Iterative-Free Quantum Approximate Optimization Algorithm Using Neural Networks

Ohad Amosy*[1], Tamuz Danzig*[2], Ely Porat[1], Gal Chechik[3,4], and Adi Makmal[2]

[1]Faculty of Computer Science, Bar-Ilan University, 52900 Ramat Gan, Israel
[2]Faculty of Engineering, Bar-Ilan University, 52900 Ramat Gan, Israel
[3]Gonda Brain research institute in Bar-Ilan University, 52900 Ramat Gan, Israel.
[4]NVIDIA Research

The quantum approximate optimization algorithm (QAOA) is a leading iterative variational quantum algorithm for heuristically solving combinatorial optimization problems. A large portion of the computational effort in QAOA is spent by the optimization steps, which require many executions of the quantum circuit. Therefore, there is active research focusing on finding better initial circuit parameters, which would reduce the number of required iterations and hence the overall execution time. While existing methods for parameter initialization have shown great success, they often offer a single set of parameters for all problem instances. We propose a practical method that uses a simple, fully connected neural network that leverages previous executions of QAOA to find better initialization parameters tailored to a new given problem instance. We benchmark state-of-the-art initialization methods for solving the MaxCut problem of Erdős–Rényi graphs using QAOA and show that our method is consistently the fastest to converge while also yielding the best final result. Furthermore, the parameters predicted by the neural network are shown to match very well with the fully optimized parameters, to the extent that no iterative steps are required, thereby effectively realizing an iterative-free QAOA scheme.

Ohad Amosy*: amosyoh@biu.ac.il
Tamuz Danzig*: tamuz.danzig@gmail.com,
* Equal contribution
Adi Makmal: adi.makmal@biu.ac.il

1 Introduction

Quantum computers are currently found in the so-called noisy intermediate-scale quantum (NISQ) era, comprising a small number of qubits and high error rates that accumulate rapidly with the number of operations [1]. A popular family of quantum algorithms in the NISQ era is the variational quantum algorithms (VQAs), which offer a heuristic approach for solving optimization problems [2]. VQAs use relatively shallow quantum circuits and are, therefore, more resilient to noise compared to standard quantum algorithms [3]. They employ quantum circuits of parameterized gates that are updated, by means of classical computation, to reach an optimum of a predefined objective function [4]. The quantum approximate optimization algorithm (QAOA) is a particular VQA designed specifically to solve approximate combinatorial optimization problems [5, 6, 7], which are of high importance for various fields in both industry and academia [8, 9].

In VQAs and QAQA in particular, each optimization step involves thousands of circuit executions, so reducing the number of optimization steps is highly desirable. Choosing a proper QAOA parameters can improve both the convergence rate [10, 11] and the accuracy of the final solution [5, 11]. Finding better initial parameters for the QAOA circuit is thus a matter of active research [7, 10, 11, 12, 13, 14, 15, 16].

Most initialization methods offer a single, common, set of parameters for all problem instances [7, 12, 10, 14]. This is a pragmatic approach and often very successful in reducing the number of required QAOA iterations. Nevertheless, it inherently ignores important relations between problem instances and their optimal parameters. These relations, which are based on the infor-

mation carried by the particular description of each problem instance, can potentially be utilized to save further iterations. Other methods that manage to personalize the initial parameters per problem instance [11, 15] show an impressive improvement, exponentially better compared to a random parameter initialization [11], yet at a high cost of many additional quantum circuit executions. These findings entail that using a different initialization per problem instance is beneficial and motivate the exploration of more cost-effective schemes for generating personalized parameters for each problem instance.

The goal of this paper is to reduce the number of required QAOA iterations without compromising algorithm performance and without invoking any additional quantum circuit executions, by using better initialization parameters.

We propose a practical approach based on neural network (NN) that predicts appropriate initialization parameters for QAOA *per problem instance*. The NN takes as input a predefined encoding of the problem instance and outputs the corresponding QAOA parameters. To that end, the NN is trained using past results of QAOA optimizations as labeled data. We evaluate our method on the standard MaxCut benchmark; The method can be similarly applied to other problems.

This study provides the following contributions: (a) it demonstrates that the QAOA's variational parameters can be learned efficiently and robustly by a simple NN for the MaxCut problem without any additional quantum computation; (b) the proposed learning scheme is empirically shown to outperform state-of-the-art QAOA initialization methods in terms of the approximation ratio and convergence speed; (c) using the proposed method may relinquish the need for any further optimization of the quantum circuit altogether, potentially saving many costly quantum circuit executions; and (d) we show that these advantages become more profound as the size of the problem (the number of nodes in the graph) increases.

The paper is structured as follows: Sec. 2 provides the relevant background. It describes the QAOA algorithm, the MaxCut problem, and previous works on parameter initialization for the QAOA circuit. Sec. 3 introduces our proposed method. Sec. 4 benchmarks several initialization techniques alongside our method and provides a detailed comparison of their performances. Finally, we discuss the central advantages of the proposed method for the NISQ era in Sec. 5.

