FLIP: A flexible initializer for arbitrarily-sized parametrized quantum circuits

Frederic Sauvage,[1,2] Sukin Sim,[3,4] Alexander A. Kunitsa,[3] William A. Simon,[3] Marta Mauri,[1] and Alejandro Perdomo-Ortiz[1,*]

[1]*Zapata Computing Canada Inc., 325 Front St W, Toronto, ON, M5V 2Y1*

[2]*Physics Department, Blackett Laboratory, Imperial College London, Prince Consort Road, SW7 2BW, United Kingdom*

[3]*Zapata Computing, Inc., 100 Federal Street, Boston, MA 02110, USA*

[4]*Department of Chemistry and Chemical Biology, Harvard University, 12 Oxford Street, Cambridge, MA 02138, USA*

(Dated: March 16, 2021)

When compared to fault-tolerant quantum computational strategies, variational quantum algorithms stand as one of the candidates with the potential of achieving quantum advantage for real-world applications in the near term. However, the optimization of the circuit parameters remains arduous and is impeded by many obstacles such as the presence of barren plateaus, many local minima in the optimization landscape, and limited quantum resources. A non-random initialization of the parameters seems to be key to the success of the parametrized quantum circuits (PQC) training. Drawing and extending ideas from the field of meta-learning, we address this parameter initialization task with the help of machine learning and propose FLIP: a FLexible Initializer for arbitrarily-sized Parametrized quantum circuits. FLIP can be applied to any family of PQCs, and instead of relying on a generic set of initial parameters, it is tailored to learn the structure of successful parameters from a family of related problems which are used as the training set. The flexibility advocated to FLIP hinges in the possibility of predicting the initialization of parameters in quantum circuits with a larger number of parameters from those used in the training phase. This is a critical feature lacking in other meta-learning parameter initializing strategies proposed to date. We illustrate the advantage of using FLIP in three scenarios: a family of problems with proven barren plateaus, PQC training to solve max-cut problem instances, and PQC training for finding the ground state energies of 1D Fermi-Hubbard models.

I. INTRODUCTION

Variational quantum algorithms (VQAs) are a class of algorithms well-suited for near-term quantum computers [1]. Their applications include quantum simulation and combinatorial optimization, as well as tasks in machine learning such as data classification, compression, and generation [2]. At the core of these near-term quantum algorithms, we encounter a parametrized quantum circuit (PQC) which acts as the quantum model we need to train to successfully perform the specific task at hand [3]. Optimizing PQCs remains an arduous task, and to date only optimizations over small circuit sizes have been realized experimentally. Several obstacles limit the scaling of VQAs to larger problems. In particular, the presence of many local minima and barren plateaus in the optimization landscape preclude successful optimizations even for moderately small problems (see, e.g., [4–6]). Furthermore, contrary to classical machine learning pipelines, the effort to obtain gradients scales linearly with the number of parameters [7] thus limiting the number of iterations one can realistically perform in practice.

Developing more efficient strategies for training PQCs is needed to unlock the full potential offered by VQAs and is an active topic of research [8–12]. Drawing and extending ideas from the field of meta-learning, in particular [13], we propose to address this problem from an initialization perspective and introduce FLIP: a **FL**exible **I**nitializer for arbitrarily-sized **P**arameterized quantum circuits. This initializer is trained on a family of problems such that, after training, it can be used to initialize the circuit parameters of *similar but new* problem instances.

The flexibility to which the name of the framework alludes takes several forms. First, rather than relying on a generic set of initial parameters as e.g., in [8], the initial parameters produced by FLIP are specially tailored for families of PQC problems and can even be conditioned on specific details of the individual problems. Secondly, the strategy is agnostic to the structure of the PQCs employed and can be used for any families of PQCs. Lastly, FLIP can accommodate circuits of different sizes (i.e., in terms of the number of qubits, circuit depth, and number of variational parameters), within the targeted family, both during its training and subsequent applications. This is in sharp contrast with previous meta-learning parameter initialization approaches [10, 14].

We demonstrate several examples in which FLIP provides practical advantages. During training, smaller circuits can be included in the dataset to help mitigating the difficulties arising in the optimization of larger ones. Once trained, it shows dramatically improved performances against random initialization, and also compared to a selected set of other meta-learning approaches while additionally being easier to train. Moreover, it is successfully applied to the initialization of larger circuits than the ones used for its training. This last feature is of particular appeal as FLIP could be trained on problem instances numerically simulated, and subsequently be used on larger problems run on a quantum device. This would allow to make the most out of cheap computational resources and to leverage the latest progress in the numerical simulation of quantum circuits in order to scale VQAs to numerically intractable problems where ultimately the advantage of VQAs is expected.

This work is organized as follows. In Sec. II we describe the theoretical and practical components of FLIP. In Sec. III we discuss the main results obtained in three different scenarios where we observe a significant advantage of using our

* alejandro@zapatacomputing.com

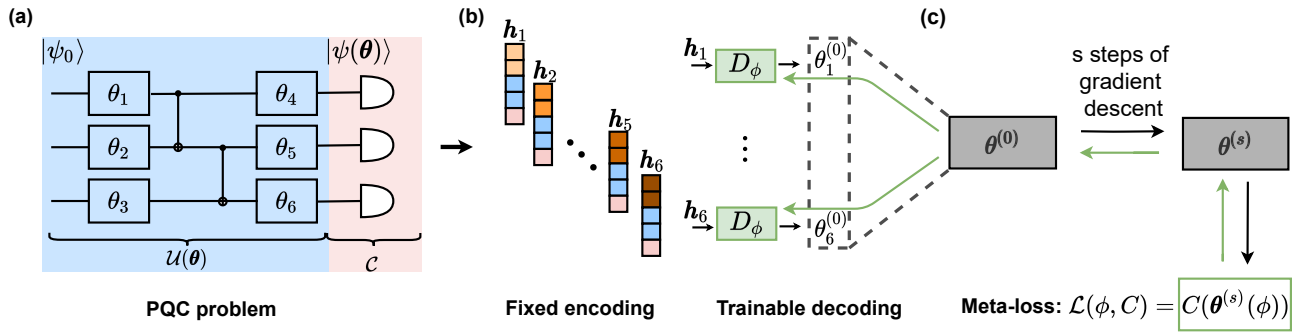

FIG. 1. **Overview of FLIP .** A generic parametrized quantum circuit (PQC) problem is composed of a parametrized circuit ansatz $\mathcal{U}(\boldsymbol{\theta})$ and an objective $\mathcal{C}$ which can be estimated through repeated measurements on the output state $|\psi(\boldsymbol{\theta})\rangle = \mathcal{U}(\boldsymbol{\theta})|\psi_0\rangle$. Solving a PQC problem corresponds to the minimization of the cost function $C(\boldsymbol{\theta}) = \mathcal{C}(|\psi(\boldsymbol{\theta})\rangle)$. **(a)** Example of a PQC problem for a system size of $n = 3$ qubits, and a quantum circuit with $K = 6$ parameters. **(b)** At the core of FLIP lies an encoding–decoding scheme which maps a PQC problem to a set of initial parameters $\boldsymbol{\theta}^{(0)}$. Each of the K parameters of the circuit $\mathcal{U}$ is first represented as an encoding vector $\boldsymbol{h}_k$. This encoding contains information about the parameter itself (orange squares), the overall circuit (blue squares) and optionally the objective (light-red squares). Importantly, each of the encodings is of fixed size (here $S = 5$) and uniquely represents each parameter. These K encodings are then decoded by a neural network, D_ϕ with weights ϕ, outputting a single value $\boldsymbol{\theta}_k^{(0)}$ per encoding $\boldsymbol{h}_k$. This encoding–decoding scheme always produces a vector of initial parameters $\boldsymbol{\theta}^{(0)}$ with dimension matching the number of circuit parameters. **(c)** These parameters $\boldsymbol{\theta}^{(0)}$ are used as the starting point of a gradient descent (GD) optimization. During training of FLIP , the weights ϕ of the decoder are tuned to minimize the meta-loss function $\mathcal{L}(\phi)$ corresponding to the value of the cost *after s steps of GD*. Gradients of this loss can be back-propagated (green arrows) to the weights ϕ of the decoder (details in the main text), which are updated accordingly. In practice FLIP is trained over PQC problems sampled from a distribution of problems $C_\tau \sim p(C)$, and tested over new problems drawn from the same or a similar distribution with, for example, problems involving larger system sizes and deeper circuits.

initialization scheme. In Sec. IV we close with an outlook for potential extensions of our work.

II. FLIP

The aim of FLIP is to accelerate optimization over targeted families of PQC problems. More efficient optimization is approached here from an *initialization* perspective: one aims at learning a set of initial parameters which can be *efficiently* refined by gradient-descent. This point-of-view has recently emerged in the field of meta-learning [13] showing promising results and has been followed by many extensions [15–17]. However, in all these works the number of parameters to be optimized is fixed, thus precluding their applicability to circuits of different sizes. To overcome this limitation, we introduce FLIP as a novel scheme which can accommodate arbitrarily-sized circuits.

We first formalize the notion of family of PQC problems in Sec. II A, then present the technical details of FLIP. The meta-learning aspect of the framework is reviewed in Sec. II B, followed in Sec. II C by a description of the encoding–decoding scheme, i.e., the strategy we developed to be able to support circuits of arbitrary sizes. As we will see, such scheme also allows to incorporate any relevant information about the problems to be optimized, thus producing fully problem-dependent initial parameters.

A. Learning over a family of related PQC problems

A generic PQC problem corresponds to a cost $C(\boldsymbol{\theta}) = \mathcal{C}(\mathcal{U}(\boldsymbol{\theta})|\psi_0\rangle)$ to be minimized, where $\mathcal{U}(\boldsymbol{\theta})$ denotes a parametrized circuit applied to an initial state $|\psi_0\rangle$, and $\mathcal{C}$ denotes an objective evaluated on the output of the circuit (Fig. 1(a)). The objective is any function which can be estimated based on measurement outcomes. For instance, it could be the expectation value $\mathcal{C}(|\psi\rangle) = \langle\psi|\mathcal{O}|\psi\rangle$ of a Hermitian operator $\mathcal{O}$ as it is often the case in VQAs [18, 19] or a distance to a target probability distribution [20, 21]. We emphasize that we have introduced a subtle distinction between the *objective* $\mathcal{C}(|\psi\rangle)$ which is agnostic to the circuit employed and the *(PQC problem) cost* $C(\boldsymbol{\theta})$ which is a function of the parameters $\boldsymbol{\theta}$ and depends both on the objective and on the choice of parametrized circuit. In the following we will mostly be interested in the latter.

