Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Hybrid Constraint Generation for the Feedback Vertex Set Problem under QAOA

Krishnan V P

krishnanvp.phdmat@rvce.edu.in

RV College of Engineering (affiliated to VTU)

Venugopal K

RV College of Engineering (affiliated to VTU)

Research Article

Keywords: Feedback vertex set, QUBO, Ising model, QAOA, constraint generation, minimum cycle basis, hybrid quantum-classical optimization

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9132952/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Hybrid Constraint Generation for the Feedback Vertex Set Problem under QAOA

Krishnan V. P.[1,2*] and K. Venugopal[1]

[1*]Department of Mathematics, RV College of Engineering (affiliated to VTU), Bengaluru, Karnataka, India.
[2]Prayoga Institute of Education Research, Bengaluru, Karnataka, India.

*Corresponding author(s). E-mail(s): krishnanvp.phdmat@rvce.edu.in;
Contributing authors: venugopalk@rvce.edu.in;

Abstract

The *Feedback Vertex Set* (FVS) problem asks for a minimum-cardinality set of vertices whose removal makes a graph acyclic. A natural integer-programming formulation enforces one constraint per cycle, but enumerating all simple cycles quickly becomes a bottleneck when constructing QUBO and Ising Hamiltonians. We study undirected graphs and present a hybrid constraint-generation pipeline that starts from a compact minimum-cycle-basis encoding, uses the Quantum Approximate Optimization Algorithm (QAOA) as a heuristic candidate generator, and augments the constraint set only when the residual graph exposes a violated cycle. A lightweight classical verification and pruning stage ensures that the final reported solution is a valid FVS. The study centers on the tradeoff between compact encodings and explicit repair: minimum-cycle-basis constraints reduce binary-variable overhead, but they do not in general enforce all cycle-hitting inequalities, so an iterative augmentation loop remains necessary. We therefore frame the method as a hybrid constrained-optimization workflow rather than a claim of quantum advantage, and evaluate it through simulator-based case studies, exact classical references on small instances, and resource metrics that track constraint growth and slack-variable overhead.

Keywords: Feedback vertex set, QUBO, Ising model, QAOA, constraint generation, minimum cycle basis, hybrid quantum-classical optimization

1 Introduction

Given an undirected graph $G = (V, E)$, a *feedback vertex set* (FVS) is a subset $F \subseteq V$ such that removing F leaves an acyclic graph (a forest). The *minimum* feedback vertex set problem asks for an FVS of smallest possible size. This problem is NP-hard even for undirected graphs [1, 2], and is a classical representative of cycle-hitting optimization problems with a long line of algorithmic work and applications [3].

FVS is motivated by the need to break cyclic structure with minimal intervention. Cycle-breaking objectives arise in deadlock-style settings modeled using dependency graphs, and in probabilistic graphical models where conditioning on a small set of variables can reduce inference in loopy networks to inference on tree-structured components (closely related to loop cutsets) [3, 4, 5]. A common theme in such uses is that feasibility is crucial: a candidate set that leaves even one cycle in the residual graph is not acceptable. For FVS, feasibility can be checked efficiently by running a cycle test on $G - F$, so it is natural to separate the goals of (i) proposing high-quality candidates in a large search space and (ii) certifying and refining feasibility and size.

Motivated by near-term quantum heuristics for combinatorial optimization, we study FVS through a hybrid quantum-classical pipeline based on QUBO and the Quantum Approximate Optimization Algorithm (QAOA). QUBO encodings provide a route to apply QAOA by mapping a discrete objective to an Ising Hamiltonian [6, 7]. However, for constrained problems, QAOA (at finite depth and finite shots) can produce high-probability or low-energy bitstrings that are infeasible for the original constraints, so feasibility must be checked explicitly in the workflow [12, 13]. For FVS in particular, the most direct formulation enforces one constraint per cycle, requiring $\sum_{v \in C} x_v \geq 1$ for every simple cycle C. Enumerating all simple cycles is expensive and quickly becomes a bottleneck, and large constraint sets inflate the QUBO size and the number of qubits needed.

In this work, we propose an *iterative constraint-generation* approach for undirected FVS that keeps the encoded cycle set compact while enforcing correctness through classical checks. We begin from a minimum cycle basis (MCB) [8], solve the resulting QUBO using QAOA as a heuristic candidate generator, then verify validity classically. If the candidate is invalid, we extract a violated cycle from the residual graph and add it as a new constraint, repeating the loop until a valid FVS is obtained. Finally, we apply a simple pruning step that greedily removes redundant vertices from the candidate while preserving validity. The key viewpoint is not that QAOA by itself certifies feasibility, but that QAOA can be embedded inside a hybrid workflow whose correctness is enforced by classical verification and whose encoding cost is controlled by starting from a compact cycle set.

Classical FVS methods remain the performance baseline for exactness and scalability. Our aim is therefore not to claim quantum advantage over state-of-the-art classical solvers, but to study a quantum-compatible constrained-optimization formulation that makes the tradeoff between compact encodings, validity repair, and binary-variable growth explicit.

1.1 Contributions

- We formulate an undirected-FVS QUBO from a *selected* cycle set, using binary slack variables to encode cycle-hitting constraints as quadratic penalties and making the binary-variable overhead explicit.
- We show why a minimum cycle basis should be viewed as a compact initializer rather than a complete surrogate for all cycle-hitting constraints, and use this observation to motivate an *iterative validity-repair loop*.
- We introduce a hybrid solve-verify-prune pipeline: QAOA proposes candidates, classical verification certifies feasibility on the original graph, violated cycles are added only when necessary, and a pruning step removes redundant deletions.
- We organize the empirical study around simulator-based case studies, exact classical references on small instances, and resource metrics that track constraint counts, slack-bit overhead, and total binary-variable growth.

