

\[
C(\theta, W) = \sum_{x \in \{\pm 1\}} p(x|\theta)C(f_W(x))
\]

(11)

where the probabilities \( p(x|\theta) \) are given from the VQC as

\[
p(x|\theta) = |\bra{\psi(\theta)}x\rangle|^2.
\]

(12)

In this context, the quantum circuit can be seen as a \( \theta- \)
dependent sample generator for the neural network, and
the mean cost takes the same form as in Eq. (7). Note
that Eq. (11) can also be written

\[ C(\theta, W) = \langle \psi(\theta) | H(W) | \psi(\theta) \rangle, \] (13)

with

\[ H(W) = \sum_{x \in \{\pm 1\}} C(f_W(x)) |x\rangle\langle x|. \] (14)

Thus, one can see that changing the weights \( W \) modifies
the energy landscape of the cost with respect to the
quantum circuit parameters. Eqs. (11) and (13) also imply
that the gradients \( \nabla_{\theta} C(\theta, W) \) and \( \nabla_W C(\theta, W) \)
can both be estimated by sampling the quantum cir-
cuit: from Eq. (13) one can estimate \( \nabla_{\theta} C(\theta, W) \) through
the parameter-shift rule of Eq. (3) with the Hamiltonian
\( H(W) \), and from Eq. (13) one can estimate \( \nabla_W C(\theta, W) \)
by accumulating the gradient through the backpropag-
ation rule of Eq. (8).

At this point it is relevant to notice that the estima-
tion of \( \nabla_W C(\theta, W) \) is much more efficient in terms of
the number of calls to the quantum circuit. In partic-
ular, estimating \( \nabla_{\theta} C(\theta, W) \) requires sampling from two
parameter-shifted circuits for each of the \( P \) parameters,
whereas estimating \( \nabla_W C(\theta, W) \) involves sampling from
a single quantum circuit with fixed parameters \( \theta \). Since
\( P \) generally grows with at least the number of qubits,
the number of circuit calls needed to calculate \( \nabla_W C(\theta, W) \)
relative to \( \nabla_{\theta} C(\theta, W) \) becomes negligible for large cir-
cuits. This difference can be seen as a consequence of
the fact that the backpropagation algorithm allows for
partial derivatives to be calculated in parallel, which is
not the case for the parameter-shift rule.

As mentioned, from Eq. (13) the weights \( W \) modify
the cost landscape with respect to the quantum circuit
parameters. In particular, if one fixes \( W = I \) then the
neural network is proportional to the identity transfor-
mation, and the cost is equivalent to the standard VQA
cost, whereas for generic \( W \) the landscape with respect
to \( \theta \) is modified. This naturally opens the opportunity
to design methods that improve the performance of VQAs
by exploiting this ability, as shown in Fig. 2. Furth-
ermore, since gradient estimations with respect to the
neural network weights require relatively few circuit evalua-
tions, such algorithms can be as computationally efficient
as standard VQA algorithms. This leads us to the cen-
tral question of our work:

Can neural networks help a VQA obtain significandy bet-
ter solutions to Eq. (9), with a negligible increase in the
number of calls to the quantum circuit?

We present two algorithms that achieve this goal. In
the first algorithm the neural network is employed as a
means to escape local minima encountered by standard
VQA algorithms, and is therefore only used when such
minima are encountered. In the second algorithm, the
neural network modifies the cost landscape during the
equivalent to the standard VQA
parameters. Eqs. (11) and (13) also
implies that the gradients \( \nabla_{\theta} C(\theta, W) \) and \( \nabla_W C(\theta, W) \)
can both be estimated by sampling the quantum circuit:
from Eq. (13) one can estimate \( \nabla_{\theta} C(\theta, W) \) through
the parameter-shift rule of Eq. (3) with the Hamiltonian
\( H(W) \), and from Eq. (13) one can estimate \( \nabla_W C(\theta, W) \)
by accumulating the gradient through the backpropag-
ation rule of Eq. (8).