Rather than considering any such PQC problem C independently, we are interested in families of related similar problems indexed by τ and drawn from a probability distribution, i.e., $C_\tau \sim p(C)$. Such distribution can be obtained by fixing the circuit ansatz $\mathcal{U}$ but varying the objective $\mathcal{C}_\tau \sim p(\mathcal{C})$, or by fixing the objective and allowing for different circuits $\mathcal{U}_\tau \sim p(\mathcal{U})$. More generally both the underlying objective and circuits are varied.

As we will aim at exploiting meaningful parameters patterns over distributions of PQC problems (this is discussed in more details in Sec. II B and II C), we will impose some restrictions in the way these distributions are defined. In the following, we will consider distributions over circuits of various sizes but with the same underlying structure, and over

objectives with the same attributes. The exact details of the distributions used are made explicit when showcasing the results.

B. Initialization-based metalearning

Meta-learning, i.e., learning how to efficiently optimize related problems, has a rich history in machine-learning [22, 23]. Here we focus on a subset of such techniques, dubbed *initialization-based* meta-learners, in which the entire knowledge about a distribution of problems $p(C)$ is summarized into a single set of parameters $\boldsymbol{\theta}^{(0)}$ which is used as a starting point of a gradient-based optimization for any problem $C_\tau \sim p(C)$.

In the original version of the method [13], these initial parameters are trained to minimize the (meta-)loss function

$$\mathcal{L}(\boldsymbol{\theta}^{(0)}) = \int p(C)\, C_\tau(\boldsymbol{\theta}_\tau^{(s)})\, dC \tag{1}$$

where the parameters $\boldsymbol{\theta}_\tau^{(s)}$, for the problem C_τ, are obtained after s steps of gradient descent For instance, for a single step, $s = 1$, of gradient descent $\boldsymbol{\theta}_\tau^{(1)} = \boldsymbol{\theta}^{(0)} - \eta \nabla_{\boldsymbol{\theta}} C_\tau(\boldsymbol{\theta}^{(0)})$. In practice, this number of steps is taken to be small ($s < 10$) but non null. The case $s = 0$ corresponds to finding good parameters *on average* rather than good *initial* parameters. It was shown [13, 16] that in some cases even $s = 1$ can produce drastically different and better parameters than the $s = 0$.

Training these initial parameters $\boldsymbol{\theta}^{(0)}$ is performed via gradient descent of the loss function Eq. 1, which requires the evaluation of the terms $\nabla_{\boldsymbol{\theta}^{(0)}} C_\tau(\boldsymbol{\theta}_\tau^{(s)})$ where $\boldsymbol{\theta}_\tau^{(s)}$ depends implicitly on $\boldsymbol{\theta}^{(0)}$. These terms can be obtained by virtue of the chain rule but involve second-order derivatives (Hessian) of the type $\nabla_{\boldsymbol{\theta}_i,\boldsymbol{\theta}_j} C_\tau(\boldsymbol{\theta})$. Evaluating these second-order terms is costly in general [13] and even more in the context of quantum circuits [24]. Fortunately, approximations of the gradients of the loss involving only first-order terms have been found to work well empirically, and we follow the approximation suggested in [16]

$$\nabla_{\boldsymbol{\theta}^{(0)}} C(\boldsymbol{\theta}_\tau^{(s)}) \approx \frac{\boldsymbol{\theta}_\tau^{(s)} - \boldsymbol{\theta}^{(0)}}{\eta}, \tag{2}$$

which has been shown to be competitive and allows for a straightforward implementation.

C. Encoding–decoding of the initial parameters

The meta-learning approach presented in the previous section requires the set of initial parameters $\boldsymbol{\theta}^{(0)}$ to be shared by any of the problems $C_\tau \sim p(C)$, which imposes the problems to have the same number of parameters. However, one would expect that good initial parameters for a given PQC problem should be informative for related problems, even if of different sizes. For instance, the ground state preparation of an N-particle Hamiltonian probably could share some resemblance with the preparation of the ground state of a similar but extended $N + \Delta N$-particle system. Likewise, optimal parameters for a circuit of depth d may inform us about an adequate range of parameter values for a deeper circuit of depth $d + \Delta d$. The existence of such circuit parameters patterns, both as a function of the size of the system and of the depth of the circuit, has been observed in the context of QAOA for max-cut problems [6] and for the long range Ising model [25]. This motivates us to extend the idea of learning good initial parameters for fixed-size circuits to learning good *patterns* of initial parameters over circuits of arbitrary sizes.

For this purpose we introduce a novel encoding– decoding scheme mapping the description of a PQC problem to a vector of parameter values with the adequate dimension. The encoding part of this map is fixed, while the decoding part can be trained to produce good initial parameter values in the spirit of Sec. II B. In addition, this mapping allows to condition these initial parameters upon the relevant details of the objective, as they can be incorporated in the description of the PQC problem produced by the encoding strategy. The general idea for a single PQC problem is illustrated in Fig. 1(b) and detailed in the following.

Each parameter, indexed by k, of an ansatz $\mathcal{U}_\tau$, is *encoded* as a vector $\boldsymbol{h}_k^\tau$ containing information about the specific nature of the parameter and of the ansatz. It includes, for example, the position and type of the corresponding parametrized gate, and the dimension of the ansatz. Several choices could be made but importantly we ensure that this encoding scheme results in encoding vectors of the same dimension S for each parameter, i.e., $\forall k, \tau,\ dim(\boldsymbol{h}_k^\tau) = S$ and that distinct parameters and circuits have distinct representations, i.e., $\forall k \neq k'$, $\boldsymbol{h}_k^\tau \neq \boldsymbol{h}_{k'}^\tau$ and $\forall \mathcal{U}_\tau \neq \mathcal{U}_{\tau'}$, $\boldsymbol{h}_k^\tau \neq \boldsymbol{h}_k^{\tau'}$. Explicit definitions of the encodings used for the results presented in Sec. III are provided in Appendix. A 1.

Once this choice of encoding is taken, any PQC problem containing an arbitrary number of parameters K is mapped to K of such encodings. These are then fed to a *decoder*, denoted D_ϕ, with weights ϕ, which is the trainable part of the scheme. This decoder is taken to be a neural network with input dimension S and output dimension one; that is, for any given encoding $\boldsymbol{h}_k^\tau$ it outputs a scalar value, and when applied to K of such encodings it outputs a vector of dimension K which contains the initial parameters $\boldsymbol{\theta}_\tau^{(0)}$ for the problem C_τ to be used in the meta-learning framework of Sec. II B. Note the extra index τ when denoting these initial parameters $\boldsymbol{\theta}_\tau^{(0)}$ as they now depend on the underlying problems.

In addition to the details of the parameters and ansatz, one can also extend the encoding to incorporate objective-specific details, that is relevant information about the objective $\mathcal{C}$. This straightforward extension allows to produce fully problem-dependent initial parameters. Finally, in Sec. II D we discuss practical aspects of the training and testing of FLIP .

D. Training and testing

Training FLIP consists of learning the weights ϕ of the decoder to minimize the loss-function

$$\mathcal{L}(\phi) = \int p(C)\, C_\tau(\boldsymbol{\theta}_\tau^{(s)}(\phi))\, dC \tag{3}$$

which is similar to Eq. 1, with the difference that the parameters $\boldsymbol{\theta}_\tau^{(s)}(\phi)$ are now obtained after s steps of gradient-descent performed from the initial parameters $\boldsymbol{\theta}_\tau^{(0)}(\phi)$ outputted by the decoder (the dependence to the decoder weights ϕ has been made explicit here). As illustrated in Fig. 1(c), the gradients needed to minimize this loss function are obtained by virtue of the chain rule:

$$\nabla_\phi C_\tau(\boldsymbol{\theta}_\tau^{(s)}) = \nabla_{\boldsymbol{\theta}_\tau^{(0)}} C_\tau(\boldsymbol{\theta}_\tau^{(s)}).\nabla_\phi \boldsymbol{\theta}_\tau^{(0)}, \tag{4}$$

where the new Jacobian term $\nabla_\phi \boldsymbol{\theta}_\tau^{(0)}$ contains derivatives of the output of the neural network D_ϕ for the different encodings $\boldsymbol{h}_k^\tau$, while the term $\nabla_{\boldsymbol{\theta}_\tau^{(0)}} C(\boldsymbol{\theta}_\tau^{(s)})$ was discussed earlier and is approximated according to Eq. 2. Training such hybrid schemes, involving backpropagation through quantum circuits and neural networks, has been facilitated by the recent development of several libraries [26, 27]. In practice each step of training of FLIP consists of drawing a small batch of problems from the problem distribution $p(C)$ and using the gradients in Eq. 4 averaged over these problems to update the weights ϕ.

Finally, once trained, the framework is applied to unseen testing problems. Testing problems, indexed by τ', are sampled from a distribution $C_{\tau'} \sim p'(C)$. When presented to a new problem $C_{\tau'}$ the encoding–decoding scheme is used to initialize the corresponding circuit $\mathcal{U}_{\tau'}$, from which s' steps (typically larger than the number of steps s used for training) of gradient descents are performed. In the result section, Sec. III, we draw these testing problems from distributions similar in nature to the training distribution but containing problems of larger sizes, involving in some cases circuits twice as deep and as wide as the training ones.

III. RESULTS AND DISCUSSION

To put FLIP into practice we start with state preparation problems using simple quantum circuits, described in Sec. III A. This will allow us to illustrate the working details of FLIP , and the construction of a distribution of problems where both the target states and sizes of the circuits are varied. Furthermore, these tasks exhibit barren plateaus and we show how such issues, arising from random initialization of the circuits, could be circumvented using FLIP .

FLIP is then applied to some of the most promising types of VQAs encountered in the literature. In Sec. III B we consider the case of the quantum approximate optimization algorithm (QAOA) [28] in the context of max-cut problems. As applications of QAOA to graph problems have been extensively studied [6, 29–31], it will allow us to thoroughly benchmark FLIP against competitive initialization alternatives and to further investigate the patterns in the initial parameters that are learned.

In contrast to QAOA ansätze, hardware-efficient ansätze [32, 33] are tailored to exploit the physical connections of quantum hardware. Typically, they aim at reducing the depth of circuits, and thus the coherence-time requirements, at the expense of introducing many more parameters to be optimized over. As such, if successfully optimized, they could offer practical applications in the near-term. In Sec. III C, we apply FLIP to the ground state preparation (VQE [34] calculations) of the one-dimensional Fermi-Hubbard model (FHM) employing the low-depth circuit ansatz (LDCA) [33]. This FHM is treated in the half-filling regime over a continuous range of interaction strengths.

