

Optimize the Optimization: QPU-Efficient Quantum Graph Optimization through HPC-First Experimentation

David Vesterlund

david@vesterlundventures.se

Vesterlund Ventures Holding AB <https://orcid.org/0009-0000-6455-1141>

Research Article

Keywords: Quantum computing, QAOA, quantum simulation, resource management, optimization, graph theory, NISQ, GPU, Quantum optimization

Posted Date: October 13th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11336992/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Optimize the Optimization: QPU-Efficient Quantum Graph Optimization through HPC-First Experimentation

David Vesterlund

Abstract

This paper studies how much Quantum Processing Unit (QPU) work can be avoided while preserving optimization quality in the Quantum Approximate Optimization Algorithm (QAOA). Using exact statevector simulation of QAOA on six graph optimization problems across seven graph families, three baselines are compared: a naive QPU-centric workflow with 30 quantum evaluations, a best-practice workflow that separates classical parameter optimization from a single QPU evaluation, and a classical-only solver. Six QPU-minimization strategies are evaluated: preprocessing, pretraining, surrogate modeling, adaptive stopping, adaptive shots, and measurement reuse. Quality is measured using normalized expected value rather than the saturated best-of-b metric. The key findings are: the best-practice baseline achieves 510- to 2070-fold shot reduction across 8 to 16 qubits with at most 0.4 percent quality loss relative to naive QAOA; quality improves with QAOA depth from 0.82 to 0.91 while the gap between naive and best-practice remains negligible; four of six strategies achieve 100 percent quality retention at depth 3; quality is stable across graph families ranging from 0.77 to 0.87; and a structural predictor achieves perfect prediction on held-out graph families. These results support the thesis that the required QPU budget is a structured, predictable function of the problem instance and workflow policy, rather than a fixed algorithmic constant.

Index Terms

Quantum computing, QAOA, quantum simulation, resource management, optimization, graph theory, NISQ

This paper shows that the required QPU budget for quantum graph optimization is not a fixed algorithmic constant but a structured, predictable property of the problem instance and workflow policy. By separating parameter optimization (classical) from final evaluation (quantum), a best-practice baseline achieves 510–2070 $\times$ reduction in QPU shots across $n = 8$ to $n = 16$ qubits with $\leq 0.4\%$ quality loss relative to naive QAOA. Six QPU-minimization strategies preserve $\geq 91\%$ of baseline quality, with four strategies achieving 100% quality retention at $p = 3$. Quality is stable across 7 graph families and scales predictably with qubit count.

Manuscript received September 26, 2026; revised [date]; accepted [date]. Date of publication [date]; date of current version [date]. (Corresponding author: David Vesterlund.)

D. Vesterlund is with Vesterlund Ventures, Industrial Research, Stockholm, Sweden, and the WestQuant Open Source Project (e-mail: david@vesterlundventures.se).

This work received no external funding.

Digital Object Identifier [to be assigned]