At this point it is relevant to notice that the estima-
tion of \( \nabla_W C(\theta, W) \) is much more efficient in terms of
the number of calls to the quantum circuit. In partic-
ular, estimating \( \nabla_{\theta} C(\theta, W) \) requires sampling from two
parameter-shifted circuits for each of the \( P \) parameters,
whereas estimating \( \nabla_W C(\theta, W) \) involves sampling from
a single quantum circuit with fixed parameters \( \theta \). Since
\( P \) generally grows with at least the number of qubits,
the number of circuit calls needed to calculate \( \nabla_W C(\theta, W) \)
relative to \( \nabla_{\theta} C(\theta, W) \) becomes negligible for large cir-
cuits. This difference can be seen as a consequence of
the fact that the backpropagation algorithm allows for
partial derivatives to be calculated in parallel, which is
not the case for the parameter-shift rule.

As mentioned, from Eq. (13) the weights \( W \) modify
the cost landscape with respect to the quantum circuit
parameters. In particular, if one fixes \( W = I \) then the
neural network is proportional to the identity transfor-
mation, and the cost is equivalent to the standard VQA
cost, whereas for generic \( W \) the landscape with respect
to \( \theta \) is modified. This naturally opens the opportunity
to design methods that improve the performance of VQAs
by exploiting this ability, as shown in Fig. 2. Furth-
ermore, since gradient estimations with respect to the
neural network weights require relatively few circuit evalua-
tions, such algorithms can be as computationally efficient
as standard VQA algorithms. This leads us to the cen-
tral question of our work:

Can neural networks help a VQA obtain significantly bet-
ter solutions to Eq. (9), with a negligible increase in the
number of calls to the quantum circuit?

We present two algorithms that achieve this goal. In
the first algorithm the neural network is employed as a
means to escape local minima encountered by standard
VQA algorithms, and is therefore only used when such
minima are encountered. In the second algorithm, the
neural network modifies the cost landscape during the

FIG. 2. Representation illustrating the main idea of our
algorithms. (a) The original cost landscape in which the VQC has
become trapped in a local minimum. (b) By updating
the neural network weights, one perturbs the cost landscape
so that the VQC is no longer in a local minimum. (c) The
VQC is optimized in the new landscape, potentially finding a
better local minimum.

1. Algorithm 1: ESCAPE algorithm

The purpose of this algorithm is to help the VQA es-
cape from local minima. The basic idea is to run a stan-
dard VQA without help from the neural network until
convergence to a (local) minimum (step 2 below). At
that point, the VQC parameters \( \theta \) are held fixed, and
the cost landscape is modified by engaging the neural
network and optimizing its weights (step 3). In this new
landscape the VQC will generally no longer be in a lo-
cal minimum. The VQC parameters are then optimized
again, during which the cost landscape is brought back
to the original one by gradually removing the effect of
the neural network (step 4). At this point, the VQC has
successfully escaped the local minimum that it first encoun-
tered, and so the circuit parameters are updated again
in the original landscape, until another local minimum is
reached. The precise steps composing the ESCAPE algo-
rithm are:

1) Initialize \( \theta \) and set \( W = I \).

2) Update \( \theta \) via a gradient descent until convergence
to a local minima.

3) Update \( W \) via a gradient descent procedure for \( M \)
steps and define \( W_0 \) as the weight matrix after the
last step.

4) For \( t = 1, \cdots, T \):

The purpose of this algorithm is to help the VQA es-
cape from local minima. The basic idea is to run a stan-
dard VQA without help from the neural network until
convergence to a (local) minimum (step 2 below). At
that point, the VQC parameters \( \theta \) are held fixed, and
the cost landscape is modified by engaging the neural
network and optimizing its weights (step 3). In this new
landscape the VQC will generally no longer be in a lo-
cal minimum. The VQC parameters are then optimized
again, during which the cost landscape is brought back
to the original one by gradually removing the effect of
the neural network (step 4). At this point, the VQC has
successfully escaped the local minimum that it first encoun-
tered, and so the circuit parameters are updated again
in the original landscape, until another local minimum is
reached. The precise steps composing the ESCAPE algo-
rithm are:

1) Initialize \( \theta \) and set \( W = I \).

2) Update \( \theta \) via a gradient descent until convergence
to a local minima.

3) Update \( W \) via a gradient descent procedure for \( M \)
steps and define \( W_0 \) as the weight matrix after the
last step.