In all these examples we demonstrate the advantage of FLIP compared to random initialization or other more sophisticated alternatives, and systematically assess its ability to successfully initialize larger circuits than the ones it was exposed to during training. All the results are obtained on numerical experiments which were carried out with Orquestra® [35], Zapata's proprietary platform for workflow and data management. We also leverage here its integration with Tensorflow Quantum [27].

A. Mitigating barren plateaus in state preparation problems

Rather than considering the preparation of a single target state $|\psi^{tgt}\rangle$, we consider a family of target states $|\psi_\tau^{tgt}\rangle$ which are computational basis states with only one qubit in the $|1\rangle$ state, at target position p_τ. This allows us to generate problems, indexed as usual by τ, where both the size of the target state n_τ and the position p_τ can be varied. For example, for $n_\tau = 3$ qubits, and a position $p_\tau = 2$, the target state reads $|\psi_\tau^{tgt}\rangle = |010\rangle$.

The circuits are composed of d_τ layers of parametrized single qubit gates $R_y(\theta)$ applied to each qubits, followed by controlled-Z gates acting on adjacent qubits (where the first and last qubits are assumed to be adjacent). The resulting parametrized circuits contain $K_\tau = n_\tau d_\tau$ variational parameters. The objective to be minimized is taken to be the negated fidelity $C_\tau(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|\mathcal{O}_\tau|\psi(\boldsymbol{\theta})\rangle$, with $\mathcal{O}_\tau = -|\psi_\tau^{tgt}\rangle\langle\psi_\tau^{tgt}|$. Distributions of problems are thus fully specified by defining how to sample the integers n_τ, d_τ and p_τ. A single problem with $n_\tau = 3$ qubits, $d_\tau = 6$ layers, and position $p_\tau = 2$ is illustrated in Fig. 2(a).

For training, we consider a distribution of problems where the integers $n_\tau \in [1, 8]$ qubits, $d_\tau \in [1, 8]$ layers, and $p_\tau \in [1, n_\tau]$ are uniformly sampled within their respective range. Further details about the hyper-parameters used during training can be found in Appendix. A. For testing, 50 new problems are sampled with $n_{\tau'} \in [4, 16]$ qubits, $d_{\tau'} \in [4, 16]$ layers, and $p_{\tau'} \in [1, n_{\tau'}]$, that is from a distribution containing problems supported by the training distribution but also larger problems (with circuits up to twice as wide and as deep as the largest circuit in the training set).

Convergence of the optimizations performed over these

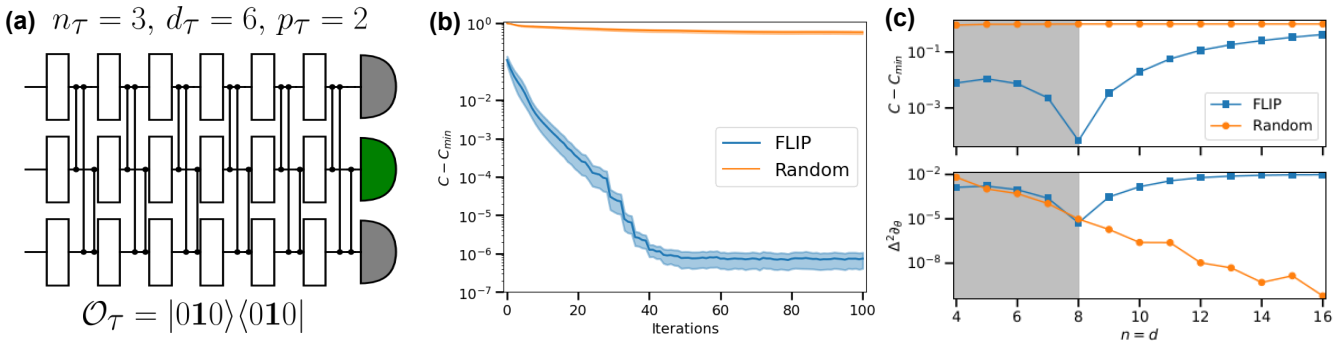

FIG. 2. **State preparation problems.** Each problem consists in the preparation of a target state $|\psi_\tau^{tgt}\rangle$ of size n_τ with all qubits in the $|0\rangle$ state except one, at target position p_τ, which is in the $|1\rangle$ state. The objective to minimize is the negated fidelity, i.e., the expectation value of the operator $\mathcal{O}_\tau = -|\psi_\tau^{tgt}\rangle\langle\psi_\tau^{tgt}|$. Circuits are composed of d_τ layers of parametrized R_y gates followed by fixed controlled-Z gates. **(a)** Problem instance for $n_\tau = 3$ qubits, $d_\tau = 6$ layers and target position $p_\tau = 2$, that corresponds to the target state $|\psi_\tau^{tgt}\rangle = |010\rangle$. FLIP is trained over circuits with sizes up to $n_\tau = d_\tau = 8$ but tested up to the case $n_\tau = d_\tau = 16$. **(b)** Results of optimizations over 50 testing problems for circuits either initialized randomly (orange curves) or with FLIP (blue curves). The deviations of the objective from its minimum are reported as a function of the number of optimization steps (iterations). **(c)** Initial values of the deviations in the objective (top panel) and variances in its gradients (bottom panel). These are obtained for various system sizes n, a number of layers $d = n$ and a target position fixed to $p = 1$. The shaded grey regions highlight system sizes used when training FLIP . In the case of random initialization, the exponential decrease of the variances as a function of the system size indicates the presence of barren plateaus.

testing problems are depicted in Fig. 2(b) with a comparison of circuits initialized with FLIP (blue curves) and randomly initialized (orange curve). In both cases, 100 steps of simple gradient descent are performed after initialization. The absolute minimum of the objective which can be reached for these state preparation problems is $C_{min} = -1$, and we report the average (and confidence interval) of the deviation $\Delta C = C - C_{min}$ from this minimum as a function of the number of optimization steps.

One can see that circuits initialized by FLIP can be quickly refined to reach an average value of $\Delta C \approx 0.1\%$ after fewer than 30 iterations. This is in contrast with optimizations starting with random initial parameters which even after 100 iterations only achieve an average $\Delta C \approx 50\%$. These average results are dissected in Appendix. B 2 where optimization traces on individual problems are displayed (Fig. 6). These individual results show that the benefit of FLIP is particularly appreciable for the largest circuits considered: for problems with $n_{\tau'} = d_{\tau'} \geq 12$, most of the optimizations starting with random parameters fail in even slightly improving the objective, while optimizations of circuits initialized with FLIP converge quickly.

These patterns in optimizations with randomly initialized parameters are symptomatic of barren plateaus. To further understand the advantage of FLIP in this context, in Fig. 2(c) we compare the *initial* values of the objective and gradients for PQCs initialized randomly (orange curves) and with FLIP (blue curves). In the top panel the deviations $\Delta C = C - C_{min}$ of the objective values are reported while the variances $\Delta^2 \partial_\theta$ of the cost function gradients are displayed in the bottom panel. Shaded regions indicate circuit sizes seen by FLIP during training.

For random initialization, the deviations of the objective value are always close to its maximum value 1, i.e., far away from the optimal parameters. Furthermore one can see that the amplitude of the gradients exponentially vanishes with the system size, thus preventing successful optimizations. Circuits initialized with FLIP (blue curves) exhibit strikingly different patterns. For problem sizes $n \leq 8$ qubits, seen during training (shaded regions), both the objective and the gradient amplitudes are small, showing that FLIP successfully learnt to initialize parameters close to the optimal ones. When the size of the circuits is increased further ($n > 8$), the objective values increase, indicating that circuits are initialized further away from ideal parameters. This degradation is expected as FLIP has to extrapolate initial parameters patterns found for small circuits to new and larger ones. Nonetheless, in all cases the values of the initial objective stay significantly better than the ones obtained for random initialization. More remarkably, the amplitudes of the initial gradients remain non-vanishing for the range of circuit sizes studied, thus allowing for the fast optimization results displayed in Fig. 2(b).

Complementary results are provided in Appendix. B. In particular, we verify that FLIP remains competitive even when trained and tested with noisy gradients (Appendix. B 4). Overall, these results illustrate the ability of FLIP to learn patterns of good initial parameters with respects to the specific objective details (here corresponding to the target state to be realized) and the circuits dimensions. For these state preparation examples, where such structure exists and is relatively simple, this lead to the ability to avoid the barren plateaus phenomenon which would have arisen from random initialization. This motivates us to apply FLIP further to more practical VQAs.

B. Max-cut graph problems with QAOA

The Quantum Approximate Optimization Algorithm (QAOA) [28] was suggested as an approximate technique

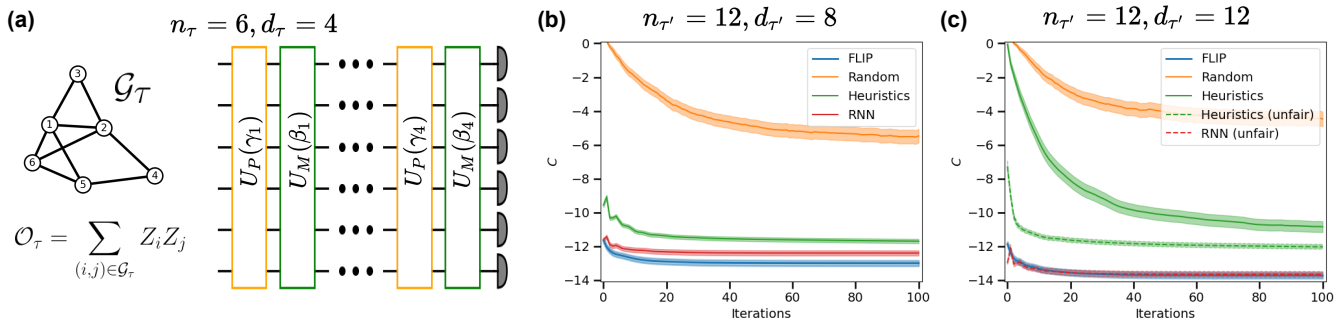

FIG. 3. **QAOA applied to max-cut problems.** **(a)** The graph instances $\mathcal{G}_\tau$ are drawn from an Erdos-Renyi distribution $P_{ER}(\mathcal{G}; e)$, with the parameter $e \in [30\%, 90\%]$ specifying the probability of any edge to be included. Example for a graph with $n_\tau = 6$ nodes and probability $e = 50\%$. Solving the max-cut of the graph $\mathcal{G}_\tau$ can be mapped to the minimization of the expectation value $\langle \mathcal{O}_\tau \rangle$ of the cost operator defined in the figure. This ground state preparation is attempted with a QAOA ansatz consisting of d_τ layers, of parametrized problem (orange) and mixer (green) unitaries. **(b-c)** Different initialization strategies (described in the main text) are compared: FLIP , recurrent neural network based meta-learning (RNN), heuristics initial parameters, and random initialization. Except for the case of random initialization each of the initialization strategies is trained beforehand. For FLIP this training is performed over circuits with $d_\tau \leq 8$ layers, while RNN and heuristics are trained on circuits containing $d_\tau = 8$ layers (as they cannot be trained on circuits of varying depths). Optimization results averaged over 100 new testing graph instances containing $n_{\tau'} = 12$ nodes and circuits with (b) $d_{\tau'} = 8$ (b) and (c) $d_{\tau'} = 12$ layers. For FLIP, the latter case corresponds to testing over more parameters than initially trained for. The RNN needs to be re-trained from scratch on these larger circuits and, thus, labelled *unfair* since it is given this advantage over FLIP which does not use training with these larger instances to make its predictions. The heuristics initial parameters trained for 8 layers can be padded with random values, or new heuristics parameters can be re-trained on the larger circuits (thus labelled unfair). FLIP is not re-trained but still outperforms other initialization strategies.

