QAOA-PCA: Enhancing Efficiency in the Quantum Approximate Optimization Algorithm via Principal Component Analysis

Owain Parry
University of Sheffield
UK

Phil McMinn
University of Sheffield
UK

ABSTRACT

The Quantum Approximate Optimization Algorithm (QAOA) is a promising variational algorithm for solving combinatorial optimization problems on near-term devices. However, as the number of layers in a QAOA circuit increases, which is correlated with the quality of the solution, the number of parameters to optimize grows linearly. This results in more iterations required by the classical optimizer, which results in an increasing computational burden as more circuit executions are needed. To mitigate this issue, we introduce QAOA-PCA, a novel reparameterization technique that employs Principal Component Analysis (PCA) to reduce the dimensionality of the QAOA parameter space. By extracting principal components from optimized parameters of smaller problem instances, QAOA-PCA facilitates efficient optimization with fewer parameters on larger instances. Our empirical evaluation on the prominent MaxCut problem demonstrates that QAOA-PCA consistently requires fewer iterations than standard QAOA, achieving substantial efficiency gains. While this comes at the cost of a slight reduction in approximation ratio compared to QAOA with the same number of layers, QAOA-PCA almost always outperforms standard QAOA when matched by parameter count. QAOA-PCA strikes a favorable balance between efficiency and performance, reducing optimization overhead without significantly compromising solution quality.

CCS CONCEPTS

• **Computer systems organization → Quantum computing**.

KEYWORDS

Quantum Computing, QAOA, PCA

ACM Reference Format:
Owain Parry and Phil McMinn. 2025. QAOA-PCA: Enhancing Efficiency in the Quantum Approximate Optimization Algorithm via Principal Component Analysis . In *Proceedings of The 2nd International Workshop on Empirical Studies for Quantum Software Engineering (E-QSE 2025).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

The authors are supported by the EPSRC grant "Test FLARE" (EP/X024539/1) and the Robust and Reliable Quantum Computing Programme (RoaRQ) .

E-QSE 2025, 17–20 June, 2025 , Istanbul, Türkiye

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Quantum computing leverages quantum mechanical phenomena to perform computations that are intractable on classical systems [18, 20]. One prominent application is combinatorial optimization, where the Quantum Approximate Optimization Algorithm (QAOA) is a promising approach to tackle problems like MaxCut [8]. The goal in MaxCut is to partition a graph's vertices to maximize the number of edges crossing the partition in the unweighted case, or to maximize the total weight of these edges in the weighted case. It finds important applications in diverse fields [26].

QAOA operates by applying p layers of alternating cost and mixing Hamiltonians parameterized by two angles, with each layer increasing solution quality [10]. A major challenge in the practical deployment of QAOA is the optimization of its $2p$ parameters. This involves a classical-quantum loop where a classical optimizer proposes parameter values used to execute the QAOA circuit on a quantum computer to evaluate the cost function at each iteration. This becomes more expensive as the number of layers, and thus parameters, increases. The burden of repeated circuit executions, and the non-convex cost landscape [14], contributes to the quantum software engineering challenge of scaling QAOA to larger problem instances (e.g., graphs with more vertices), especially for weighted MaxCut due to the proliferation of local minima [25].

Recent research has identified parameter concentration, where optimal parameters cluster around specific values, and transferability, where parameters optimized for one instance are applicable to similar instances. This suggests redundancy in the parameter space, implying that reduced dimensionality could improve efficiency without sacrificing performance [4, 6, 9, 10, 16, 23, 24, 27]. Building on this, we propose QAOA-PCA, a novel reparameterization technique to improve the efficiency of QAOA. Rather than optimizing the full QAOA parameter set, QAOA-PCA leverages Principal Component Analysis (PCA) [2] to extract key features from the optimal parameters of a training set of smaller instances. These principal components form a lower-dimensional subspace, resulting in fewer parameters to optimize when tackling larger instances, leading to faster convergence and fewer circuit executions.

