

# When Should a Quantum Workflow Call the QPU? - A Systematic Review and Taxonomy of Semantic QPU Minimization

David Vesterlund

david@vesterlundventures.se

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

---

## Research Article

**Keywords:** quantum computing, QPU minimization, hybrid quantum-classical, systematic mapping review, semantic QPU minimization, resource displacement, workflow optimization

**Posted Date:** October 13th, 2026

**DOI:** <https://doi.org/10.21203/rs.3.rs-11337050/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.

---

# When Should a Quantum Workflow Call the QPU?

## A Systematic Review and Taxonomy of Semantic QPU Minimization

David Vesterlund  
Industrial Research at Vesterlund Ventures  
WestQuant Open Source Project  
david@vesterlundventures.se

This paper shows that no existing quantum workflow system automatically determines whether a planned QPU operation needs to run on quantum hardware at all. While 60 studies (2018–2026) demonstrate methods to reduce QPU consumption, they operate in isolation: 62.1% remain at simulation level only, only 8.6% combine multiple techniques, and classical resource costs are systematically underreported (0% report energy, 5.8% GPU time, 11.5% CPU time). We formalize *Semantic QPU Minimization* as a multi-objective optimization problem and show that the gap between *scheduling* (“where should this task run?”) and *semantic minimization* (“does this quantum task need to exist?”) is the central open problem.

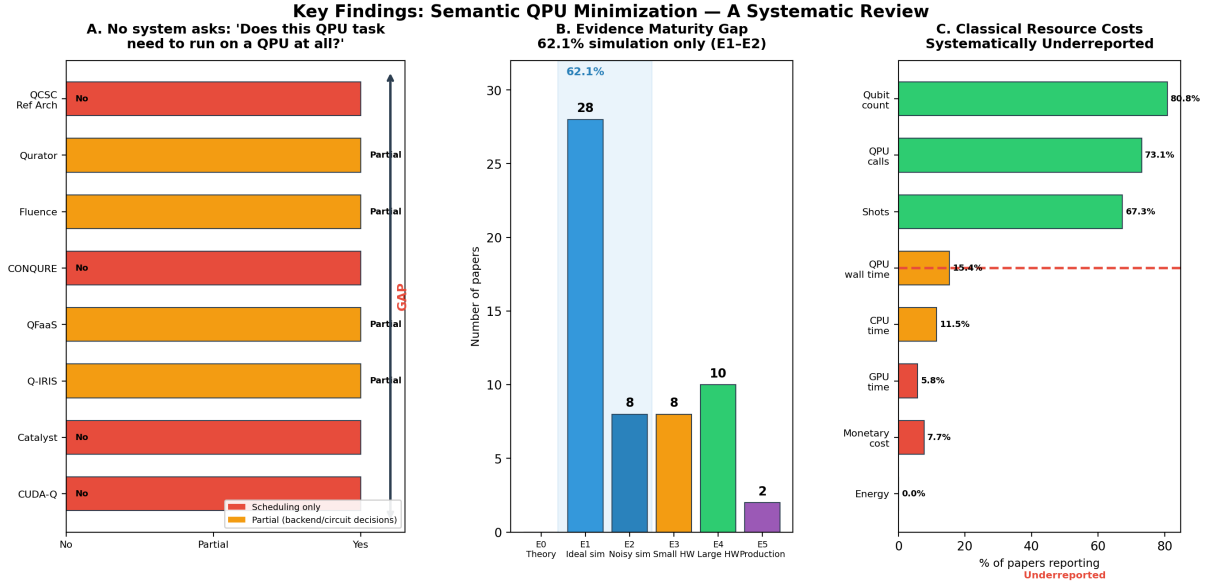


Figure 1: **Key findings of this review.** (A) No system in the corpus performs automatic semantic QPU minimization; all existing systems (QCSC, Qurator, Fluence, CONQURE, QFaaS, Q-IRIS, Catalyst, CUDA-Q) perform scheduling or partial backend selection. (B) 62.1% of method papers operate at simulation level only (E1–E2); only 17.2% reach hardware evidence level E4+. (C) Classical resource costs are systematically underreported: 0% of papers report energy consumption, 5.8% GPU time, and 11.5% CPU time, making full resource-displacement assessment impossible for most methods.