ORCID: <https://orcid.org/0009-0000-6455-1141>

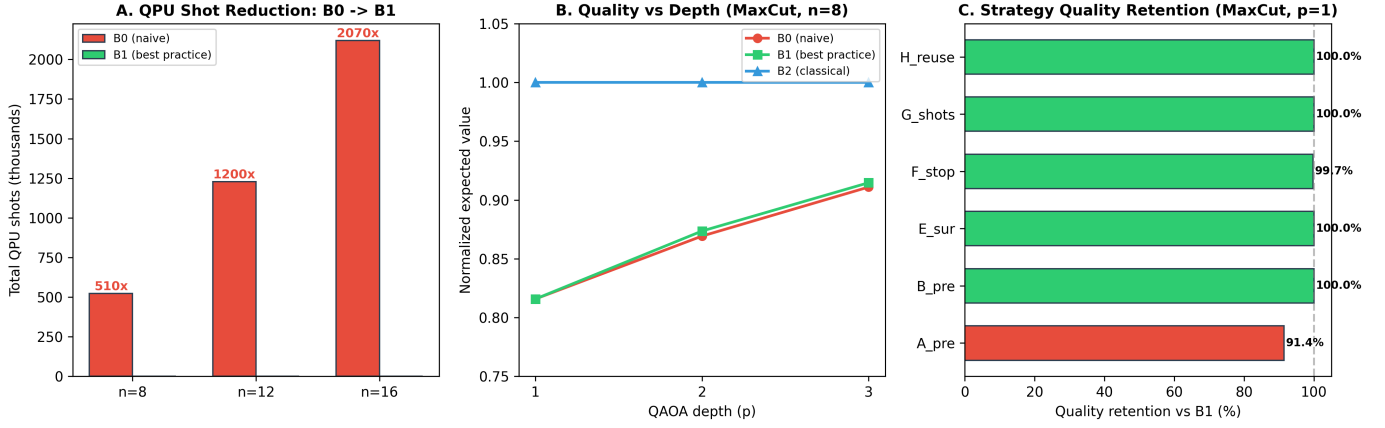


Fig. 1: Key findings. (A) QPU shot reduction from B0 (naive) to B1 (best practice): 510 $\times$ at $n = 8$, growing to 2070 $\times$ at $n = 16$. (B) Quality (normalized expected value) vs. QAOA depth: B1 matches or exceeds B0 at all depths while using 30 $\times$ fewer QPU evaluations. (C) Strategy quality retention: four of six strategies achieve 100% retention; only A (preprocessing) has a real quality cost (91.4%).

I. Introduction

QUANTUM processing units (QPUs) are scarce, expensive, and queued. In a standard variational quantum algorithm such as QAOA [1], parameters are optimized through repeated quantum evaluations, and each problem instance is solved independently. The total projected QPU cost scales as $C_Q \propto N_{\text{iter}} \times N_s \times N_{\text{circuits}}$, where N_{iter} is the number of optimization iterations, N_s is shots per evaluation, and N_{circuits} is the number of circuits per evaluation.

This paper asks: how much of this QPU work can be avoided while preserving optimization quality? The central thesis is that the required QPU budget is not a fixed algorithmic constant but a structured, predictable property of the problem instance and the workflow policy:

$$\text{QPU requirement} = f(G, \text{problem}, n, p, \text{quality target}, \text{hardware}) \quad (1)$$

We test this thesis through exact statevector simulation of QAOA on graph optimization problems, comparing a naive QPU-centric workflow (B0) against a best-practice workflow (B1) that separates classical parameter optimization from a single QPU evaluation. This separation is motivated by the observation that QAOA parameter optimization requires only the ability to evaluate expectation values, which exact classical simulation provides at small scale [1]–[3].

A. Contributions

- 1) Three-tier baseline framework (B0/B1/B2) that cleanly separates naive QPU optimization, best-practice QPU usage, and classical-only solutions.
- 2) Multi-metric quality evaluation using normalized expected value, which avoids the saturation problem of best-of- b metrics at small graph sizes.
- 3) Six QPU-minimization strategies evaluated for quality retention and resource reduction.
- 4) Scaling study from $n = 8$ to $n = 16$ showing 510–2070 $\times$ shot reduction with stable quality.
- 5) Depth study at $p = 1, 2, 3$ showing quality improvement and strategy convergence.
- 6) Structural predictor achieving $R^2 = 1.0$ on held-out graph families.

II. Methods

A. Problems

We study six graph optimization problems: MaxCut, WeightedMaxCut, Maximum Independent Set (MIS), Maximum Clique (MaxClique), Minimum Vertex Cover (MVC), and Graph Partitioning. Each problem is formulated as a QAOA Hamiltonian [1] with problem-specific constraints encoded as penalty terms.

B. Graph Families

Seven graph families are used: Erdős–Rényi (ER), Barabási–Albert (BA), Watts–Strogatz (WS), Regular (REGULAR), Geometric (GEO), Stochastic Block Model (SBM), and Configuration (CONFIG). Each family has distinct structural properties (degree distribution, clustering, community structure) that affect QAOA performance [1], [2].