We conducted an extensive empirical evaluation of QAOA-PCA supported by robust statistical testing. We collected all connected, non-isomorphic graphs on 5–7 vertices to form an unweighted training set of 986 graphs. To create a weighted training set, we assigned random edge weights to these. We ran QAOA with 2, 4, and 8 layers on all graphs in both training sets. For these six configurations of circuit depth and training set, we applied PCA to extract the principal components from the optimized parameters. We sampled 1,000 8-vertex weighted graphs to form an evaluation set. Instead of optimizing the full QAOA parameters for these graphs, we optimized the coefficients of the most important principal components

as substitute parameters. We did this for various numbers of components for each of the six training configurations. We ensured at least a 50% reduction in the number of parameters compared to standard QAOA with the same number of layers, which we used as a baseline. As another baseline, we used standard QAOA with the same number of parameters as components and thus fewer layers.

Our results demonstrate that QAOA-PCA requires significantly fewer iterations, and thus circuit executions, compared to standard QAOA of the same depth. With the same number of parameters, QAOA-PCA generally requires slightly fewer. QAOA-PCA achieves only a slightly reduced approximation ratio [22] compared to standard QAOA with the same number of layers. It almost always outperforms standard QAOA when matched by parameter count. By broadly preserving the advantage of deeper QAOA circuits while avoiding excessive optimization overhead, QAOA-PCA achieves a favorable trade-off between efficiency and performance.

The main contributions of this study are as follows:

Contribution 1: QAOA-PCA. We introduce QAOA-PCA, a novel reparameterization technique that leverages PCA to reduce the dimensionality of the QAOA parameter space, improving efficiency.

Contribution 2: Empirical Evaluation. We provide a rigorous empirical evaluation of QAOA-PCA compared to standard QAOA.

Contribution 3: Results. We demonstrate that QAOA-PCA achieves a favorable trade-off between performance and efficiency.

2 BACKGROUND

The Quantum Approximate Optimization Algorithm (QAOA) is a quantum-classical algorithm for solving combinatorial optimization problems [8]. It employs a parameterized circuit of depth p alternating between unitaries generated by the cost Hamiltonian H_C (encoding the problem) and the mixing Hamiltonian H_B (facilitating computational state transitions). The resulting state is

$$|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \prod_{i=1}^{p} e^{-i\beta_i H_B} e^{-i\gamma_i H_C} |+\rangle^{\otimes n},$$

where $|+\rangle^{\otimes n}$ denotes a uniform superposition over all computational basis states, and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ are the variational parameters. These parameters are optimized classically to minimize the expectation value of the cost Hamiltonian:

$$(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*) = \arg\min_{\boldsymbol{\gamma}, \boldsymbol{\beta}} \langle \psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) | H_C | \psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle.$$

MaxCut aims to cut (partition) the vertices of a graph $G = (V, E)$ into two sets to maximize the cut value, i.e., the total weight of the edges between them. A cut is represented by $\boldsymbol{z} = (z_1, \ldots, z_n)$, with $z_i \in \{+1, -1\}$ indicating the set of vertex i. The cut value of $\mathbf{z}$ is

$$C(\mathbf{z}) = \sum_{(i,j) \in E} w_{ij} \frac{1 - z_i z_j}{2},$$

where w_{ij} is the weight of edge (i, j), or 1 in the unweighted case. The ground state of H_C corresponds to the optimal cut. QAOA performance is measured by the approximation ratio

$$r(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \frac{\langle \psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) | H_C | \psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle}{C_{\min}},$$

where $C_{\min}$ is the minimum energy of H_C, corresponding to the maximum cut value. This ratio quantifies how close the obtained solution is to the optimal one: a value near 1 indicates near-optimal performance, while a lower value reflects a suboptimal cut.