2 Background

2.1 The quantum approximate optimization algorithm (QAOA)

The QAOA can be regarded as a time-discretization of adiabatic quantum computation [5, 7, 17], whose p-layers circuit Ansatz constructs the following state:

$$\left|\psi_p(\vec{\beta},\vec{\gamma})\right\rangle = \left[\prod_{l=1}^{p} e^{-i\beta_l H_M} e^{-i\gamma_l H_C}\right] |+\rangle^{\otimes N} \quad (1)$$

where N is the number of qubits. H_C is called the problem Hamiltonian, which is defined uniquely by the specific problem we are trying to solve (see below for the MaxCut problem example), and H_M is called the mixer Hamiltonian, provided in the same form for all problems, and given by $H_M = \sum_{n=1}^{N} \sigma_n^x$, where σ_n^j is the Pauli operator j that acts on qubit n. Finally, $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is the $+1$ eigenstate of σ^x. The p-dimensional vectors $\vec{\beta}$ and $\vec{\gamma}$ are the variational real parameters that correspond to H_M and H_C, respectively. The objective cost function is determined by the expectation value of the problem Hamiltonian

$$F_p(\vec{\beta},\vec{\gamma}) = \left\langle \psi_p(\vec{\beta},\vec{\gamma}) \right| H_C \left| \psi_p(\vec{\beta},\vec{\gamma}) \right\rangle \quad , \quad (2)$$

and we denote the optimal parameters by $(\vec{\beta}^*,\vec{\gamma}^*)$, with which the optimal solution is attained:

$$(\vec{\beta}^*,\vec{\gamma}^*) = \arg\operatorname*{opt}_{\vec{\beta},\vec{\gamma}} F_p(\vec{\beta},\vec{\gamma}). \quad (3)$$

QAOA is an iterative algorithm: it starts with an initial guess of the $(\vec{\beta},\vec{\gamma})$ parameters, then the expectation value of the problem Hamiltonian of Eq. 2 is evaluated by repeated measurements of the same circuit to reach a certain level of statistical accuracy. Once the cost function is calculated, the $(\vec{\beta},\vec{\gamma})$ parameters are updated towards the next iteration by a classical optimizer so as to optimize $F_p(\vec{\gamma},\vec{\beta})$. The overall QAOA iterative process requires many executions of the quantum circuit; first, reaching a satisfactory level of statistical error ϵ typically requires $O(\epsilon^{-2})$ shots [18].

To keep the same level of accuracy, the number of shots grows with the system size; it was recently estimated to grow exponentially with size [19]; second, the optimization process requires additional circuit executions: if the optimizer is gradient-based, then the derivative of the cost function with respect to the $(\vec{\beta},\vec{\gamma})$ parameters must also be calculated at each iteration, requiring many additional executions of the quantum circuit to evaluate the gradients, e.g., by following the so-called "parameter shift rule" [20, 21]. Alternatively, gradient-free optimizers can avoid the calculations of the derivatives, yet at a high cost of many more optimization iterations [22]. The iterative QAOA process thus requires many executions of the quantum circuit and each optimization step that can be avoided translates directly into a significant reduction in computational resources.

2.2 The MaxCut problem

The maximum-cut (MaxCut) problem is an NP-hard combinatorial problem that has become the canonical problem to benchmark QAOA [10, 23]. It is defined over an undirected graph $G=(V,E)$, where $V=1,2,...,N$ denotes the set of nodes, and E is the set of edges. The (unweighted) *maximum cut* objective is the partition of the nodes into two groups $\{+1,-1\}$, such that the number of edges connecting nodes from the two different groups is maximal.

Within QAOA, the problem Hamiltonian H_C that corresponds to the MaxCut problem is given by [5]:

$$H_C=\frac{1}{2}\sum_{(i,j)\in E}\left(1-\sigma_i^z\sigma_j^z\right) \quad , \tag{4}$$

such that an edge (i,j) contributes to the sum if and only if the (i,j) qubits are measured anti-aligned. The common performance metric of QAOA for the MaxCut problem is the approximation ratio:

$$r=\frac{F_p(\vec{\beta}^*,\vec{\gamma}^*)}{C_{max}} \tag{5}$$

where C_{max} is the maximum cut of the graph.

2.3 QAOA initialization techniques

The attempt to find optimal parameters for QAOA, which would require a minimum number of optimization steps for solving the MaxCut problem, is currently an extensive research topic.

The most intuitive initialization scheme is linear, where the $(\vec{\beta},\vec{\gamma})$ parameters vary linearly from one layer to the other, such that β is gradually turned off and γ is turned on. This linear approach is inspired by adiabatic quantum computation [17]: the circuit begins from the ground state of the mixer Hamiltonian and aims at reaching the ground state of the problem Hamiltonian. While very simple and computationally efficient, the linear solution can be sub-optimal, see e.g., [10, 23]. As an alternative approach, several initialization methods selected the same set of parameters for common type graph instances, such as regular graphs [7, 12, 10, 14, 24]. For example, it was suggested in [7] to use a batches optimization method, in which initial parameters are found by optimizing batches of graphs in parallel. This way, the parameters fit multiple graphs and should also fit new graphs. The problem, however, with such homogeneous methods is that in practice, the optimal parameters change from one graph instance to another, even within the same family of graphs [10]. Such non-personalized methods ignore by construction possible relations between specific problem instances and their corresponding optimal parameters.

Indeed, other studies have suggested using a different, personalized set of parameters for each problem instance. One method suggested initializing a $p+1$ layers circuit based on the optimized p layers circuit. This proved useful [10]: for 3-regular graphs, the number of optimization steps reduced exponentially, from $2^{O(p)}$ for random parameter initialization to $O(poly(p))$ [10]. However, it required the full optimization of the p layers circuit. Similarly, Ref. [15] used a regression model to predict the initial parameters of QAOA for a circuit with $p\geq 2$, given the optimized parameters for the corresponding single layer circuit, i.e. $p=1$. This requires the generation of an optimized dataset for both the one-layer circuit and the desired p-layers circuit; Moreover, given a new problem instance, this method requires full optimization of its single-layer circuit. Recently, it was shown that reusing the optimal parameters of small graph instances can be beneficial for solving the MaxCut problem of larger graph instances, when using single-layer QAOA circuits ($p=1$) [25]. This is due to the confined combi-

natorial number of small regular subgraphs with radius p (the connection between the number of circuit's layers and the performance on p-radius graphs was analyzed in [5] and is also addressed here, in Sec. 4.4). However, for $p > 1$ the combinatorial number of subgraphs increases rapidly and the method becomes less practical.

