Meta-learning digitized-counterdiabatic quantum optimization

Pranav Chandarana, Pablo Suárez Vieites, Narendra N Hegade, Enrique Solano, Yue Ban, and Xi Chen

1 Department of Physical Chemistry, University of the Basque Country UPV/EHU, Apartado 644, 48080 Bilbao, Spain
2 EHU Quantum Center, University of the Basque Country UPV/EHU, 48940 Leioa, Spain
3 International Center of Quantum Artificial Intelligence for Science and Technology (QuArtist) and Physics Department, Shanghai University, 200444 Shanghai, People’s Republic of China
4 Kipu Quantum, Greifswalderstrasse 226, 10405 Berlin, Germany
5 IKERBASQUE, Basque Foundation for Science, Plaza Euskadi 5, 48009 Bilbao, Spain
6 TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Spain
7 Co-first authors.
* Author to whom any correspondence should be addressed.
E-mail: chenxi1979cn@gmail.com

Keywords: meta-learning, parameter concentration, shortcuts to adiabaticity (STA), counterdiabatic driving, QAOA, digitized-counterdiabatic quantum computing

Abstract

The use of variational quantum algorithms for optimization tasks has emerged as a crucial application for the current noisy intermediate-scale quantum computers. However, these algorithms face significant difficulties in finding suitable ansatz and appropriate initial parameters. In this paper, we employ meta-learning using recurrent neural networks to address these issues for the recently proposed digitized-counterdiabatic quantum approximate optimization algorithm (QAOA). By combining meta-learning and counterdiabaticity, we find suitable variational parameters and reduce the number of optimization iterations required. We demonstrate the effectiveness of our approach by applying it to the MaxCut problem and the Sherrington–Kirkpatrick model. Our method offers a short-depth circuit ansatz with optimal initial parameters, thus improving the performance of the state-of-the-art QAOA.

1. Introduction

Hybrid quantum–classical optimization has been of such importance in quantum algorithms that has captured the attention of the quantum computing community, particularly because of its relevance for the noisy intermediate-scale quantum devices [1–6]. Variational quantum algorithms (VQAs) leverage quantum circuits with variational parameters to minimize a cost function, assisted by classical optimization routines. Among them, the quantum approximate optimization algorithm (QAOA) [7] has proved to be a promising algorithm to tackle combinatorial optimization problems [5]. However, while QAOA has been proposed as a demonstration of quantum supremacy [8], there still remain several challenges that must be addressed.

One of the key challenges for any variational quantum optimization algorithm is to enhance the expressibility of the circuit ansatz while ensuring the trainability, thereby avoiding the barren-plateau [9, 10]. To combat this challenge, an approach is to add terms to the circuit ansatz to enlarge the solution space. As QAOA closely resembles the quantum adiabatic evolution, the use of counterdiabatic (CD) protocols has been proposed [11–15]. CD protocols are the well-known methods of shortcuts to adiabaticity (STA) [16, 17], that are utilized to circumvent the need for slow driving, and also improve the state-of-the-art QAOA. Implementing approximate counterdiabaticity in quantum many-body systems [18, 19] has shown significant improvement in adiabatic computing [20], factorization [21], portfolio optimization [22], preparing entangled states [23–26], and quantum annealing (QA) [27–29]. Particularly, a recent study has reported a polynomial enhancement over the current methods in solving a class of Ising spin glass optimization problems, where CD interaction serves as a non-stoquastic catalyst [30]. In practical applications, counterdiabaticity has been incorporated in the scheduling for QAOA-like algorithms [31], and further optimized through reinforcement learning (RL) [32] and variational circuit learning [33]. In the
digitized-counterdiabatic QAOA, (DC-QAOA) [34], appropriate CD terms are supplemented to the circuit ansatz, corresponding to the adiabatic gauge potentials [19], to improve the efficiency in reaching the ground state. By introducing additional CD terms with free parameters, the expressibility at a constant \( p \) is increased, providing a larger solution space to explore within a single algorithm layer. These algorithms are still an active subject of research, as there are numerous possibilities for further improvement.

Another challenge in quantum optimization is local optimization. Gradient-based optimization routines [35], such as quantum back-propagation [36] or finite-difference gradients [37, 38], can be expensive as they require additional circuit depth and a large number of measurements, respectively. One solution is to set suitable initial parameters that are close to the global minima. This can significantly reduce the number of optimization iterations needed. Several strategies for parameter initialization in QAOA have been proposed in the literature [5, 39–43]. Specifically, long short-term memory (LSTM), one of the most popular recurrent neural networks (RNNs), has been implemented for finding initial parameters in QAOA [43].

Indeed, the integration of quantum algorithms within machine learning has proven to be a powerful method for leveraging performance. RL algorithms [44, 45], such as Q learning [46, 47], policy gradient [48–50] and Alphazero [51], have been extensively applied in VQAs. Also, the policy gradient as an alternative optimizer has been exploited for QAOA [52]. Q-learning, proximal policy optimization, and hybrid optimization by Monte Carlo search [53] can be utilized to tackle combinatorial problems [54] and finding the ground state of the transverse Ising model [55]. In addition, LSTM has been employed to optimize the hyperparameters of a gradient-descent-based algorithm [56], resulting in an improvement over different variations of the gradient descent algorithm. Moreover, the ability of RNNs to capture temporal dependencies in data makes them well-suited for learning the dynamics of the optimization process in QAOA-like algorithm [57], which involves non-commuting operators that evolve alternately at discrete time intervals. It has been demonstrated that a single step of the algorithm mimics a step of a gradient optimizer, and the use of RNNs in optimizing the hyperparameters of gradient descent algorithms has already shown good results. Thus, the similarity between DC-QAOA and QAOA suggests that the use of RNNs in optimizing the parameters in DC-QAOA seems promising.