2 Preliminaries

2.1 Feedback Vertex Set

A set $F \subseteq V$ is an undirected feedback vertex set (FVS) of $G = (V, E)$ if deleting F and all incident edges makes the residual graph $G - F$ acyclic (a forest). We write $|F|$ for the minimum cardinality of an FVS. Given a candidate set F, verifying that $G - F$ is acyclic is polynomial-time (for example via a DFS/BFS cycle test). In this paper, we use this fact as a classical verification step inside a hybrid pipeline.

2.2 Classical and quantum context

Classical work on FVS includes exact algorithms, parameterized and kernelization-based methods, integer-programming approaches, and approximation or heuristic methods [2, 3]. These methods remain the natural benchmark for large-scale performance. Our use of QAOA should therefore be interpreted as an investigation of a quantum-compatible constrained-optimization workflow, not as a replacement for the strongest classical solvers.

On the quantum side, QUBO and Ising encodings enable the use of gate-model heuristics such as QAOA, but feasibility is generally not guaranteed by the variational loop alone. For constrained graph problems, it is therefore important to separate three tasks: proposing candidates, certifying validity, and quantifying the resource overhead introduced by the encoding itself. This paper focuses on that separation.

2.3 QUBO and Ising Hamiltonians

A QUBO problem minimizes a quadratic polynomial over binary variables [6] $x \in \{0, 1\}^n$:

$$\min_{x \in \{0,1\}^n} \; x^\top Q x + c, \tag{1}$$

with Q symmetric. To run QAOA we map the QUBO to an Ising Hamiltonian by the standard transformation $x_i = (1 - z_i)/2$ with $z_i \in \{+1, -1\}$, yielding

$$H = \sum_i h_i Z_i + \sum_{i<j} J_{ij} Z_i Z_j + \text{const} \cdot I. \tag{2}$$

2.4 QAOA

QAOA is a variational quantum algorithm for approximately minimizing the expected energy of an Ising Hamiltonian by optimizing parameters in a layered ansatz [7]. In our pipeline, QAOA serves as a heuristic optimizer for the QUBO encoding of FVS.

2.5 Cycle Bases and Minimum Cycle Bases

A *cycle basis* of an undirected graph is a set of simple cycles that spans the cycle space. A *minimum cycle basis* (MCB) minimizes the total cycle length among all bases and can be computed in polynomial time [8]. In our pipeline, the MCB provides a compact initial constraint set. However, cycle-basis constraints do not generally imply cycle-hitting constraints for *all* cycles, because the inequalities are not closed under symmetric difference. This motivates iterative constraint generation.

2.6 Problem Statement

Given an undirected graph $G = (V, E)$, our goal is to find a minimum-cardinality set $F \subseteq V$ such that $G - F$ is acyclic. Equivalently, F must intersect every simple cycle of G. The next section shows how we encode this requirement using a selected cycle set and then enforce validity through verification and constraint generation.

3 From Compact Cycle Sets to Validity Repair

For FVS, the conceptually simplest encoding enforces one cycle-hitting inequality for every simple cycle. That model is faithful, but it is often too large to be attractive for QUBO-based quantum heuristics because both constraint enumeration and slack-bit overhead grow quickly. This motivates starting from a smaller cycle set.

A minimum cycle basis is a natural compact initializer, but it must be used carefully. The cycle space is closed under symmetric difference, whereas the inequalities $\sum_{v \in C} x_v \geq 1$ are not. Consequently, satisfying all basis-cycle constraints does not imply that every simple cycle is hit. The central design choice of this paper is therefore to combine a compact initial encoding with an explicit validity-repair loop, rather than to treat basis constraints as sufficient by themselves.

4 QUBO Encoding of Cycle-Hitting Constraints

4.1 Cycle-hitting constraints

Let $x_v \in \{0,1\}$ indicate whether vertex v is deleted ($x_v = 1$) or kept ($x_v = 0$). For a cycle $C \subseteq V$, the cycle-hitting constraint is

$$\sum_{v \in C} x_v \geq 1. \tag{3}$$

A direct model uses (3) for every simple cycle, which is typically too large.

4.2 Slack-variable equality encoding

Given a selected set of cycles C, we encode each inequality (3) as an equality with a nonnegative slack integer y_C:

$$\sum_{v \in C} x_v - 1 - y_C = 0, \qquad y_C \in \{0, 1, \ldots, |C| - 1\}. \tag{4}$$

We represent y_C in binary using $m_C = \lceil \log_2 |C| \rceil$ slack bits $s_{C,0}, \ldots, s_{C,m_C-1} \in \{0,1\}$:

$$y_C = \sum_{k=0}^{m_C-1} 2^k s_{C,k}. \tag{5}$$

Define $\lambda > 0$ as the objective weight and $P > 0$ as the penalty weight. The QUBO energy for a cycle set C is

$$E_C(x, s) = \lambda \sum_{v \in V} x_v \; + \; P \sum_{C \in C} \left(\sum_{v \in C} x_v - 1 - \sum_{k=0}^{m_C-1} 2^k s_{C,k} \right)^2. \tag{6}$$

Minimizing (6) encourages small deletion sets while enforcing (4) for the chosen cycles.