for optimizing combinatorial problems. Since its proposal, QAOA has received a lot of attention with recent works focusing on aspects of its practical implementation and scaling [6, 25, 36, 37]. Alternating-type ansätze such as QAOA as well as the Hamiltonian Variational Ansatz [18, 38] have the advantage of being parameter-efficient. Still, optimizing such ansätze can be challenging and it was found that even for small problem sizes, the optimization landscape is filled with local minima [6, 31]. This has motivated many works [6, 10, 11, 39] aiming at devising more efficient optimization strategies. We first briefly recall the definition of max-cut problems and of the QAOA ansatz, then apply FLIP and compare it to random initialization and other more sophisticated initialization strategies.

Consider a graph $\mathcal{G}$ with a set of n nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. The maximum cut of this graph is defined as the partition $(\mathcal{V}_1, \mathcal{V}_2)$ of $\mathcal{V}$, which maximizes the number of edges having both an end-point in $\mathcal{V}_1$ and $\mathcal{V}_2$. A max-cut problem can be mapped to the n–qubit operator $\mathcal{O} = \sum_{(i,j)\in\mathcal{E}} Z_i Z_j$, whose ground state provides a solution to the problem. While in principle this ground state preparation could be attempted with any type of PQCs, it is typical to resort to QAOA ansätze for these problems. A QAOA ansatz is formed of repeating composition of *problem* and *mixer* unitaries, defined respectively as $U_P(\gamma) = exp(-i\gamma\mathcal{O})$ and $U_M(\beta) = exp(-i\beta\sum_{i=1}^n X_i)$. For d of such layers the overall ansatz reads $\mathcal{U}(\boldsymbol{\theta}) = \prod_{l=d}^1 U_M(\beta_l)U_P(\gamma_l)$ where $\boldsymbol{\theta} = (\gamma_1, \beta_1, \ldots, \gamma_d, \beta_d)$ is the set of the $K = 2d$ parameters to be optimized over.

In the following we consider randomly generated graph instances $\mathcal{G}_\tau \sim P_{ER}(\mathcal{G}; e)$ drawn from an Erdos-Renyi distribution with parameter e. This parameter e specifies the probability of any edge to belong to the graph. We follow [10] and rather than keeping this probability fixed we also sample it uniformly from $e_\tau \in [30\%, 90\%]$ each time a new graph is drawn. When training FLIP , we consider a number of graph nodes $n_\tau \in [2, 8]$ and a number of circuit layers $d_\tau \in [2, 8]$, both uniformly sampled. An instance of a QAOA max-cut problem belonging to the training data set is illustrated in Fig. 3, for the case where $d_\tau = 4$ layers, $n_\tau = 6$ nodes and $e_\tau = 50\%$.

As in the previous case, we will compare optimizations with circuits initialized by FLIP against random initializations, but we will also include more competitive baselines. In [29], it was reported that the objective values of QAOA ansätze concentrate for fixed parameters but different problem instances. In other words, good parameters obtained for a given graph are typically also good for other similar graphs. While these results are obtained for 3–regular graphs and a number of layers smaller than the number of nodes, this motivates us to build a simple general initialization strategy. This strategy - that we call *heuristics initialization* - consists in:

(i) performing optimization over randomly drawn training problems,

(ii) selecting the set of optimal parameters resulting in the best *average* objective value over the training problems,

(iii) reusing these parameters as initial parameters when optimizing new problems.

The two-step training part of the strategy allows to mitigate for (i) optimizations trapped in local minima by repeated optimizations, and (ii) to ensure that the selected parameters are typically good for many other problems.

In addition, we also include results obtained with the recurrent neural network (RNN) meta-learner approach [10]. A RNN is trained to act as a black-box optimizer: at each step it receives the latest evaluation of the objective function and suggests a new set of parameters to try. After the training, this RNN can be used on new problem instances for a small number of steps. The best set of parameters found over these preliminary steps is subsequently used as initial parameters of a new optimization. For its implementation we follow [10].

In contrast with FLIP , both the heuristics and the RNN initializer require that all the problem instances share the same number of parameters. For QAOA circuits, this restricts the circuits employed to be of fixed depth, although the number of graph nodes considered can be varied as it does not relate directly to the number of parameters involved. Hence, when training these alternative initializers, we consider a similar training distribution as the one used for FLIP , with the exception that all circuits are taken to be of fixed depth, $d_\tau = 8$ layers.

A first batch of testing problems are generated for graphs with $n_{\tau'} = 12$ nodes and circuits with $d_{\tau'} = 8$ layers. Average optimization results (and confidence intervals) over 100 of such testing problem instances are reported in Fig. 3(b). The four different initialization strategies previously discussed are compared. One can see that the simple heuristic strategy already provides a significant improvement compared to random initialization, thus highlighting the importance of informed initialization of the circuit parameters. An extra improvement is achieved when using the RNN initializer. Finally circuits initialized with FLIP exhibit the best final average performance over these problems. While initial objective values are similar for circuits initialized by FLIP and RNN, the initial parameters produced by FLIP are found to be more auspicious to further optimization.

Results for new testing problems with an increased depth of $d_{\tau'} = 12$ layers are displayed in Fig. 3(c). The heuristics initial parameters trained on circuits with $d_\tau = 8$ layers, are adapted to these larger circuits by padding the missing additional parameters entries with random values. However, there is no straightforward way to fairly extend the RNN trained on circuits with $d_\tau = 8$ layers to these larger circuits. Hence, we include the heuristics and RNN initializer re-trained from scratch on problems with $d_\tau = 12$ layers. These are labeled as "unfair" in the legend as they are trained on circuits 50% deeper than the largest ones seen by FLIP during its training. Remarkably, even in this challenging set-up, FLIP outperforms all the other approaches, albeit only showing an almost indistinguishable advantage compared to the RNN trained on the $d_\tau = 12$ layered circuits ($\Delta C \approx 0.06$).

We also investigate the patterns in the initial parameters found by the framework. These are plotted for different circuit sizes ranging from $d = 2$ to $d = 15$ layers, in Fig. 9 of the Appendix. C. In particular, we found similar patterns as the ones discovered and exploited in [6]. In contrast to [6], these patterns are learnt during a single phase of training, without requiring neither sequential growing of the circuits nor the use of handcrafted extrapolation rules.

Having shown the benefits of FLIP for initializing QAOA circuits, we now consider its application to a circuit ansatz with a more involved structure.

C. Initializing LDCA for the 1D Fermi-Hubbard Model

The Fermi-Hubbard model (FHM) is a prototype of a many-body interacting system, widely used to study collective phenomena in condensed matter physics, most notably high-temperature superconductivity [40]. Despite its simplicity, FHM features a broad spectrum of coupling regimes that are challenging for the state-of-the-art classical electronic structure methods [41]. In the context of VQAs, various classes of FHMs were used to benchmark VQE optimizers [11] and parameter initialization heuristics [10]. Recently Cade et al. [42] analyzed the prospect of achieving quantum advantage for the large scale VQE simulations of the two-dimensional FHM, emphasizing the need for efficient circuit parameter optimization techniques, including those based on meta-learning. Here we consider the one-dimensional FHM (1D FHM), which describes a system of fermions on a linear chain of sites with length L. The 1D FHM Hamiltonian is defined as the following in the second quantization form:

$$\begin{aligned} H_{\text{1D FHM}} = &- t \sum_{\sigma=\uparrow,\downarrow} \sum_{j=1}^{L-1} (a^\dagger_{j+1,\sigma} a_{j,\sigma} + a^\dagger_{j,\sigma} a_{j+1,\sigma}) \\ &+ U \sum_{j=1}^{L} n_{j,\uparrow} n_{j,\downarrow} - \mu \sum_{\sigma=\uparrow,\downarrow} \sum_{j=1}^{L} n_{j,\sigma}, \end{aligned} \tag{5}$$

where j indexes the sites and σ indexes the spin projection. The first term quantifies the kinetic energy corresponding to fermions hopping between nearest-neighboring sites and is proportional to the tunneling amplitude t. The second term accounts for the on-site Coulomb interaction with strength U. Symbols $n_{j,\sigma}$ refer to number operators. Lastly, the third term is the chemical potential μ that determines the number of electrons or the filling. For the half-filling case, in which the number of electrons N is equal to L, μ is set to $\frac{U}{2}$.

For the infinite 1D FHM, the ground state energy density per site is exactly solvable using the Bethe ansatz [43]. Consequently, equipped with verifiable results, this model has been proposed as a benchmark system for near-term quantum computers [44]. In this work, ground state energies of the 1D FHM over a range of chain lengths are systematically estimated using the VQE algorithm. With increasing system size (chain length) and corresponding increase in the circuit resources (e.g., depth or gate count) of the respective VQE ansatz, the noise in the quantum device deteriorates the quality of the solutions. The maximum chain length before the device noise dominates the quality of VQE solutions informs the maximum capability of the particular quantum device at solving related algorithm tasks.