Trotterized Quantum Annealing (TQA) provides a heuristic for initializing QAOA parameters. In quantum annealing, the system is gradually driven from H_B to H_C over a total time T. Discretizing this evolution into p steps of duration $\Delta t = T/p$ via the Suzuki-Trotter decomposition produces the initial QAOA parameters

$$\gamma_i = \frac{i}{p} \Delta t, \quad \beta_i = \left(1 - \frac{i}{p}\right) \Delta t, \quad i = 1, \ldots, p.$$

An appropriate choice of Δt can guide the optimization towards the global minimum while avoiding poor local minima [21].

Principal Component Analysis (PCA) reduces the dimensionality of a dataset while retaining most of its variance [2]. Given a $n \times m$ mean-centered data matrix X, the covariance matrix is $\Sigma = \frac{1}{n-1} X^\top X$. The principal components are the eigenvectors associated with the largest eigenvalues of Σ.

3 APPROACH

QAOA shows promise in addressing combinatorial optimization problems [8]. Its application is hindered by the need to optimize numerous parameters as the number of layers increases. We introduce QAOA-PCA to mitigate this, a reparameterization technique leveraging PCA to exploit redundancies in QAOA parameters.

Recent research has identified parameter concentration and transferability. Concentration is the tendency of optimal QAOA parameters to cluster around specific values as the problem size (e.g., vertex count) increases, indicating that parameters optimized on smaller problem instances can be effective on larger ones [4, 6, 9, 27]. Transferability denotes the applicability of parameters optimized for one instance to different instances, which is predictable from graph properties when solving MaxCut [9, 10, 16, 23]. These insights imply that the effective dimensionality of the parameter space is lower than its nominal size [24], presenting an opportunity to reduce the number of parameters requiring optimization.

QAOA-PCA capitalizes on concentration and transferability by employing PCA to identify the principal components of optimal QAOA parameters from a training set of smaller problem instances. By efficiently narrowing the search space, QAOA-PCA reduces computational cost while maintaining high solution quality when addressing larger instances. The process involves four stages:

Stage 1: Data Collection. Gather optimized QAOA parameters from a diverse set of smaller problem instances.

Stage 2: Principal Component Analysis. Apply PCA to the optimized parameters to identify the principal components that capture the most significant variations in the parameter space.

Stage 3: Reparameterization. Represent the QAOA parameters of new, larger problem instances as linear combinations of the identified principal components. This approach reconstructs the full parameter vector from a much smaller set of variables.

Stage 4: Optimization. Optimize only the coefficients of these principal components, thereby reducing the dimensionality of the optimization problem and facilitating convergence.

4 EVALUATION

This section details the methodology of our empirical evaluation aimed at addressing our research questions regarding QAOA-PCA:

RQ1: Efficiency. How does the number of optimizer iterations required by QAOA-PCA compare with that of standard QAOA?
RQ2: Performance. How does the approximation ratio achieved by QAOA-PCA compare with that of standard QAOA?

We created a Python script to automate all aspects of our empirical evaluation. We make this available in our replication package [1].

Training. We collected all connected, undirected, non-isomorphic graphs on 5, 6, and 7 vertices to form an unweighted training set of 986 graphs [15]. Our script applied random edge weights to every graph to form a weighted training set. It ran QAOA with 2, 4, and 8 layers on all graphs in both training sets using ideal quantum simulation provided by Qiskit [13], initializing the parameters using TQA, and optimizing them using COBYLA [19]. For each combination of circuit depth and graph, our script repeated the process for five values of TQA time step Δt (0.1, 0.3, 0.5, 0.7, 0.9) and retained only the results of the run with the greatest approximation ratio. For the six configurations of training set and circuit depth p, our script applied PCA to extract the principal components from the $986 \times 2p$ matrix of the optimized parameters of every graph.