4.3 Qubit count

If $|V| = n$ and the selected cycles are C, the number of binary variables is

$$N = n + \sum_{C \in C} \lceil \log_2 |C| \rceil \,. \tag{7}$$

This directly controls the number of qubits required to represent the Ising Hamiltonian for QAOA.

5 Iterative Constraint Generation With Verification and Pruning

5.1 High-level loop

We start from an initial cycle set C_0 given by a minimum cycle basis of G. At iteration t, we solve the QUBO (6) for C_t using QAOA and obtain a candidate deletion set F_t from measurement samples. We then classically verify whether F_t is a valid feedback vertex set by checking if $G-F_t$ is acyclic. If not, we extract *one* cycle from the residual graph $G - F_t$ and add it to the constraint set:

$$C_{t+1} \leftarrow C_t \cup \{C_{\text{viol}}\}. \tag{8}$$

The process stops once a valid FVS is found or a maximum iteration budget is reached.

5.2 Candidate selection from samples

QAOA yields a parameterized circuit; we sample it with a finite number of shots to obtain an empirical distribution over measured bit strings. Since slack bits can vary while the *vertex* pattern is the same, we aggregate samples by the vertex bits. For each vertex-bit string x, we record (i) its total sampled probability mass and (ii) the minimum *observed* QUBO energy among all sampled slack assignments consistent with x (the slack bits are auxiliary variables, so we care about the best slack completion for a fixed x). We use two orderings: a probability-first ordering for reporting and diagnostics (what QAOA produces most often), and an energy-first ordering for analysis (what the QUBO objective rates best among sampled completions).

For *selection*, we scan the top-K vertex patterns ranked by probability mass, validate each candidate on the original graph (i.e., check whether the selected vertices form a feedback vertex set), and then apply classical pruning to remove any redundant deletions. Among the valid pruned candidates, we select the one with the smallest cardinality; ties are broken by larger probability mass. If no candidate in the top-K set is valid, we use the most probable candidate to identify a violated cycle constraint, add it to the QUBO, and repeat in the next iteration.

5.3 Classical pruning

Once a candidate set F passes verification, we apply a simple pruning heuristic: we attempt to remove vertices from F one at a time and keep the removal if the set remains a valid FVS. This never turns a valid FVS into an invalid one, and can reduce the reported size. Importantly, pruning is local improvement; it does not solve the global optimization problem by itself.

5.4 Pseudocode

Algorithm 1 Constraint-generation QAOA pipeline for undirected FVS

Require: Undirected graph $G = (V, E)$; QAOA parameters; max iterations T; top-K sample budget K.

$C \leftarrow$ Minimum Cycle Basis(G)
for $t = 0$ to $T - 1$ **do**
  Build QUBO $E_C(x, s)$ using (6) and map to Ising H_C
  Run QAOA on H_C and sample the optimized circuit
  Aggregate samples by vertex bits; obtain a ranked list of candidate sets F_{cand}
  for each $F \in$ top-K candidates in F_{cand} **do**
    if Is Acyclic$(G - F)$ **then**
      $F \leftarrow$ Prune(G, F) ▷ greedy removal while preserving acyclicity
      return F ▷ valid feedback vertex set
    end if
  end for
  Find any cycle C_{viol} in $G - F$ for the best-ranked (energy-first) candidate F
  if no such cycle exists **then**
    return F ▷ failsafe: residual is acyclic
  end if
  $C \leftarrow C \cup \{C_{\text{viol}}\}$
end for
return best valid F found (if any) else failure

6 Experimental Setup

6.1 Implementation

We implement the pipeline in Python using NetworkX for graph handling and cycle routines [9], and Qiskit for building the Ising Hamiltonian and running QAOA [10]. We use statevector-based simulation during QAOA parameter optimization and finite-shot sampling for reporting and candidate extraction. The appendix-level hardware runs follow the same workflow, using IBM Quantum device backends for sampling instead of the simulator. For those runs, we use the IBM Quantum backends `ibm_fez`, `ibm_torino`, and `ibm_marrakesh`; when multiple constraint-generation iterations are required, the backends are assigned in round-robin order across successive iterations.

6.2 Parameters

All experiments use a fixed penalty policy P chosen conservatively from the number of vertices. We report the QAOA depth p (reps), shot count, and optimizer settings for each instance. For simulator-side QAOA parameter optimization, we use the derivative-free classical optimizer COBYLA:

Unless stated otherwise, the global QAOA settings table above applies to all simulator and hardware runs. For the broader simulator benchmark introduced in this

Table 1 Global QAOA settings used in all experiments.

Parameter	Value
QAOA depth p (reps)	5
Shots (simulator)	4096
Shots (hardware)	4096
Classical optimizer	COBYLA
Max iterations (COBYLA)	200
Tolerance (COBYLA)	10^{-3}

revision, we use a lighter benchmark profile than in the deep case studies: QAOA depth $p = 3$, 2048 shots, COBYLA with 60 iterations, and probability-ranked validation over the top 12 aggregated vertex candidates. The goal of this benchmark is not to maximize variational performance, but to provide a reproducible nine-graph comparison among augmenting-MCB, MCB-only, and selectively executed full-cycle baselines while retaining exact classical references on all executed instances.

6.3 Evaluation layers

The empirical study has three complementary parts. The first consists of five illustrative case studies that preserve the original narrative of the paper and expose instance-level behaviors such as symmetry, pruning, and violated-cycle augmentation. The second is a systematic small-graph simulator benchmark designed to compare baselines across a fixed set of structured and random instances while keeping the execution profile uniform. The third is a resource-growth study that tracks how minimum-cycle-basis and full-cycle encodings differ in constraint count, slack-bit count, and QUBO coupling growth even when full QAOA execution is no longer practical.