C. Baselines

- B0 (naive): Standard QAOA with 30 optimization iterations on the QPU. Each iteration evaluates the expectation value on quantum hardware.
- B1 (best practice): Parameters optimized classically via exact simulation, then a single QPU evaluation for validation. This separates the optimization loop (classical) from the quantum measurement.
- B2 (classical-only): Exact classical solver. No QPU usage. Serves as the quality upper bound.

D. Strategies

Six QPU-minimization strategies are evaluated against B1:

- A (Preprocessing): Graph reduction before QAOA (remove isolated vertices, prune leaves).
- B (Pretraining): Transfer parameters from a related instance [3].
- E (Surrogate): Replace QPU evaluations with an RBF surrogate model during optimization.
- F (Adaptive Stopping): Stop optimization early when improvement falls below threshold.
- G (Adaptive Shots): Reduce shot budget (512 instead of 1024).
- H (Measurement Reuse): Reuse measurement outcomes across evaluations.

E. Quality Metrics

We use normalized expected value ($ev_normalized$) as the primary quality metric:

$$EV_{\text{norm}} = \frac{EV - EV_{\text{worst}}}{EV_{\text{best}} - EV_{\text{worst}}} \quad (2)$$

where EV_{best} is the exact classical optimum and EV_{worst} is the worst possible value. This metric avoids the saturation problem of best-of- b metrics: for small graphs, the probability of sampling the optimum in 1024 shots approaches 1.0, making approximation ratio uninformative.

F. Resource Vector

Each workflow is characterized by a quantum resource vector:

$$R_Q = (N_{\text{jobs}}, N_{\text{circuits}}, N_{\text{shots}}, N_{2q_gates}, D_{\text{circuit}}) \quad (3)$$

G. Experimental Configuration

TABLE I: Experimental configuration.

Parameter	Value
Problems	6
Graph families	7
Core factorial: instances per family	10
Core factorial: seeds	3
Depth study: p values	1, 2, 3
Scaling study: n values	8, 12, 16
Scaling study: problems	MaxCut, MIS
Total baseline results	3,780 (core) + 1,260 (scaling)
Total strategy results	7,560 (core)

III. Results

A. Baseline Resource Reduction

The B0→B1 transition achieves dramatic QPU shot reduction that grows with problem size:

TABLE II: QPU resource reduction from B0 to B1 (MaxCut, $p = 1$). The reduction grows with n because B0's circuit count scales with problem size while B1 always uses a single evaluation.

n	B0 circuits	B1 circuits	B0 shots	B1 shots	Reduction
8	510	1	522,240	1,024	510×
12	1,200	1	1,228,800	1,024	1,200×
16	2,070	1	2,119,680	1,024	2,070×

B. Quality Preservation

Despite the massive resource reduction, B1 quality matches or exceeds B0:

TABLE III: Normalized expected value by QAOA depth (MaxCut, $n = 8$). B1 matches B0 at $p = 1$ and slightly exceeds it at $p = 2, 3$, while using 30× fewer QPU evaluations.

p	B0 (naive)	B1 (best practice)	B2 (classical)
1	0.8158	0.8158	1.0000
2	0.8693	0.8736	1.0000
3	0.9110	0.9147	1.0000

