

MLQAOA: Graph Learning Accelerated Hybrid Quantum-Classical Multilevel QAOA

Bao Bach

*Computer and Information Sciences
Quantum Science and Engineering
University of Delaware
Newark DE, USA
baobach@udel.edu*

Jose Falla

*Physics and Astronomy
University of Delaware
Newark DE, USA
jfalla@udel.edu*

Ilya Safro

*Computer and Information Sciences
Physics and Astronomy
University of Delaware
Newark DE, USA
isafro@udel.edu*

May 2, 2024

Abstract

Learning the problem structure at multiple levels of coarseness to inform the decomposition-based hybrid quantum-classical combinatorial optimization solvers is a promising approach to scaling up variational approaches. We introduce a multilevel algorithm reinforced with the spectral graph representation learning-based accelerator to tackle large-scale graph maximum cut instances and fused with several versions of the quantum approximate optimization algorithm (QAOA) and QAOA-inspired algorithms. The graph representation learning model utilizes the idea of QAOA variational parameters concentration and substantially improves the performance of QAOA. We demonstrate the potential of using multilevel QAOA and representation learning-based approaches on very large graphs by achieving high-quality solutions in a much faster time.

Reproducibility: Our source code and results are available at <https://github.com/bachbao/MLQAOA>

1 Introduction

A general strategy for tackling many large-scale computational science problems on various hardware architectures, including fundamental combinatorial optimization problems on graphs, is the use of multilevel algorithms (also known as multiscale, multiresolution, and multigrid methods) [1]. The multilevel approach starts by coarsening the problem to create a series (also known as a hierarchy) of progressively simpler, related problems at coarser levels, which are more feasible for the currently available hardware. This strategy is particularly useful in quantum computing where the number of qubits is limited as well as their overall quantum circuit fidelity, depth, and qubit connectivity. As a result, the hybrid quantum-classical algorithm developers put a lot of effort into making the circuit more compact. In particular, this is highly relevant to variational quantum algorithms [2] (e.g., many versions of the variational quantum eigensolver and quantum approximate optimization algorithm) when the same or slightly updated quantum circuit is reparametrized at each iteration and executed many times.

The motto of the multilevel algorithms is “Think globally but act locally”. At each coarse level i , the best-found solution serves as an initialization for the next finer solution at level $i - 1$. This initialization is enhanced through what is commonly referred to as “local processing” (also known as a refinement), a cost-effective series of fast steps that involve only a few variables at a time but collectively revisit all variables of that level multiple times. This method fits well with quantum computers, as it allows for solving parts of the problem within the constraints of limited qubit numbers.

In the classical domain, typical examples of such local processing include several iterations of classical relaxation methods like Gauss-Seidel or Jacobi relaxations when solving systems of equations, a few Monte Carlo passes in statistical physics simulations, or Kernighan-Lin or Fiduccia-Mattheyses search for optimization on graphs [1]. In the quantum context, such multilevel framework was explored for the graph partitioning, clustering, and the maxcut problems. In both cases, the main local processing component was the quantum approximate optimization algorithm (QAOA). While these multilevel frameworks played their role and allowed increasing the instance graph size, the overall running time was suffering from the variational loop slowness. In this paper, we introduce graph representation learning-based hybrid quantum-classical multilevel QAOA and quantum-inspired recursive optimization frameworks that break the performance barrier and improve the overall solution quality.

The QAOA [3] stands out as a promising candidate, offering the potential to achieve a speedup to certain combinatorial problems or classes of instances. However, the current landscape of quantum computers, characterized by the NISQ (Noisy Intermediate-Scale Quantum) era, poses various challenges [4]. Large-scale implementation of QAOA remains impractical [5] due to physical limitations inherent in current quantum devices, such as constraints related to connectivity and noise. These hardware limitations present significant obstacles to realizing the full potential of QAOA in real-world applications. Taking into consideration the constraints and limitations of current quantum hardware, the hybrid architecture of QAOA emerges as one of the most promising approaches to mitigate the impact of hardware constraints by incorporating classical counterparts [2].

Our contribution We introduce a scalable hybrid quantum-classical multilevel scheme integrated with two QAOA-inspired solvers and accelerated by the graph representation learning for fast parametrization of QAOA [6]. This scheme allows scalability for fast small-scale QAOA solvers by decomposing the original problem across the different scales of coarseness into sub-problems and constructing the global solution from the sub-problem solutions. This resolves the huge time complexity that was a significant barrier in previous hybrid quantum-classical decomposition-based algorithms [7–9] and makes them comparable even to purely classical algorithms.

Specifically, our work focuses on addressing the maximum cut (MAXCUT) problem on large-scale graphs using the graph representation learning-based parameter transferability for QAOA [6] and quantum-informed recursive optimization [10] as accelerators. These methods enhance the execution time of the sub-problem solver, resulting in an overall speed-up of the multilevel scheme with two orders of magnitude compared with [7–9] approaches on mid-scale graphs and *way more on larger that (to the best of our knowledge) no hybrid quantum-classical approach was able to tackle in a reasonable computational time*. Our approach not only yields faster runtimes but also improves the performance of multilevel QAOA (it was impossible to run other QAOA-based methods due to the slow performance and/or huge memory requirements). To validate our approach, we evaluate it on diverse sets of graphs, including real-world problems like social networks, optimization instances, graphs that are hard for MAXCUT, and those that are particularly hard for the Goemans-Williamson MAXCUT approximation [11–14]. Even in the QAOA simulation mode, our experimental results demonstrate competitive quality compared to the MAXCUT dedicated state-of-the-art classical solvers, with comparable runtime.

2 Preliminaries and Notations

2.1 QAOA and MAXCUT

Quantum approximate optimization algorithm (QAOA) [3] is a hybrid quantum algorithm focusing on combinatorial optimization problems. The quantum heuristic algorithm aims to produce an approximation of the problem's solution by alternating between cost-function-based Hamiltonian and mixing Hamiltonian. The QAOA variational loop consists of p parameterized layers of alternating unitary operators and a classical optimizer. The role of the optimizer is to find the best set of parameters that minimize the cost function of the problem.

Given a combinatorial optimization problem with a cost function $f(x)$ where $x \in \{0, 1\}^n$, QAOA alternately apply unitaries drawn from two Hamiltonian families, cost-function-based unitary $U_P(\gamma) = e^{-i\gamma H_f}$ and mixing unitary $U_M(\beta) = e^{-i\beta H_B}$ parametrized by $\gamma = \{\gamma_i\}$ and $\beta = \{\beta_i\}$, $1 \leq i \leq p$, respectively. Hamiltonian H_f is a cost function-based Hamiltonian where the information of cost function $f(x)$ is embedded while Hamiltonian H_B is a fixed mixing Hamiltonian. With H_f as the observable, QAOA prepares the quantum state expressed in (1) and performs optimization concerning the expectation value $\langle H_f \rangle = \langle \gamma, \beta | H_f | \gamma, \beta \rangle$.

$$|\gamma, \beta\rangle = U_M(\beta_p)U_P(\gamma_p) \dots U_M(\beta_1)U_P(\gamma_1) |+\rangle^{\otimes n} \quad (1)$$

The MAXCUT problem on a graph is the first QAOA demonstration [3] and the algorithm's most common benchmark. This problem is NP-complete [15]. The problem involves finding a cut that splits the graph nodes into two disjoint parts V_1 and V_2 such that the weighted sum of edges ij connecting two parts is maximized. The MaxCut problem is often formulated as a quadratic unconstrained binary optimization problem (QUBO) by assigning a binary value x_i to every node based on its part. Given a graph $G(V, E, w)$ where V is the set of nodes, E is the set of edges and w is the edge weighting function, the MAXCUT problem is defined in (2).

$$\max_{x \in \{-1, 1\}^n} \sum_{ij \in E} w_{ij} \frac{(1 - x_i x_j)}{2} \quad (2)$$

For this QUBO problem, the Hamiltonian can be constructed by applying the mapping $x_i \mapsto \frac{1}{2}(1 - Z_i)$, where Z is the Pauli operator Z .