The Trotterized quantum annealing (TQA) is an initialization protocol that personalizes the initial parameters for a new problem instance by running the quantum circuit on many different linear initializations and choosing the initial parameters that achieve the best result [11]. Another approach proposed in [24] is a multi-start method in which the objective function is evaluated in different places in the parameters domain by using gradient-free optimizers. They show that it is possible to reach good minima with finite number of circuit evaluations. These methods perform well but at a high computational cost per problem instance, as they optimize the initial parameters for each new problem instance. In particular, they do not leverage the knowledge obtained from previous QAOA optimizations.

Several methods used machine learning techniques to improve the performance of QAOA. For example, several studies have attempted to facilitate the optimization step by replacing the standard classical optimizers with reinforcement learning methods and meta-learning algorithms [13, 26, 27, 28, 29, 30, 31]. Egger et al. proposed the warm-start QAOA algorithm, where the qubits are initialized in an approximated solution instead of the uniform superposition as typically done in QAOA [32]. Jain et al. used a graph neural network (GNN) for predicting the probability of each node being on either side of the cut in order to initialize warm-start QAOA [33]. Unlike these methods, we use NN only to predict the initial parameters. Since we do not alter the optimization process, our method can be applied together with any of the other methods.

3 The Method

Given a combinatorial optimization problem, we predict the best initialization parameters for QAOA for a new particular problem instance, as illustrated in Fig. 1. We use a simple, fully connected neural network (NN) that takes as an input an encoding of the problem instance and predicts an initialization parameters for that specific instance. The encoding of the problem instances may vary from one combinatorial problem to another. In our case of solving the MaxCut problem we encode each graph instance by its adjacency matrix, which is then fed to the network, as described below.

The method is based on the assumption that the QAOA was already applied on n different problem instances $S = \{s_i\}_{i=1}^n$, and that both the set of instances and the set of the corresponding final parameters $P = \{\vec{\beta}_i, \vec{\gamma}_i\}_{i=1}^n$ were saved and can be used as labeled data for the NN training. This allows us to exploit past QAOA calculations for predicting the initial parameters $(\vec{\beta}_{new}, \vec{\gamma}_{new})$ for a new problem instance.

Formally, we train a neural network f_θ to map each problem instance from S to its optimal circuit parameters from P. The input of the NN encodes the problem instance and the NN output is a vector of size $2p$, where p is the number of layers in the quantum circuit: p parameters for $\vec{\beta}$ and p parameters for $\vec{\gamma}$. The neural network parameters θ are trained to minimize the $L2$ norm

$$\theta = \arg\min_\theta \sum_{i=1}^{n} ||f_\theta(s_i) - (\vec{\beta}_i, \vec{\gamma}_i)||_2. \qquad (6)$$

Finally, once the neural network is trained, predicting the initial parameters for a QAOA calculation of a new problem instance is done by setting $(\vec{\beta}_{new}, \vec{\gamma}_{new}) = f_\theta(s_{new})$.

In comparison to previous methods, which either offer a fixed set of parameters for all problem instances or personalize the parameters per problem instance, but at a high cost of extra quantum circuit execution, our scheme manages, by design, to personalize the parameters per problem instance with no such extra cost. This is an *a-priori* advantage. In the next section the empirical performance of the methods is examined.

3.1 Applying to the MaxCut problem

We exemplify our method on the MaxCut problem. We encode each problem instance, i.e., a graph, by its adjacency matrix. The matrix is reshaped into a one-dimensional vector of length $\frac{N(N-1)}{2}$, which is given as an input to the NN.

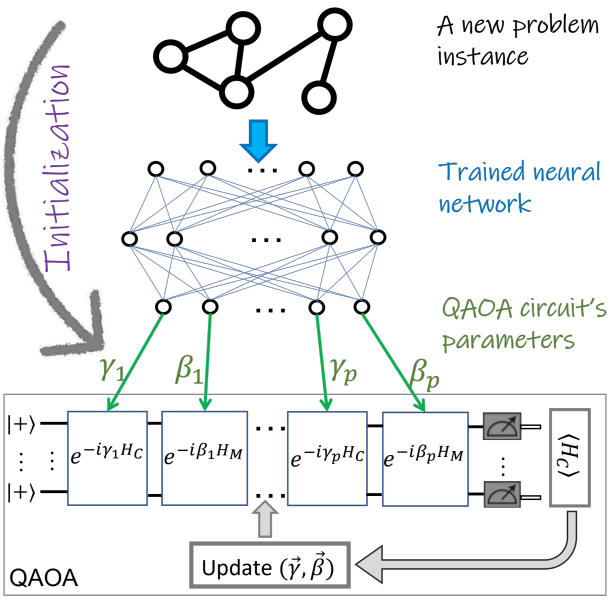


Figure 1: An Illustration of the proposed method: each new problem instance (e.g., a graph in the MaxCut problem) is encoded (e.g., by its adjacency matrix) and inserted as an input into a trained neural network. The latter predicts better initial parameters for QAOA, specifically tailored to the given problem instance (the graph).