In this work, we aim to enhance the classical optimization part of the DC-QAOA and utilize RNNs as a black-box optimizer to find good initial DC-QAOA parameters. Specifically, we apply RNNs in the context of meta-learning [58], allowing them to learn from a dataset of graphs to predict optimal initial parameters for DC-QAOA. Complementary to the previous work [43], here we focus on the introduction of the DC-QAOA ansatz in the RNN learning paradigm, in order to improve conventional QAOA and analyze how the additional parameters and terms affect the training process. We also compare the performance of the LSTM and gated recurrent unit (GRU) architectures for the initialization of DC-QAOA parameters. Lastly, we clarify that the parameter concentration effect is still present in DC-QAOA even after new terms with free parameters are introduced.

The article is organized as follows: section 2 provides a brief overview of DC-QAOA and discusses its integration with RNNs. In section 3, we present the results of our experiments on the MaxCut problem for both unweighted and weighted instances, as well as on the Sherrington–Kirkpatrick (SK) model. Section 4 is dedicated to discussions and future directions of this work.

2. Meta-learning DC-QAOA

We commence by providing a brief overview of DC-QAOA, which is a more generalized version of QAOA and falls under the category of VQAs. In VQAs, a quantum circuit with variational parameters is utilized along with classical optimization routines to optimize a cost function,

\[
F(\gamma, \beta) = \langle \psi(\gamma, \beta) | H_p | \psi(\gamma, \beta) \rangle,
\]

which corresponds to the expectation value of the problem Hamiltonian \( H_p \) with respect to the quantum state \( | \psi(\gamma, \beta) \rangle \).

A \( p \)-depth QAOA circuit ansatz involves the iterative application (\( p \) times) of two unitaries:

\[
U_0(\beta) = e^{-iH_p} \quad \text{and} \quad U(\gamma) = e^{-iH_m},
\]

where \( H_m \) is the mixing Hamiltonian and \( H_p \) is the problem Hamiltonian. Assuming \( H_m = \sum_i \sigma_i^z \) for simplicity, the final state \( | \psi(\gamma, \beta) \rangle \) is given by

\[
| \psi(\gamma, \beta) \rangle = U(\gamma, \beta) | \psi_i \rangle,
\]

with

\[
U(\gamma, \beta) = U_p(\beta_p)U_c(\gamma_p)U_b(\beta_{p-1})U_c(\gamma_{p-1}) \ldots U_b(\beta_1)U_c(\gamma_1),
\]
where \(|\psi_i\rangle = |+\rangle^\otimes S\) is the initial state in the computational basis and \(S\) is the system size. Here, \((\beta, \gamma)\) are free parameters to be optimized by classical optimization routines, which can be either local or global. A systematic comparison of different types of optimizers is presented in [59]. The choice of QAOA ansatz is advantageous due to its scalability and good trainability [60, 61]. However, implementing high-depth QAOA circuits can be impractical experimentally due to various sources of noise [62]. Therefore, improving QAOA with low-depth circuits has been actively pursued [63–65].

In DC-QAOA, an additional term \(U_D(\alpha)\) is added to each layer, together with the standard \(U_b(\beta)\) and \(U_d(\gamma)\) terms. The \(U_D(\alpha)\) term is calculated using CD protocols. The inclusion of the \(U_D(\alpha)\) term in DC-QAOA brings two main advantages. Firstly, the additional free parameter \(\alpha\) and the non-commuting CD term make the ansatz more expressive. Secondly, the CD unitary effectively decreases the number of circuit ansatz layers required to reach a target state, which helps to reduce the depth of the quantum circuit and to improve its performance.

However, implementing the exact CD term can be challenging, especially when the prior knowledge of the system eigenstates is not available [11, 13, 66]. Instead, an approximate CD term can be calculated using the nested commutator (NC) method. This involves finding an approximate gauge of potential \(\lambda A^{(l)}\), using a finite expansion order \(l\), given by

\[
A^{(l)} = i \sum_{k=1}^{l} \alpha_k(t) [H, [H, \ldots [H, \partial_\lambda H]]],
\]

where \(H = (1 - \lambda)H_\gamma + \lambda H_p\) with time-varying scheduling function \(\lambda\) from 0 to 1, and \(l\) determines the order of expansion. The variational parameters are then obtained by minimizing the action, \(S = \text{Tr}[G_l]\) with \(G_l = \partial_\lambda H - i [H, A^{(l)}_\lambda]\) [19]. Here we truncate to two-local terms with \(l = 2\) and optimize their coefficients using classical optimizers. This approach limits the pool of operators to \(A_\lambda = \{\sigma^x, \sigma^y, \sigma^z, \sigma^x\sigma^y, \sigma^y\sigma^z, \sigma^z\sigma^x\}\), as the higher-order terms increase circuit depth and make implementation more difficult. To use DC-QAOA, we can randomly initialize its parameters and feed them into the circuit to compute \(F(\beta, \gamma, \alpha)\). A classical optimizer then updates \((\beta, \gamma, \alpha)\) to \((\beta', \gamma', \alpha')\) such that \(F(\beta', \gamma', \alpha')\) is minimized. However, randomly chosen initial DC-QAOA parameters can be far from the global minimum, making the classical optimization process extremely difficult. To address this issue, we use RNNs to find optimal initial parameters for DC-QAOA (as shown in Figure 1). While we focus on DC-QAOA parameters in this work, all the techniques presented in this section can also be applied to QAOA for comparison.
Figure 2. Schematic diagram depicting the flow of information on an LSTM (a) and GRU (b) cell for a time step $t$. The gates of the RNN modulate the amount of information that is kept from previous iterations and input vectors. The hidden state $h_t$, the cell state $c_t$, and the input $\theta_t$ (in our case, the optimizable parameters) are also shown. The GRU uses one less gate and does not utilize the cell state $c_t$. The symbols $\odot$, $\sigma$, and tanh denote the Hadamard product, the sigma function, and the hyperbolic tangent function, respectively.