2.2 Multilevel Methods

The multilevel methods for optimization on graphs are inspired by the multigrid methods that were originally devised to tackle boundary value problems in spatial domains [1]. Choosing a set of grids makes these problems discrete and consists of algebraic equations associated with the grid points. The essence of the multigrid method lies in iteratively increasing the grid point spacing, transforming the original problem into coarser versions, and leveraging solutions from these coarser problems to aid in finding the final solution. The theory behind this scheme is based on two observations. First, the standard iterative methods have smoothing properties, this means they are effective at relaxing oscillatory components of error while leaving smooth components unchanged. Second, the smooth modes on a fine grid look less smooth on a coarse grid. Those observations imply that incorporating coarse grids during computation can make the smooth components of error of the finest grid look more oscillatory and be eliminated by the relaxation of the iterative method. The idea of such elimination and some ways of constructing the coarse problems were inherited from the multilevel methods.

Nowadays, multigrid-inspired multilevel methods are applied to a broader spectrum of problems regardless of their connection with physical grids. A broader organizational framework has emerged, replacing the concept of a coarse grid with a more generalized notion known as the multilevel method [1]. The approach proved itself to be useful in the (hyper)graph context where the multilevel method utilizes the graph structure to curtail large-scale graphs into their coarser representations. Examples of problems successfully tackled by multilevel methods include graph partitioning [16], visualization [17], linear ordering [18], and generation [19].

While there are several types of the coarsening-uncoarsening schedules in multilevel methods (e.g., V-, W-, or FMG cycles [1]), in this work we use the simplest V-cycle to eliminate additional advantages of the classical computing in the hybrid framework. In this setting, we generate a hierarchy of next coarser graphs

$$\{G_l = (V_l, E_l, w_l)\}_{l=0}^L, \quad (3)$$

where l is the index of level, G_0 is the original large-scale graph, and G_L is the coarsest smallest graph. The coarsening consists of (1) relaxation-based grouping pairs of nodes based on the recently introduced maximization version [9] of the algebraic distance for graph coarsening [20], and (2) edge weight accumulation.

After the hierarchy is created, the MAXCUT at the coarsest level is solved and the solution is gradually uncoarsened all the way up to the finest level. At each step of the uncoarsening, the l th level solution is initialized by that from level $l + 1$ and further refined by various QAOA approaches. In the end, the final solution of the original problem is obtained.

2.3 Graph Representation Learning

Graph representation learning techniques have shown great promise in addressing graph analytic tasks, such as node classification, link prediction, and community detection by transforming graph features (nodes, edges, edge weights, etc.) into non-linear, low-dimensional, dense, continuous, and highly informative vector spaces [21]. In these low-dimensional graph representations, if two graph instances possess common structural features, they exhibit closeness with respect to some distance function, such as a Euclidean distance function. In this work, this idea is utilized for the scalable transferability of QAOA parameters.

Of these graph representation techniques, some involve node-level embedding at various scales (microscopic, mesoscopic, and macroscopic node embedding), while others (that are relevant to this work) involve whole graph embedding. Whole graph embedding techniques, as previously mentioned, are useful when analyzing whole networks [22–25], particularly when trying to determine the structural similarity between graphs.

Methods for graph representation learning can be grouped into a few categories. Among these categories, a classic family of methods involves graph kernels, with examples including the Weisfeiler-Lehman [26], random walk [27], shortest path [28], and deep graph [29] kernels. Another family of methods involves graph embedding for learning vector representation of graphs, with examples including Graph2Vec [25], which uses the Weisfeiler-Lehman kernel to extract rooted subgraph features to obtain the embeddings; also, GL2Vec [30], which is an extension of Graph2Vec that includes line graphs to account for edge features. More recently, an approach to graph representation learning involves graph neural networks, which employ machine learning methods, with some examples including GCN [31], SGCN [32], GIN [33], and Causal GraphSAGE [34], to name a few. Finally, there is a family of methods that use information from the graph’s Laplacian matrix and its eigenvalues to generate embeddings, such as SF [35], NetLSD [36], and FGSD [37]. Our method belongs to the latter type of method.

3 Related works

3.1 QAOA-based Approaches

At the heart of this algorithm lies the utilization of QAOA to optimize sub-problem graphs, which refines the final solution on the finest graph. Hence, the quality of the QAOA solution serves as a benchmark for our refinement performance. In this study, we employ vanilla QAOA with three layers, bypassing the exhaustive optimization process by incorporating graph learning techniques for parameter transferability [6, 38]. This approach is one of many attempts to enhance the performance of QAOA, some noticeable attempts include using warm-start for good initialization [39], noise-directed adaptive mapping for a higher-quality solution based on previous step result [40], finding the best QAOA ansatz architecture for the given Hamiltonian with bayesian optimization [41] and the connection between underlying symmetry of the objective function and QAOA [42].

3.2 Hybrid Quantum-Classical Algorithms For Large Graphs

Quantum algorithms like the Quantum Approximate Optimization Algorithm (QAOA) hold significant promise in addressing complex graph problems. However, their efficacy is hindered by the current limitations of quantum hardware, which often suffer from noise and architectural constraints. Consequently, to overcome these limitations and scale up the application of quantum algorithms such as QAOA to tackle larger graph instances, researchers explore strategies to downscale the original problems into manageable subproblems. The first attempts to utilize multilevel methods in a hybrid quantum-classical setting, [8, 9] introduced the coarsening-uncoarsening approach and solved multiple small graphs at each level to enhance the final solution of the original larger graph problem. This work extended the single-level decomposition-based approaches [7]. However, already with the subproblem graph size of 22 nodes, the solvers were too slow due to the parametrization in variational loops. This scalability issue is addressed in this work.

In a similar vein, [43] employs the graph decomposition technique to scale down the original graph to a manageable size, producing a high-quality approximation of the final solution from the simplified problem. In addition to the down-scaling approach, the divide-and-conquer method stands out as another candidate. As exemplified by [44], the work demonstrates the potential of tackling divided subgraphs in parallel and then merging these solutions to derive the final solution. This approach is comparable to the two-level scheme if viewed as a multilevel algorithm.

From the circuit cutting, [45] creates a hybrid scalable approach CutQC to distributing big quantum circuits that can be run on smaller quantum processing units (QPU), this allows breaking the limit of classical simulation and achieving a larger quantum circuit evaluation. Similarly, [46] introduces SuperSim which uses Clifford-based circuit cutting. By isolating (cutting) the Clifford circuit within the big non-clifford circuit, the approach leads to the utilization of Clifford simulation that greatly reduces runtime.

3.3 Graph Learning For Parameter Transferability

The task of finding good QAOA parameters is challenging in general. For example, determining whether the optimized solution corresponds to a local or a global minimum in the energy landscape, or due to encountering barren plateaus [47, 48]. Furthermore, while the QAOA solution improves as the depth of the circuit is increased, it only does so marginally at the cost of increasing the computational complexity of optimizing the variational parameters [49]. For this reason, acceleration of optimal parameter search for a given QAOA depth p is a key in demonstrating quantum advantage. Examples of optimal parameter search acceleration include warm- and multi-start optimization [39, 50], problem decomposition [7], instance structure analysis [42], and multigrid inspired parameter learning [51]. In particular, optimal QAOA parameter transferability has shown great promise in circumventing the problem of finding good QAOA parameters [38, 52]. Based on structural graph features, successful parameter transferability can be achieved between a donor graph and an acceptor graph, with a very small reduction in the approximation ratio.