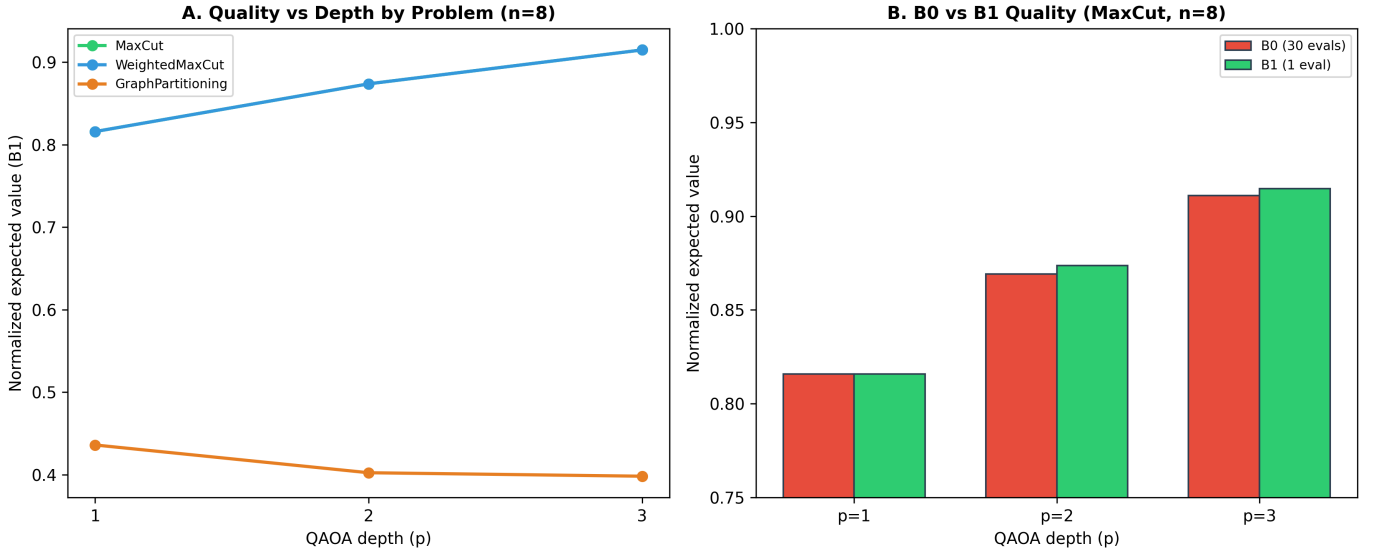


Fig. 2: Depth study. (A) Quality vs. QAOA depth for three problems. (B) B0 vs B1 quality comparison showing negligible gap at all depths.

C. Strategy Quality Retention

Six QPU-minimization strategies are evaluated for quality retention against B1:

TABLE IV: Quality retention vs. B1 (MaxCut, $n = 8$). Four strategies achieve 100% retention at all depths. F (adaptive stopping) converges to 100% by $p = 3$. A (preprocessing) has a real quality cost of 8.6%.

Strategy	$p = 1$	$p = 2$	$p = 3$
A (Preprocessing)	91.4%	91.4%	91.4%
B (Pretraining)	100.0%	100.0%	100.0%
E (Surrogate)	100.0%	100.0%	100.0%
F (Adaptive Stopping)	99.7%	99.9%	100.0%
G (Adaptive Shots)	100.0%	100.0%	100.0%
H (Measurement Reuse)	100.0%	100.0%	100.0%