Evaluation. Our script randomly sampled 1,000 connected, undirected, non-isomorphic graphs on 8 vertices and applied random edge weights to form an evaluation set. It ran QAOA on these as before with 1, 2, 4, and 8 layers. This represents standard QAOA. For each of the six sets of principal components that it extracted earlier, our script ran QAOA on the evaluation graphs again. This time, it optimized the coefficients of the most important principal components as substitute parameters, trying five random initializations. This represents QAOA-PCA with 2, 4, and 8 layers as trained on the optimal parameters of both unweighted and weighted graphs. In each case, our script repeated the process for different numbers of principal components, ensuring at least a 50% reduction in the number of parameters compared to standard QAOA with the same number of layers. For QAOA-PCA with 2 layers, it tried 2 principal components. For 4 layers, it tried 2 and 4 components. For 8 layers, it tried 2, 4, and 8. This resulted in 12 configurations of QAOA-PCA.

Comparison. We compared each configuration of QAOA-PCA against two baselines. The first was against standard QAOA with the same number of layers and thus twice as many parameters. The second was against standard QAOA with the same number of parameters as principal components and thus half as many layers. For each comparison, our script performed two-tailed Wilcoxon signed-rank tests with rank-biserial correlation (RBC) effect sizes. It did this regarding the number of optimizer iterations and the approximation ratio to compare both the efficiency and performance respectively of QAOA-PCA to standard QAOA. To calculate approximation ratio, our script obtained $C_{\min}$ by brute-force.

Threats. Our results may not reflect the behavior of QAOA-PCA on real quantum computers due to our use of ideal simulation. However, we also used it for the standard QAOA baseline, so the comparison is fair. It is possible that our script did not converge on the globally optimal set of parameters when performing QAOA. We mitigated this risk by trying multiple initializations, which increases the probability. Our results could be biased by our choice of optimizer, COBYLA. We mitigated this by selecting an optimizer that is popular and well-regarded in this domain [7, 11, 12].

5 RESULTS

For the 12 configurations of QAOA-PCA, Table 1 gives the training set, the number of layers, and the number of principal components that determines the number of parameters. It gives the median of either the number of iterations (RQ1) or the approximation ratio (RQ2) attained by QAOA-PCA over the evaluation set, followed by the results of the comparisons against both standard QAOA baselines. It gives the median attained by the both baselines, and the p-value and RBC effect sizes. A p-value less than 0.01 indicates that we can reject the null hypothesis of no difference at the 99% confidence interval. An RBC of -1 indicates QAOA-PCA always has a lower value, 1 indicates the opposite, and 0 indicates no difference.

Each subfigure of Figure 1 corresponds to a row in Table 1 as indicated by the captions. They illustrate the distribution of the number of iterations (x-axis) and the approximation ratio (y-axis) across the evaluation set for QAOA-PCA (blue), for standard QAOA with the same number of layers (orange), and for standard QAOA with the same number of parameters (green).

Results for RQ1: Efficiency. Table 1 shows that the p-values for the comparisons between the number of iterations required by QAOA-PCA and standard QAOA with the same number of layers are significant (<0.01) for every configuration. The effect sizes are all -1, indicating that QAOA-PCA always requires fewer iterations. The magnitude of this is shown by the large differences between the medians in favor of QAOA-PCA. It is clear in every subfigure of Figure 1 as the large gap along the x-axis between the data points for QAOA-PCA and standard QAOA with the same number of layers.

With respect to standard QAOA with the same number of parameters, all but one of the p-values is significant. While the majority of the effect sizes are negative, there is significant variance in magnitude. This is illustrated in Figure 1, where the data points for QAOA-PCA are generally more in line along the x-axis with those of standard QAOA with the same number of parameters.

Conclusion for RQ1. QAOA-PCA consistently requires far fewer iterations than standard QAOA with the same layers. With the same number of parameters, the improvement is less pronounced.