4 Our methods

The multilevel MAXCUT QAOA approach, as introduced in [9], encounters a significant challenge stemming from the substantial computational overhead incurred in refining numerous sub-problem graphs by slow variational loops. In some cases, this issue leads to inefficiencies even when compared to general-purpose global solvers such as Gurobi [53]. To address this limitation, we leverage the potential of graph learning techniques for QAOA parameter transferability, as outlined in [6] and the use of single and second correlations to simplify the sub-problem graph [54]. Our proposed method (MLQAOA) introduces a reinforcement for this multilevel scheme. By using a graph learning model tailored to the weighted graphs at each level or quantum-informed recursive optimization, the model then effectively approximates high-quality ansatz parameters, generating high-quality solutions for the sub-problems. Figure 1 demonstrates our method.

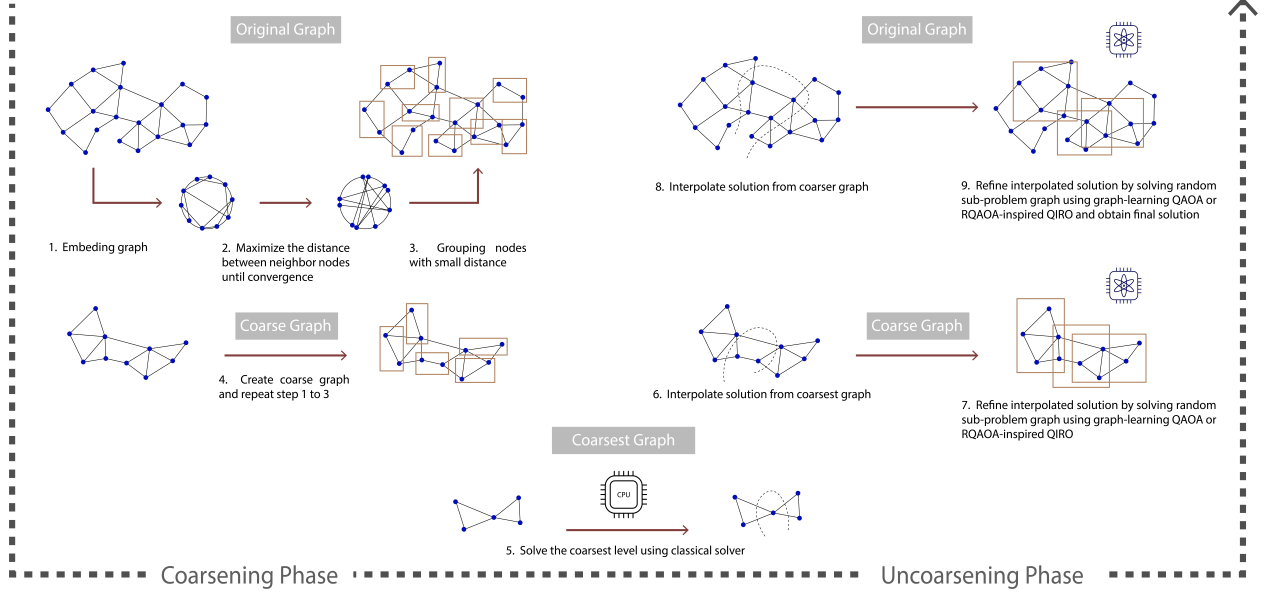


Figure 1: MLQAOA scheme using V-cycle. The original graph is iteratively coarsened in the Coarsening Phase. The coarsest graph is solved using a classical solver. Building upon this solution, the Uncoarsening Phase leverages the previous level solution through interpolation and performs refinement by solving sub-problem graphs.

4.1 Coarsening Phase

At each level of coarseness (i.e., at each G_l in (3)), the coarsening phase is started by constructing a d -dimensional unit sphere with every node from the graph G_l embedded and further relaxed in. This is essential for establishing a distance metric between nodes for multilevel algorithms, as discussed in [55]. Every node is initially placed in a random position on the surface of the sphere or d -dimensional hypercube embedded in the sphere. Denote p_i^t as the position of node i at iteration t . For example, for the initial hypercube embedding $\forall i \in V_l, p_i^0 \leftarrow \text{rand}[-1, 1]^d$.

Following the initialization, several node-wise correction iterations are applied to maximize within the sphere the total weighted distance between each node and its neighbors:

$$\forall i \in V_l, p_i^{t+1} \leftarrow \max \sum_{j \in N(i)} w_{ij} \|p_i^t - p_j^t\|_2. \quad (4)$$

Here, $N(i)$ represents the set of neighbors of node i , and w_{ij} is the weight of the edge ij . This iterative scheme continues until convergence is achieved.

Subsequently, by leveraging the embedding, the nodes are paired for further coarsening using a K-D tree. A fast search within the K-D tree facilitates pairing the node with its nearest unpaired neighbor. This search finalizes the formation of the new coarse graph by contracting every paired node in the matching and creating the coarse nodes for V_{l+1} . Algorithm 1 describes the coarsening scheme. This process is repeated until the desired coarsest graph size is achieved. Finally, by grouping into pairs of nodes at each level, the original graph is broken into approximately $\log |V_0|$ number of increasingly coarse graphs.

4.2 Uncoarsening Phase

The uncoarsening phase starts with the coarsest graph G_L obtained in the last coarsening step. The number of nodes in G_L is based on the sub-problem size that is defined by the user. The solution to the problem

Algorithm 1 Coarsening

Require: Graph at level l , $G_l = (V_l, E_l, w_l)$

```
1:  $A_l \leftarrow$  Adjacency matrix of  $G_l$ 
2: Initialize embedding of  $G_l$  into sphere
3:  $\mathfrak{E} \leftarrow$  Apply iterations of (4)  $\triangleright V_l$  is embedded into sphere
4:  $matched \leftarrow \emptyset$ ;  $pairs \leftarrow \emptyset$ 
5: for  $i \in V_l$  do  $\triangleright$  Pair node with nearest neighbor in  $\mathfrak{E}$ 
6:   if  $i \notin matched$  then
7:      $j \leftarrow \mathfrak{E}$ -nearest not matched neighbor of  $i$ 
8:      $matched \leftarrow matched \cup \{i, j\}$ 
9:      $pairs \leftarrow pairs \cup (i, j)$ 
10:  end if
11: end for
12:  $\triangleright$  Now each pair is a node in  $V_{l+1}$  and  $|V_{l+1}| = |pairs|$ 
13:  $P \leftarrow 0^{|V_{l+1}| \times |V_l|}$   $\triangleright$  Initialize multilevel restriction operator. For details see [55].
14:  $q \leftarrow 0$   $\triangleright$  Initialize iterator over coarse nodes
15: for  $(i, j) \in pairs$  do  $\triangleright$  Construct coarse graph
16:    $P_{q,i} \leftarrow 1$ ;  $P_{q,j} \leftarrow 1$ 
17:    $q \leftarrow q + 1$ 
18: end for
19:  $A_{l+1} \leftarrow P^T A_l P$   $\triangleright$  This is the algebraic multigrid formulation to create the coarse graph with weighted edges that are accumulated from the fine level. In general, algebraic multigrid is not restricted to having only two fine nodes to form a coarse node. For details see [55].
20: return  $G_{l+1}$  obtained from  $A_{l+1}$ 
```

of the coarsest graph is found through state-of-the-art classical solver MQLib employing the BURER2002 heuristic [56] or extensively optimized QAOA. Usually, the larger $|V_L|$ could be, the better it affects the final results.