Figure 3. Trainable parameters of RNN’s weight matrices and biases as a function of the number of layers for QAOA and DC-QAOA.

Meta-learning, also known as learning to learn, has recently gained popularity as a way to improve machine learning algorithms [67]. RNNs have proven to be particularly useful in this context [68], as they allow the network to exhibit temporal behavior by enabling inputs and outputs to depend on previous iterations. Unlike feed-forward networks, where an input vector is processed by different weight layers to produce an output, RNNs use a hidden state $h_t$ to keep information from previous iterations. In the case of the DC-QAOA problem (shown in figure 1), $\hat{\theta}_{t-1}$ represents the guess of the parameters in time step $t-1$, which serves as the input vector in step $t$ of the network such that $\theta_t = \hat{\theta}_{t-1}$. As a result, the output of an RNN at iteration $T$ yields a joint probability distribution, as shown in

$$\hat{\theta}_T = \prod_{i=1}^{T} p(\theta_i|\theta_{i-1}, \ldots, \theta_1) \equiv g(h_T),$$

where the function $g$ is chosen based on the problem being modeled and the network weights and biases.

LSTMs [69] and GRUs [70] are two popular types of RNNs that have been widely used in various applications. LSTMs are designed to identify both long-term and short-term dependencies in the input data, while GRUs are a simplified version of LSTMs that use fewer gate operations to achieve similar functionality. These networks partially solve the issue of vanishing and exploding gradients, which can occur when training RNNs using backpropagation [71, 72]. Figure 2 provides a schematic diagram of LSTM and GRU cells, and a detailed discussion on how information flows through these networks is given in appendix A. Figure 3 shows a comparison of the optimizable RNN parameter requirements for LSTM and GRU with both QAOA and DC-QAOA, as a function of the number of layers. The resources required for the DC-QAOA algorithm scale significantly faster than those required for QAOA. This is because DC-QAOA requires three parameters per layer, while QAOA requires only two. This difference in parameter requirements becomes a critical factor when considering the scalability of the technique.

The RNN takes as input $\theta$, the parameters in DC-QAOA, with all parameters $\theta_0$ set near zero for the first RNN iteration. At each RNN iteration, the network outputs the guesses of the parameters $\hat{\theta}_t = g(h_t)$, where $h_t$ is the hidden state at step $t$. We choose $g(h_t) = \pi h_t = \theta_t = \theta_{t+1}$, so that all the different parameters $\theta_t$ in DC-QAOA that the RNN output at step $t$ are on the interval $[-\pi, \pi]$, as $h_t \in [-1, 1]$. The hidden state $h_t$ of an RNN iteration depends on the flow of RNN gates described in appendix A, which in turn depends on RNN...
parameters, i.e. weights $W$ and biases $b$. In general, the loss function $L(\hat{\theta})$ used to train the RNN parameters can be formulated in different ways. In this work, we use the following expression:

$$L(\hat{\theta}) = \sum_{i=1}^{T} \omega_i F(\hat{\theta}_i), \tag{5}$$

where $T$ is the time horizon, $F(\hat{\theta}_i)$ is the expected value of the problem Hamiltonian for DC-QAOA (similar to equation (1)), and $\omega_i$ are arbitrary weights assigned depending on the relative importance of each iteration.

In order to train the RNN parameters for predicting the DC-QAOA parameters, we start by selecting a pool of $N$ graphs. For each graph, we run the RNN for $T$ iterations to obtain the outputs $\hat{\theta}_i$, which are then used to compute the loss function as shown in equation (5). The expected value of the problem Hamiltonian $F(\hat{\theta}_i)$ for each $\hat{\theta}_i$ depends on the adjacency matrix of the corresponding graph. We then use back-propagation \cite{73} to compute the gradient of the RNN parameters with respect to the loss function and update the RNN parameters using gradient descent optimizers. This process is repeated for each graph in the dataset and can be iterated for a total of $M$ times over the whole dataset until certain stop criteria for the loss function is met, as shown in the first step of the training process in figure 1.

After training the RNN parameters, further fine-tuning optimization based on the test graph instance is required to find the optimal parameters for the DC-QAOA. This is achieved using a local optimizer, which provides instance-based fine-tuning to the parameters to obtain better approximate solutions based on the specific graph. Using a local optimizer is advantageous as it allows us to reduce the time horizon $T$, thereby decreasing the computational cost needed to train the RNNs. Even if the RNN is trained for only a few time steps, the additional local optimizer can adjust the parameters and successfully solve the test problems to find good approximate solutions.