7 Results

We organize the evaluation in three complementary parts. First, we retain a small set of illustrative case studies to show how the hybrid pipeline behaves on representative graph structures and to highlight situations in which augmentation matters. Second, we report a systematic simulator benchmark that compares the augmenting-MCB pipeline with MCB-only and selectively executed full-cycle baselines under a uniform execution profile. Third, we present a resource-growth study that tracks how cycle counts, slack-bit overhead, and QUBO coupling counts grow as the graph family becomes denser or larger.

The case studies remain useful because they expose behaviors that are easy to lose in aggregate summaries, including symmetry between equivalent solutions and instances where verification reveals a violated residual cycle. However, they are no longer the sole empirical evidence for the paper.

Table 2 Nine-graph simulator benchmark. Mean gap is measured against the exact classical optimum, Mean selected rank is the average probability rank of the valid candidate eventually reported after validation and pruning, and "Top-rank invalid" counts runs in which the most probable candidate failed FVS validation.

Baseline	Runs	Mean gap	Mean selected rank	Top-rank invalid
Augmenting-MCB	9	0.0	1.00	4/9
MCB-only	9	0.0	1.22	3/9
Full-cycle	3	0.0	0.00	0/3

7.1 Illustrative case studies

All five structured example graphs are now collected in Appendix A. That appendix includes a compact visual overview of the butterfly graph, the 2×3 ladder, K_4, W_6, and K_5, together with the detailed butterfly sanity check and the K_5 augmentation example that remains the clearest instance of explicit violated-cycle repair in the present study.

The examples still play an important qualitative role. They show that pruning often collapses supersets of valid solutions to minimum FVSs, that symmetric graphs can yield different but equivalent pruned solutions, and that the hybrid loop should be interpreted as solve–verify–prune rather than as raw QAOA output alone. With that qualitative background deferred to the appendix, the main text can focus on the aggregate simulator benchmark and the resource-growth comparison.

7.2 Systematic simulator benchmark

To complement the hand-picked case studies, we ran a systematic simulator benchmark using the refactored pipeline and the benchmark profile described in Section 6.2. The benchmark consists of nine graphs: the ladders 2×3 and 2×4, K_4, K_5, W_6, and four connected Erdős–Rényi instances with nontrivial cycle structure. For each graph we ran the proposed augmenting-MCB pipeline and an MCB-only encoding without augmentation. We additionally ran the full-cycle baseline on the three smallest graphs for which the all-cycle encoding remained practical in the same workflow: the ladder 2×3, K_4, and one small Erdős–Rényi instance.

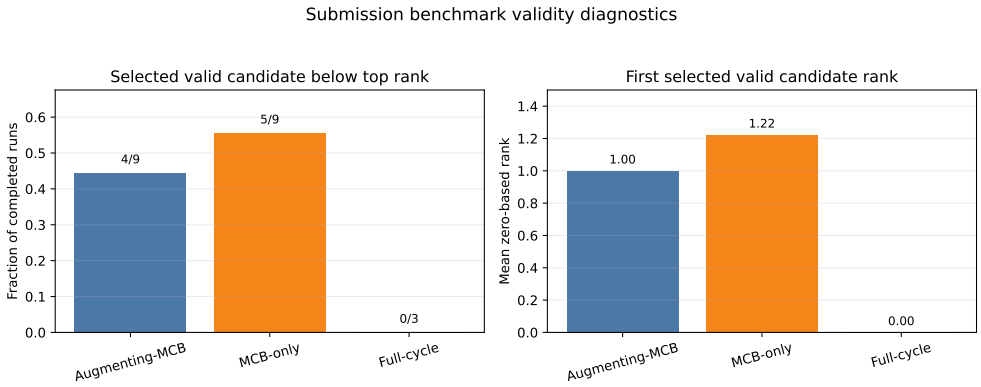


Fig. 1 Validation diagnostics for the nine-graph simulator benchmark. Left: fraction of completed runs in which the final reported valid candidate was not the most probable sample. Right: mean zero-based probability rank of the first valid candidate selected after validation and pruning.

Figure 1 makes the operational role of validity checking more explicit than the aggregate size summary alone. Across the 18 MCB-based runs, the final reported valid candidate appeared below the most probable sample in 9 cases: 4/9 for augmenting-MCB and 5/9 for MCB-only. The corresponding mean zero-based selected-candidate ranks are 1.00 and 1.22, whereas the full-cycle baseline remained at 0.00 on the three instances where that encoding was still executable. These outcomes reinforce the paper's central interpretation of QAOA as a candidate generator rather than a validity guarantee. On these small graphs, compact MCB encodings are often sufficient to reach an exact final answer, but the reported answer still depends on explicit validation and pruning.

Table 2 records the complementary optimality result. On all 21 executed runs, the final pruned solution matched the exact classical optimum for that graph. The most probable sample itself was invalid in 7 of the 18 MCB-based runs, so even in this small benchmark the workflow's correctness still comes from classical validation and pruning rather than from the raw QAOA ranking. This is slightly different from the selected-rank diagnostic above: a final reported candidate can appear below the top probability either because the top-ranked sample is invalid or because a lower-ranked valid sample becomes preferable after the validation-and-pruning scan.