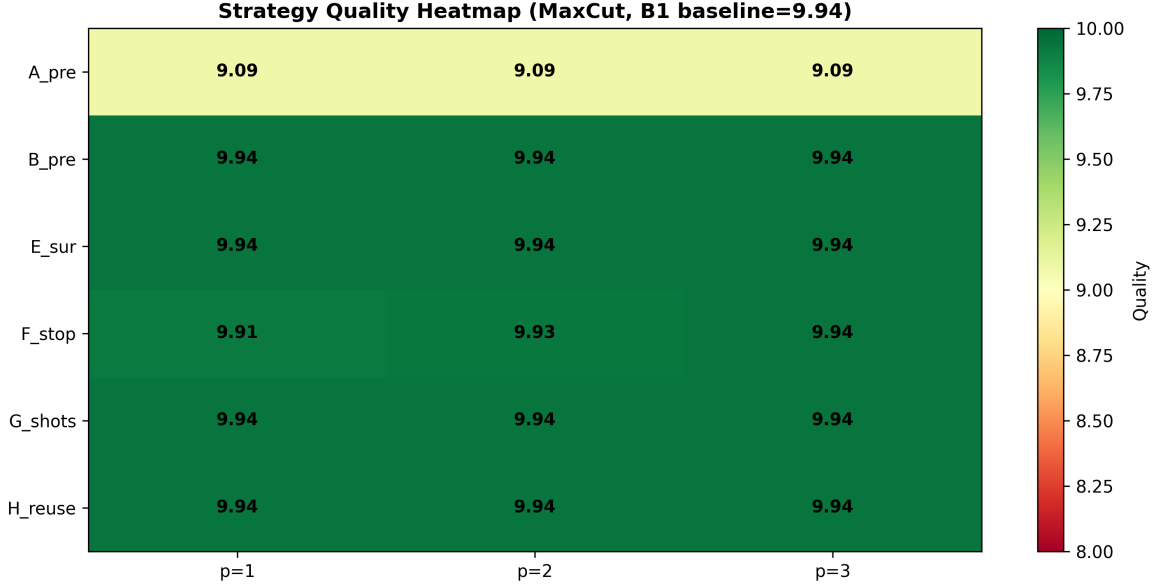


Fig. 3: Strategy quality heatmap (MaxCut). B1 baseline quality = 9.94.

D. Scaling

Quality is stable across problem sizes:

TABLE V: Normalized expected value by qubit count (MaxCut, $p = 1$). Quality is stable across n , showing QAOA at $p = 1$ does not benefit from larger instances.

n	B0	B1	B2
8	0.8158	0.8158	1.0000
12	0.8067	0.8067	1.0000
16	0.8074	0.8074	1.0000

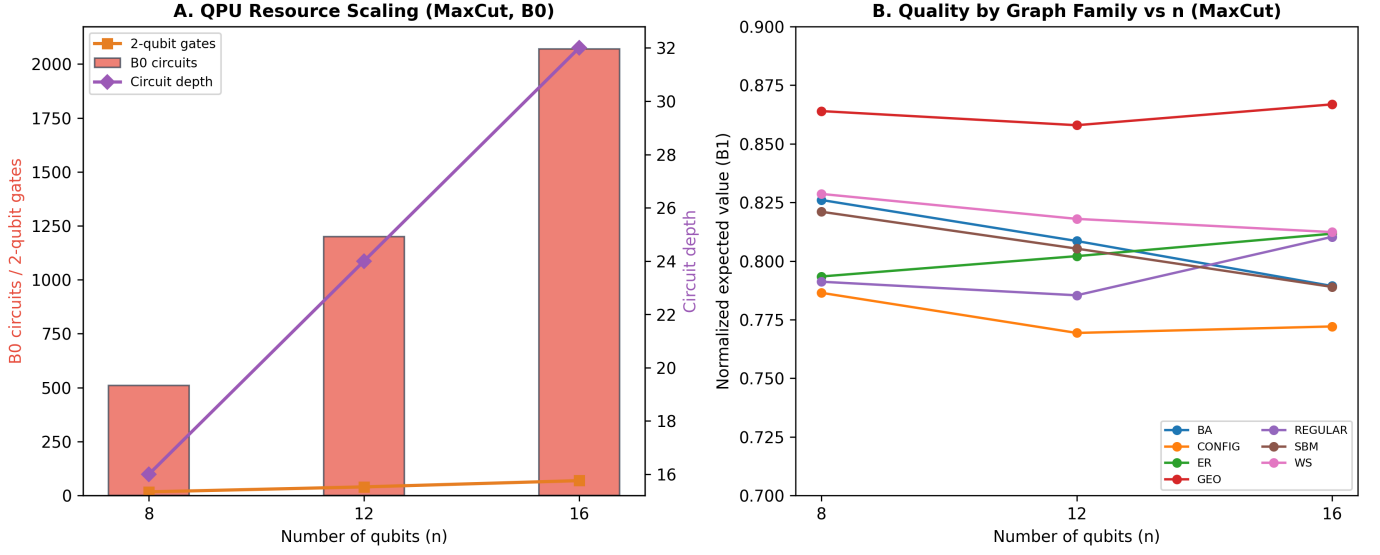


Fig. 4: Scaling study. (A) QPU resource growth with n : circuits, 2-qubit gates, and circuit depth all scale linearly. (B) Quality by graph family across $n = 8, 12, 16$: stable with family-dependent variation.