Building upon the initial solution of G_L , the uncoarsening phase progresses towards a finer level through linear interpolation from the previous coarser level solution. This interpolation is a surjective mapping $F : V_{l+1} \rightarrow V_l$ (i.e., mapping of a fine node to its aggregating coarse node), and the initial solution at the fine level is initialized by $\forall i \in V_{l+1}, x_i = x_{F(i)}$. This yields an initial approximation for further refinement.

The refinement scheme begins with the computation of the gain of every node. This gain is then used to estimate the impact of each node on the final energy at the current level and update the MAXCUT objective. This gain is efficiently tracked by updating it based on the edges that enter or leave the cut.

$$\forall i \in V_l, gain(i) \leftarrow \sum_{j \in N(i)} w_{ij} (-1)^{2x_i x_j - x_i - x_j} \quad (5)$$

During the refinement stage, sub-problems are iteratively generated and solved to improve the final solution. In each iteration, n random nodes are sampled and ranked by their gains, and K nodes with the highest gain are then used to construct the sub-problem graph. This construction involves creating a graph with two super-nodes, along with the selected nodes. Each super-node aggregates the nodes not chosen for the sub-problem, and weighted edges are added between the super-nodes and chosen nodes.

Given the sub-problem graph, a sub-problem graph solver scheme must be devised. In this work, instead of using exhaustive learning on vanilla QAOA, we study how the graph learning for parameter transferability for QAOA with depth $p = 3$ [6] and quantum-informed recursive optimization algorithm [10] works as a sub-problem solver. This investigation follows the spirit of the classical multilevel algorithm. By leveraging an excellent efficient, small-scale problem solver, the multilevel approach enables the scalability of the problem solver with good approximation.

After the relaxation, if the sub-problem solution enhances the objective function, the solution and ob-

jective are updated. The iteration counter is reset whenever there is an improvement. To prevent excessive iterations, a restriction is set: if three consecutive iterations do not yield improvement or if there have been a total of ten iterations, the algorithm accepts the current solution and proceeds to the next level. Algorithm 2 outlines the process.

Algorithm 2 Refinement

Require: Graph $G_l(V_l, E_l)$, Subproblem graph size K , Initial solution from previous level S_{l+1}

```

1:  $\mathfrak{G} \leftarrow$  compute gain for all nodes as in Eq. (5)
2:  $maxIter, count \leftarrow 0$ 
3:  $\mathcal{O} \leftarrow$  compute objective inherited from  $S_{l+1}$ 
4: while  $count < 3$  and  $maxIter < 10$  do
5:   if  $count = 0$  then
6:      $n \leftarrow |V_l|$ 
7:   else
8:      $n \leftarrow \max(0.3|V_l|, 2K)$ 
9:   end if
10:   $subset \leftarrow K$  highest gain nodes from randomly sampled  $n$  nodes from  $V_l$ 
11:   $P \leftarrow \text{constructMAXCUTSubproblem}(subset)$ 
12:  Solve  $P$  and compute new objective  $\mathcal{O}^{new}$ 
13:  Update  $\mathfrak{G}$ 
14:   $maxIter \leftarrow maxIter + 1$ 
15:  if  $\mathcal{O}^{new} \geq \mathcal{O}$  then
16:     $\mathcal{O} \leftarrow \mathcal{O}^{new}$ 
17:     $S_l \leftarrow$  new solution derived from  $\mathcal{O}^{new}$ 
18:     $count \leftarrow 0$ 
19:  else
20:     $count \leftarrow count + 1$ 
21:  end if
22: end while
23: return  $S_l$ 

```

4.3 Graph Representation Using Spectral Learning For Parameter Transferability On Weighted Graphs

We begin the graph representation by defining the Laplacian matrix:

$$L = D - A, \quad (6)$$

where D is the degree matrix and A the adjacency matrix. Denoting the vector $w = (w_1, \dots, w_n)$ as the weights for each node $(1, \dots, n)$, $D = \text{diag}(d)$, where $d = Aw$. The Laplacian is a symmetric, positive semi-definite matrix. Observing that,

$$\forall v \in \mathbb{R}^n, \quad v^T L v = \sum_{i < j} A_{ij} (v_j - v_i)^2 \quad (7)$$

the spectral theorem yields:

$$L = U \Lambda U^T, \quad (8)$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ is the diagonal matrix of eigenvalues of L , with $0 = \lambda_1 < \lambda_2 \leq \dots \leq \lambda_n$, and $U = (u_1, \dots, u_n)$ is the matrix of corresponding eigenvectors, with $U^T U = I$.

For our graph representation model, we consider the following normalized version of the Laplacian, referred to as the *weighted* Laplacian:

$$L_W = W^{-\frac{1}{2}} L W^{\frac{1}{2}}, \quad (9)$$

where $W = \text{diag}(w)$. Once again, this is a symmetric, positive semi-definite matrix, and the spectral theorem yields:

$$L_W = \hat{U} \hat{\Lambda} \hat{U}^T, \quad (10)$$

where $\hat{\Lambda} = \text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_n)$ is the diagonal matrix of the eigenvalues of L_W , with $0 = \hat{\lambda}_1 < \hat{\lambda}_2 \leq \dots \leq \hat{\lambda}_n$, and $\hat{U} = (\hat{u}_1, \dots, \hat{u}_n)$ is the matrix of corresponding eigenvectors, with $\hat{U}^T \hat{U} = I$.

To determine the similarity between two graph instances, we compute the Euclidean distance between the first non-trivial $\hat{u}_2$ (second) eigenvector¹ of the weighted Laplacian matrix 10 of each of the graphs. The reason for this is that the first non-trivial eigenvector of a graph's Laplacian contains information about the graph's connectivity, and therefore, its structure. We are interested in determining graph similarity based on structure since our previous work has shown that structural similarity is a key feature in determining successful parameter transferability between two graph instances [38].

The construction of the model for parameter transferability begins by generating a corpus of 22-node graphs. This corpus of graphs contains $\sim 5,000$ various types of graphs, including random graphs with weight distributions ranging from $w \in [0, m * 5)$ with $m \in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$. Additionally, the corpus includes graphs that are structurally similar to those found in our multi-level approach. Each of the corpus 'graphs' optimal parameters is optimized on 20 independent multi-starts. For the 20 multi-starts, the best solution to the expectation value of the cost Hamiltonian $\langle H_f \rangle = \langle \gamma, \beta | H_f | \gamma, \beta \rangle$ is taken as the graph's optimal solution, and the optimal parameters $(\vec{\gamma}, \vec{\beta})$ are stored². The circuit used for graph optimization is constructed as per Section 2.1 using Cirq [57], and the quantum-classical optimization loop is performed with a COBYLA solver for 300 iterations, or until convergence. In the end, the model consists of an $N \times n$ matrix, where N is the number of graphs in the corpus, and n is the number of nodes, with each row corresponding to the first non-trivial eigenvector of each graph in the corpus.

4.4 Quantum-Informed Recursive Optimization Algorithm

QAOA is a *local* algorithm, a characteristic that has been demonstrated to constrain its overall performance. To address this problem, recursive QAOA [54] introduced non-local updates by iteratively eliminating variables using the single and double correlation information between them. This iterative process systematically prunes variables from the original optimization problem while creating new connections between previously distant pairs of nodes. The change in graph structure introduces a non-local effect that counteracts the inherent locality of QAOA. Based on this foundation, Quantum-Informed Recursive Optimization Algorithm (QIRO) [10] emerges as a family of approaches that leverages quantum information to derive the potential classical problem-specific reduction, thus recursively simplifying the original problem.