4 Results

4.1 Setup, benchmarking, and implementation details

Setup

We test the proposed method by solving the MaxCut problem for Erdős–Rényi graphs of $N = 6 - 16$ nodes. In an Erdős–Rényi graph, each edge (i, j) exists with a certain probability, independently from all other edges. We consider two different graph ensembles: (a) **constant Erdős–Rényi graphs** - where the existing probability for each edge is constant and equals $p = 0.5$, for all graphs; (b) **random Erdős–Rényi graphs** - where the edge probability changes between graphs but remains the same for all edges within a graph: for each graph, we uniformly sample a probability in the range $p \in \{0.3, 0.9\}$ and assign it to all the edges. While the first, constant probability setup is the most commonly considered in the literature, see e.g., [11], the latter setup with a random edge probability, better represents natural, real-world scenarios, where the probability for creating a connection between nodes depends on variables that may vary across graphs. For example, if the nodes of the graph represent people and a connection indicates an interaction between two people, then the probability of the interaction usually varies depending on external variables, such as place, age, culture, etc., and varies between different communities.

Benchmarking

To evaluate the proposed initialization method, we compare its performance to those of four initialization methods: (a) the batches optimization method [7]; (b) the Trotterized quantum annealing (TQA) initialization procedure [11] with a predefined Δt; (c) a simple linear method; and (d) an average method. The first two methods [7, 11] are state-of-the-art methods, described in Sec. 2.3, whereas the latter two are simpler yet natural baselines. We chose to focus on the batches and the TQA optimization methods because they are the best initialization methods known to date that, given a new test graph, do not require any additional computational effort, as is also the case in our proposal.

To enable the comparison, we implemented all four benchmark methods, as follows. In the batches optimization method, the QAOA initial parameters are determined by finding a fixed, optimized set of parameters for a large training set of graphs that are optimized in parallel. This set of optimized parameters is then used as the initialization point for new test graphs. We implemented the batches optimization method, where we used 200 training graphs for each setup choice, i.e., {graph ensemble (random or constant ER), number of qubits, and number of layers}. This is the same training set size used in Ref. [7], which we explicitly verified is sufficient, in the sense that optimizing over a larger set did not yield parameters that performed better on the test set.

The TQA initialization procedure takes the following linear form:

$$\beta_l = \left(1 - \frac{l}{p}\right)\Delta t, \quad \gamma_l = \frac{l}{p}\Delta t, \tag{7}$$

where $l = 1...p$ indicates the layer's number in the quantum circuit. In the complete TQA protocol, the hyperparameter Δt is determined by performing a simple grid search that optimizes the overall performance for each graph. This requires extra quantum circuit executions for each new graph instance, which we wanted to avoid to enable a fair comparison. It was numerically observed in

Ref. [11] that for graph ensembles of similar nature (e.g., regular unweighted graphs, weighted regular graphs, and constant ER graphs) the optimal value of Δt does not vary by much. Therefore, in our experiments, for each kind of ensemble (constant and random ER), number of nodes n, and number of layers p, we performed the complete TQA protocol over 50 training graphs and averaged their results to find a single Δt^* that optimizes the overall performance. We then used this optimized Δt^* to evaluate the performance of TQA on new graphs. See Sec. A in the appendix for more details.

We also implemented the simpler baselines: (a) a linear initialization in which β decreases linearly from $\frac{\pi}{4}(1-\frac{1}{p})$ to $\frac{\pi}{4p}$ and γ increases linearly from $\frac{\pi}{p}$ to $\pi(1-\frac{1}{p})$. This schedule is similar to the TQA one but simpler, as it employs no further hyper-parameters. The reason for this particular choice is the periodicity of $\beta_i \in [0, \frac{\pi}{2}]$ and $\gamma_i \in [0, 2\pi]$ when plugged into Eq. 2. Note that a similar choice was made in [10, 34]; (b) an average initialization, which takes the optimal parameters of 100 training graphs and averages them. This initialization method has not yet been addressed in the literature. Yet, it is natural to consider it once we have independent sets of optimized $(\vec{\beta}, \vec{\gamma})$ parameters, as we do in this work. Moreover, our numerical results indicate that, given its simplicity and performance, this is a relatively good choice. We further tried a random initialization scheme, but it performed poorly, starting at a low approximation ratio and reaching suboptimal local minima after optimization. Thus, it is not shown here.

Implementation Details
We built a dataset of $5,000$ graphs for each setup combination, namely the size of the graph, number of layers, and sampling technique, as described above. In particular, we considered Erdős–Rényi (ER) graphs of sizes 6 to 16 nodes, whose edges are sampled randomly at a probability of 0.5 or using a random uniform probability in the range $[0.3, 0.9]$. Then, we optimized each graph's MaxCut solution by the QAOA algorithm, using the BFGS classical optimizer [35, 36, 37, 38], where we used the full TQA algorithm to initialize our $(\vec{\beta}, \vec{\gamma})$ parameters. Using this procedure, we built a labeled training set of {graph, optimized $(\vec{\beta}, \vec{\gamma})$ parameters} pairs, with which we trained the neural network. All our QAOA circuit executions were performed on Qiskit's noiseless statevector simulator [39].