3. Results

3.1. MaxCut

In this section, we demonstrate the application of meta-learning to DC-QAOA for the MaxCut and SK models, showing that merging these two paradigms leads to a significant advantage in terms of reducing the number of local optimization iterations and achieving higher success rates in finding the ground state.

We start by considering a graph $G = (V, U)$, where $V$ represents the set of vertices and $U$ the set of edges. In the MaxCut problem, the objective is to find a subset of vertices $V_0$ and its complement $V_1$ that maximizes the number of edges between $V_0$ and $V_1$. Regarding the complexity, MaxCut is an NP-complete problem \cite{74}. To encode MaxCut into a Hamiltonian, we use $z = z_1 z_2 \ldots z_n$ such that if $z_i = 1$, the $i$th vertex is in set $V_0$ and if $z_i = -1$, the $i$th vertex is in set $V_1$. The cost Hamiltonian that we minimize is given by

$$H_p = -\frac{1}{2} \sum_{(i,j) \in U} w_{ij} (1 - \sigma_i^z \sigma_j^z), \tag{6}$$

where $w_{ij}$ are the weights assigned to edges.

While QAOA has shown promising results for the MaxCut problem, it is important to note that achieving quantum supremacy in terms of time may require hundreds of qubits. In \cite{75}, a classical solver is compared with QAOA for MaxCut in 3-regular graphs, highlighting the need for a large number of qubits in certain cases. The computational cost of solving the optimization of variational parameters in QAOA with constant $p$ for the MaxCut problem was shown to scale exponentially with the number of qubits, as estimated with noisy superconducting qubits data.

For our DC-QAOA computations, we choose $U_D(\alpha) = e^{-\frac{i}{2} \alpha \sum_i \sigma_i^z \sigma_i^z}$, where $i, j$ run over the edges of the respective graph. The performance of various CD terms is compared (see appendix B), from which the best-performing CD term is selected in our simulations. We conduct simulations on two types of problems: a 10-node unweighted 3-regular case and a 10-node complete weighted case. We generate random graph instances and train a type of RNN for each problem and variational circuits with or without CD terms. A set of 10 random graphs was chosen and optimized with random and LSTM/GRU initialization. The time horizon (number of RNN iterations) is set to $T = 6$. In each RNN iteration, we input the DC-QAOA parameters obtained from previous iterations, the hidden state, and the cell state in the case of the LSTM cell as inputs. The RNN parameters are initialized to zero every time we start the RNN, and the initial DC-QAOA parameters are near zero. We choose Adam optimizer \cite{76} for the unweighted case and Adagrad optimizer \cite{77} for the weighted one, both with a learning rate of 0.1 for training the RNNs. The pool of graphs size used for RNN training is 100 for the unweighted case and 300 for the weighted case. A detail of the optimizers
used in this work can be found in appendix C. The maximum number of epochs for RNN training (number of times we iterate over the graph dataset) is chosen to be $M = 10$. During the computations, we kept a tolerance $\tau = 0.01$ such that if $|L_{\text{epoch}}^{t+1} - L_{\text{epoch}}^{t}| \leq 0.01$, then the training process is stopped, where $L_{\text{epoch}}^{t}$ is the mean value obtained for the loss function in equation (5) overall graph dataset during the epoch. At this point, it is worthwhile to mention that the RNNs are supposed to be trained with data from actual quantum hardware. Training RNNs with data from quantum hardware is a challenging task, as measurement overhead and noise can disrupt the learning process. However, in this case, the number of measurements is low since the problems are classical, and all the Pauli terms commute. The introduced tolerance $\tau$ also helps to stop the RNN training once a certain precision is reached, reducing the number of measurements required. Moreover, the fact that the RNN only needs to be trained once and can be used to initialize any graph test makes it a better option than random initialization. The use of initialization in RNNs also helps to reduce the number of local optimization steps required to reach convergence. Therefore, the overall algorithm can help to reduce the measurement overhead. However, the study of noise in near-term quantum hardware is an important issue that requires further investigation.

In order to quantify the performance, we use the relative error $\mathcal{E}$ metric given by

$$\mathcal{E} = \frac{F(\beta, \gamma, \alpha) - E_0}{|E_0|}, \quad \text{(7)}$$

where $E_0$ is the energy of the ground state configuration. In figure 4, $\mathcal{E}$ is plotted as a function of the number of local optimizer steps for the 10-qubit unweighted and weighted MaxCut problems. The results compare $p = 2$ layers with randomly initialized QAOA/DC-QAOA parameters and LSTM/GRU initialized QAOA/DC-QAOA parameters. With the average of the 10 studied graphs instances, the shaded region shows the standard error, and the bar plots in the right corner show the results of the last step of local optimization. Adagrad optimizer with a learning rate of 0.1 is used as a classical local optimization routine for both QAOA and DC-QAOA. We observe that both methods successfully find the initial parameters and require only a few iterations to converge to the minima by the classical local optimizer.