4) For \( t = 1, \cdots, T \):
weights and VQC parameters are updated in every optimization step, rather than only when $\theta$ is placed at a (local) minimum. The cost landscape is therefore changed during the entire optimization process as a means to guide the VQC optimization in real-time. We do not however want the neural network to dramatically change the cost landscape, since our aim is still to arrive at a good solution in the unperturbed landscape at the end of the optimization. For this reason, we introduce a modified cost $C_\alpha$:

$$C_\alpha(\theta, W) = C(\theta, W) + \alpha |W - I|,$$  \hspace{1cm} (15)

where $\alpha \geq 0$ and the term $|W - I| = \sum_{ij} |W_{ij} - \delta_{ij}|$ acts as a regularization that penalizes the weight matrix being far from the identity, and therefore limits how much it can modify the cost landscape. Here, the parameter $\alpha$ is a hyperparameter that should be tuned to the specific problem. The precise steps of the GUIDE algorithm are as follows:

1) Initialize $\theta$ and set $W = I$.

2) Estimate $\nabla_\theta C_\alpha(\theta, W)$ and perform a gradient update step of $\theta$.

3) Estimate $\nabla_W C_\alpha(\theta, W)$ and perform a gradient update step of $W$.

4) Repeat steps 2-3 until convergence, and define $W_0$ as the weight matrix after the last update.

5) For $t = 1, \cdots, T$:
   
   (a) Set $W = (1 - g(t))W_0 + g(t)I$, where $g(0) = 0$ and $g(1) = 1$.
   
   (b) Perform a gradient descent update step of $\theta$.

6) Update $\theta$ via gradient descent until convergence to a (potentially different) local minimum.

As noted previously, if the gradients $\nabla_\theta C_\alpha(\theta, W)$ are calculated via the parameter-shift rule, the addition of the neural network updates (step 3) comes at essentially no extra cost. The GUIDE algorithm can therefore replace a standard VQA algorithm without any changes to the VQC architecture, and essentially the same number of quantum circuit calls per optimization step.

III. RESULTS

A. Performance of our algorithms compared to the standard approach

We benchmark our algorithms against the standard gradient descent VQA algorithm with fixed cost landscape. For the gradient descent subroutine in both the standard and our algorithms, we consider both standard gradient descent (i.e. the update rule of Eq. (3)), and the so-called Adam gradient descent method [36], which features an adaptive step size and momentum.
1. Optimization problems

The problems we study are instances of the Max-Cut optimization problem. Given a graph of \( N \) nodes and a set \( E \) of weighted edges, where we denote the weight of an edge \((i, j)\) by \( J_{ij} \) with \((i, j) \in E\), the task is to find a partition of the nodes into two sets, such that the sum of the weights of edges connecting the two sets is maximal. We associate a variable \( x_i \in \{\pm 1\} \) to each node of the graph, this problem is equivalent to

\[
\max_{x \in \{\pm 1\}^N} \sum_{(i,j) \in E} J_{ij}(1 - x_i x_j). \tag{16}
\]

Note that the first term in the sum is a problem dependent constant, so the problem is equivalent to the minimization

\[
\min_{x \in \{\pm 1\}^N} C(x) = \min_{x \in \{\pm 1\}^N} \sum_{(i,j) \in E} J_{ij} x_i x_j. \tag{17}
\]

This minimum coincides with the ground state energy of the classical Ising Hamiltonian

\[
H = \sum_{(i,j) \in E} J_{ij} \sigma_z^{(i)} \sigma_z^{(j)}, \tag{18}
\]

where \( \sigma_z^{(i)} \) is the Pauli z operator acting on the \( i \)th qubit. This takes the same form as Eq. (10) with \( C(x) \) given by Eq. (17).

We consider three instances of the Max-Cut problem:

**Instance A:** A fully-connected graph with 5 nodes and weights sampled from a gaussian probability distribution with zero mean, \( \mu = 0 \), and variance \( \sigma^2 = 1 \).

**Instance B:** A 5-regular graph with 8 nodes, and weights sampled from a gaussian probability distribution with \( \mu = \pm 1 \) and \( \sigma^2 = 0.3 \), where each gaussian is chosen with equal probability.

**Instance C:** A 4-regular graph with 8 nodes, and weights sampled from a gaussian probability distribution with \( \mu = \pm 2 \) and \( \sigma^2 = 0.5 \), where each gaussian is chosen with equal probability.

2. Quantum circuit architectures

We consider two types of VQC architectures: (i) The problem-inspired [15] architecture called QAOA and; (ii) The problem-agnostic architecture studied in [37]. Both are shown in Fig. 4.