At the same time, the systematic benchmark does not exhaust the augmentation story. In this executed nine-graph set, all augmenting-MCB runs terminated without adding a violated cycle because a valid candidate already appeared within the scanned top-12 probability-ranked set. The clearest augmentation-sensitive behavior therefore remains in Appendix A, especially the supporting K_5 example, rather than in the broad benchmark table alone.

7.3 Resource-growth observations

The broader resource study quantifies how quickly the encoding size grows even before hardware noise or transpilation cost is considered. Table 3 reports representative examples from the generated resource tables. The gap between MCB-based and full-cycle encodings is already visible on small graphs such as the ladder 2×4 and K_4, and becomes dramatic on denser graphs such as K_5, K_6, and W_6.

Table 3 Representative encoding-growth examples from the resource study. "Bits" counts vertex plus slack variables; "quadratic terms" counts nonzero off-diagonal QUBO couplings. Full-cycle counts are reported only when simple-cycle enumeration remained feasible.

Graph	$\lvert C_0 \rvert$	All cycles	MCB bits	Full bits	MCB quad.	Full quad.
2×3 ladder	2	3	10	13	29	54
2×4 ladder	3	6	14	23	43	124
K_4	3	7	10	18	27	61
W_6	5	21	16	59	45	308
K_5	6	37	17	91	52	431
K_6	10	197	26	532	85	3116

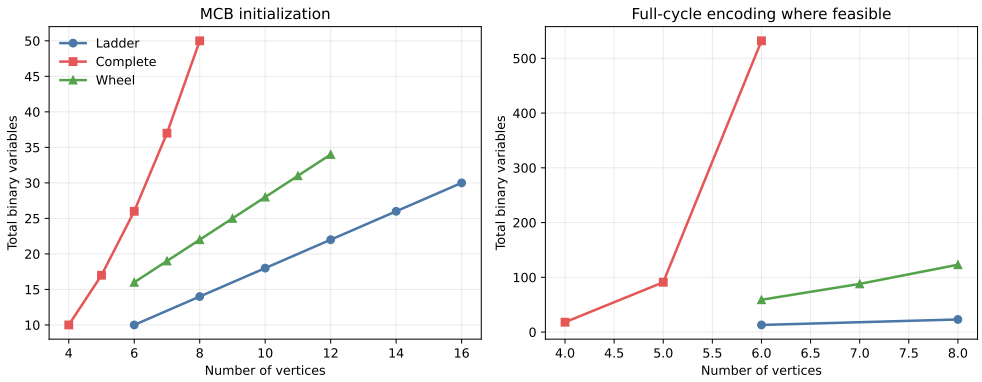


Fig. 2 Encoding-growth comparison across deterministic graph families. Left: total binary variables under minimum-cycle-basis initialization for ladder, complete, and wheel graphs. Right: total binary variables under full-cycle encoding on the deterministic instances where full-cycle enumeration remained feasible.

Figure 2 makes the scaling contrast more explicit than the representative table alone. The left panel shows that MCB initialization still grows with graph size, but at a comparatively moderate rate on the deterministic families considered here. The right panel shows why full-cycle encodings quickly become the limiting comparison point: even where enumeration remains feasible, the total binary-variable count rises much more sharply, especially for complete and wheel graphs.

Together, Table 3 and Figure 2 support the central modeling tradeoff of the paper. Starting from a minimum cycle basis does not remove the need for validity checks, but it can substantially reduce binary-variable and coupling growth relative to a full-cycle encoding. The augmented pipeline therefore separates two roles that are often conflated in constrained variational optimization: compactness of the initial encoding and correctness of the final reported solution.

8 Discussion

8.1 What the benchmark shows

The simulator benchmark establishes three concrete outcomes for the proposed workflow. First, across all executed benchmark instances, the final reported solution agrees with the exact small-instance optimum after validation and pruning. Second, the resource study confirms that minimum-cycle-basis initialization preserves a much smaller executable problem than a full-cycle encoding on the same graph families. Third, the sampled candidate with the highest probability is often not the final valid solution that should be reported, which shows why solution selection must include explicit graph-level verification rather than rely on sample ranking alone.

Taken together, these results support the main design choice of the paper: correctness should be enforced at the hybrid-workflow level, not assumed from the compact initial encoding alone. QAOA plays the role of a heuristic candidate generator, while classical validation certifies acyclicity and pruning removes redundant vertices before the final FVS is reported. This separation is useful because it allows the encoding to remain compact at the start while still preserving the ability to return a valid solution on the original graph.

The case studies complement the aggregate benchmark by showing where the workflow becomes most informative. The butterfly, ladder, and K_4 examples illustrate baseline behavior and candidate ranking effects, while W_6 and K_5 highlight the more demanding dense and augmentation-sensitive settings that motivated the iterative formulation. Viewed together with the resource-growth plots, the empirical picture is consistent: compact initialization is valuable, verification is essential, and the cost of enforcing all cycles upfront grows quickly enough to justify a constrained-generation strategy.

8.2 Scope and next steps

The present results define a clear baseline for future extensions in hybrid quantum optimization for FVS. The paper shows that the formulation is internally consistent, that it can recover exact optima on nontrivial small instances, and that it exposes a measurable compactness-versus-completeness tradeoff between MCB initialization and full-cycle enforcement. That is a useful result for quantum optimization studies on graph problems because it identifies where the quantum layer contributes and where classical post-processing remains indispensable.

Several natural next steps follow directly from this baseline. Broader simulator studies can test how frequently violated-cycle augmentation activates as graph size and

density increase. Improved penalty-selection strategies and alternative slack encodings may reduce logical overhead without sacrificing validity. On the implementation side, noise-aware experiments and transpilation-level resource accounting would connect the logical QUBO analysis reported here to executable circuit costs on larger backends. Within that broader program, the current paper contributes the formulation, the validation logic, and the first systematic evidence that this hybrid workflow behaves coherently on both structured and random undirected FVS instances.