For 3-regular MaxCut, the results presented in figure 4(a) reveal that LSTM-initialized DC-QAOA outperforms the others as it can reach lower $\mathcal{E}$ values with fewer iterations. On the other hand, GRU with QAOA performs better than LSTM with QAOA, and with an increasing number of iterations, GRU DC-QAOA surpasses GRU QAOA. In the case of a complete weighted case, a recent study on the optimization landscape for QAOA for weighted and unweighted MaxCut [78] states that weighted edges lead

![Figure 4. Relative error $\mathcal{E}$ as a function of the number of local optimizer steps for (a) 3-regular unweighted MaxCut and (b) first 50 steps of complete weighted MaxCut problem. Different colored lines compare the results of QAOA and DC-QAOA for $p = 2$ with randomly initialized parameters and with LSTM/GRU initialized parameters. The bar plots in the corner show the results of the last step of local optimization. The shaded region shows standard error. Results are all obtained on an ideal quantum simulator.](image-url)
Figure 5. Relative error $\mathcal{E}$ as a function of the nodes for 3-regular unweighted MaxCut problem with $p = 2$. DC-QAOA (QAOA) ’initialized’ shows $\mathcal{E}$ values where the optimal DC-QAOA (QAOA) parameters obtained with LSTM for a 10-qubit system were used as the initial parameters for higher system DC-QAOA (QAOA) whereas 'random' shows random initialization. Results are the average over 10 random instances of graphs and 100 local optimizer steps with error bars showing standard error.

1. Parameter concentration is an effect where for certain types of problems, fixing the parameter values leads to different instances with almost the same objective function values [80]. In other words, the optimal parameter values for such instances are similar. Based on the analysis of the parameter concentration for QAOA in MaxCut [80] and the SK model [81], it has been observed that meta-learning of RNNs exhibits similar behavior [43], which aids in alleviating barren plateaus [82]. These concentrations can serve as a good initialization strategy, as the optimal parameters of small system sizes can be utilized as initial parameters for larger complex systems. Inspired by this, we further examine the parameter concentration effect in DC-QAOA for 3-regular unweighted MaxCut problems, to confirm that this effect persists after adding the CD-term and offers an advantage over QAOA. In detail, the dependence of $\mathcal{E}$ on the nodes of the graph for 3-regular unweighted MaxCut problems with $p = 2$ layers is presented in figure 5. The 10-qubit initialized graphs are compared to randomly initialized graphs, where the optimal parameters for 10-node systems obtained with LSTM are used as initial parameters for 14, 16, and 18-node systems. $\mathcal{E}$ represents the mean of 10 graph instances with 100 steps of the local optimizer. For both QAOA and DC-QAOA, we can observe the parameter concentration effect, where the $\mathcal{E}$ values for the 10 qubit-initialized graphs have relatively lower values compared to randomly initialized graphs. As expected, the $\mathcal{E}$ values for DC-QAOA are the lowest, implying that the concentrations strategy combined with DC-QAOA has advantages over QAOA. Furthermore, the $\mathcal{E}$ values start to saturate with increasing the system size. This can be attributed to the fact that for higher system sizes, the expressibility for QAOA and DC-QAOA decreases significantly, and thus the ansatz might not be expressive to explore a larger solution space. A similar conclusion was made in [83], where the authors showed the parameter concentration effect and the approximation ratio for different $p$.

However, this does not contradict the observations made in QA where time-to-solution scales roughly exponentially with system size for NP-complete problems, since in QA, the goal is to find the global solution rather than an approximate solution. Performance might improve if optimal parameters from a 12 or 14 nodes system are considered instead of those from 10 node system. In conclusion, the parameter concentration effect is also present in DC-QAOA and serves as a good initialization strategy in the sense that LSTM or GRU can be implemented for a smaller system, and the optimal DC-QAOA parameters can be used for initializing larger systems.
3.2. The SK model

The second benchmark for our method is the SK model, a classical spin model with couplings that possess all-to-all connectivity [84, 85]. Investigating this model is important because lots of optimization problems can be encoded into spin glass systems [86]. The Hamiltonian of the SK model is defined as follows:

$$H_p = -\sum_{i,j} J_{ij} \sigma_i^z \sigma_j^z,$$

where $J_{ij}$ are coupling coefficients associated to spin $i$ and spin $j$. We examine a 10 qubit SK model with $J_{ij} \in \{-1, 1\}$ with mean 0 and variance 1. The CD unitary that works best for this case is again $U_D(\alpha) = e^{-i\alpha \sum_i \sigma_i^z \sigma_i^y}$, see appendix B. For RNNs training, we select 200 graphs with all-to-all connectivity and randomly generate the coupling coefficients $J_{ij} \in \{-1, 1\}$. As with the MaxCut problem, we limit the maximum epochs for training the RNN to $M = 10$, with $\tau = 0.01$, and time horizon $T = 6$. We use an Adam optimizer with step size 0.01 to train the network and an Adagrad optimizer with a 0.1 learning rate for local optimization.

Figure 6 displays the mean $\mathcal{E}$ of 10 randomly generated graph instances with respect to the number of steps for different initializations of QAOA and DC-QAOA parameters for $p = 2$. We observe that the LSTM and GRU initialization of optimizations for DC-QAOA problems results in the fastest convergence and lowest final relative error. Notably, as in the MaxCut case, GRU-QAOA exhibits faster convergence than LSTM-QAOA, suggesting better performance than the former in the QAOA case. Once again, we find that DC-QAOA outperforms QAOA for the same initialization. Similarly to the MaxCut case, the optimization of the network itself approaches values quite close to the final relative error.