The framework of QIRO is shown in algorithm 3, where the original problem undergoes successive simplification until reaching a desired size. This reduction process exploits single and double correlations concerning a low-energy quantum state (lines 6, 7) and a problem-specific simplification rule (line 9). In this study, we adopt the simplification rule for the MAXCUT problem from [54], where the rule is defined with the use of correlation matrix M . Initially, we find the edge $(i, j) \in E_{it}$ exhibiting the largest magnitude of M_{ij} . Subsequently, we treat $\hat{Z}_i$ and $\hat{Z}_j$ as correlated if $M_{ij} > 0$ and anti-correlated if $M_{ij} < 0$. By assigning $\hat{Z}_j = \text{sgn}(M_{i,j})Z_i$, we effectively eliminate variable Z_j from our Hamiltonian, resulting a new Hamiltonian depending on only $|V| - 1$ variable or a graph G with $|V| - 1$ nodes. We remark on the possibility that there may exist a better simplification strategy for the MAXCUT problem that we are not aware of.

To ensure RQAOA-inspired QIRO maintains a competitive runtime, we employ QAOA with a single layer, as the single and double correlation can be efficiently calculated [58].

¹Since we are working with connected graphs, the first eigenvector of the Laplacian is trivial.

²For a depth of $p = 3$, there are 3 optimal γ and 3 optimal β parameters.

Algorithm 3 General scheme of QIRO algorithm

Require: Graph $G_0(V_0, E_0)$ **Require:** Smallest problem size s

```
1:  $l \leftarrow 0$ 
2:  $S \leftarrow \emptyset$ 
3: while  $|V_l| > s$  do
4:   Prepare low-energy quantum state  $|\psi\rangle$ 
5:    $M \leftarrow 0^{|V_l| \times |V_l|}$ 
6:    $\forall i \in V_l, M_{ii} \leftarrow \langle \psi | \hat{Z}_i | \psi \rangle$ 
7:    $\forall (i, j) \in E_l, M_{ij} \leftarrow \langle \psi | \hat{Z}_i \hat{Z}_j | \psi \rangle$ 
8:    $l \leftarrow l + 1$ 
9:    $G_l(V_l, E_l), S \leftarrow \text{Simplification}(G_{l-1}, M, S)$ 
10: end while
11:  $S \leftarrow \text{Classical\_Solver}(G_l)$ 
12: return  $S$ 
```

5 Computational Results

Our source code and results are available at <https://github.com/bachbao/MLQAOA>. To evaluate the performance and scalability of MLQAOA alongside the proposed sub-problem graph solvers, we conducted numerical simulations using large instance graphs sourced from diverse datasets. We present a detailed outline of our experimental setup, including hyperparameter configurations for MLQAOA and both sub-problem solvers, namely, Graph Learning Parameter Transfer and RQAOA-inspired QIRO. Subsequently, we offer a comprehensive comparison between our approach and classical solvers, focusing on approximation ratio and runtime metrics.

The configuration for MLQAOA sub-problem solvers is as follows: The sub-problem graph size, denoted by K , is set to 20. For parameter transferring using graph learning on weighted graphs, the QAOA entails a 22-qubit circuit, comprising 20 chosen nodes and 2 supernodes, with 3 layers. For RQAOA-inspired QIRO, the smallest size problem is 10. The simulations are executed on the `ibmq_qasm_simulator` with 10240 sampling shots.

We simulate our method on three different graph instance sets: (1) the well-known public MAXCUT dataset G_{set} [11], (2) the Karloff graphs [12] and (3) larger graphs sourced from SuiteSparse Matrix Collection [13] and the Network Repository [14]. The selection of these graphs is based on their widespread use in MAXCUT problems, their diversity in size, degree distributions, and edge weights, and their challenging nature, as demonstrated by their effectiveness as benchmarks for existing solvers. Especially, the Karloff collection contains challenging graphs for the Goemans-Williamson MAXCUT algorithm [59], the polynomial-time approximation algorithm.

In all results, we use either the approximation ratio (best known or optimal solution vs those by our algorithms) or the best objective cut as performance metrics to compare our proposed scheme with other solvers. Specifically, given graph G and MAXCUT problem, the best objective cut achieved by method $\mathcal{A}$ is denoted as $C_{\mathcal{A}}$ while the optimal cut (if known) is denoted as C^* . The approximation ratio of method $\mathcal{A}$ is defined as $r_{\mathcal{A}} = \frac{C_{\mathcal{A}}}{C^*}$. As finding the optimal cut for large graphs is not always possible, we primarily use the approximation ratio metric for graphs with known optimal cuts.

G_{set} graphs Table 1 shows a comparison between the proposed reinforced schemes MLQAOA and exhaustive learning multilevel MAXCUT QAOA [9] with respect to the optimal solution. In the first column, we show the details of graphs (number of vertices and edges) used from the G_{set} dataset. In the second, third, and fourth columns, the approximation ratio of each method is recorded with the form: average approximation ratio / best approximation ratio. The final column shows the optimal cut (best objective value) for the MAXCUT problem for specific graphs. We observe that MLQAOA outperforms the exhaustive learning

multilevel MAXCUT QAOA with a much shorter runtime. The running times of Graph Learning MLQAOA and RQAOA-QIRO MLQAOA are comparable (Fig. 3), so while the quality of the latter is better, the gap with the former is not very significant.

Karloff graphs We analyze the performance of MLQAOA against the Goemans-Williamson MAXCUT algorithm for the Karloff graphs in Table 2. The first column gives the details of Karloff graphs (number of vertices and edges), while the second column shows the average and best cuts from the Goemans-Williamson algorithm. The third and fourth columns demonstrate our average and best approximation ratio and the last column shows the optimal cut. The MLQAOA demonstrates competitive performance against the Goemans-Williamson algorithm for this type of graph, reaching the optimal cut in many cases.

To provide a detailed insight into the performance of MLQAO on G_{set} dataset and Karloff dataset, figure 2 shows the boxplot of the graph learning MLQAOA and RQAOA-inspired QIRO MLQAOA. The boxes are drawn from the 25% percentile to the 75% percentile with a horizontal line drawn in the middle to denote the median. The x -axis represents the graph instances and the y -axis represents the approximation ratio. The RQAOA-inspired QIRO MLQAOA exhibits more consistent performance with less variance and achieves a slightly higher approximation ratio in most cases compared to graph-learning MLQAOA.

G_{set} continued We further compare our method with the QAOA-in-QAOA approach [44] for solving more graphs from the G_{set} dataset in table 3. Our method demonstrates comparable performance with classical solvers like a dualscaling SDP solver (DSDP) [60] or a physics-inspired graph neural network (PI-GNN) method [61], outperforming the QAOA-in-QAOA approach but yielding worse results than breakout local search (BLS) [62]. These three graphs are separated from the previous graphs from G_{set} due to the unavailability of the QAOA-in-QAOA code.

Larger graphs Finally, to demonstrate the robustness of our method, we sample 25 graphs from the SuiteSparse Matrix Collection [13] and the Network Repository [14] and present their results in Table 4. These graphs span different domains, sizes, and degrees, providing a comprehensive overview of our algorithm’s performance compared to a systematic implementation of MAXCUT and QUBO heuristics in MQLib [63]. Among the classical heuristics, we choose the best performing to the MAXCUT problem including “BURER2002” [64] (the best heuristic evaluated by different metrics from [63]), “DUARTE2005” [65], “GLOVER2010” [66] and “DESOUSA2013” [67]. The reinforcement of MLQAOA gives a comparable result with classical solvers, especially for huge graphs as shown in table 4. This holds a promise for hybrid-quantum algorithms to address other quadratic unconstrained binary optimization problems (QUBO) such as Maximum independent set or max- k satisfiability.