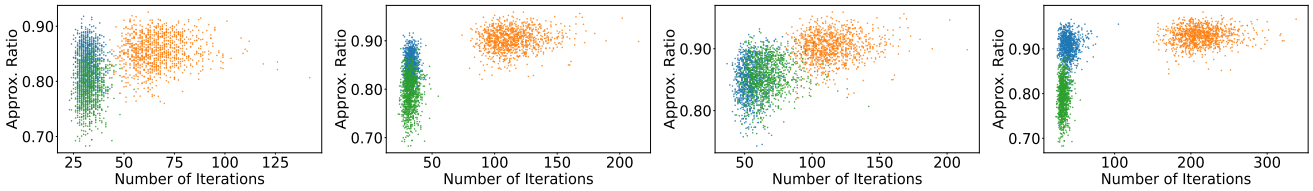
Results for RQ2: Performance. The difference between the approximation ratio of QAOA-PCA and standard QAOA with the same number of layers is always significant. The negative effect sizes with varying magnitudes indicate QAOA-PCA often performs worse. However, comparing medians, and the y-axis positions of the data points for QAOA-PCA and standard QAOA with the same number of layers, highlights that the difference is generally minor.

As for standard QAOA with the same number of parameters, the difference in performance is once again statistically significant for every configuration. In this case, all but one of the effect sizes are positive and are generally close to 1. This indicates that QAOA-PCA very frequently performs better, as illustrated by the generally higher y-axis positions of the data points for QAOA-PCA compared to those for standard QAOA with the same number of parameters.

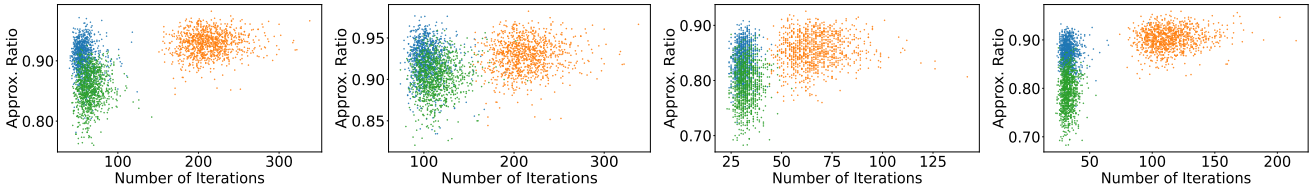
Conclusion for RQ2. QAOA-PCA generally achieves slightly lower approximation ratios than standard QAOA with the same layers. However, with the same number of parameters, QAOA-PCA almost always outperforms standard QAOA.

Table 1: Comparison of QAOA-PCA and standard QAOA across 12 configurations. For each, the table gives the median (Med.) number of optimizer iterations (RQ1) and approximation ratio (RQ2) for QAOA-PCA and the two standard QAOA baselines (Same # Layers, Same # Param.). Wilcoxon test results include p-values (P-Val.) and rank-biserial correlation (RBC) effect sizes.