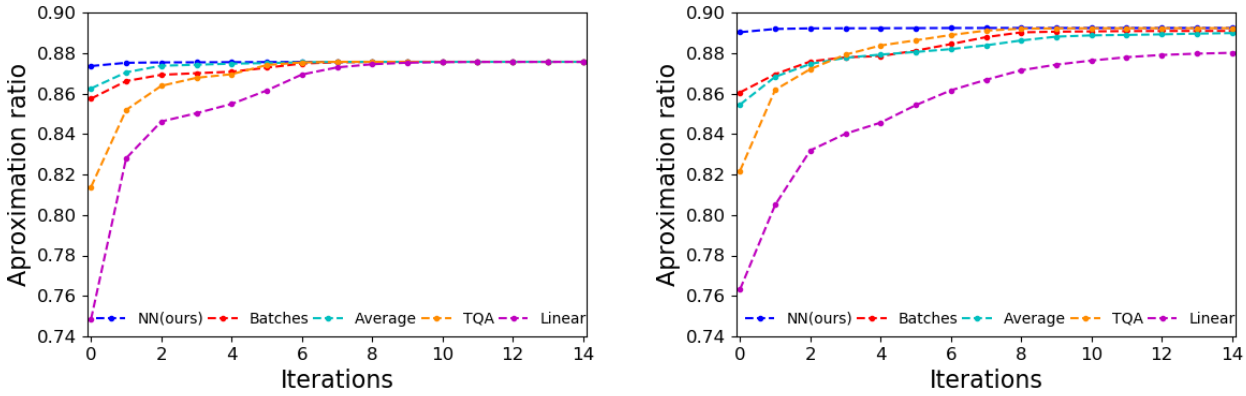
For all setups, we used the same simple 3-layer network architecture: the input layer is composed of $\frac{N(N-1)}{2}$ neurons, so as to encode the adjacency matrix of an undirected graph of N nodes, the hidden layer has 100 neurons, and the output layer has $2p$ neurons that encode the learned $(\vec{\beta}, \vec{\gamma})$ parameters of the optimized QAOA circuit.

4.2 Constant Erdős–Rényi

We begin with ER graphs with $N = 14$ nodes generated with a constant edge probability of 0.5. Fig. 2a compares the performance of the proposed NN method with those of: (a) the batches optimization method [7]; (b) the TQA initialization procedure [11]; (c) the linear method, and (d) the average method, as described above. Each point in the graph indicates the averaged approximation ratio achieved by a 2-layer QAOA circuit over 50 test graphs as a function of the QAOA optimization step iterations. The standard error of the mean is less than 0.47% and is not displayed in the figure for visual clarity.

It is seen that prior to the QAOA optimization steps, at the 0'th iteration, each method reaches a different approximation ratio and that all methods improve from one iteration to another until converging to the same approximation ratio. Yet, each method converges at a different rate. Fig. 2a demonstrates that, compared to all other methods, the NN method begins at a better approximation ratio and converges the fastest. This means that the NN learns to predict adequate $(\vec{\beta}, \vec{\gamma})$ parameters, given the adjacency matrix of a test graph. In fact, it can be observed that the initial approximation ratio performance of our NN method is so close to the final parameters that hardly any optimization step is required.

It is also observed that the simple average method is the second best approach for the constant ER ensemble with an edge probability of $p = 0.5$. This implies that the distribution over the $(\vec{\beta}, \vec{\gamma})$ parameters is rather confined. What would be the effect of a more widespread ensemble distribution on our results? To check that, we next examine the performance of the benchmark initialization methods, alongside the NN one, on a random ER ensemble.



(a) Constant ER with edge probability $p = 0.5$

(b) Random ER with edge probability $p \in [0.3, 0.9]$

Figure 2: The approximation ratio during QAOA optimization, starting from initial parameters obtained by the different methods, averaged over 50 graphs of $N = 14$ nodes. The QAOA circuit is composed of two layers. All graphs are sampled using ER: (a) with constant edge probability; (b) with random edge probability. The proposed NN method converges the fastest in both cases. A more significant improvement is observed in the case of graphs with random edge probability.

4.3 Random Erdős–Rényi

So far we tested our method in the standard setup: graphs are randomly drawn from ER with an edge probability of 0.5. We now expand this setup by assigning each graph with an edge probability that is sampled uniformly from the interval [0.3,0.9]. This way, we increase the diversity between the graphs, making it more realistic and challenging to guess good initial parameters [10, 11, 25].

Fig. 2b shows the performance of all our benchmark methods in the case of such a random-ER graph ensemble. The results resemble those of the constant-ER ensemble, with similar trends. In particular, the NN method begins from the highest approximation ratio and converges faster than all other methods. Yet, in the random-ER case, the differences between the methods are much more pronounced. First, it is seen that, in contrast to the constant-ER case, not all methods converge to the same approximation ratio, with the simple linear method reaching the lowest approximation ratio. Moreover, all methods, except for the NN one, exhibit a much slower convergence rate than the constant-ER scenario. The performance of the NN method is manifested in the achievable approximation ratio at the zero'th iteration, showing a significant approximation ratio gap of roughly 3% from the second-best result obtained by the batches optimization method.

This is not surprising: the NN method is designed to predict the best $(\vec{\beta}, \vec{\gamma})$ parameters per graph instance, i.e., to personalize its performance. This is in contrast to all the benchmark methods, which essentially suggest a fixed set of $(\vec{\beta}, \vec{\gamma})$ parameters for all the test graphs, independent of the particular graph's structure. It should be noted that, in principle, the TQA has the capacity of personalizing over a particular graph, by searching for the optimal dt that maximizes its results. However, as explained in Sec. 4.1, this search of Δt for each graph comes at a very high computational cost per graph, which we aim to avoid.

It is seen in Fig. 2b that also in the more challenging case of random-ER graphs, the NN method requires merely a single iteration to converge. In contrast, the TQA, which converges the fastest out of all other methods, requires about 8-12 iterations (depending on accuracy) to reach the same level of approximation ratio. We demonstrate the computational saving of our method by estimating the quantum circuit executions in our experiments. In our case, using the BFGS optimizer, we observed that even in the small $p = 2$ circuit depth, each intermediate iteration requires, on average, 12 estimations of the cost function $F_p(\vec{\beta}, \vec{\gamma})$, each with different $(\vec{\beta}, \vec{\gamma})$ parameters, so as to evaluate the relevant gradients. As each cost function estimation typically requires thousands of shots to get a meaningful statistical accuracy, we get an overall saving of hundreds of thousands of quantum circuit exe-

cutions. This saving is non-negligible and is expected to grow linearly with the circuit's depth.