To conclude the experiments and numerical simulation, Figure 3 illustrates the runtime of our reinforced method for various graph sizes. This log-scale graph highlights the scalability and potential of our method when executed on quantum hardware. The execution time shown here is from simulation and can be significantly reduced when implemented on quantum devices.

6 Discussion

The main goal successfully achieved in this work was to break the variational quantum algorithm complexity barriers to tackle large-scale combinatorial optimization instances without losing the solution quality. As one can see from the results both Graph Learning MLQAOA and RQAOA-QIRO MLQAOA are highly scalable, and pretty similar to each other in terms of the solution quality (while RQAOA-QIRO MLQAOA is usually slightly better). They both often reach optimality (if it is known for the comparison) and are identical to the MAXCUT dedicated top heuristics. It is important to mention that no optimization of hyperparameters in this work has been done and it is clear that these results could be further improved. At the same time, we note that no comparison with generic solvers (such as Gurobi) has been presented because they are either way slower than MLQAOA or exhibit low quality. Below we discuss several important lessons learned and obstacles we encountered during this work.

Table 1: G_{set} graphs approximation ratio between three different methods: exhaustive learning MLQAOA, graph-learning MLQAOA, and RQAOA-inspired QIRO MLQAOA. The approximation ratio is recorded over 20 runs. The exhaustive MLQAOA [9] has not been executed multiple times due to the incomparable slowness.

$G(V , E)$	MLQAOA [9]	MLQAOA Graph-Learning	MLQAOA RQAOA-QIRO	Optimal Cut
$G_1(800, 19716)$	-/0.984	0.976/0.985	0.989/0.993	11624
$G_2(800, 19716)$	-/0.984	0.978/0.984	0.989/0.993	11620
$G_3(800, 19716)$	-/0.982	0.978/0.986	0.990/0.995	11622
$G_4(800, 19716)$	-/0.978	0.980/0.988	0.990/0.994	11646
$G_5(800, 19716)$	-/0.983	0.979/0.989	0.989/0.994	11631

Table 2: Karloff graphs approximation ratio between three different methods: Goemans-Williamson MAX CUT algorithm, graph learning MLQAOA, and RQAOA-inspired QIRO MLQAOA. The approximation ratio is recorded over 20 runs

$G(V , E)$	Classical	Quantum approach		
	GW [59]	MLQAOA Graph-Learning	MLQAOA RQAOA-QIRO	Optimal Cut
$K(252, 3150)$	0.881/0.925	0.984/ 1.0	0.997/1.0	2520
$K(252, 12600)$	0.940/0.941	0.981/0.997	0.997/1.0	7560
$K(924, 16632)$	0.879/0.913	0.991/ 1.0	0.994/1.0	13860
$K(924, 103950)$	0.912/0.922	0.988/0.999	0.995/1.0	69300
$K(3432, 84084)$	0.879/0.937	0.992/1.0	0.989/ 1.0	72072
$K(3432, 756756)$	0.897/0.927	0.969/1.0	0.968/ 1.0	540540

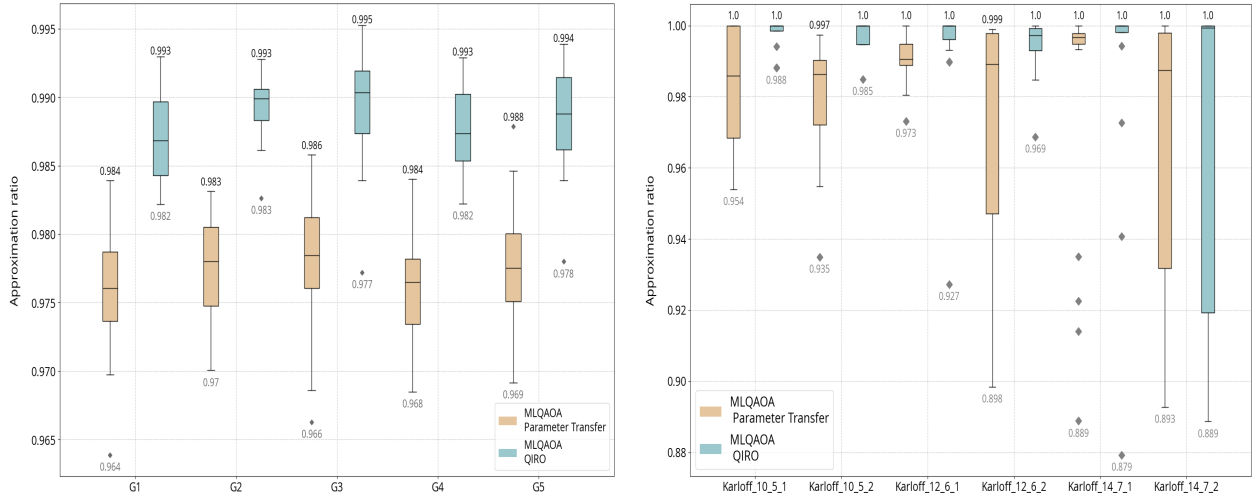


Figure 2: Approximation ratio of graph learning MLQAOA RQAOA-inspired QIRO MLQAOA on G_{set} and Karloff graph. The whisker lines are drawn up (down) to the largest (lowest) observed data point from the dataset that falls within the 1.5 interquartile range (IQR) value from the upper (lower) quartile. The upper (lower) notated approximation ratio of each box indicates the highest (lowest) approximation ratio over 20 run

Graph representation learning One unexpected result was a relatively low quality of the graph representation learning based on the Weisfeiler-Leman (W-L) graph isomorphism test (e.g., graph2vec algorithm)

Table 3: Extended G_{set} graphs approximation ratio between different quantum and classical methods, including graph-learning MLQAOA, RQAOA-inspired QIRO MLQAOA, and QAOA-in-QAOA. The best approximation ratio for both classical and quantum algorithms is recorded. The best results in their categories are in bold.

Graph	$ V $	$ E $	Quantum approach				Classical approach		
			MLQAOA Graph Learning	MLQAOA RQAOA-QIRO	QAOA ² $p = 1$	QAOA ² $p = 4$	DSDP [60]	PI GNN [61]	BLS [62]
G_{14}	800	4694	2994	3026	2593	2596	2922	3026	3064
G_{15}	800	4694	2977	3026	2596	2579	2938	2990	3050
G_{22}	2000	19990	13122	13174	10664	10559	12960	13181	13359

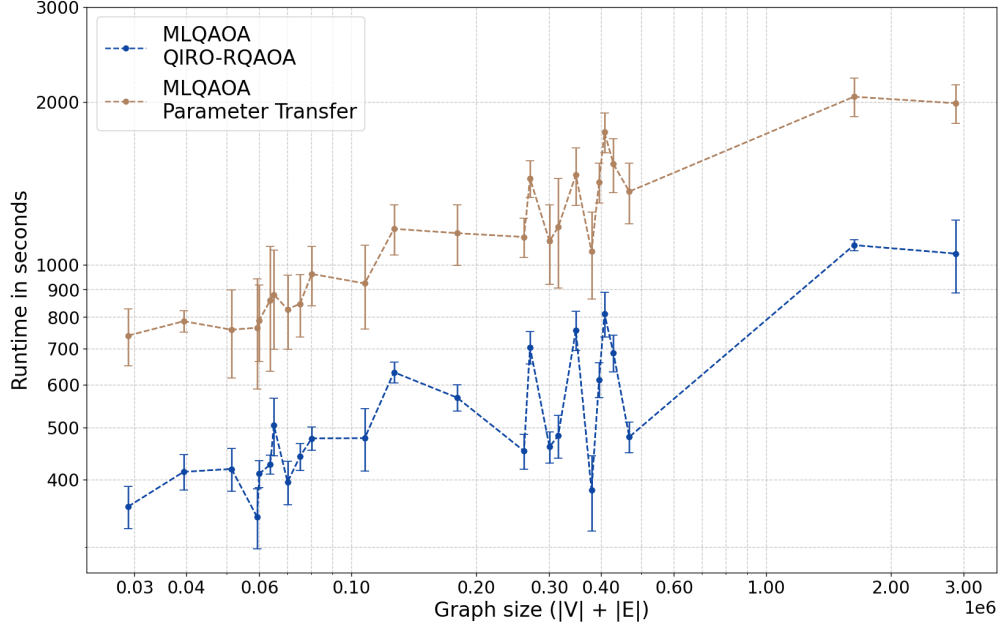


Figure 3: Average run time in seconds of graph learning MLQAOA and RQAOA-inspired QIRO MLQAOA on graphs from table 4 over 20 runs. The x -axis and y -axis are logarithmic scales with x -axis denoted the runtime and y -axis denoted the size of the graph calculated by $|V| + |E|$.