Implementation of such VQE benchmark on near-term devices requires a careful design of a variational ansatz with low circuit depth. A candidate for such ansatz is the "Low-Depth Circuit Ansatz" (LDCA), a linear-depth hardware-inspired ansatz for devices with linear qubit connectivity and tunable

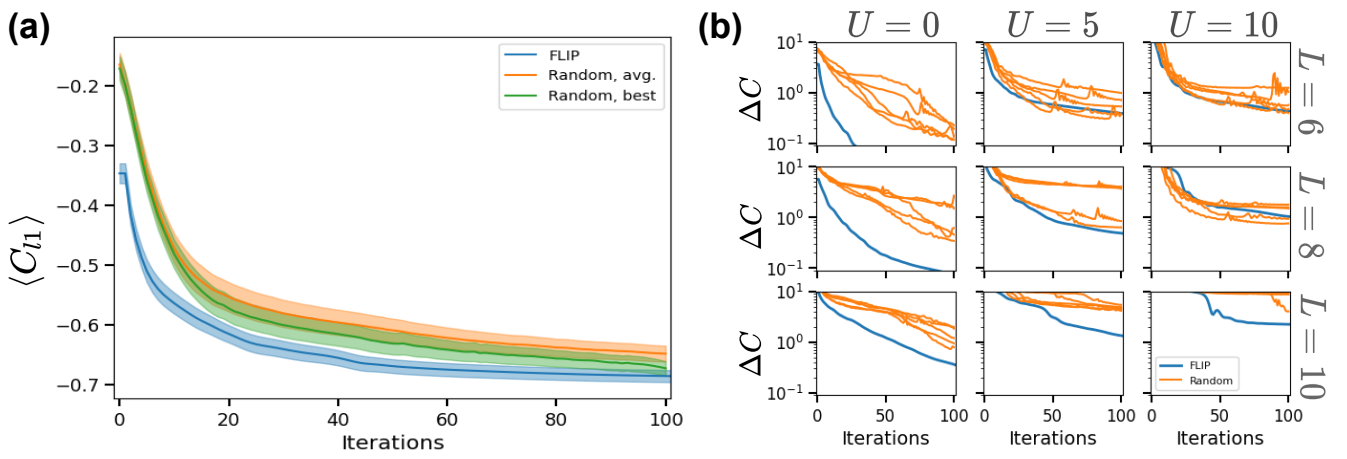

FIG. 4. **Optimization results for the 1D Fermi-Hubbard model (1D FHM).** FLIP is trained on LDCA ansätze with a number of layers $d_\tau \leq 6$ and a number of sites $L_\tau \leq 6$. All 1D FHM instances are in the half-filling regime with positive interaction strength $U \in [0, 10]$. For testing these numbers are increased to $d_{\tau'} = 8$ and $L_{\tau'} \in \{6, 8, 10\}$. Optimizations of the circuits initialized by FLIP (blue curves) or randomly (with 5 restarts) are compared. **(a)** Convergence of the $l1-$ normalized energies $\langle C_{l1} \rangle$, averaged over the testing problems. For random initialization the average (orange curve), and the best per problem over the 5 restarts (green curve) are reported. **(b)** Results for individual testing problems with interaction strengths $U = 0, 5$ and 10 (from left to right column) and a number of sites $L = 6, 8$ and 10 (from top to bottom row). Deviations $\Delta C = C - C_{min}$ of the optimized raw energies from the true ground state energies are reported. Each of the five traces correspond to the repeated random initialization (orange curves).

couplers [33]. While LDCA was shown to be effective in estimating ground state energies of strongly correlated fermionic systems, its application has been limited to small problem sizes due to the quadratic scaling of parameters with the system size and the corresponding difficulty in parameter optimization. A recent work proposed an optimization method for parameter-heavy circuits such as LDCA, but the reported simulations for LDCA required many energy evaluations on the quantum computer [45]. This prompts a need for better parameter initialization strategies to reduce the number of energy evaluations. For a parameter-heavy ansatz like LDCA, in which the role of each parameter (and its corresponding gate) is not easily understood, it would especially be beneficial to have an initialization strategy that is effective across a family of related problem instances. This poses an opportunity for a strategy like FLIP.

In the following paragraphs, we describe details and results for applying FLIP to initialize number-preserving LDCA for 1D FHM problem instances of varying chain lengths L and numbers d of circuit layers (named "sublayers" in LDCA). The structure of LDCA used in this study, including the definition of a sublayer, can be found in the Appendix of Ref. [45]. In all cases the ansatz circuit is applied to non-interacting antiferromagnetic initial states with two electrons per occupied lattice site. For this version of LDCA, which conserves the particle number, there are $K = 3d(n-1) + n$ parameters where the system size $n = 2L$, i.e., two qubits are used per lattice site.

Training instances for FLIP were generated for a number of sites $L_\tau \in [1, 6]$, a value of the interaction $U_\tau \in [0, 10]$ and a number of LDCA sublayers $d_\tau \in [1, 6]$. For each new training problem, these values are sampled uniformly within their respective discrete or continuous ranges and the cost function is taken to be to the expectation value of the $l1$-normalized version of the problem Hamiltonian, Eq. 5. For testing, both the number of sites and of circuit layers are increased to $L_{\tau'} \in \{6, 8, 10\}$ and $d_{\tau'} = 8$ and values of U are taken at regular intervals in the range $[0, 10]$. Extended details on the training and testing can be found in Appendix. A.

Optimization results averaged over the testing problems are reported in Fig.4(a). For random parameter initialization, each problem optimization is restarted five times. Results corresponding to the average over these repetitions (orange) or to the best per problem (green) are compared to optimizations of circuits initialized with FLIP (blue). After only $c.30(50)$ steps of optimization, circuits initialized with FLIP achieve similar convergence when compared to 100 steps of optimizations from random initialization for the average (best of five) case. When looking at individual cases, as reported in Fig.4(b), one can see that again the advantage of FLIP is the most prominent for the largest circuits (last row) over which it was applied. Importantly, FLIP outperforms random initialization in the strong coupling regime (i.e., away from $U = 0$), where the non-interacting initial state generally provides a poor starting point for VQE optimization. We emphasize that prior to this study there was no method for initializing parameters of LDCA circuits other than assigning random values.

Finally, in Appendix. D we report results obtained for an extended number of optimization steps performed after initialization. We also include results for the case where the interaction strength can adopt both positive and negative values. In particular, we obtained similar positive results for the case $U \in [-3, 10]$ but less of an advantage found on the extended range $U \in [-10, -3]$. There, we discuss potential modifications to FLIP to boost its performance in that region as well.

IV. OUTLOOK

In its simplest form, FLIP can be applied to a single VQA objective but with circuits of varying depths. Rather than growing the circuits sequentially and making incremental adjustments of parameters [6, 12, 25], which may fail in certain cases [46], FLIP aims at capturing and exploiting patterns in the parameter space and thus can provide a more robust approach. This flexibility towards learning over circuits of different sizes is one of the outstanding features of our initialization scheme. Still, as illustrated in the three case studies presented here, the full capability of FLIP appear in scenarios where both the circuits and the objectives (corresponding to the target states in Sec. III A, the graph instances in Sec. III B, and the interaction strengths in Sec. III C) are varied. Rather than considering each task individually, FLIP provides a unified framework to learn good initial parameters over many problems, resulting in overall faster convergence.

Although the first use case of applying FLIP to mitigate barren plateaus was demonstrated in a simple synthetic setting similar to those previously studied in the literature, we further demonstrated that FLIP is a promising initialization technique for more complex problems. The intuition behind a successful parameter initialization with FLIP is that as long as there is some (hidden) structure in the parameter space that can be learned by the framework, this can be exploited to adequately initialize new circuits even when these circuits are larger than the ones seen during training. A clear demonstration of learning such patterns was the application of FLIP to the max-cut instances in QAOA, in which it outperformed other proposed initialization techniques. We also observed an enhancement over random initialization in the application to the 1D FHM instances, where the structure in the parameter space is not obvious even after training. We are currently working on extending FLIP to other application domains, especially those that lack a "problem Hamiltonian" to guide the construction of the circuit ansatz. This is, for example, the case for probabilistic generative modeling with Quantum Circuit Born Machines (QCBMs) [20].

We highlight that our proposed encoding–decoding scheme allows to fully condition the initial parameters with respect to specific details of the task-at-hand, e.g., the interaction strength of the parametrized Hamiltonians in Sec. III C. Such task-dependent feature was also proposed in [14] but their method was limited to fixed-size circuits and relied on a more rigid encoding scheme. This possibility to easily incorporate informative details of the task can be further explored and exploited. For the QAOA graph problems, we intend to extend the encoding of Sec. III B to also incorporate information about the graph instances, e.g., their densities.

The meta-learning aspect of FLIP we have adapted here [13] is a well-studied paradigm for which many extensions have been proposed [15–17, 47]. These could be readily incorporated. In particular, the ability to train the learning rates to be used after initialization [15], in addition to training the initial parameters, could further contribute to more efficient optimizations (this is discussed further in Appendix. D).

Finally, we note that a recent work reported well-behaved optimization landscapes for over-parametrized circuits in the case of employing the Hamiltonian Variational Ansatz [38]. In this work, the authors observed that for circuits with depths scaling at most polynomially with the system size, low-quality local minima in corresponding objective landscapes disappear, and optimization becomes relatively easy. It would be interesting to apply FLIP to these problems and assess if this onset of "easy trainability" can be even further enhanced with better initialization strategies. In general, we expect that informed initialization of the parameters can accelerate convergence and thus reduce the overall number of circuits to be run, which is critical for extending the application of VQAs to larger problem sizes. As gate-based quantum computing technologies mature, initialization techniques which embrace this unique flexibility will be essential to mitigate the challenges in trainability posed for PQC-based models and eventually scale to their application in real-world applications settings.

ACKNOWLEDGMENTS

F.S would like to acknowledge Zapata Computing for hosting his Quantum Applications Internship. All authors would like to acknowledge access to the Orquestra® software platform where all simulations where performed.

[1] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational Quantum Algorithms, arXiv e-prints , arXiv:2012.09265 (2020), arXiv:2012.09265 [quant-ph].

[2] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, *et al.*, Noisy intermediate-scale quantum (nisq) algorithms, arXiv preprint arXiv:2101.08448 (2021).

[3] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Science and Technology **4**, 043001 (2019), arXiv:1906.07682.

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

[5] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost-function-dependent barren plateaus in shallow quantum neural networks, arXiv (2020), arXiv:2001.00550.

[6] 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 **10**, 021067 (2020).