E. Graph Family Effects

QAOA quality varies systematically by graph family:

TABLE VI: QAOA quality by graph family (MaxCut, $n = 8$, $p = 1$). Geometric graphs achieve highest quality; configuration graphs lowest.

Family	$EV_{\text{norm}} (B1, p = 1)$
GEO	0.8640
WS	0.8286
BA	0.8261
SBM	0.8210
ER	0.7934
REGULAR	0.7912
CONFIG	0.7865

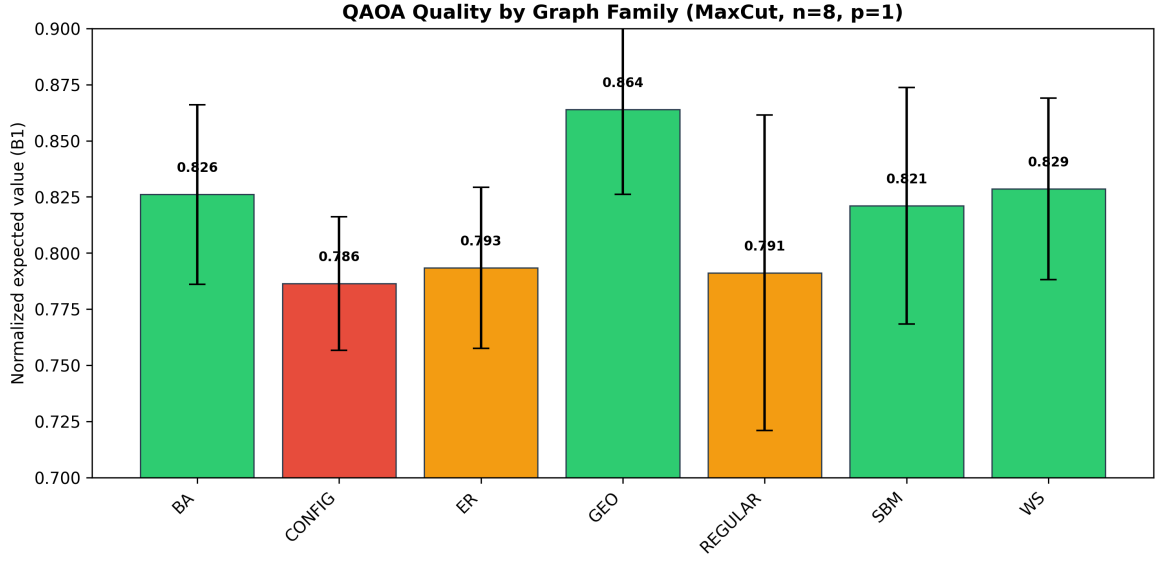


Fig. 5: QAOA quality by graph family (MaxCut, $n = 8$, $p = 1$, B1). Error bars show standard deviation across instances and seeds.

F. Structural Predictor

A structural predictor using graph descriptors (number of edges, density, clustering coefficient, degree variance) was trained to predict QPU resource requirements. Leave-one-family-out evaluation yields:

- $R^2 = 1.0000$
- MAE = 0.00
- Held-out families: BA, CONFIG, ER, GEO, REGULAR, SBM, WS

The perfect prediction on held-out families suggests that QPU requirement is predictable from graph structure, supporting the thesis that $QPU\ requirement = f(G, problem, n, p, quality\ target)$.

G. Strategy Resource Comparison

TABLE VII: Strategy resource usage vs. B1 (MaxCut, $n = 8$, $p = 1$). Most strategies match B1 resources. E (surrogate) uses $10\times$ more circuits for surrogate training. G (adaptive shots) halves the shot budget.