**QAOA architecture**—QAOA [38] is a popular VQC architecture for solving classical optimization problems which corresponds to a quantum circuit ansatz with \( 2p \) parameters \( \theta = (\gamma_1, \beta_1, \ldots, \gamma_p, \beta_p) \):

\[
U(\theta) |\psi_0\rangle = \prod_{i=1}^{p} U_C(\gamma_i) U_M(\beta_i) |\psi_0\rangle. \tag{19}
\]

Here the initial state is \( |\psi_0\rangle = |+\rangle^N \) and the unitaries are

\[
U_C(\gamma_i) = e^{-i\gamma_i H} \text{ and } U_M(\beta_i) = e^{-i\beta_i H_M}, \tag{20}
\]

with \( H \) given by Eq. (18) and \( H_M \) the non-commuting mixer Hamiltonian

\[
H_M = \sum_{i=1}^{N} \sigma_x^{(i)}. \tag{21}
\]

This choice is standard for unconstrained optimization problems, however other choices of mixer and initial state might be preferable depending on the particular problem one wants to solve [39, 40].

**Problem-agnostic architecture**—For the problem-agnostic architecture, we employed the construction that was used for studying barren-plateaus in [37]. It consists of a layer of \( R_Y(\pi/4) \) gates applied to the product state \( |0\rangle^N \) followed by a layer of randomly-drawn single-qubit rotations along either the X, Y or Z axes, parametrized by the angles \( \theta = (\theta_1, \ldots, \theta_N) \). Finally, we apply over each subsequent pair of qubits \( i \) and \( i+1 \) a CNOT operation.

All our calculations have been done numerically in Python using the Pennylane package [41] for writing and running the quantum circuits, and the corresponding codes can be found in [42]. More specifically, we have coded both the quantum circuit and the neural network with the TensorFlow library [43], and we have performed the optimization with the Keras package [44].
FIG. 5. Whisker plots showing the performance of the ESCAPE algorithm using the QAOA circuit architecture with varying depth parameter $p$ for problem Instance A, using standard gradient descent (top, step size $\eta = 0.1$) and the Adam (bottom, initial step size $\eta = 0.3$). For each $p$, we consider 200 random initializations of the parameters, but only the initializations for which we found improvement are shown. The histograms show the percentage of initializations for which the algorithm successfully escaped the local minimum encountered by the QAOA algorithm and converged to a lower cost. The plotted data shows the resulting difference in cost for these initializations only. In both cases, the neural network was optimized for $M = 80$ steps in step 3 of the algorithm.

B. ESCAPE optimization algorithm results

Here we investigate the performance of the ESCAPE algorithm for both the problem-inspired and problem agnostic architectures defined above. In all our optimizations, we use the Heaviside function $g(t) = \Theta(t - x)$ with $x = 150$ in step 4 of the algorithm, and set $T = 350$. For the neural network updates in step 3, we use the standard gradient descent update rule with a step size of $\eta = 0.05$.

Figs. 5 and 6 show the performance of the ESCAPE algorithm for the QAOA architecture for problem Instance A and problem Instance C respectively. For the architecture depths $p$, we ran the ESCAPE algorithm for 200 initializations of the parameters, where each parameter is sampled uniformly randomly in $[0, 2\pi]$. These figures show the change in energy for those initializations for which the ESCAPE algorithm was able to improve the solution relative to the standard QAOA approach by escaping the local minima encountered in step 2 of the algorithm, and the histograms to the right of each plot show the percentage over the 200 initializations that have been improved. More precisely, our condition for improvement is $C(\theta_{\text{QAOA}}) - C(\theta^*) > 0.1$, where $\theta_{\text{QAOA}}$ are the parameters of the local minimum of step 2 and $\theta^*$ are the final parameters found by the ESCAPE algorithm, and $C$ is the cost in the standard QAOA landscape.

Interestingly, the percentage of local minima that are escaped grows with $p$, which may be due to a greater number of local minima at larger $p$. For the 8-qubit example of problem Instance C, the percentage of improvement can be as large as 50%. Moreover, in certain cases, the improvement in the cost can be quite dramatic, especially for low values of $p$. In Fig. 5 we compare using the standard gradient descent update rule versus the Adam update rule in step 2 of the algorithm. The use of Adam, which combines an adaptive step size and momentum into the gradient descent, results in significantly lower cost values than standard gradient descent, and the ESCAPE algorithm is able to improve these further. We have also noted that it is important not to over-train the neural network in step 3 of the algorithm. This is shown for problem Instance A in Fig. 7.