9 Conclusion

We presented a hybrid constraint-generation pipeline for the undirected feedback vertex set problem that combines QAOA-based candidate generation with compact cycle-set initialization, graph-level verification, pruning, and iterative augmentation when needed. A key outcome of the paper is that a minimum-cycle-basis start can be turned into a coherent solve-verify-refine workflow whose behavior is interpretable in both solution-quality and resource-growth terms.

The study shows that this workflow can recover exact small-instance optima across the executed benchmark set, that compact MCB encodings are materially more executable than full-cycle formulations, and that explicit validity checks remain essential even when the most probable sampled candidate looks competitive. These points give the method a clear role within hybrid quantum optimization: it is a structured way to balance compact encodings against correctness requirements for constrained graph problems.

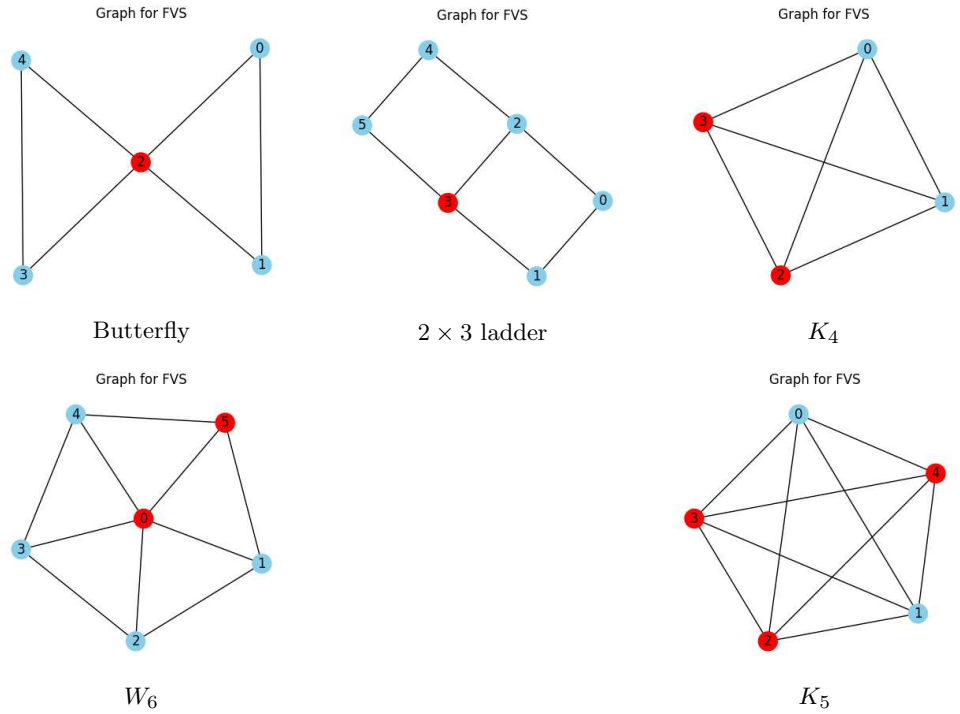
Future work can now build on this foundation in a targeted way by enlarging the simulator benchmark, strengthening augmentation-sensitive test sets, refining penalty and slack-variable design, and extending the analysis from logical problem size to noise-aware circuit execution.

Appendix A Supporting Case-Study Material

This appendix collects the detailed case-study material deferred from the main text. Figure A1 gives a compact visual overview of all five structured examples used throughout the paper, and the appendix summary table that follows collects their main encoding and solution statistics. The detailed butterfly sanity check and the K_5 augmentation example are then retained because they are the most informative representative examples for the present manuscript.

A.1 Overview of the five illustrative examples

The five structured examples serve different roles. The butterfly graph is the simplest sanity check. The 2×3 ladder highlights symmetry between equivalent single-vertex solutions. K_4 is the smallest dense example in which the final valid candidate need not be the most probable sampled candidate. W_6 shows that validation and pruning remain important even when the final optimum has size two. K_5 is the strongest augmentation-sensitive example retained in this paper, especially in the supporting hardware run.



Butterfly | 2×3 ladder | K_4

W_6 | K_5

Fig. A1 Compact visual overview of the five structured simulator examples. Red vertices indicate one representative pruned FVS returned by the workflow.

A.2 Butterfly graph (simulator)

We consider the butterfly graph on $|V| = 5$ vertices and $|E| = 6$ edges, formed by two triangles sharing a single articulation vertex. Using the minimum cycle basis $C_0 = \{(0, 1, 2), (2, 3, 4)\}$, the QUBO objective used in our experiments is

$$\begin{aligned} E(x, s) = \lambda \sum_{v=0}^{4} x_v \\ + P\Big(\big(x_0 + x_1 + x_2 - 1 - (s_{10} + 2s_{11})\big)^2 \\ + \big(x_2 + x_3 + x_4 - 1 - (s_{20} + 2s_{21})\big)^2 \Big), \end{aligned} \tag{A1}$$

where (s_{10}, s_{11}) are the slack bits for cycle $(0, 1, 2)$ and (s_{20}, s_{21}) for cycle $(2, 3, 4)$. In the reported runs we set $\lambda = 1$ and used the default penalty policy $P = 2|V| + 1 = 11$. The constraint-generation loop terminates at Iteration 1, and after verification and greedy pruning the reported feedback vertex set is $F = \{2\}$.