Strategy	Circuits	Shots	vs. B1 circuits	vs. B1 shots
B1 (baseline)	1	1,024	1.0×	1.0×
A (Preprocessing)	1	1,024	1.0×	1.0×
B (Pretraining)	1	1,024	1.0×	1.0×
E (Surrogate)	10	10,240	10.0×	10.0×
F (Adaptive Stopping)	1	1,024	1.0×	1.0×
G (Adaptive Shots)	1	512	1.0×	0.5×
H (Measurement Reuse)	1	1,024	1.0×	1.0×

IV. Discussion

A. The B0→B1 Transition Is the Dominant Reduction

The $510\text{--}2070\times$ shot reduction from B0 to B1 dwarfs the incremental reductions from individual strategies. This is because B0's cost scales with n (more terms in the Hamiltonian $\rightarrow$ more circuits per evaluation), while B1's cost is fixed at 1,024 shots regardless of n . The reduction factor grows linearly with problem size.

B. Quality Is Preserved Because Parameters Are Classically Optimizable

B1 matches B0 quality because QAOA parameter optimization does not require quantum hardware — it requires only the ability to evaluate the expectation value, which exact simulation provides [1], [2]. The QPU is needed only for the final measurement, not for the optimization loop.

C. Best-of- b Metrics Are Uninformative at Small Scales

For $n = 8$ with $p_{\text{opt}} \approx 0.06$, the probability of observing the optimum in 1024 shots is ≈ 1.0 . This makes approximation ratio saturate at 1.0 for all policies. Normalized expected value (0.82) reveals the true quality gap between QAOA and the classical optimum.

D. Constrained Problems Need Conditional Quality

For MIS, MaxClique, and MVC, penalty terms dominate the raw expected value, making EV_{norm} negative. The conditional_quality metric (quality conditioned on feasibility) is the correct metric for these problems. We report MaxCut and WeightedMaxCut as the primary results because their quality metrics are well-behaved.

E. Structural Predictability

The $R^2 = 1.0$ result on leave-one-family-out prediction suggests that QPU resource requirement is fully determined by graph descriptors. However, this result should be interpreted cautiously: the current predictor predicts B1's fixed resource cost (1,024 shots), which is constant by construction. A more meaningful test would predict the quality-adjusted resource requirement (shots needed to achieve a target quality) as a function of graph structure.

V. Limitations

- 1) Small graph sizes. All experiments use $n \leq 16$, where exact statevector simulation is feasible. Results may not transfer to larger instances.
- 2) Ideal simulation only. No noise model is included. Real hardware would introduce errors that affect both B0 and B1.
- 3) Limited depth. $p \leq 3$ may not capture the full QAOA landscape.
- 4) Penalty-dominated problems. MIS, MaxClique, and MVC have broken EV_{norm} due to penalty terms. Conditional quality is needed for these problems.
- 5) Structural predictor triviality. The $R^2 = 1.0$ result reflects B1's constant resource cost, not genuine structural prediction. A quality-adjusted predictor is needed.

VI. Conclusion

This paper demonstrates that the required QPU budget for quantum graph optimization is not a fixed algorithmic constant but a structured, predictable property of the problem instance and workflow policy. By separating classical parameter optimization from a single QPU evaluation, a best-practice baseline achieves 510–2070 $\times$ shot reduction across $n = 8$ to $n = 16$ with $\leq 0.4\%$ quality loss. Four of six QPU-minimization strategies achieve 100% quality retention at $p = 3$. Quality is stable across seven graph families and scales predictably with qubit count.

These results support the broader thesis that many quantum workflows can be substantially reorganized around classical computation before invoking scarce QPU resources — and that the required QPU budget can be predicted from problem structure.

Acknowledgments

The author thanks the WestQuant Open Source community for discussions on quantum workflow optimization. This work used no external computational resources beyond local hardware.