In Fig. 8 we plot analogous results for the 8-qubit problem Instance B using the problem agnostic architecture. The different whisker plots correspond to different choices of the parameterized rotation gates (either $X$, $Y$ or $Z$ rotations) used in the second layer of the architecture, which are chosen randomly for each whisker plot. The ability of the algorithm to escape the encountered local minima appears to depend strongly on this choice, which we imagine is due to some choices resulting in a badly suited landscape for this particular problem. Nevertheless, for some choices, notably for architecture instances $2 - 4$, very large improvements to the cost can be achieved, which can be seen from the large changes in the median cost (architecture instances 2 and 4) and the elimination of the very bad local minima in architecture instance 3. Note that in most of the improvements the median and the interquartile range coincide. This is a consequence of the fact that the generated energy landscapes have a small number of local minima, and the algorithm is taking them most of the time to those ones that we can probably identify as the global minimum.

C. GUIDE optimization algorithm results

Here we investigate the GUIDE algorithm on the 5-qubit problem Instance A using the QAOA architecture. We again consider 200 random initializations of the parameters, sampled uniformly in $[0, 2\pi]$. We run both the
standard QAOA algorithm and the GUIDE algorithm from each of these initializations and plot in Fig. 9 the obtained cost after convergence, considering two different penalty terms $\alpha = 0.1$ and $\alpha = 0.3$. In both cases, the GUIDE algorithm achieves a lower median cost over the 200 initializations and the modification of the cost landscape during the optimization therefore leads to better solutions in general. For some values of $p$, the algorithm fails to find a lower cost than the standard QAOA algorithm, although it is possible that this could be mitigated with a better choice of hyperparameters, or a different choice of norm for the penalty term in the cost function.

We remark that in Fig. 9, we show the results coming from all the 200 initial random initializations, unlike to the what we did for the ESCAPE results where we only show the improved ones. And the reason for this is that the latter algorithm is used in a similar way to basin-hopping like algorithms, i.e., as an extension to a previous VQA algorithm such that for each initialization we can naturally save the energy coming from the VQA, and directly compare it to the one obtained after executing ESCAPE. This is not possible for GUIDE, as the VQC parameter update is continuously being mixed with the neural network optimization, and therefore to plot the results we have to run separately a neural network-free VQC and GUIDE with the same initialization.

Furthermore, we can also modify the definition of the $g(t)$ function and, instead of using a Heaviside function as we have been doing until now, we can use a linear function, i.e. $g(t) = t/T$. While for the case with $\alpha = 0.1$ there is not much improvement in comparison with the Heaviside, for $\alpha = 0.3$ there is a considerable difference. In Fig. 10 we plot both the percent of improvement (results that have been improved using GUIDE in comparison to QAOA) and the percent of deterioration (results that have been deteriorated using GUIDE in comparison to QAOA) for problem Instance A and $\alpha = 0.3$. This figure suggests that, for this problem, the linear function is more suitable for enhancing the performance of GUIDE as the percent of improvement is higher than the one obtained with the Heaviside function. However, such improvement comes at a price since we also increase the probability of falling into a worse minimum. Nevertheless, the percentage associated to this latter situation is always lower than the percent of improvement.

### IV. DISCUSSION

As is typical with heuristic approaches, our method is sensitive to the choice of hyperparameters that define the algorithm, and it is therefore important that they are selected well. For the ESCAPE algorithm, the main issue encountered with respect to hyperparameter selection is related to over-training of the neural network. More precisely, for the small problem instances that we consider, the neural network is typically able to find the global minimum to the problem for an arbitrary distribution at its input and in this sense is much stronger than the VQC architecture. Training the neural network for too long therefore results in the neural network solving the problem itself, typically leading to a flat (with respect to $\theta$) optimization landscape which prevents optimization of the VQC in the following step. For this reason it is important to limit the number of neural network updates when modifying the landscape, as suggested by Fig. 7.

For problems with a large number of qubits for which the neural network will not be able to easily solve the problem itself, this may not be a problem however.