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

[8] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, An initialization strategy for addressing barren plateaus in

parametrized quantum circuits, Quantum **3**, 214 (2019).
[9] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, An adaptive variational algorithm for exact molecular simulations on a quantum computer, Nature Communications **10**, 3007 (2019).
[10] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni, Learning to learn with quantum neural networks via classical neural networks, arXiv:1907.05415 (2019).
[11] M. Wilson, S. Stromswold, F. Wudarski, S. Hadfield, N. M. Tubman, and E. Rieffel, Optimizing quantum heuristics with meta-learning, arXiv:1908.03185 (2019).
[12] A. Skolik, J. R. McClean, M. Mohseni, P. van der Smagt, and M. Leib, Layerwise learning for quantum neural networks, arXiv:2006.14904 (2020).
[13] C. Finn and S. Levine, Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm, arXiv:1710.11622 (2017).
[14] A. Cervera-Lierta, J. S. Kottmann, and A. Aspuru-Guzik, The meta-variational quantum eigensolver (meta-vqe): Learning energy profiles of parameterized hamiltonians for quantum simulation, arXiv:2009.13545 (2020).
[15] Z. Li, F. Zhou, F. Chen, and H. Li, Meta-sgd: Learning to learn quickly for few-shot learning, arXiv preprint arXiv:1707.09835 (2017).
[16] A. Nichol, J. Achiam, and J. Schulman, On first-order meta-learning algorithms, arXiv:1803.02999 (2018).
[17] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, Meta-learning with latent embedding optimization, arXiv:1807.05960 (2018).
[18] D. Wecker, M. B. Hastings, and M. Troyer, Progress towards practical quantum variational algorithms, Phys. Rev. A **92**, 042303 (2015).
[19] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New Journal of Physics **18**, 023023 (2016).
[20] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, A generative modeling approach for benchmarking and training shallow quantum circuits, npj Quantum Information **5**, 45 (2019).
[21] J.-G. Liu and L. Wang, Differentiable learning of quantum circuit born machines, Phys. Rev. A **98**, 062324 (2018).
[22] C. Lemke, M. Budka, and B. Gabrys, Metalearning: a survey of trends and technologies, Artificial Intelligence Review **44**, 117 (2015).
[23] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, Meta-Learning in Neural Networks: A Survey, arXiv e-prints , arXiv:2004.05439 (2020), arXiv:2004.05439 [cs.LG].
[24] A. Mari, T. R. Bromley, and N. Killoran, Estimating the gradient and higher-order derivatives on quantum hardware, Phys. Rev. A **103**, 012405 (2021).
[25] G. Pagano, A. Bapat, P. Becker, K. S. Collins, A. De, P. W. Hess, H. B. Kaplan, A. Kyprianidis, W. L. Tan, C. Baldwin, L. T. Brady, A. Deshpande, F. Liu, S. Jordan, A. V. Gorshkov, and C. Monroe, Quantum approximate optimization of the long-range ising model with a trapped-ion quantum simulator, Proceedings of the National Academy of Sciences **117**, 25396 (2020), https://www.pnas.org/content/117/41/25396.full.pdf.
[26] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, *et al.*, Pennylane: Automatic differentiation of hybrid quantum-classical computations, arXiv preprint arXiv:1811.04968 (2018).
[27] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, M. Y. Niu, R. Halavati, E. Peters, *et al.*, Tensorflow quantum: A software framework for quantum machine learning, arXiv preprint arXiv:2003.02989 (2020).
[28] S. G. Edward Farhi, Jeffrey Goldstone, A quantum approximate optimization algorithm, arXiv:1411.4028 (2014).
[29] F. G. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven, For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances, arXiv preprint arXiv:1812.04170 (2018).
[30] G. E. Crooks, Performance of the quantum approximate optimization algorithm on the maximum cut problem, arXiv preprint arXiv:1811.08419 (2018).
[31] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, Benchmarking the quantum approximate optimization algorithm, Quantum Information Processing **19**, 197 (2020).
[32] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549**, 242 (2017).
[33] P.-L. Dallaire-Demers, J. Romero, L. Veis, S. Sim, and A. Aspuru-Guzik, Low-depth circuit ansatz for preparing correlated fermionic states on a quantum computer, Quantum Science and Technology **4**, 045005 (2019).
[34] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nature Communications **5**, 4213 EP (2014).
[35] Https://www.orquestra.io/.
[36] N. Lacroix, C. Hellings, C. K. Andersen, A. Di Paolo, A. Remm, S. Lazar, S. Krinner, G. J. Norris, M. Gabureac, J. Heinsoo, A. Blais, C. Eichler, and A. Wallraff, Improving the performance of deep quantum optimization algorithms with continuous gate sets, PRX Quantum **1**, 110304 (2020).
[37] M. P. Harrigan, K. J. Sung, M. Neeley, K. J. Satzinger, F. Arute, K. Arya, J. Atalaya, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, D. Eppens, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, A. Ho, S. Hong, T. Huang, L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, M. Leib, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, C. Neill, F. Neukart, M. Y. Niu, T. E. O'Brien, B. O'Gorman, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, A. Skolik, V. Smelyanskiy, D. Strain, M. Streif, M. Szalay, A. Vainsencher, T. White, Z. J. Yao, P. Yeh, A. Zalcman, L. Zhou, H. Neven, D. Bacon, E. Lucero, E. Farhi, and R. Babbush, Quantum approximate optimization of non-planar graph problems on a planar superconducting processor, Nature Physics 10.1038/s41567-020-01105-y (2021).
[38] R. Wiersema, C. Zhou, Y. de Sereville, J. F. Carrasquilla, Y. B. Kim, and H. Yuen, Exploring entanglement and optimization within the hamiltonian variational ansatz, arXiv:2008.02941 (2020).
[39] L. Li, M. Fan, M. Coram, P. Riley, and S. Leichenauer, Quantum optimization with a novel gibbs objective function and ansatz architecture search, Phys. Rev. Research **2**, 023074 (2020).
[40] E. Dagotto, Correlated electrons in high-temperature superconductors, Rev. Mod. Phys. **66**, 763 (1994).

[41] P. F. LeBlanc, A. E. Antipov, F. Becca, I. W. Bulik, G. K. L. Chan, C. M. Chung, Y. Deng, M. Ferrero, T. M. Henderson, C. A. Jiménez-Hoyos, E. Kozik, X. W. Liu, A. J. Millis, N. V. Prokof'ev, M. Qin, G. E. Scuseria, H. Shi, B. V. Svistunov, L. F. Tocchio, I. S. Tupitsyn, S. R. White, S. Zhang, B. X. Zheng, Z. Zhu, and E. Gull, Solutions of the two-dimensional hubbard model: Benchmarks and results from a wide range of numerical algorithms (2015).

[42] C. Cade, L. Mineh, A. Montanaro, and S. Stanisic, Strategies for solving the Fermi-Hubbard model on near-term quantum computers, Physical Review B **102**, 235122 (2020), arXiv:1912.06007.

[43] E. H. Lieb and F. Wu, The one-dimensional Hubbard model: a reminiscence, Physica A: Statistical Mechanics and its Applications **321**, 1 (2003), arXiv:0207529 [cond-mat].

[44] P.-L. Dallaire-Demers, M. Stechły, J. F. Gonthier, N. T. Bashige, J. Romero, and Y. Cao, An application benchmark for fermionic quantum simulations, arXiv preprint arXiv:2003.01862 (2020), arXiv:2003.01862.

[45] S. Sim, J. Romero Fontalvo, J. F. Gonthier, and A. A. Kunitsa, Adaptive pruning-based optimization of parameterized quantum circuits, Quantum Science and Technology 10.1088/2058-9565/abe107 (2021), arXiv:2010.00629.

[46] E. Campos, A. Nasrallah, and J. Biamonte, Abrupt transitions in variational quantum circuit training, arXiv preprint arXiv:2010.09720 (2020).

[47] S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell, Meta-learning with warped gradient descent, arXiv preprint arXiv:1909.00025 (2019).

[48] We refer to a layer of nearest-neighboring two-qubit gates that start with acting on the first and second qubits as an even layer, and one that start with acting on the second and third qubits as an odd layer.

[49] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[50] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, On the variance of the adaptive learning rate and beyond, in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* (OpenReview.net, 2020).

[51] T. Albash and D. A. Lidar, Adiabatic quantum computation, Rev. Mod. Phys. **90**, 015002 (2018).

Appendix A: Hyper-parameters

In this appendix we report the design choices, commonly referred as hyper-parameters, used for the results presented in Sec. III. For both the state preparation example of Sec. III A and the FHM of Sec. III C, FLIP is only compared to random initialization. For the max-cut problems of Sec. III B, FLIP is compared to other trainable initialization strategies (heuristics and RNN) for which the hyper-parameters used are also reported here. These hyper-parameters are grouped in three categories: Encoder-Decoder for FLIP (A 1), training of the initializers (A 2) and testing (A 3).

1. Encoder – Decoder

As detailed in Sec. II C, the encoding part of FLIP consists of a fixed mapping from any parameter belonging to a PQC problem to an encoding vector of size S. Crucially, this encoding vector is taken such that any pair of parameter and circuit is mapped to a distinct vector. These encodings are then fed to a trainable decoder producing initial values of the parameters.

For the state preparation problems in Sec. III A, each encoding vector contains the location of the parametrized gate, i.e., its qubit and layer index, and the size of the overall ansatz, i.e., its number of qubits and layers. In addition to these 4 values, the target position which specifies the state to be prepared p_τ is also included, accounting for a total size of $S = 5$ values per encoding.

For the QAOA circuits in Sec. III B, the layer index, the total number of layers, and a Boolean value indicating the gate type (problem or mixer unitary) are part of the encoding, giving a total of $S = 3$ entries.

At last, for the LDCA circuits in Sec. III C, each encoding includes the index of the first qubit the corresponding parametrized gate acts on, the type of gate, and its layer index. In this case, eight different types of gates are involved and are encoded using a one-hot encoding of size 8. The eight gate types are the following:

- R_z acting on initial states (1) $|0\rangle$ and (2) $|1\rangle$,
- Gate operations $e^{-i\theta XX/2}$ or $e^{-i\theta YY/2}$ in (3) even and (4) odd circuit layers [48],
- Gate operation $e^{-i\theta ZZ/2}$ in (5) even and (6) odd layers,
- Gate operations $e^{-i\theta XY/2}$ or $e^{-i\theta YX/2}$ in (7) even and (8) odd layers.

We refer the readers to [45] for further details on the construction of the ansatz. Additionally, the size of the ansatz, i.e., its number of layers and qubits, and the value of the interaction strength U are also included, giving an overall dimension of $S = 13$.

The decoder consists of a simple feed-forward neural network, with ReLu activation functions and linear output, of dimensions (number of layers $\times$ neurons per layer) 6×30 for the state preparations, 4×30 for the max-cut, and 4×20 for the FHM problems.

In machine learning, it is good practice to make sure that inputs and outputs of neural networks have reasonable scale. For this purpose, the outputs of the decoder, which correspond to rotation angles, are systematically rescaled by a factor π and elements of the encoding vectors (except for the gate types) are divided by a factor in between 10 to 15, as we consider circuits up to maximum of c.20 qubits and layers. Finally in all the examples the costs used during training are normalized using the $l1$-norm of the corresponding operators.