4.4 Varying the size of the graph

In previous sections we examined the performance of the proposed NN method on graphs with a fixed number of $N = 14$ nodes. Next, we examine the method on random-ER graphs with varying number of nodes and draw our attention to the first QAOA evaluation, without applying any optimization steps. To that end, we generated 200 different graphs for each graph size, from $N = 6$ to 16. Then, for each graph, we obtained the initial parameters using all the different methods, and ran a 2-layer QAOA circuit using those parameters (zeroth iteration only). Fig. 3 shows the average approximation ratio achieved by each method as a function of graph size.

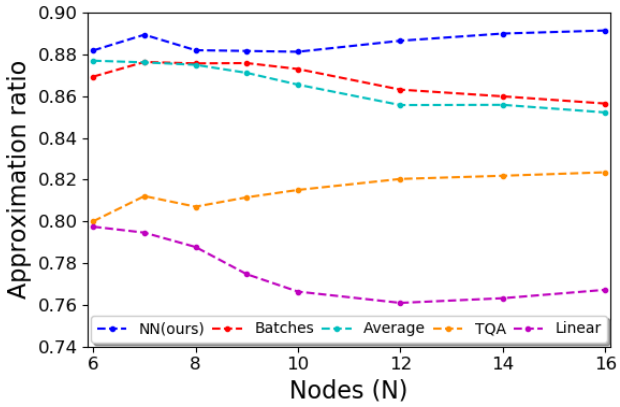


Figure 3: The approximation ratio attained by each initialization method shown as a function of the number of nodes in the graph. Graphs were sampled using ER with a random edge probability for each graph size as in Fig. 2b. The approximation ratios were calculated at the zeroth iteration, i.e., no optimization steps were taken. The number of layers is $p = 2$. It is observed that the relative performance of our proposed method increases with graph size.

It is seen that the NN-method outperforms all other methods, irrespective of the graphs' size. Moreover, the preference of the NN-method over the other methods increases with the number of nodes, especially with respect to the batches and the average optimization methods. In comparison to the TQA method, the NN shows a rather fixed, large performance gap of roughly 8% in approximation ratio in favor of the NN method.

These results indicate that as we increase the graph size N for a fixed number of layers p, the advantage of our method becomes more significant. To understand this trend, we go back to the structure of the QAOA algorithm and consider what happens when the QAOA circuit is kept fixed while the number of nodes increases, in approximating the MaxCut problem. We recall that QAOA is a p-local algorithm: in the case of the MaxCut problem, the algorithm's objective function, see Eq. 2, is a sum of the expectation value terms of all edges in the graph (i.e., sum of $\left\langle \psi_p(\vec{\gamma},\vec{\beta}) \right| \sigma_i^z \sigma_j^z \left| \psi_p(\vec{\gamma},\vec{\beta}) \right\rangle$ terms); following the commutation relation of the involved operators reveals that each expectation value term is influenced only by nodes that are at most p edges away from the calculated edge term, i.e. all subgraphs with radius of at most p [5, 40]. Zhou et al. showed that for certain families of graphs, such as the 3-regular graphs, where the possible number of subgraphs with confined radius p is finite, the spread of the optimal parameters vanishes in the limit $N \to \infty$ [10]. This may justify a non-personalized approach which initializes a single set of $(\vec{\beta},\vec{\gamma})$ parameters for all graph instances. In contrast, in random ER graphs, the number of confined subgraphs is not finite, but rather grows with N. Accordingly, the set of optimal parameters is not expected to converge to a single set as N grows. This explains the decline of the approximation ratio as N grows for methods that try to initialize with the same parameters for all graphs and emphasises the power of a personalized approach as proposed here. Next, we examine further the effect of personalizing the initial parameters.

4.5 Personalization

The key objective of our method is to create personalized initial parameters per problem instance. Fig. 4 depicts the $(\vec{\beta},\vec{\gamma})$ values as a function of the layers in a 4-layers circuit for three individual graphs (graph-1, graph-2, graph-3) with $N = 12$ nodes, sampled from the random ER ensemble. The exact optimized parameters are marked in dotted-blue lines of different shades and different markers: graph-1 is marked in dark blue circles, graph-2 in blue squares, and graph-3 is shown in light blue triangles. The NN parameters are shown in solid lines with corresponding shades and markers. The simple average method, the batches-method, and the TQA, all produce a single set of parameters, depicted in red, brown, and