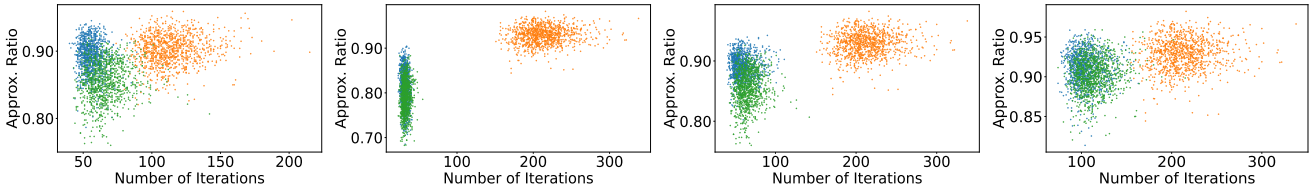
			Number of Iterations (RQ1)							Approximation Ratio (RQ2)						
				Same # Layers			Same # Param.				Same # Layers			Same # Param.		
Training Set	# Layers	# Param.	Med.	Med.	P-Val.	RBC	Med.	P-Val.	RBC	Med.	Med.	P-Val.	RBC	Med.	P-Val.	RBC
Unweighted	2	2	32	65	<0.01	-1.00	32	<0.01	-0.14	0.83	0.85	<0.01	-0.89	0.79	<0.01	1.00
Unweighted	4	2	33	112	<0.01	-1.00	32	<0.01	0.11	0.85	0.90	<0.01	-1.00	0.79	<0.01	1.00
Unweighted	4	4	54	112	<0.01	-1.00	65	<0.01	-0.73	0.85	0.90	<0.01	-1.00	0.85	<0.01	-0.26
Unweighted	8	2	39	210	<0.01	-1.00	32	<0.01	0.78	0.91	0.93	<0.01	-0.97	0.79	<0.01	1.00
Unweighted	8	4	57	210	<0.01	-1.00	65	<0.01	-0.57	0.91	0.93	<0.01	-0.96	0.85	<0.01	0.99
Unweighted	8	8	101	210	<0.01	-1.00	112	<0.01	-0.55	0.93	0.93	<0.01	-0.30	0.90	<0.01	0.94
Weighted	2	2	31	65	<0.01	-1.00	32	<0.01	-0.32	0.84	0.85	<0.01	-0.77	0.79	<0.01	1.00
Weighted	4	2	32	112	<0.01	-1.00	32	0.07	0.07	0.87	0.90	<0.01	-0.96	0.79	<0.01	1.00
Weighted	4	4	55	112	<0.01	-1.00	65	<0.01	-0.76	0.89	0.90	<0.01	-0.38	0.85	<0.01	0.99
Weighted	8	2	32	210	<0.01	-1.00	32	<0.01	-0.20	0.82	0.93	<0.01	-1.00	0.79	<0.01	0.88
Weighted	8	4	57	210	<0.01	-1.00	65	<0.01	-0.46	0.89	0.93	<0.01	-1.00	0.85	<0.01	1.00
Weighted	8	8	103	210	<0.01	-1.00	112	<0.01	-0.49	0.91	0.93	<0.01	-0.95	0.90	<0.01	0.65



(a) Unweighted, 2 Layers, 2 Param. (b) Unweighted, 4 Layers, 2 Param. (c) Unweighted, 4 Layers, 4 Param. (d) Unweighted, 8 Layers, 2 Param.



(e) Unweighted, 8 Layers, 4 Param. (f) Unweighted, 8 Layers, 8 Param. (g) Weighted, 2 Layers, 2 Param. (h) Weighted, 4 Layers, 2 Param.



(i) Weighted, 4 Layers, 4 Param. (j) Weighted, 8 Layers, 2 Param. (k) Weighted, 8 Layers, 4 Param. (l) Weighted, 8 Layers, 8 Param.

Figure 1: Distributions of the number of iterations (x-axis) and approximation ratios (y-axis) for QAOA-PCA (blue), standard QAOA with the same number of layers (orange), and standard QAOA with the same number of parameters (green).

Discussion. Our results demonstrate that QAOA-PCA consistently requires fewer iterations than standard QAOA, achieving a substantial efficiency gain. While this comes at the cost of a slight reduction in approximation ratio compared to QAOA with the same number of layers, QAOA-PCA often outperforms standard QAOA when matched by parameter count. This suggests that QAOA-PCA strikes a favorable balance between efficiency and performance, reducing optimization overhead without significantly compromising solution quality. This is illustrated in Figure 1, where the data points for QAOA-PCA generally occupy the upper-left of each subfigure, which represents the optimum region. Our results do not indicate any obvious differences between training on unweighted versus

weighted graphs. This may seem surprising given the complexity of weighted MaxCut, however, previous research demonstrated parameter transferability between unweighted and weighted graphs [23].