2. Training

For training, we report in Table. I the number N of problem instances used and their sizes, i.e., the number of qubits n, the depth d, and the number of parameters K of the circuits. In the case of FLIP these values are indicated as a range, as

training is performed on problems of different sizes. In addition we include the learning rate α and number of epochs e used for training. All the initializers are trained using Adam [49] with learning rate α. For FLIP the value s, used in Eq. 1, corresponding to the meta-learning aspect is fixed to 5, and η is reported in Table I.

TABLE I. Training hyper-parameters.

	N	n	d	K	e	α	η
State preparation							
FLIP	150	[1,8]	[1,8]	[1,64]	100	4.10^{-3}	10^{-1}
QAOA							
FLIP	200	[6,9]	[1,8]	[1,16]	90	4.10^{-3}	10^{-1}
Heuristics	200	[6,9]	8	16		4.10^{-3}	10^{-1}
RNN	200	[6,9]	8	16	400	4.10^{-3}	10^{-1}
Fermi-Hubbard							
FLIP	300	[4,12]	[2,6]	[8,276]	100	10^{-3}	2.10^{-2}

3. Testing

Similarly, in Table II we report the number of problem instances N used to produce the averaged testing results and their sizes. After initialization (either with FLIP , randomly or other initializer presented in Sec. III B), optimizations are performed for 100 steps with standard gradient descent (III A) or Adam (Sec. III B, III C). The learning rates α used are also reported. In each case we ensured that appropriate learning rates were chosen. For random initialization, these were found to be larger than the rates used after informed initialization.

TABLE II. Testing hyper-parameters.

	N	n	d	K	α
State preparation					
FLIP	50	[4,16]	[4,16]	[22,210]	10^{-1}
Random	"	"	"	"	3.10^{-1}
QAOA					
FLIP	100	12	8(12)	16(24)	2.10^{-2}
Heuristics	"	"	"	"	2.10^{-2}
RNN	"	"	"	"	2.10^{-2}
Random	"	"	"	"	10^{-1}
Fermi-Hubbard					
FLIP	21	[12,20]	8	[268,476]	2.10^{-2}

Appendix B: State preparations

In Sec. III A we compared optimizations for state preparation problems, ran for PQCs initialized both randomly and with FLIP . Results presented in Fig. 2 show the advantage of FLIP over random initialization on average over 50 testing problems. Here we further characterize this advantage. First, in Appendix. B 1, we present extended data exhibiting vanishing gradient patterns, i.e., barren plateaus, for randomly initialized circuits. Then, individual optimization results are reported in Appendix. B 2. In line with Sec. III A, these results are obtained with standard gradient descent routines, however we notice that one could seemingly escape barren plateaus in the case of random initialization by using more refined optimization techniques, i.e., Adam [49]. This is reported and discussed in Appendix. B 3. However, as showcased in Sec. B 4, when noise is taken into account, this workaround collapses. Still, even with the addition of noise during training and testing, FLIP remains highly competitive.

All the results in this appendix section correspond to the problems presented in Sec. III A and, except when explicitly stated otherwise, we resort to the exact same version of FLIP , that is without any additional training. For the sake of simplicity we only illustrate results for the case where the target position is fixed to $p = 1$, which corresponds to the task of preparing a state $|10\dots0\rangle$ of arbitrary size, but other values of p exhibit similar performances.

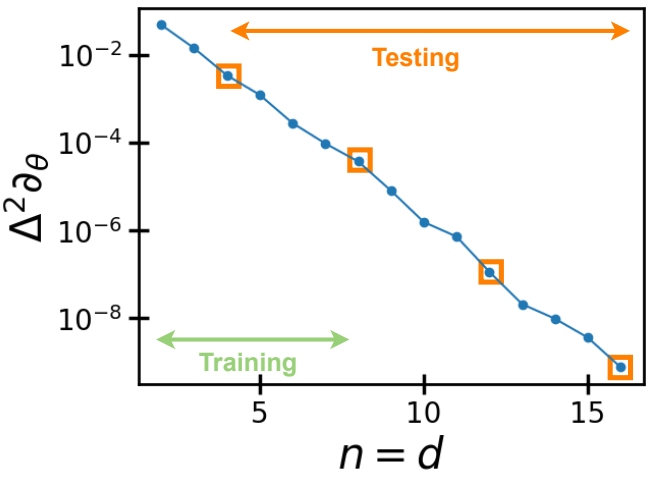

FIG. 5. Empirical variances in the gradients of the state preparation objectives for randomly initialized circuits. These variances are obtained for different system sizes from $n = 2$ to 16 qubits and a circuit depth scaling linearly with the size, $n = d$. Each circuit is repeated 250 times with random parameters, and we report the average of the variances of the gradient $\Delta^2\partial_\theta$ over each parameter. Optimization of the circuits with sizes highlighted in orange squares are studied individually in Fig. 6,7, 8.

1. Vanishing gradients

In Fig. 5(c) we report empirical variances of the gradients as a function of the system size. For these data we take the number of layers to be equal to the system size. Variances $\Delta^2\partial_\theta$ are obtained over 250 random initializations for each circuit size and averaged over each of the parameters. We observe an exponential decay of these variances as a function of the system size n, indicating the emergence of barren plateaus: for

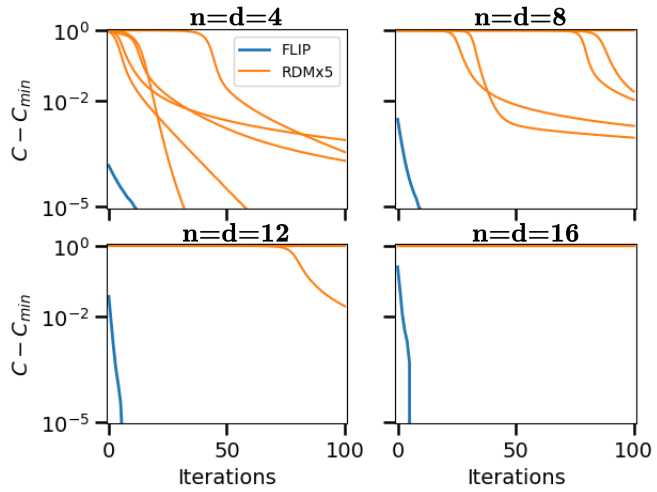

FIG. 6. Individual optimization results for the state preparation problems with circuits of different sizes from $n = d = 4$ to $n = d = 16$. Optimizations starting with FLIP (blue curves) are compared to random initialization (orange curves). For each problem random initialization is repeated 5 times.

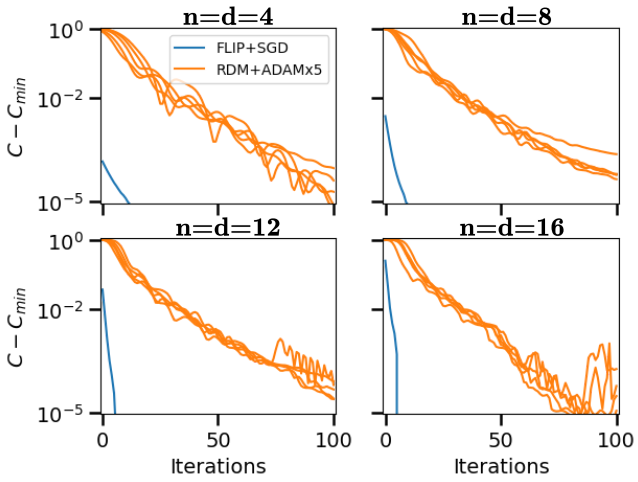

FIG. 7. Optimization with Adam. Results similar to Fig.6, but for randomly initialized circuits the gradient-descent method Adam is used.

even moderately system sizes random parameters will most likely correspond to a vanishing-gradient region of the optimization landscape. Note that reducing the depth of the circuits and choosing a different cost can in principle alleviate barren plateaus for this specific example [5]. Still, as we aim at testing FLIP in challenging situations, we did not resort to such alternatives.

2. Individual optimizations

A selected subset of optimizations is plotted in Fig. 6, spanning problems of different sizes from $n = 4$ to 16 qubits, with a circuit depth equal to the number of qubits, i.e., $n = d$. For each of these 4 individual problems, the optimizations starting from random parameters are repeated 5 times.

Looking at optimization traces for randomly initialized circuits, one can see that convergence quickly deteriorates when the size of the circuits involved is increased. Already for system size $n = 8$ qubits, only two out of the five runs show progress before c.50 iterations. For the case $n = 12$ qubits only one out of the five optimizations converged reasonably close to the minimum, while for $n = 16$ qubits none of the runs manage to even slightly improve the objective value.

In contrast, for all the problem instances, circuits initialized with FLIP converged quickly with an advantage over random initialization which is most appreciable for the largest circuits considered. The case $n = d = 16$ qubits corresponds to problems twice as deep and as wide (with four times more parameters) than the largest circuit seen during training. This probes the ability of FLIP to correctly identify successful parameters patterns based on problems of moderate sizes which can be extrapolated to larger ones.

3. Optimization with Adam

Results presented in Fig. 6 were obtained with standard gradient descent, where the step in parameter space taken at each iteration is obtained by multiplying the gradient vector by a fixed scalar value (the learning rate). Many variations of this simple routine have been explored, especially in the context of machine learning. In particular, Adam [49] has become a popular optimizer used when training neural networks. In Fig. 7 we reproduce the results displayed in Fig. 6, with the exception that optimizations starting from random parameters (green curves) are now performed using Adam. In contrast to Fig. 6, despite the presence of barren plateaus, optimizations with Adam are able to progress for any of the sizes studied. Note that even in this case optimizations with circuits initialized with FLIP , followed by standard gradient descent, converge significantly faster.

Adam relies on adaptive learning rates, which can effectively be quite large especially in the first steps of optimizations [50]. That is, in principle vanishing gradients could be magnified by re-scaling them by an arbitrarily large scalar value or tensor. However we expect that when noise is included and dominate the magnitude of these gradients (or of the higher order derivatives [24]), such approach should fail as it will consist of taking large steps in almost random directions. In the next section, we verify that it is indeed the case for Adam.