4. Discussion and conclusion

We have proposed a meta-learning technique for initializing DC-QAOA by employing RNNs, specifically LSTMs, and GRUs, to design a black-box classical optimization routine. The routine learns from a set of training graphs and outputs optimal initial DC-QAOA parameters. We have benchmarked this approach on unweighted 3-regular MaxCut, weighted complete MaxCut, and the SK model. In all cases, the RNN initialization combined with DC-QAOA outperforms QAOA by reaching lower relative errors $\mathcal{E}$ in fewer iterations. Moreover, we have also investigated the parameter concentration effect on DC-QAOA, which serves as a good alternative initialization strategy.

Our proposed method has addressed two major challenges of the near-term era: achieving higher success and finding optimal tunable parameters to reduce optimization overload. We have shown that GRUs can perform as well as LSTMs with a lower number of cell operations, which suggests that GRUs might be a better choice of RNNs for this kind of meta-learning technique.

For future perspectives, this method can be extended to potential industrial problems, such as portfolio optimization and logistical networks. Additionally, exploring the performance of higher circuit layers and qubit systems could be interesting. In conclusion, digitized-CD quantum computing is a promising paradigm that has already shown improvement over current adiabatic optimization methods in many
studies [20–22, 30, 34, 79]. Still, there is plenty of room for improvement in these algorithms, and implementing sophisticated machine learning techniques might be key in reaching closer to quantum advantage in the current noisy intermediate-scale quantum era.

Data availability statement

The data cannot be made publicly available upon publication because they are not available in a format that is sufficiently accessible or reusable by other researchers. The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgment

This work was financially supported by EU FET Open Grant EPIQUIS (899368), the Basque Government through Grant No. IT1470-22, the Project Grant PID2021-126273NB-I00 funded by MCIN/AEI/10.13039/501100011033 and by ‘ERDF A way of making Europe’ and ‘ERDF Invest in your Future’, NSFC (12075145), STCSM (Grants No. 2019SHZDZX01-ZX04), by the Basque Government through the ELKARTEK Program under Grant KK-2022/00041, ‘BRTA QUANTUM: Hacia una especialización armonizada en tecnologías cuánticas en BRTA’. X C acknowledges ayudas para contratos Ramón y Cajal–2015-2020 (RYC-2017-22482).

Appendix A. Information flow in LSTM and GRU

The standard equations that describe the information flow of one layer of the LSTM network are shown below:

\[ f_t = \sigma (W_f \theta_t + R_f h_{t-1} + b_f), \]  
\[ i_t = \sigma (W_i \theta_t + R_i h_{t-1} + b_i), \]  
\[ \tilde{c}_t = \tanh (W_c \theta_t + R_c h_{t-1} + b_c), \]  
\[ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \]  
\[ o_t = \sigma (W_o \theta_t + R_o h_{t-1} + b_o), \]  
\[ h_t = o_t \odot \tanh (c_t). \] (A1f)

Here, \( \theta_t \) is the input vector on each time step, and \( h_{t-1} \) is the previous hidden state. \( W \) and \( R \) are weights to the input and hidden state, and \( b \) are the biases vectors. These parameters are learned during the training of the network by minimizing some loss function \( \mathcal{L}(\theta) \). The functions \( \sigma \) and \( \tanh \) are the sigmoid and hyperbolic tangent functions, respectively, and their ranges are \([0, 1]\) and \([-1, 1]\). They are useful for keeping or discarding information and regulating the network. The \( \odot \) symbol denotes the Hadamard product.

Equation (A1) show how the information flows throughout the network. The forget gate \( f_t \) decides how much information from the previous cell state \( c_{t-1} \) is kept at time \( t \). Similarly, the input gate \( i_t \) multiplies the update gate \( \tilde{c}_t \) in order to regulate the information given by \( \theta_t \) and \( h_{t-1} \). Finally, the output gate \( o_t \) decides the information kept in the hidden state for the next time step \( h_t \). For the case of the GRU cell, the equations have different forms, from which we implement the following:

\[ z_t = \sigma (W_z \theta_t + R_z h_{t-1} + b_z + d_z), \]  
\[ r_t = \sigma (W_r \theta_t + R_r h_{t-1} + b_r + d_r), \]  
\[ \tilde{h}_t = \tanh (W_h \theta_t + r_t \odot (R_h h_{t-1} + d_h) + b_h), \]  
\[ h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t. \] (A2d)
As can be seen, the gates will be different now. First of all, another bias vector $d$ is introduced that will help to modulate the calculation of $h_t$. Update gate $z_t$ will select information from the previous hidden state. Similarly, the reset gate $r_t$ will select which information to discard from previous hidden states. $h_t$ is known as a candidate state gate and it is used to generate a candidate for a new hidden state. It can be seen how the reset gate helps to forget unimportant information from previous iterations. Finally, to compute the new hidden state $h_t$, the update gate will be considered, and will model how much weight is given to the candidate state $h_{t-1}$ and to the previous hidden state $h_{t-1}$.

### Appendix B. Comparison of different CD terms

In this study, we obtain the CD terms using the NC method, as shown in equation (3). Instead of RL [32] or variational circuit learning [33], we use numerical calculation to select the most efficient CD term with two-body interaction. As an example, we consider an 8-qubit system with $p = 1$ and compare all the two-local CD terms in figure 7. The results show that $A_{1}^{(2)} = \sigma^z\sigma^x$ performs exceptionally well for all the models. Therefore, we implement the same approach for our numerical analysis of higher qubit systems in MaxCut and the SK model.