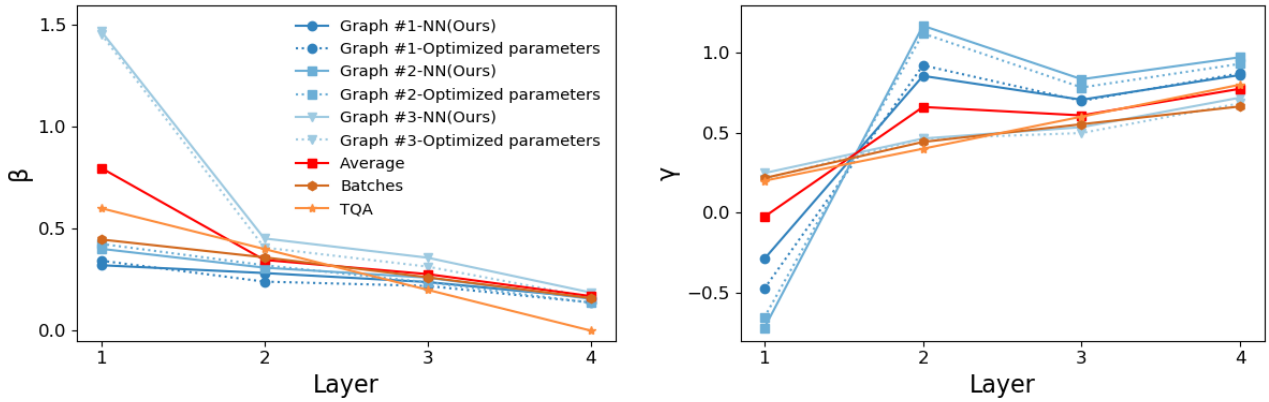


Figure 4: Initial β (left figure) and γ (right figure) parameters as a function of layers in the QAOA circuit, for three different individual graphs. A comparison is made between the prediction of the NN (solid blue lines), the optimized parameters (dashed blue lines), and the other non-personalized baselines: the simple average method (solid red line with square marks), the Batches method (solid brown line with circle marks), and the TQA (solid orange with star marks). The NN predictions are shown to follow closely the optimized curves for all three graphs.

orange, respectively. It is evident that the $(\vec{\beta}, \vec{\gamma})$ parameters obtained by our NN approach closely resemble the optimal parameters **per graph**, thus performing the personalization successfully. In contrast, other approaches, that do not have the flexibility to personalize the initial parameters, predict parameters that are approximately linear and are further away from the optimal parameters of the individual graphs.

5 Discussion and Outlook

This work joins an existing effort to find proper initialization for the QAOA circuit's variational parameters. We showed that a simple neural network could be trained to predict initial $(\vec{\beta}, \vec{\gamma})$ parameters for QAOA circuits that solve the MaxCut problem, per graph instance. Moreover, we showed that these predicted parameters match very well with the optimal parameters, to which the QAOA iterative scheme eventually converges. This enables an effectively iterative-free approach, where the QAOA circuit is executed only once, with the predicted parameters, thus saving significant amount of computational resources. We demonstrated that our approach requires up to 85% fewer iterations compared to current state-of-the-art initialization methods to reach optimized results for solving the MaxCut problem on both constant and random Erdős–Rényi graphs.

Our method assumes the availability of previous QAOA optimizations for different instances of the same problem. The neural network we employed is a simple, fully-connected 3-layer network and the classical burden of training the network is negligible. Given a new problem instance, our method directly predicts the corresponding variational parameters without needing to execute the quantum circuit. In addition, as the training set grows, the neural network can rapidly adjust to the new data by few classical-learning iterations. Thus, in contrast to other methods, our method does not require any auxiliary executions on quantum devices. In this paper we employed the simplest deep network possible. Other NN architectures, like graph neural networks (GNN), may be better for this problem, and we expect this to be a topic of future research.

The ability of the proposed NN method to personalize the predicted parameters per graph instance becomes more significant as the graph ensemble is more varied and the spread of optimal parameters is broader. We demonstrated this property by showing that while the proposed method outperforms all benchmark methods for constant-edge Erdős–Rényi graphs, it suppresses them even further when the edge probability is taken to be random. We thus conjecture that our method will provide an even more significant benefit for classes of graphs that show larger distributions, e.g., for random and weighted graphs, for which the optimal parameters are known to have a wider distribution [10]. Such graphs are

prevalent in many practical applications, such as VLSI degisn, and social networking, see e.g., [41].

Another manifestation of the same observation is that for the realistic scenario of a finite number of layers p and a growing number of graph nodes N, our method leads to better performance compared to other methods (see Fig. 3) for approximating the MaxCut of random Erdős–Rényi graphs; As quantum computers develop and have more qubits, they are able to solve larger instance problems. Yet, for practical solutions, the number of layers in QAOA must be finite. This makes our method especially practical in the NISQ era, where quantum devices increase in size but are too noisy for executing deep circuits.

Finally, in this work we focused on the MaxCut problem but we believe that our method is beneficial for any optimization problem, solved via a variational quantum algorithm (not necessarily QAOA), that incorporates an underlying mapping between the different instances of the problem and their corresponding optimized variational parameters.

Acknowledgement

We would like to thank Yoni Zimmermann and Yehuda Naveh for the fruitful discussions.

References

[1] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[2] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[3] Jarrod R McClean, Mollie E Kimchi-Schwartz, Jonathan Carter, and Wibe A De Jong. Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states. *Physical Review A*, 95(4):042308, 2017.

[4] Giacomo Nannicini. Performance of hybrid quantum-classical variational heuristics for combinatorial optimization. *Physical Review E*, 99(1):013304, 2019.

[5] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[6] Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674*, 2016.

[7] Gavin E Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv preprint arXiv:1811.08419*, 2018.

[8] Bernhard H Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial optimization*, volume 1. Springer, 2011.

[9] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[10] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, 2020.

[11] Stefan H Sack and Maksym Serbyn. Quantum annealing initialization of the quantum approximate optimization algorithm. *arXiv preprint arXiv:2101.05742*, 2021.

[12] Fernando GSL Brandao, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven. For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances. *arXiv preprint arXiv:1812.04170*, 2018.

[13] Sami Khairy, Ruslan Shaydulin, Lukasz Cincio, Yuri Alexeev, and Prasanna Balaprakash. Reinforcement learning for quantum approximate optimization. *Research Poster, accepted at Supercomputing*, 19, 2019.

[14] Vishwanathan Akshay, Daniil Rabinovich, Ernesto Campos, and Jacob Biamonte. Parameter concentrations in quantum approximate optimization. *Physical Review A*, 104(1):L010401, 2021.