in comparison to the spectral learning technique. In the previous work [6], the QAOA parameter transferability based on the W-L test was more successful on the unweighted graphs than anything else. Although we observe the high quality and scalability of the spectral representation learning transferability, more investigation is required to build a weighted graph transferability model.

Coarsening scheme The current multilevel scheme was significantly simplified to understand the quantum-based refinement effects. However, similar to many multilevel solvers for graphs, this can be improved. Introducing advanced algebraic multigrid coarsening [1, 16] opens an opportunity to preserve the spectral properties of the original problem at all levels of coarseness much better. As a result, potentially fewer refinement steps are anticipated.

RQAOA-QIRO Using the idea of recursive QAOA for simplification rule leads to the QIRO scheme demonstrating excellent performance on MAXCUT. The QIRO was chosen as a baseline because it was consistently outperforming its competitors on small-scale graphs. However, similar to many other methods

Table 4: Best objective cut of graph-learning MLQAOA (MLQAOA GL) and RQAOA-inspired QIRO (MLQAOA QIRO) compared with classical heuristic algorithm “BURER2002” [64], “DUARTE2005” [65], “GLOVER2010” [66] and “DESOUSA2013” [67]. The graphs are drawn from SuiteSparse Matrix Collection [13] and the Network Repository [14].

Graph	$ V $	$ E $	Quantum approaches		Classical approaches			
			MLQAOA GL	MLQAOA QIRO	BURER02	DUARTE05	GLOVER10	DESOUSA13
soc-buzznet	101,163	2,763,066	2,069,342	2,070,569	2,071,445	2,071,612	2,071,427	1,390,645
c-72	84,064	395,811	311,299	311,623	296,238	303,000	289,878	158,740
c-71	76,638	468,096	390,832	391,333	391,007	390,937	360,192	198,915
soc-slashdot	70,068	358,647	277,664	278,124	277,675	278,445	276,640	182,254
c-68	64,810	315,408	250,405	250,547	242,676	245,123	233,846	127,692
dixmaanl	60,000	179,999	96,299	96,659	99,555	98,281	89,411	61,713
soc-brightkite	56,739	212,945	149,739	150,183	149,328	151,057	145,405	108,795
copter2	55,476	407,714	218,486	219,595	225,982	222,862	213,634	178,894
c-64	51,035	384,438	328,421	328,471	328,475	328,475	326,302	169,408
3dtube	45,330	1,629,474	1,033,982	1,044,174	1,065,693	1,066,489	1,066,489	797,620
c-62	41,731	300,537	258,698	258,805	258,759	258,802	245,026	132,373
c-59	41,282	260,909	219,481	219,618	218,368	219,193	216,117	112,342
shock-9	36,476	71,290	66,702	69,168	69,772	68,717	64,780	37,034
big_dual	30,269	44,929	41,658	41,933	43,280	42,489	39,801	23,551
rajat10	30,202	80,202	38,776	38,989	39,847	39,316	36,028	26,223
aug2dc	30,200	40,000	38,221	38,407	39,805	38,670	31,920	21,080
soc-epinions	26,588	100,120	68,934	69,282	69,496	69,850	67,541	51,705
dtoc	24,993	34,986	33,915	34,070	34,590	33,951	32,454	18,497
rajat09	24,482	64,982	31,514	31,588	32,334	31,874	29,136	21,305
aug3d	24,300	34,992	34,784	34,785	34,992	33,866	27,590	18,462
biplane-9	21,701	42,038	39,740	39,892	41,373	40,660	41,120	22,129
rajat08	19,362	51,362	24,913	24,972	25,581	25,183	23,228	16,968
ex3stal	16,782	34,7890	180,530	181,141	183,535	183,141	180,975	168,116
rajat07	14,842	39,342	19,078	19,144	19,575	19,303	17,749	13,115
rajat06	10,922	28,922	14,034	14,083	14,418	14,211	13,203	9732

with the goal of creating a more compact circuit, this method encounters scalability challenges in terms of both time and space complexity when applied to large instance graphs. An interesting observation emerges regarding the RQAOA-QIRO approach, the approach produces consistent results for large graphs, indicating its tendency to converge to local optima. Conversely, the graph learning approach yields more fluctuated results with a higher degree of randomness, leading to a better chance of escaping local points and heading toward global minimum. On average, the difference between graph learning MLQAOA and RQAOA-QIRO MLQAOA is only 0.7% on the set of larger graphs, i.e., they are both highly competitive.

Acknowledgment

This work was supported with funding from the Defense Advanced Research Projects Agency (DARPA) under the ONISQ program.

References

- [1] Achi Brandt and Dorit Ron. Multigrid solvers and multilevel optimization strategies. *Multilevel optimization in VLSICAD*, pages 1–69, 2003.

- [2] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, August 2021.
- [3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.
- [4] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermann Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, 94(1), February 2022.
- [5] Gian Giacomo Guerreschi and A. Y. Matsuura. Qaoa for max-cut requires hundreds of qubits for quantum speed-up. *Scientific Reports*, 9:6903, 2019.
- [6] Jose Falla, Quinn Langfitt, Yuri Alexeev, and Ilya Safro. Graph representation learning for parameter transferability in quantum approximate optimization algorithm. *arXiv preprint arXiv:2401.06655*, 2024.
- [7] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Christian FA Negre, Ilya Safro, Susan M Mniszewski, and Yuri Alexeev. A hybrid approach for solving optimization problems on small quantum computers. *Computer*, 52(6):18–26, 2019.
- [8] Hayato Ushijima-Mwesigwa, Ruslan Shaydulin, Christian FA Negre, Susan M Mniszewski, Yuri Alexeev, and Ilya Safro. Multilevel combinatorial optimization across quantum architectures. *ACM Transactions on Quantum Computing*, 2(1):1–29, 2021.
- [9] Anthony Angone, Xiaoyuan Liu, Ruslan Shaydulin, and Ilya Safro. Hybrid quantum-classical multilevel approach for maximum cuts on graphs. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2023.
- [10] Jernej Rudi Finžgar, Aron Kerschbaumer, Martin JA Schuetz, Christian B Mendl, and Helmut G Katzgraber. Quantum-informed recursive optimization algorithms. *arXiv preprint arXiv:2308.13607*, 2023.
- [11] Yuan Ye. Gset - a suite-style benchmark for graph processing systems. <https://web.stanford.edu/~yyye/yyye/Gset/>, 2003.
- [12] Howard Karloff. How good is the Goemans-Williamson MAX CUT algorithm? In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 427–434, 1996.
- [13] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.
- [14] Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Mar. 2015.
- [15] Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, 1974.
- [16] Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. In *International Symposium on Experimental Algorithms*, pages 369–380. Springer, 2012.
- [17] Yifan Hu and Lei Shi. Visualizing large graphs. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(2):115–136, 2015.
- [18] Ilya Safro, Dorit Ron, and Achi Brandt. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, 60(1):24–41, 2006.