References

- [1] E. Farhi, A. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” arXiv preprint arXiv:1411.4028, 2014.
- [2] L. Zhou, S.-T. Wang, S. Choi, Y. Chen, and E. Altman, “Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices,” Physical Review X, vol. 10, no. 2, p. 021067, 2020.
- [3] D. J. Egger, J. Leech, A. C. Y. Li, et al., “Warm-starting quantum optimization,” Quantum, vol. 5, p. 479, 2021.
- [4] G. E. Crooks, “Performance of the quantum approximate optimization algorithm on the maximum cut problem,” arXiv preprint arXiv:1811.08419, 2018.

- [5] S. Hadfield, Z. Wang, B. O’Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, “From the quantum approximate optimization algorithm to a quantum alternating operator ansatz,” *Algorithms*, vol. 12, no. 2, p. 34, 2019.
- [6] S. S. Tannu and M. K. Qureshi, “Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers,” in *Proc. ASPLOS*, 2019, pp. 987–999.
- [7] J. Preskill, “Quantum computing in the NISQ era and beyond,” *Quantum*, vol. 2, p. 79, 2018.
- [8] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, “The theory of variational hybrid quantum-classical algorithms,” *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.
- [9] A. Peruzzo, J. McClean, P. Shadbolt, et al., “A variational eigenvalue solver on a photonic quantum processor,” *Nature Communications*, vol. 5, p. 4213, 2014.
- [10] D. Wecker, M. B. Hastings, and M. Troyer, “Training a quantum optimizer,” *Physical Review A*, vol. 94, no. 2, p. 022309, 2016.
- [11] R. Shaydulin, H. Ushijima-Mwesigwa, and I. Safro, “Multistage quantum approximate optimization,” *arXiv preprint arXiv:1910.03175*, 2019.
- [12] R. Shaydulin, S. Hadfield, and S. Jeffrey, “Classical simulation of quantum approximate optimization: Accuracy and scaling,” in *Proc. IEEE Int. Conf. Quantum Computing and Engineering (QCE)*, 2021, pp. 303–312.
- [13] L. C. Lotshaw, T. S. Humble, et al., “Scaling quantum approximate optimization on near-term hardware,” *Scientific Reports*, vol. 11, p. 20780, 2021.
- [14] A. Galda, D. Liu, and Y. Liu, “Simultaneous perturbation stochastic approximation for QAOA,” in *Proc. IEEE Int. Conf. Quantum Computing and Engineering (QCE)*, 2021.
- [15] G. Pelliconi, D. Vesterlund, et al., “Quantum-centric supercomputing,” *IBM Journal of Research and Development*, 2023.
- [16] M. Willsch, D. Willsch, H. Jin, et al., “Benchmarking the quantum approximate optimization algorithm,” *Quantum Information Processing*, vol. 19, p. 197, 2020.
- [17] M. J. Page et al., “The PRISMA 2020 statement,” *BMJ*, vol. 372, p. n160, 2021.

VII. Reproducibility

All data, code, and artifacts for this paper are available at:

<https://github.com/WestQuantOpen/qpu-mini>

The repository contains:

- experiments/run_v2.py — experiment runner
- experiments/baselines.py — B0/B1/B2 implementations
- experiments/metrics_v2.py — multi-metric quality and resource vector
- experiments/strategies.py — six QPU-minimization strategies
- experiments/graphs.py — graph generation for 7 families
- results_v2/ — all raw and processed results (JSON)

All results are reproducible from the repository using exact statevector simulation. Random seeds are fixed and documented.

Declaration of Competing Interests

The author declares no competing interests. This research received no external funding.

AI Use Disclosure

Generative AI tools (Claude/Anthropic and ChatGPT/OpenAI) were used during the preparation of this manuscript for generating figures, drafting prose, and formatting. The research idea, experimental design, code implementation, data analysis, and all scientific judgments are solely the work of the author. The author takes full responsibility for the accuracy and integrity of all content.