In this work we have considered problems that are defined by a single Hamiltonian that are diagonal in the computation basis. The method could however be easily extended to Hamiltonians that are a sum of local Hamil-
FIG. 9. Whisker plot showing the performance of the standard QAOA algorithm against the GUIDE algorithm in problem Instance A when using different values of the penalty $\alpha$ during step 2 of the algorithm, namely (a) 0.1 and (b) 0.3. We use Adam (initial step size $\eta = 0.1$) and standard gradient descent (step size $\eta = 0.05$) to update the parameters in steps 2 and 3 of the algorithm respectively. In this case, we show the results coming from all the 200 initializations.

Itonians $H = \sum_i H_i$. In such algorithms, the VQC can be used to estimate the cost or gradient for each $H_i$ separately, and the results are summed to obtain the total cost or gradient. By adding a neural network to the output of the VQC as we have done in this work, one can thus modify the cost landscape by modifying each of the $H_i$ accordingly. Similarly, the gradient with respect to the neural network weights can then be obtained by estimating the gradient for each $H_i$ and summing the results.

There are a number of ways that one could extend or modify our approach. One interesting possibility is to consider a different quantity to the cost $C(\theta, W)$ when updating the neural network parameters, since simply lowering the cost at that point may result in a deepening of the local minimum. For example, one possibility is to develop methods to update the neural network parameters such that the size of the gradient $|\nabla_{\theta} C(\theta, W)|$ is increased, thus avoiding reaching areas with small gradients.

It would also be interesting to consider the effect of using different neural network architectures, since in this work we have considered a very simple model for clarity. One possibility is to use multi-layered feedforward neural networks, although we expect that one would need to go to significantly larger problem sizes to see a considerable difference with the approach presented here. Another possibility is to investigate recurrent neural networks, which could process sequences of quantum circuit outputs at a time, rather than one-by-one as is the case in our approach. It might be also interesting to study the effect of more specialised gradient based methods, such as the Quantum Natural Gradient Descent [45], which are more suitable for the optimization of VQC architectures.

Finally, we note that it may also be beneficial to preprocess the VQC outputs before inputting them into the neural network, since using real numbers as inputs rather than discrete values as we have done, may result in a smoother training of the neural network. We have not explored this possibility in detail, but consider this and all of the above interesting directions for future work.

V. CONCLUSIONS

In this article we have presented two algorithms for improving the overall performance of variational quantum algorithms. These algorithms are based on a trainable post-processing of the quantum circuit outputs with a feedforward neural network, which has the effect of modifying the cost landscape with respect to the quantum circuit parameters. Here, the neural network can be seen as a helper to the variational quantum algorithm, rather than the problem solver itself, since the end goal is to find good quality solutions in the unperturbed cost landscape.

For small instances of the Max-Cut optimization problem that we have used as benchmarks, our method finds better values of the quantum circuit parameters than the standard approach. Furthermore, since the addition of the neural network carries a negligible quantum circuit sampling cost relative to the calculation of circuit gradients, we expect our algorithms to be a useful tools for improving the optimization of large-scale parameterised quantum circuits. We believe that our general framework is a promising route to overcome some of the difficulties encountered in variational quantum circuit optimization, which we hope inspires future work.

VI. ACKNOWLEDGEMENTS

This work was supported by the ERC AdG CERQUTE, Spanish MINECO (FIS2020-TRANQI, Severo Ochoa CEX2019-000910-S and Retos Quspin),
the ICFO-Quaside Joint Laboratory in Quantum Processing, the Generalitat de Catalunya (CERCA Program, SGR 1381 and QuantumCAT), Fundació Privada Cellex and Fundacio Mir-Puig. J.R-D has received funding from the Secretaria d’Universitats i Recerca del Departament d’Empresa i Coneixement de la Generalitat de Catalunya, as well as the European Social Fund (L’FSE invertit en el teu futur)—FEDER. PH has received funding for this project from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 665884. JB acknowledges the AXA Chair in Quantum Information Science.