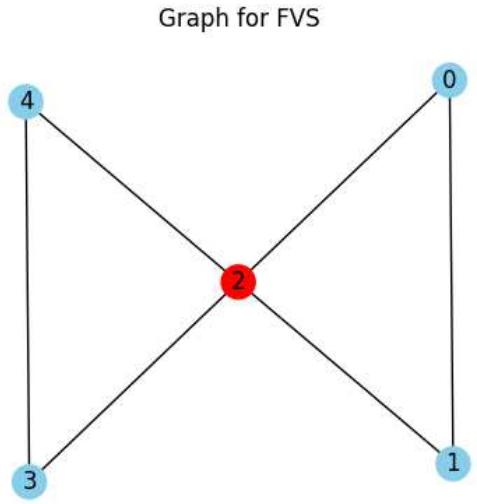


Fig. A2 Butterfly graph used in the simulator sanity check.

Fig. A3 Top-12 candidates for the butterfly instance (simulator). Candidates are ranked by sampled probability; each valid candidate is greedily pruned, and the final reported FVS is taken after pruning.

Rank	Candidate F	Prob.	Valid?	Pruned F	$\lvert F\rvert$
0	$\{2\}$	0.079102	Yes	$\{2\}$	1
1	$\{0, 1, 2\}$	0.065430	Yes	$\{2\}$	1
2	$\{2, 3, 4\}$	0.056641	Yes	$\{2\}$	1
3	$\{0, 1, 3, 4\}$	0.054688	Yes	$\{1, 4\}$	2
4	$\{3, 4\}$	0.049805	No		
5	$\{1\}$	0.044922	No		
6	$\{0\}$	0.044922	No		
7	$\{0, 1\}$	0.042969	No		
8	$\{1, 2, 3, 4\}$	0.041992	Yes	$\{2\}$	1
9	$\{4\}$	0.041992	No		
10	$\{0, 2, 3, 4\}$	0.041016	Yes	$\{2\}$	1
11	$\{3\}$	0.039062	No		

A.3 K_5 augmentation example (simulator and hardware)

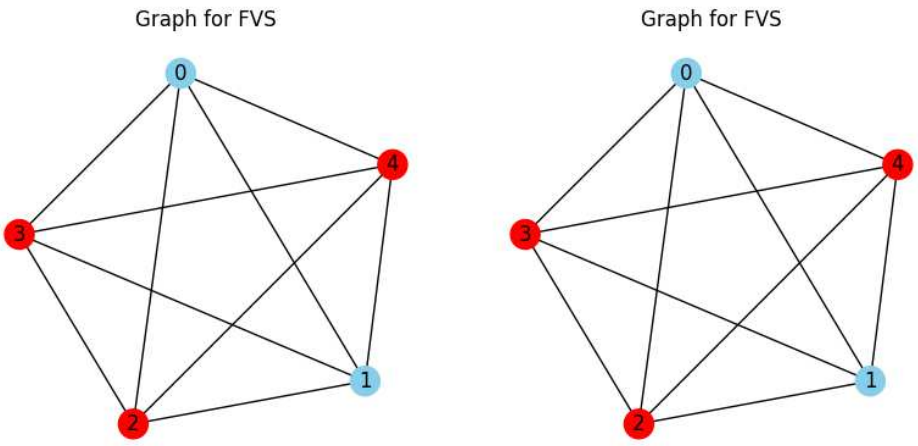
We consider the complete graph K_5 with $|V| = 5$ vertices and $|E| = 10$ edges. A minimum feedback vertex set has size 3: deleting any 3 vertices leaves a K_2 (acyclic), while deleting only 2 vertices leaves a K_3, which contains a cycle. Using the minimum cycle basis $C_0 = \{(1, 3, 0), (2, 3, 0), (1, 4, 0), (3, 4, 0), (1, 2, 0), (2, 4, 0)\}$, the base QUBO objective is

$$E_0(x, s) = \lambda \sum_{v=0}^{4} x_v + P \sum_{i=1}^{6} \Big(x_{u_i} + x_{v_i} + x_0 - 1 - (s_{i0} + 2s_{i1}) \Big)^2, \tag{A2}$$

where (u_i, v_i) are the two nonzero vertices in the i-th triangle of C_0 and each triangle uses two slack bits. Because all basis cycles include vertex 0, deleting vertex 0 alone satisfies E_0 but leaves a cyclic residual K_4 on vertices $\{1, 2, 3, 4\}$; our cycle-augmentation step therefore appends one violated triangle from the residual graph, such as $(1, 2, 3)$, yielding an additional penalty term

$$P\Big(x_1 + x_2 + x_3 - 1 - (t_0 + 2t_1) \Big)^2. \tag{A3}$$

In the reported runs we set $\lambda = 1$ and used the default penalty policy $P = 2|V| + 1 = 11$. In the simulator run, a valid FVS appears among the top candidates already at Iteration 1 and is pruned to size 3. In the hardware run, none of the top-12 candidates at Iteration 1 is a valid FVS; the algorithm therefore adds one violated cycle from the residual graph and re-optimizes, after which Iteration 2 yields valid solutions that prune to $|F| = 3$. In both runs, the reported pruned solution is the minimum-size set $F = \{2, 3, 4\}$ (one representative minimum FVS).



Simulator: $F = \{2, 3, 4\}$. Hardware: $F = \{2, 3, 4\}$.