### Appendix C. Classical optimizers

Gradient descent techniques are used to optimized both RNN, QAOA and DC-QAOA parameters. The basic idea is to take steps in the opposite direction of the gradient, as it is the direction of the steepest descent. The basic functioning of gradient descent is as follows:

Let us consider an objective function $J(\theta)$, which is parameterized by some variables $\theta \in \mathbb{R}^d$ that need to be minimized. Here, $\nabla_\theta J(\theta)$ denotes the gradient of the objective function with respect to the parameters, and $\eta$ is a learning rate parameter that controls the updating of variational parameters at each step. The core of a gradient descent algorithm is to perform on each step of the operation:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta),$$

where in the case of DC-QAOA, the function $J(\theta)$ is the problem Hamiltonian, similar to equation (1). Having that said, let us see how we can describe Adagrad and Adam algorithms based on equation (C1):

#### C.1. Adagrad optimizer

Let us start by defining $g_{t,i}$ to be the gradient of the objective function with respect to the parameter $\theta_i$ at time step $t$:

$$g_{t,i} = \nabla_{\theta_i} J(\theta_{t,i}).$$

Also, let $\eta$ be the learning rate, $\epsilon$ a small positive number of order $10^{-9}$ and $G_t \in \mathbb{R}^{d \times d}$ a diagonal matrix such that the element $i,i$ is the sum of the squares of the gradient with respect to $\theta_i$ up to step $t$. We will have then that on each step, the parameter $\theta_i$ will be updated as:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,i,i} + \epsilon}} \cdot g_{t,i}.$$  

We see from the last equation how introducing the term $G_{t,i,i} + \epsilon$ in the denominator is useful. This modification helps to adjust the learning rate as the algorithm progresses, allowing for larger changes in the early stages and smaller changes in later stages as $G_{t,i,i}$ becomes larger. A common value for $\epsilon$ is $10^{-8}$.

#### C.2. Adam optimizer

For this optimizer, for all variational parameters $\theta_i$ we define one vector $m$ and one scalar $v$, which work as estimators of the first and second moment (mean and variance), respectively. These are computed as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t \quad (C4a)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2 \quad (C4b)$$

with $\beta_1, \beta_2$ being positive real numbers close to 1. The moments are corrected on each step so that:

$$\tilde{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (C5a)$$
Figure 7. Comparison of performance of different CD terms. Mean $\mathcal{E}$ for five graphs vs. the number of local optimizer steps is plotted for an 8 qubit system and $p = 1$ layer. A training sample of 100 graphs was used for training the RNNs.

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t},$$  \hspace{1cm} \text{(C5b)}$$

where $\beta_1^t, \beta_2^t$ mean $\beta_1, \beta_2$ to the power of $t$. So we are able now to provide the parameter update formula in Adam algorithm as:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t,$$  \hspace{1cm} \text{(C6)}$$
we have shown a summary of the different optimizers used on each case for this work (the selection of the optimizer for each case was made heuristically).

In table 1 we have shown a summary of the different optimizers used on each case for this work (the selection of the optimizer for each case was made heuristically).

**ORCID iDs**

Pranav Chandarana  [https://orcid.org/0000-0002-3890-1862](https://orcid.org/0000-0002-3890-1862)

Narendra N Hegade  [https://orcid.org/0000-0002-9673-2833](https://orcid.org/0000-0002-9673-2833)

Enrique Solano  [https://orcid.org/0000-0002-8602-1181](https://orcid.org/0000-0002-8602-1181)

Yue Ban  [https://orcid.org/0000-0003-1764-4470](https://orcid.org/0000-0003-1764-4470)

Xi Chen  [https://orcid.org/0000-0003-4221-4288](https://orcid.org/0000-0003-4221-4288)

**References**

[1] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love P J, Aspuru-Guzik A and O’Brien J I 2014 A variational eigenvalue solver on a photonic quantum processor Nat. Commun. 5 1

[2] Killoran N, Bromley T R, Arrazola J M, Shuld M, Quesada N and Lloyd S 2019 Continuous-variable quantum neural networks Phys. Rev. Res. 1 033003

[3] Wecker D, Hastings M B and Troyer M 2015 Progress towards practical quantum variational algorithms Phys. Rev. A 92 042303

[4] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 Quantum machine learning Adv. Quantum Technol. 1 1-11

[5] Zhou L, Wang S-T, Choi S, Pichler H and Lukin M D 2020 Quantum approximate optimization algorithm: performance, mechanism and implementation on near-term devices Phys. Rev. X 10 021067

[6] Bharti K et al 2022 Noisy intermediate-scale quantum algorithms Rev. Mod. Phys. 94 015004

[7] Farhi E, Goldstone J and Gutmann S 2014 A quantum approximate optimization algorithm (arXiv:1411.4028)

[8] Farhi E and Harrow A W 2016 Quantum supremacy through the quantum approximate optimization algorithm (arXiv:1602.07674)

[9] McClean J R, Boixo S, Smelyanskiy V N, Babbush R and Neven H 2018 Barren plateaus in quantum neural network training Phys. Rev. Lett. 120 050503

[10] Holmes Z, Sharma K, Cerezo M and Coles P J 2022 Connecting ansatz expressibility to gradient magnitudes and barren plateaus Phys. Rev. Res. 4 032753

[11] Demirplak M and Rice S A 2003 Adiabatic population transfer with control fields J. Phys. Chem. A 107 9937

[12] Demirplak M and Rice S A 2005 Assisted adiabatic passage revisited J. Phys. Chem. B 109 6638