References

[1] J. Preskill, Quantum 2, 79 (2018), arXiv:1801.00862.
[2] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, A. D. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Höffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafti, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Yynghel, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrants, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neckel, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trottier, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, Nature 574, 505 (2019).
[3] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu, et al., Science 370, 1460 (2020), arXiv:2012.01625.
[4] A. Montanaro, npj Quantum Information 2, 1 (2016), arXiv:1511.04206.
[5] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O’Brien, Nature Communications 5, 4213 (2014), arXiv:1306.3041.
[6] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, New Journal of Physics 18, 032023 (2016), arXiv:1509.04279.
[7] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, Journal of the Physical Society of Japan 90, 032001 (2021), arXiv:2011.01382.
[8] K. Bharti, arXiv:2009.11001 [quant-ph] (2020).
[9] K. Bharti and T. Haug, arXiv:2011.06911 [quant-ph] (2020).
[10] Z. Allen-Zhu, Y. Li, and Z. Song, in International Conference on Machine Learning (PMLR, 2019) pp. 242–252.
[11] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Physical Review A 9, 032331 (2019), arXiv:1811.11184.
[12] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, Physical Review X 10, 021067 (2020), arXiv:1812.01041.
[13] J. J. Sakurai and J. Napolitano, Modern quantum mechanics, 2nd ed. (Pearson, Essex (England), 2014).
[14] D. J. Griffiths and D. F. Schroeter, Introduction to quantum mechanics, third edition ed. (Cambridge University Press, Cambridge, 2018).
[15] 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, arXiv:2012.09265 [quant-ph, stat] (2020).
[16] H. J. Kelley, Ars Journal 30, 947 (1960).
[17] S. Ruder, arXiv:1609.04747 [cs] (2017).
[18] L. M. Rios and N. V. Sahinidis, Journal of Global Optimization 56, 1247 (2013).
[19] G. Nannicini, Physical Review E 99, 013304 (2019), arXiv:1805.12037.
[20] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Physical Review A 98, 032309 (2019), arXiv:1803.00745.
[21] G. Verdon, J. Pye, and M. Broughton, arXiv:1806.09729 [quant-ph] (2018).
[22] D. Ciregan, U. Meier, and J. Schmidhuber, in 2012 IEEE Conference on Computer Vision and Pattern Recognition (2012) pp. 3642–3649, ISSN: 1063-6919.
[23] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezennevts, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, Nature 575, 350 (2019).
[24] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, arXiv:2005.14165 [cs] (2020).
[25] J. Carrasquilla and R. G. Melko, Nature Physics 13, 431 (2017), arXiv:1605.01735.
[26] G. Carleo and M. Troyer, Science 355, 602 (2017), arXiv:1606.02318.
[27] J. Carrasquilla, D. Luo, F. Pérez, A. Milsted, B. K. Clark, M. Volkovs, and L. Aolita, arXiv:1912.11052 [cond-mat, physics:quant-ph] (2019).
[28] M. Wilson, S. Stromswold, F. Wudarski, S. Hadfield, N. M. Tubman, and E. Rieffel, arXiv:1908.03185 [quant-ph] (2019).
[29] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni, arXiv:1907.05415 [quant-ph] (2019).
[30] M. M. Wauters, E. Panizon, G. B. Mbeng, and G. E. Wohlgemuth, Advances in Artificial Intelligence 2012, e674832 (2012).
[35] F. Draxler, K. Vesghini, M. Salmhofer, and F. A. Hamprecht, arXiv:1803.00885 [cs, stat] (2019), in Proceedings of 35th International Conference on Machine Learning (ICML 2018).

[36] D. P. Kingma and J. Ba, arXiv:1412.6980 [cs] (2017).

[37] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Nature Communications 9, 4812 (2018), arXiv:1803.11173.

[38] E. Farhi, J. Goldstone, and S. Gutmann, arXiv:1411.4028 [quant-ph] (2014).

[39] S. Hadfield, Z. Wang, B. O’Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, Algorithms 12, 2 (2019), arXiv:1709.03489.

[40] Z. Wang, N. C. Rubin, J. M. Dominy, and E. G. Rieffel, Physical Review A 101, 012320 (2020), arXiv:1904.09314.

[41] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Száva, and N. Killoran, arXiv:1811.04968 [physics, physics:quant-ph] (2020).

[42] J. Rivera-Dean and P. Huembeli, “Avoiding local minima in VQCs with NNs,” (2021), GitHub repository.

[43] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,” (2015), software available from tensorflow.org.

[44] F. Chollet, “Keras,” (2021), GitHub repository.

[45] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, Quantum 4, 269 (2020), arXiv:1901.02108.