Fig. A4 Complete graph K_5. Red vertices are the pruned feedback vertex set returned by our method; removing them eliminates all cycles in the remaining graph.

Fig. A5 Top-12 candidates for the K_5 instance (simulator). Candidates are ranked by sampled probability; validity is checked on $G - F$ and valid sets are greedily pruned.

Rank	Candidate F	Prob.	Valid?	Pruned F	$\lvert F \rvert$
0	$\{0, 1, 2, 3, 4\}$	0.052734	Yes	$\{2, 3, 4\}$	3
1	$\{1, 2\}$	0.045898	No		
2	$\{1, 2, 3, 4\}$	0.044922	Yes	$\{2, 3, 4\}$	3
3	$\{0\}$	0.044922	No		
4	$\{1\}$	0.040039	No		
5	$\{1, 2, 3\}$	0.038086	Yes	$\{1, 2, 3\}$	3
6	$\{\}$	0.035156	No		
7	$\{0, 2\}$	0.03418	No		
8	$\{0, 4\}$	0.033203	No		
9	$\{1, 3\}$	0.033203	No		
10	$\{0, 1\}$	0.032227	No		
11	$\{2, 4\}$	0.032227	No		

Fig. A6 Top-12 candidates for the K_5 instance on hardware at Iteration 1. None of the top-12 candidates is a valid FVS, triggering one violated-cycle augmentation before Iteration 2.

Rank	Candidate F	Prob.	Valid?	Pruned F	$\lvert F \rvert$
0	$\{\}$	0.059082	No		
1	$\{4\}$	0.056396	No		
2	$\{2\}$	0.04541	No		
3	$\{0, 4\}$	0.044189	No		
4	$\{0\}$	0.043457	No		
5	$\{3, 4\}$	0.041748	No		
6	$\{1\}$	0.041016	No		
7	$\{3\}$	0.040527	No		
8	$\{2, 4\}$	0.036621	No		
9	$\{1, 4\}$	0.036621	No		
10	$\{0, 3\}$	0.032959	No		
11	$\{0, 2\}$	0.032227	No		

Fig. A7 Top-12 candidates for the K_5 instance on hardware at Iteration 2 after adding one violated-cycle constraint. Valid sets are greedily pruned; the final reported solution has $|F| = 3$.

Rank	Candidate F	Prob.	Valid?	Pruned F	$\lvert F\rvert$
0	$\{0, 1, 2, 3, 4\}$	0.041016	Yes	$\{2, 3, 4\}$	3
1	$\{1, 2, 3, 4\}$	0.039795	Yes	$\{2, 3, 4\}$	3
2	$\{2, 4\}$	0.039551	No		
3	$\{0, 2, 3, 4\}$	0.0354	Yes	$\{2, 3, 4\}$	3
4	$\{1, 2, 4\}$	0.0354	Yes	$\{1, 2, 4\}$	3
5	$\{0, 1, 3, 4\}$	0.035156	Yes	$\{1, 3, 4\}$	3
6	$\{0, 2, 4\}$	0.034912	Yes	$\{0, 2, 4\}$	3
7	$\{0, 1, 2, 4\}$	0.03418	Yes	$\{1, 2, 4\}$	3
8	$\{2, 3\}$	0.03418	No		
9	$\{2, 3, 4\}$	0.033936	Yes	$\{2, 3, 4\}$	3
10	$\{2\}$	0.033203	No		
11	$\{0, 2, 3\}$	0.032471	Yes	$\{0, 2, 3\}$	3

Acknowledgments

We thank IBM for providing the Qiskit software stack and access to IBM Quantum services for the hardware experiments reported in this work. We also acknowledge RV College of Engineering and Prayoga Institute of Education Research for their institutional support.

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: W. H. Freeman, 1979.

[2] L. Brunetta, F. Maffioli, and M. Trubian, "Solving the feedback vertex set problem on undirected graphs," *Discrete Applied Mathematics*, vol. 101, no. 1–3, pp. 37–51, 2000, doi: 10.1016/S0166-218X(99)00180-8.

[3] Y. Zhang, Y. Xia, and X. He, "Feedback vertex set: A survey," *Algorithms*, vol. 6, no. 4, pp. 519–539, 2013, doi: 10.3390/a6040519.

[4] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan Kaufmann, 1988.

[5] A. Becker and D. Geiger, "A sufficiently fast algorithm for finding close to optimal loop cutsets in belief networks," in *Proc. 10th Conf. on Uncertainty in Artificial Intelligence (UAI)*, 1994, pp. 407–414, doi: 10.1016/B978-1-55860-332-5.50013-4.

[6] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using QUBO models," *4OR*, vol. 17, pp. 335–371, 2019, doi: 10.1007/s10288-019-00424-y.

[7] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," arXiv:1411.4028, 2014, doi: 10.48550/arXiv.1411.4028.

[8] J. D. Horton, "A polynomial-time algorithm to find the shortest cycle basis of a graph," *SIAM Journal on Computing*, vol. 16, no. 2, pp. 358–366, 1987, doi: 10.1137/0216026.

[9] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proc. 7th Python in Science Conf. (SciPy)*, 2008, pp. 11–15.

[10] H. Abraham *et al.*, "Qiskit: An open-source framework for quantum computing," 2019, doi: 10.5281/zenodo.2562111.

[11] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, p. 5, 2014, doi: 10.3389/fphy.2014.00005.

[12] 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, doi: 10.3390/a12020034.

[13] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm," *Quantum Information Processing*, vol. 19, p. 197, 2020, doi: 10.1007/s11128-020-02692-8.