4. Optimization with noise

So far, we have assumed noiseless gradients. However in practice these gradients are estimated through a finite number of measurements and statistical noise needs to be accounted for. For that purpose we now include additive i.i.d Gaussian noise with standard deviation $\sigma_n = 0.01$ and report results

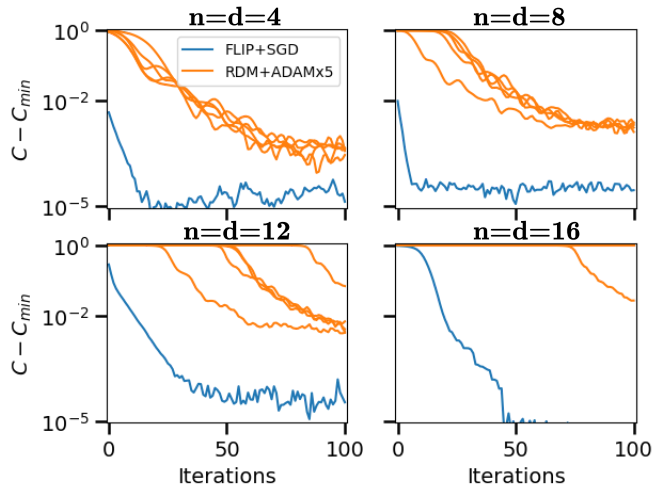

FIG. 8. Optimization with noisy gradients. Exact same optimizations as shown in Fig.7, except for the addition of noise in the gradients. This noise is taken to be i.i.d Gaussian distributed with zero mean and a standard deviation of $\sigma_n = 0.01$. This noise is included in the gradients during optimization and also when training FLIP .

of optimization performed in this more realistic scenario in Fig. 8.

For random initialization (green curves) we resort to Adam as it was found beneficial in the noiseless scenario. Noticeably the ability to overcome barren plateaus seen in the noiseless case is now suppressed, as displayed in Fig. 8 for the cases $n = d \geq 12$ (bottom row). This confirms our intuition that strategies relying on re-scaling gradients by large values are probably not viable in realistic conditions.

In the case of FLIP (blue curves), both its training and testing are subjected to noisy gradients. When compared to Fig. 7, one can notice a slight decrease in the overall rate of convergence. Still the main conclusions drawn for the noiseless case remain unchanged: (i) for circuit sizes pertaining to the training distributions circuits are initialized already close to the optimal parameters (ii) for larger circuits, despite starting further away, the parameters initialized with FLIP remains in a region where the gradient signal is strong enough such that the initial parameters can be quickly refined by gradient-descent. As such, these results confirm the ability of FLIP to mitigate the barren plateau phenomenon as long as underlying patterns in the optimal parameters over varied circuit sizes and objectives can be learnt.

Appendix C: Max-cut with QAOA

Here we present the patterns in the initial parameters learnt by FLIP over the distribution of max-cut problems with the QAOA circuits discussed in Sec. III B. We plot in Fig. 9 the initial parameter values, produced by FLIP , for circuits of dimension $d_\tau \in [1, 12]$ layers (colours). The absolute values (rescaled by π) of the parameters β_l and γ_l are reported in Fig. 9(a), left and right panels respectively, as a function of the

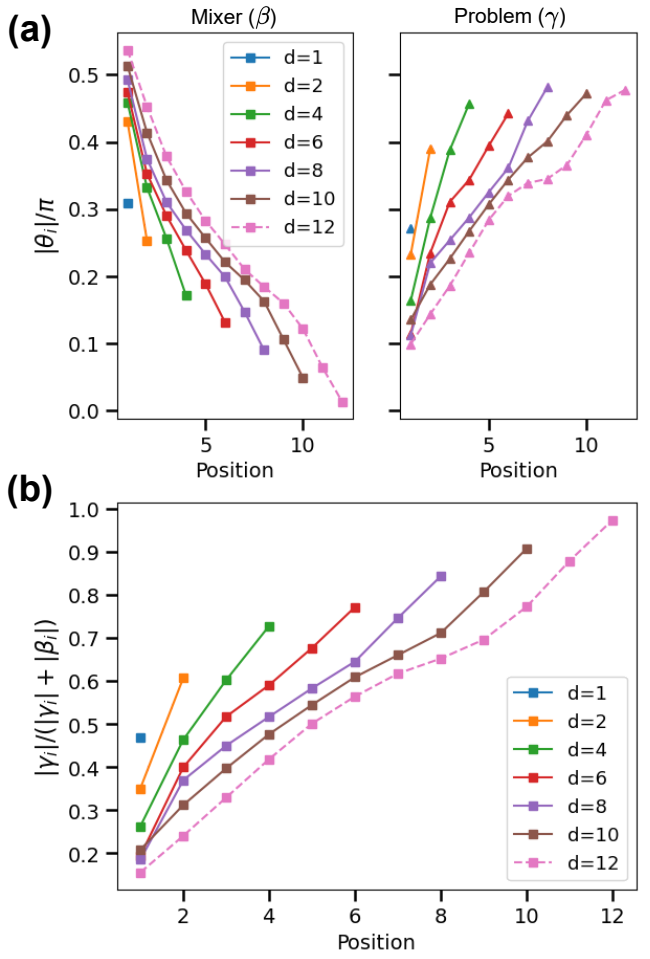

FIG. 9. Patterns in the initial parameters learnt by FLIP for the max-cut problems with QAOA circuits. **(a)** Absolute values of the parameters θ_l for varied circuit sizes within $1 \leq d \leq 12$ layers (colours in legend) and varied layer position l (x-axis). These are grouped by parameters pertaining to the mixer (β_l) and problem (γ_l) unitaries, in the left and right panels respectively. The values (y-axis) are reported as a fraction of π. **(b)** Ratio $|\gamma_l|/(|\gamma_l| + |\beta_l|)$.

layer position l. In Fig. 9(b) we plot the ratio $|\gamma_l|/(|\gamma_l| + |\beta_l|)$.

Clear patterns emerge: at fixed circuit depth the absolute values of β_l (γ_l) decrease (increase) for increasing layer position l. In addition, one can also inspect changes in the initial values of a fixed parameter but for circuits of varied sizes. In this case β_l (γ_l) are also found to increase (decrease) for increasing circuit depths. These patterns are qualitatively similar to the patterns discovered and exploited in the context of max-cut QAOA problems [6] and are reminiscent of smooth schedules used in adiabatic quantum computing [51].

In [6] the authors conducted a thorough study of optimal parameters found for different QAOA circuit sizes over many repeated optimizations. Insights gained on the structure of these optimal parameters for small problems were subsequently exploited to engineer a sequential optimization strategy. A circuit is grown one layer at a time and the initial parameters of new circuits are obtained by extrapolating the optimized pa-

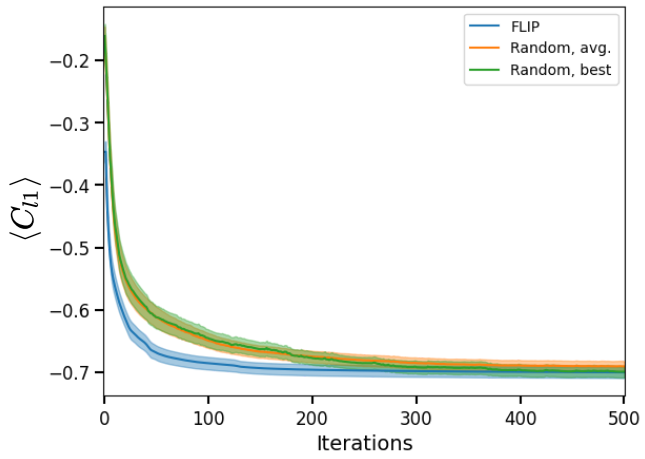

FIG. 10. Fermi-Hubbard model with 500 optimization steps. Similar to Fig. 4, the average of the normalized costs for circuits initialized with FLIP and randomly are compared, but with a larger number of iterations (500 steps instead of 100).

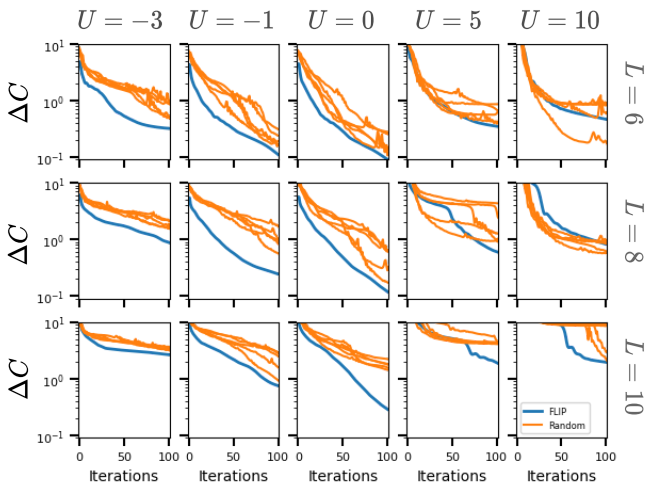

FIG. 11. Fermi-Hubbard model with interaction strengths $U \in [-3, 10]$. Results of individual optimizations with randomly and FLIP -initialized circuits. The training of FLIP is also performed based on this extended range of interaction values.

rameters of the previous circuit. Directly learning these patterns over many circuit sizes, as made possible using FLIP , avoids such sequential procedure and handcrafted extrapolation rules.

Appendix D: Fermi-Hubbard model

In this appendix we provide additional results obtained for the Fermi-Hubbard model described in Sec. III C.

In Fig. 10 we report results of optimizations of circuits either initialized with FLIP or randomly with an extended number of optimization steps (500 instead of the 100 iterations showcased in the main text). Similarly to Fig. 4(a) the values of the normalized costs are averaged over all the testing problems and displayed as a function of the number of iterations performed. One can see that the gap between the different initializers shrinks only after a large number of steps, $c.300$. Circuits initialized with FLIP converge faster and show (slightly) better final convergence in average, even when compared to the best out of 5 random initializations per problem.

At last, we also consider the case where interaction strengths U of the FHM can adopt both positive and negative values. Results for the case $U \in [-3, 10]$ are reported in Fig. 11 for individual testing problems spanning both positive and negative interaction cases. Details of the training and testing are the same as in Sec. III C, except for the interaction strength which is now drawn uniformly within its new range. These results are in line with the ones reported in the main text: in almost all cases circuits initialized with FLIP converge to lower values than the best out of 5 random initializations.

However when considering the larger interval of interactions $U \in [-10, 10]$, we found it harder to train FLIP and in some cases our strategy under-performs random initialization. Inspecting the absolute values of the normalized costs (used for training) as a function of the interaction strengths, we notice that they were significantly smaller for the range $U = [-10, -3]$ than for the range of interaction strengths $U = [-3, 10]$. We believe that it explains the degradation in performance of FLIP and expect that a better choice of normalization could resolve these difficulties. Additionally, it would be interesting to extend FLIP to also learn optimal learning rates [15] to be used after initialization. Similarly to the initial parameters, these could be made dependent on the underlying details of the problems, and in this case compensate for the changes in magnitudes of the cost for varied values of interactions.