6 RELATED WORK

Acampora et al. proposed a clustering approach that assigns fixed parameters to new problem instances based on their similarity to previously addressed instances from a training set [3]. Along a similar vein, Moussa et al. investigated unsupervised machine learning techniques, particularly clustering, to set QAOA parameters for MaxCut without explicit optimization [17]. While this removes the need for classical optimization, it relies on the assumption that new instances closely resemble those in the training set, with no mechanism for further refinement. Unlike these approaches, which trade all such adaptability for efficiency, QAOA-PCA retains some optimization capability while still reducing computational cost. Instead of assigning parameters from a set of clusters, QAOA-PCA applies PCA, another technique in unsupervised machine learning, to identify a lower-dimensional representation of QAOA parameters. It reduces the number of variables that require optimization while still allowing adjustments for diverse unseen instances.

Amosy et al. introduced a neural network-based method for initializing QAOA parameters, which predicts parameters based on previously optimized instances with the aim of eliminating further optimization [5]. Their approach is effective for the problem sizes used in training, but their study does not evaluate generalizability to larger instances. QAOA-PCA retains the ability to refine parameters after initialization while reducing the number of parameters, enabling it to efficiently adapt to larger instances. Additionally, it provides direct control over the parameter count to adjust the balance between efficiency and performance. These differences make QAOA-PCA a viable alternative when full optimization is too costly but entirely eliminating optimization is undesirable.

7 CONCLUSION AND FUTURE WORK

Our empirical evaluation demonstrates that QAOA-PCA substantially reduces the number of optimizer iterations. Compared to standard QAOA, it consistently improves efficiency without significantly compromising performance. These results highlight the feasibility of QAOA-PCA as a promising approach in quantum software engineering for scaling QAOA to larger problem instances.

Future Work. We aim to develop an adaptive strategy in which QAOA-PCA begins with a small number of principal components and expands dynamically when optimization progress stagnates. This approach should reduce initial computational overhead while allowing for increased expressivity when necessary, helping the optimizer escape plateaus and refine solutions more effectively. We also plan to conduct a broader empirical evaluation by applying QAOA-PCA to larger graphs and additional problems. We will evaluate a greater range of circuit depths, training set configurations, and numbers of components to more rigorously determine the conditions under which QAOA-PCA provides the best trade-off between efficiency and performance. Most critically, we will evaluate QAOA-PCA against noisy quantum simulators and run experiments on real quantum hardware. Finally, we intend to investigate alternate initialization schemes for QAOA-PCA. In our empirical evaluation, we relied on random initialization. We will evaluate the applicability of prior machine learning-based approaches [5] to produce a hybrid technique that benefits from both parameter reduction and data-driven starting points in the optimization process.