[13] Berry M V 2009 Transitionless quantum driving J. Phys. A: Math. Theor. 42 165303

[14] del Campo A 2013 Shortcuts to adiabaticity by counterdiabatic driving Adv. At. Mol. Opt. Phys. 62 117

[15] Sels D and Polkovnikov A 2017 Minimizing irreversible losses in quantum systems by local counterdiabatic driving Proc. Natl Acad. Sci. 114 E3909

[16] Guéry-Odelin D, Ruscilla A, Torrontegui E, Martínez-Garaot S and Muga J G 2019 Shortcuts to adiabaticity: concepts, methods and applications Rev. Mod. Phys. 91 045001

[17] Torrontegui E, Ibáñez S, Martínez-Garaot S, Modugno M, del Campo A, Guéry-Odelin D, Ruscilla A, Chen X and Muga J G 2013 Shortcuts to adiabaticity Adv. At. Mol. Op. Phys. 62 117

[18] Sels D and Polkovnikov A 2019 Floquet-engineering counterdiabatic protocols in quantum many-body systems Phys. Rev. Lett. 123 090602

[19] Hegade N N, Paul K, Ding Y, Sanz M, Albarrán-Arriagada F, Solano E and Chen X 2021 Shortcuts to adiabaticity in digitized adiabatic quantum computing Phys. Rev. Appl. 15 042038

[20] Hegade N N, Paul K, Albarrán-Arriagada F, Chen X and Solano E 2021 Digitized adiabatic quantum factorization Phys. Rev. A 104 L050403

[21] Hegade N, Chandarana R, Paul K, Chen X, Albarrán-Arriagada F and Solano E 2021 Portfolio optimization with digitized-counterdiabatic quantum algorithms (arXiv:211208347)

[22] Opatrn T and Mølmer K 2014 Partial suppression of nonadiabatic transitions New J. Phys. 16 015025

[23] Petzold R, Dube B, Carretta S, Mannella R, Mintsert F and Wimberger S 2019 Accelerating quantum-adiabatic protocols for entangling two qubits in circuit QED Phys. Rev. A 99 042315

[24] Zhou H, Ji Y, Nie X, Yang X, Chen X, Bian J and Peng X 2020 Experimental realization of shortcuts to adiabaticity in a nonintegrable spin chain by local counterdiabatic driving Phys. Rev. Appl. 13 044059

[25] Ji Y, Bian J, Chen X, Li J, Nie X, Zhou H and Peng X 2019 Experimental preparation of Greenberger-Horne-Zeilinger states in an Ising spin model by partially suppressing the nonadiabatic transitions Phys. Rev. A 99 032323

[26] Passarelli G, Cataridda V, Fazio R and Lucignano P 2020 Counterdiabatic driving in the quantum annealing of the p-spin model: a variational approach Phys. Rev. Res. 2 013283

[27] Takahashi K 2017 Shortcuts to adiabaticity for quantum annealing Phys. Rev. A 95 012309
[70] Cho K, Van Merriënboer B, Bahdanau D and Bengio Y 2014 On the properties of neural machine translation: encoder-decoder approaches (arXiv:1409.1259)

[71] Lechner M and Hasani R 2020 Learning long-term dependencies in irregularly-sampled time series (arXiv:2006.04413)

[72] Noh S-H 2021 Analysis of gradient vanishing of RNNs and performance comparison (arXiv:2009.06459)

[73] Rumelhart D E, Hinton G E and Williams R J 1986 Learning representations by back-propagating errors Nature 323 533

[74] Karp R M 1972 Reducibility among combinatorial problems Complex. Comput. Comput. 85 748–56

[75] Guerreschi G G and Matsuura A Y 2019 QAOA for Max-Cut requires hundreds of qubits for quantum speed-up (arXiv:1903.04612)

[76] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)

[77] Duch J, Hazan E and Singer Y 2011 Adaptive subgradient methods for online learning and stochastic optimization (arXiv:1107.4287)

[78] Shaydulin R, Lotshaw P C, Larson J, Ostrowski J and Humble T S 2022 Parameter transfer for quantum approximate optimization of weighted MaxCut (arXiv:2201.11785)

[79] Chandarana P, Hegade N N, Montalban I, Solano E and Chen X 2022 Digitized-counterdiabatic quantum algorithm for protein folding (arXiv:2212.13511)

[80] Brandao F G, Broughton M, Farhi E, Gutmann S and Neven H 2018 For fixed control parameters the quantum approximate optimization algorithm’s objective function value concentrates for typical instances (arXiv:1812.04176)

[81] Akshay V, Rabinovich D, Campos E and Biamonte J 2021 Parameter concentrations in quantum approximate optimization Phys. Rev. A 104 L010401

[82] Mele A A, Mbeng G B, Santoro G E, Collura M and Torta P 2022 Avoiding barren plateaus via transferability of smooth solutions in Hamiltonian variational ansatz (arXiv:2206.01982)

[83] Wurtz J and Lykov D 2021 The fixed angle conjecture for QAOA on regular MaxCut graphs (arXiv:2107.00677)

[84] Binder K and Young A P 1986 Spin glasses: experimental facts, theoretical concepts and open questions Rev. Mod. Phys. 58 801

[85] Sherrington D and Kirkpatrick S 1975 Solvable model of a spin–glass Phys. Rev. Lett. 35 1792

[86] Lucas A 2014 Ising formulations of many NP problems Front. Phys. 2 5