[15] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. Accelerating quantum approximate optimization algorithm using machine learning. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 686–689. IEEE, 2020.

[16] Daniil Rabinovich, Richik Sengupta, Ernesto Campos, Vishwanathan Akshay, and Jacob Biamonte. Progress towards analytically optimal angles in quantum approximate optimisation. *Mathematics*, 10(15):2601, 2022.

[17] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[18] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. SIAM, 2019.

[19] Phillip C Lotshaw, Thien Nguyen, Anthony Santana, Alexander McCaskey, Rebekah Herrman, James Ostrowski, George Siopsis, and Travis S Humble. Scaling quantum approximate optimization on near-term hardware. *arXiv preprint arXiv:2201.02247*, 2022.

[20] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[21] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[22] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.

[23] Madita Willsch, Dennis Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19:1–24, 2020.

[24] Ruslan Shaydulin, Ilya Safro, and Jeffrey Larson. Multistart methods for quantum approximate optimization. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8. IEEE, 2019.

[25] Alexey Galda, Xiaoyuan Liu, Danylo Lykov, Yuri Alexeev, and Ilya Safro. Transferability of optimal qaoa parameters between random graphs. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 171–180. IEEE, 2021.

[26] Matteo M Wauters, Emanuele Panizon, Glen B Mbeng, and Giuseppe E Santoro. Reinforcement-learning-assisted quantum optimization. *Physical Review Research*, 2(3):033446, 2020.

[27] Sami Khairy, Ruslan Shaydulin, Lukasz Cincio, Yuri Alexeev, and Prasanna Balaprakash. Learning to optimize variational quantum circuits to solve combinatorial problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2367–2375, 2020.

[28] Sami Khairy, Ruslan Shaydulin, Lukasz Cincio, Yuri Alexeev, and Prasanna Balaprakash. Reinforcement-learning-based variational quantum circuits optimization for combinatorial problems. *arXiv preprint arXiv:1911.04574*, 2019.

[29] Max Wilson, Rachel Stromswold, Filip Wudarski, Stuart Hadfield, Norm M Tubman, and Eleanor G Rieffel. Optimizing quantum heuristics with meta-learning. *Quantum Machine Intelligence*, 3(1):1–14, 2021.

[30] Guillaume Verdon, Michael Broughton, Jarrod R McClean, Kevin J Sung, Ryan Babbush, Zhang Jiang, Hartmut Neven, and Masoud Mohseni. Learning to learn with quantum neural networks via classical neural networks. *arXiv preprint arXiv:1907.05415*, 2019.

[31] Haibin Wang, Jiaojiao Zhao, Bosi Wang, and Lian Tong. A quantum approximate optimization algorithm with metalearning for maxcut problem and its simulation via tensorflow quantum. *Mathematical Problems in Engineering*, 2021, 2021.

[32] Daniel J Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, 2021.

[33] Nishant Jain, Brian Coyle, Elham Kashefi, and Niraj Kumar. Graph neural network

initialisation of quantum approximate optimisation. *arXiv preprint arXiv:2111.03016*, 2021.

[34] Franz G Fuchs, Herman Øie Kolden, Niels Henrik Aase, and Giorgio Sartor. Efficient encoding of the weighted max k-cut on a quantum computer using qaoa. *SN Computer Science*, 2(2):1–14, 2021.

[35] Charles George Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

[36] Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

[37] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

[38] David F Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

[39] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, et al. Qiskit: An open-source framework for quantum computing. 16, 2019.

[40] Boaz Barak and Kunal Marwaha. Classical algorithms and quantum limitations for maximum cut on high-girth graphs. *arXiv preprint arXiv:2106.05900*, 2021.

[41] Paul Spirakis, Sotiris Nikoletseas, and Christoforos Raptopoulos. Max cut in weighted random intersection graphs and discrepancy of sparse random set systems. *LIPIcs: Leibniz International Proceedings in Informatics*, 2021.

A Finding an optimal Δt for the TQA method

In our experiments, for each number of nodes N, number of layers p, and graph ensemble (constant and random ER graphs, as in the main text), we applied a grid search to find the optimal Δt (Eq. 7), as in Ref. [11]. We averaged the best Δt's over 50 different graphs to obtain a single initialization that would fit as well as possible the variety of graphs. Fig. 5 shows the optimal Δt for different graph sizes, with $p = 2$. For the random ER graphs, the spread of the optimal Δt is wider, as the distribution of the graphs is broader.

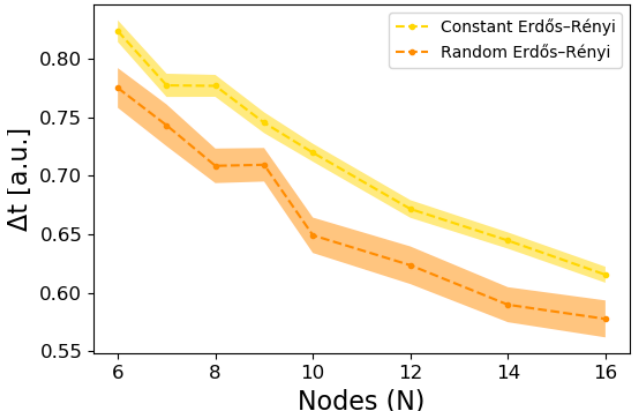


Figure 5: Optimal Δt for TQA initialization for each graph size (i.e. the number of nodes), with $p = 2$. Constant Erdős–Rényi graph ensemble is drawn in yellow and random Erdős–Rényi graph ensemble in orange. The shades mark the standard error of the mean (SEM).