- [19] Varsha Chauhan, Alexander Gutfraind, and Ilya Safro. Multiscale planar graph generation. *Applied Network Science*, 4:1–28, 2019.
- [20] Jie Chen and Ilya Safro. Algebraic distance on graphs. *SIAM Journal on Scientific Computing*, 33(6):3468–3490, 2011.
- [21] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, September 2018.
- [22] Lili Wang, Chenghan Huang, Weicheng Ma, Xinyuan Cao, and Soroush Vosoughi. Graph Embedding via Diffusion-Wavelets-Based Node Feature Distribution Characterization. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3478–3482. ACM, October 2021.
- [23] Chen Cai and Yusu Wang. A simple yet effective baseline for non-attributed graph classification, May 2022. arXiv:1811.03508 [cs, stat].
- [24] Alexis Galland and Marc Lelarge. Invariant embedding for graph classification. In *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data*, 2019.
- [25] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning Distributed Representations of Graphs, July 2017. arXiv:1707.05005 [cs].
- [26] Nino Shervashidze. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 2011.
- [27] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Learning Theory and Kernel Machines*, pages 129–143, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [28] K.M. Borgwardt and H. Kriege. Shortest-Path Kernels on Graphs. In *Fifth IEEE International Conference on Data Mining (ICDM’05)*, pages 74–81, Houston, TX, USA, 2005. IEEE.
- [29] Pinar Yanardag and S.V.N. Vishwanathan. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, Sydney NSW Australia, August 2015. ACM.
- [30] Hong Chen and Hisashi Koga. Gl2vec: Graph embedding enriched by line graphs with edge features. In Tom Gedeon, Kok Wai Wong, and Minh Lee, editors, *Neural Information Processing*, pages 3–14, Cham, 2019. Springer International Publishing.
- [31] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [32] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.
- [33] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- [34] Tao Zhang, Hao-Ran Shan, and Max Little. Causal graphsage: A robust graph method for classification based on causal sampling. *Pattern Recognition*, 128:108696, 08 2022.
- [35] N De Lara and E Pineau. A simple baseline algorithm for graph classification. *arXiv preprint arXiv:1810.09155*, pages 1–7, 2018.

- [36] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. Netlsd: Hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2347–2356, New York, NY, USA, 2018. Association for Computing Machinery.
- [37] Saurabh Verma and Zhi-Li Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [38] Alexey Galda, Eesh Gupta, Jose Falla, Xiaoyuan Liu, Danylo Lykov, Yuri Alexeev, and Ilya Safro. Similarity-based parameter transferability in the quantum approximate optimization algorithm. *Frontiers in Quantum Science and Technology*, 2, 2023.
- [39] Daniel J. Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, June 2021.
- [40] Filip B Maciejewski, Jacob Biamonte, Stuart Hadfield, and Davide Venturelli. Improving quantum approximate optimization by noise-directed adaptive remapping. *arXiv preprint arXiv:2404.01412*, 2024.
- [41] Trong Duong, Sang T Truong, Minh Pham, Bao Bach, and June-Koo Rhee. Quantum neural architecture search with quantum circuits metric and bayesian optimization. In *ICML 2022 2nd AI for Science Workshop*, 2022.
- [42] Ruslan Shaydulin, Stuart Hadfield, Tad Hogg, and Ilya Safro. Classical symmetries and the quantum approximate optimization algorithm. *Quantum Information Processing*, 20(11):359, 2021.
- [43] Moises Ponce, Rebekah Herrman, Phillip C Lotshaw, Sarah Powers, George Siopsis, Travis Humble, and James Ostrowski. Graph decomposition techniques for solving combinatorial optimization problems with variational quantum algorithms. *arXiv preprint arXiv:2306.00494*, 2023.
- [44] Zeqiao Zhou, Yuxuan Du, Xinmei Tian, and Dacheng Tao. Qaoa-in-qaoa: Solving large-scale maxcut problems on small quantum machines. *Phys. Rev. Appl.*, 19:024027, Feb 2023.
- [45] Wei Tang and Margaret Martonosi. Cutting quantum circuits to run on quantum and classical platforms. *arXiv preprint arXiv:2205.05836*, 2022.
- [46] Kaitlin N. Smith, Michael A. Perlin, Pranav Gokhale, Paige Frederick, David Owusu-Antwi, Richard Rines, Victory Omole, and Frederic Chong. Clifford-based circuit cutting for quantum simulation. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.
- [47] Eric R. Anschuetz and Bobak T. Kiani. Beyond Barren Plateaus: Quantum Variational Algorithms Are Swamped With Traps. *Nature Communications*, 13(1), 2022.
- [48] Samson Wang, Enrico Fontana, M. Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications*, 12(1):6961, November 2021.
- [49] Ruslan Shaydulin and Yuri Alexeev. Evaluating quantum approximate optimization algorithm: A case study. In *2019 tenth international green and sustainable computing conference (IGSC)*, pages 1–6. IEEE, 2019.
- [50] 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.

- [51] Christo Meriwether Keller, Stephan Eidenbenz, Andreas Bärtschi, Daniel O'Malley, John Golden, and Satyajayant Misra. Hierarchical multigrid ansatz for variational quantum algorithms. *arXiv preprint arXiv:2312.15048*, 2023.
- [52] 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.
- [53] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021.
- [54] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to variational quantum optimization from symmetry protection. *Physical Review Letters*, 125(26), December 2020.
- [55] Dorit Ron, Ilya Safro, and Achi Brandt. Relaxation-based coarsening and multiscale graph organization. *Multiscale Modeling & Simulation*, 9(1):407–423, 2011.
- [56] Iain Dunning, Swati Gupta, and John Silberholz. What works best when? a systematic evaluation of heuristics for max-cut and qubo. *INFORMS Journal on Computing*, 30(3), 2018.
- [57] Cirq Developers. Cirq. *version v0*, 12, 2021.
- [58] Asier Ozaeta, Wim van Dam, and Peter L McMahon. Expectation values from the single-layer quantum approximate optimization algorithm on ising problems. *Quantum Science and Technology*, 7(4):045036, 2022.
- [59] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, nov 1995.
- [60] Steven J. Benson, Yinyu Ye, and Xiong Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM Journal on Optimization*, 10(2):443–461, 2000.
- [61] Martin J. A. Schuetz, J. Kyle Brubaker, and Helmut G. Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.
- [62] Una Benlic and Jin-Kao Hao. Breakout local search for the max-cutproblem. *Engineering Applications of Artificial Intelligence*, 26(3):1162–1173, 2013.
- [63] Iain Dunning, Swati Gupta, and John Silberholz. What works best when? a systematic evaluation of heuristics for max-cut and QUBO. *INFORMS Journal on Computing*, 30(3), 2018.
- [64] Samuel Burer, Renato Monteiro, and Yin Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization*, 12, 07 2001.
- [65] Abraham Duarte, Angel Sanchez, Felipe Fernández, and Raúl Cabido. A low-level hybridization between memetic algorithm and vns for the max-cut problem. pages 999–1006, 06 2005.
- [66] Fred Glover, Zhipeng Lü, and Jin-Kao Hao. Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR*, 8:239–253, 2010.
- [67] Samuel de Sousa, Yll Haxhimusa, and Walter G. Kropatsch. Estimation of distribution algorithm for the max-cut problem. In *Graph-Based Representations in Pattern Recognition*, pages 244–253, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.