REFERENCES

[1] 2025. Replication Package, https://doi.org/10.5281/zenodo.15269564.
[2] H. Abdi and L. J Williams. 2010. Principal Component Analysis. *Wiley Interdisciplinary Reviews: Computational Statistics* (2010).
[3] G. Acampora, A. Chiatto, and A. Vitiello. 2023. Fuzzy Clustering for QAOA Complexity Reduction. In *Proc. FUZZ.* 1–7.
[4] V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte. 2021. Parameter Concentrations in Quantum Approximate Optimization. *Physical Review A* (2021).
[5] O. Amosy, T. Danzig, O. Lev, E. Porat, G. Chechik, and A. Makmal. 2024. Iteration-Free Quantum Approximate Optimization Algorithm Using Neural Networks. *Quantum Machine Intelligence* (2024).
[6] F. G. S. L. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven. 2018. For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances. *arXiv preprint arXiv:1812.04170* (2018).
[7] C. Campbell and E. Dahl. 2022. QAOA of the Highest Order. In *Proc. ICSA-C.* 141–146.
[8] E. Farhi, J. Goldstone, and S. Gutmann. 2014. A Quantum Approximate Optimization Algorithm. *arXiv preprint arXiv:1411.4028* (2014).
[9] A. Galda, E. Gupta, J. Falla, X. Liu, D. Lykov, Y. Alexeev, and I. Safro. 2023. Similarity-Based Parameter Transferability in the Quantum Approximate Optimization Algorithm. *Frontiers in Quantum Science and Technology* (2023).
[10] A. Galda, X. Liu, D. Lykov, Y. Alexeev, and I. Safro. 2021. Transferability of Optimal QAOA Parameters Between Random Graphs. In *Proc. QCE.* 171–180.
[11] T. Hao, Z. He, R. Shaydulin, J. Larson, and M. Pistoia. 2024. End-to-End Protocol for High-Quality QAOA Parameters With Few Shots. *arXiv preprint arXiv:2408.00557* (2024).
[12] Z. He, R. Shaydulin, D. Herman, C. Li, R. Raymond, S. H. Sureshbabu, and M. Pistoia. 2024. Parameter Setting Heuristics Make the Quantum Approximate Optimization Algorithm Suitable for the Early Fault-Tolerant Era. *arXiv preprint arXiv:2408.09538* (2024).
[13] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, Johnson B. R., and Gambetta J. M. 2024. Quantum Computing With Qiskit. *arXiv preprint arXiv:2405.08810* (2024).
[14] I. Lyngfelt and L. García-Álvarez. 2025. Symmetry-Informed Transferability of Optimal Parameters in the Quantum Approximate Optimization Algorithm. *Physical Review A* (2025).
[15] B. D. McKay. 1983. Applications of a Technique for Labelled Enumeration. *Congressus Numerantium* (1983).
[16] J. A. Montanez-Barrera, D. Willsch, and K. Michielsen. 2024. Transfer Learning of Optimal QAOA Parameters in Combinatorial Optimization. *arXiv preprint arXiv:2402.05549* (2024).
[17] C. Moussa, H. Wang, T. Bäck, and V. Dunjko. 2022. Unsupervised Strategies for Identifying Optimal Parameters in Quantum Approximate Optimization Algorithm. *EPJ Quantum Technology* (2022).
[18] M. A. Nielsen and I. L. Chuang. 2010. *Quantum Computation and Quantum Information.*
[19] A. Pellow-Jarman, I. Sinayskiy, A. Pillay, and F. Petruccione. 2021. A Comparison of Various Classical Optimizers for a Variational Quantum Linear Solver. *Quantum Information Processing* (2021).
[20] J. Preskill. 2018. Quantum Computing in the NISQ Era and Beyond. *Quantum* (2018).
[21] S. H. Sack and M. Serbyn. 2021. Quantum Annealing Initialization of the Quantum Approximate Optimization Algorithm. *Quantum* (2021).
[22] T. Schwägerl, Y. Chai, T. Hartung, K. Jansen, and S. Kühn. 2024. Benchmarking Variational Quantum Algorithms for Combinatorial Optimization in Practice. *arXiv preprint arXiv:2408.03073* (2024).
[23] R. Shaydulin, P. C. Lotshaw, J. Larson, J. Ostrowski, and T. S. Humble. 2023. Parameter Transfer for Quantum Approximate Optimization of Weighted MaxCut. *Transactions on Quantum Computing* (2023).
[24] K. Shi, R. Herrman, R. Shaydulin, S. Chakrabarti, M. Pistoia, and J. Larson. 2022. Multiangle QAOA Does Not Always Need All Its Angles. In *Proc. SEC.* 414–419.
[25] S. H. Sureshbabu, D. Herman, R. Shaydulin, J. Basso, S. Chakrabarti, Y. Sun, and M. Pistoia. 2024. Parameter Setting in Quantum Approximate Optimization of Weighted Problems. *Quantum* (2024).
[26] Rui-Sheng W. and Li-Min W. 2010. Maximum Cut in Fuzzy Nature: Models and Algorithms. *J. Comput. Appl. Math.* (2010).
[27] H. Zeng, F. Meng, T. Luan, X. Yu, and Z. Zhang. 2024. Improved Quantum Approximate Optimization Algorithm for Low-Density Parity-Check Channel Decoding. *Advanced Quantum Technologies* (2024).