### Abstract

Quantum processing units (QPUs) are scarce, expensive, and queued. As quantum algorithms mature, a growing body of work asks not “how can we improve quantum algorithms?” but rather “which parts of the quantum experimental lifecycle can be moved away from the QPU?” This paper presents a systematic mapping review of 60 studies (2018–2026) that explicitly reduce QPU resource consumption across the quantum experimental lifecycle. We introduce a multidimensional taxonomy with three independent axes: (1) **intervention points** organized in 4 layers (Problem/Representation, Algorithm/Circuit, Execution/Measurement, Amortization/Hybridization) with 12 specific intervention points; (2) **transformation semantics** — Eliminate, Replace, Compress, Reuse/Amortize, Defer, Batch — describing what each method does to QPU operations, formalized as operations on a workflow DAG; and (3) **evidence dimensions** — semantic guarantee (G), evidence maturity (E), hardware scale (S), and deployment maturity (D) — as independent classification axes. We formalize **Semantic QPU Minimization** as a multi-objective Pareto optimization problem over workflow transformations, minimizing a quantum resource vector  $R_Q$  and classical displacement cost  $C_C$  subject to explicit utility, classical-resource, and wall-time constraints. Our key findings: (1) 62.1% of method papers operate at simulation level only (E1–E2), with 17.2% reaching hardware evidence level E4+; (2) the largest reported reductions in individual QPU-resource dimensions occur among Replace and Compress transformations; (3) only 8.6% of papers combine multiple techniques; (4) existing workflow systems schedule already-defined tasks but none automatically transforms an end-to-end hybrid workflow to eliminate, replace, or compress planned QPU operations subject to an explicit task-utility constraint. We identify 10 research gaps, including the lack of standardized resource-displacement metrics, the absence of semantic QPU-necessity analysis, and the systematic underreporting of classical resource costs.

**Keywords:** quantum computing, QPU minimization, hybrid quantum-classical, systematic mapping review, semantic QPU minimization, resource displacement, workflow optimization

## 1 Introduction

Quantum computing has reached a stage where quantum processing units (QPUs) with 100+ qubits are available through cloud platforms. However, QPU access remains a scarce resource: queue times routinely exceed computation times by orders of magnitude, each circuit evaluation consumes shots and incurs financial cost, and the total QPU budget for any given research project is limited.

The conventional quantum computing workflow treats the QPU as the primary compute engine. In a standard variational quantum algorithm (VQA) such as QAOA or VQE, parameters are optimized through repeated quantum evaluations, measurements are taken at fixed shot budgets, and each problem instance is solved independently.

A growing body of work challenges this convention. These works ask: *which parts of the quantum experimental lifecycle can be moved away from the QPU?* The unifying target is QPU resource minimization while retaining useful scientific or computational quality.

This paper conducts a systematic mapping review of this emerging literature. Unlike broad surveys of hybrid quantum-classical computing or quantum-centric supercomputing (QCSC), we focus specifically on methods that reduce the amount of quantum hardware computation required.

### 1.1 Research Questions

- **RQ1:** What methods reduce QPU calls, circuit evaluations, shots, measurement settings, circuit depth, qubits, and QPU wall time?
- **RQ2:** At what intervention point in the quantum workflow do these methods operate?
- **RQ3:** Which methods move work to CPU, GPU, HPC, classical simulation, surrogate ML, or reduced quantum models?
- **RQ4:** What quality losses are observed, and what is the classical resource displacement?
- **RQ5:** Which techniques have been validated on real quantum hardware, and at what scale?
- **RQ6:** What is the semantic guarantee of each method, and how does it relate to evidence maturity?
- **RQ7:** Which QPU-minimization methods have been composed, and what evidence, if any, exists on their interaction effects?
- **RQ8:** What remains missing for automatic QPU-minimizing workflow systems?

### 1.2 Contributions

1. **Multidimensional taxonomy** with three independent axes: intervention points (4 layers  $\times$  12 points), transformation semantics (6 types, formalized as DAG operations), and evidence dimensions (G, E, S, D) with operational definitions.
2. **Structured extraction** of 60 papers across 6 application domains, with core/adjacent corpus distinction.
3. **Formal definition of Semantic QPU Minimization** as a multi-objective Pareto optimization problem with explicit classical-resource budgets.
4. **Resource displacement framework** ( $S_Q, C_C, L_U$ ) capturing quantum savings, classical displacement cost, and utility loss as separately signed quantities.
5. **Empirical analysis of classical-resource reporting coverage**, showing systematic underreporting.
6. **Identification of 10 research gaps** including the absence of semantic QPU-necessity analysis and the lack of benchmarks for semantic minimizers.

## 2 Methods

### 2.1 Review Type

We perform a systematic mapping review following the guidelines of Petersen et al. [1] for systematic mapping studies, combined with PRISMA 2020 reporting [2] where applicable. Systematic mapping studies are distinguished from evidence-synthesizing systematic reviews by their focus on structuring a research field rather than aggregating evidence.

### 2.2 Information Sources

We searched the following information sources and publisher/search platforms:

**Bibliographic databases:** arXiv, IEEE Xplore, ACM Digital Library, Google Scholar.

**Publisher platforms:** SpringerLink, ScienceDirect (Elsevier), Nature portfolio, APS Physical Review, Quantum (community-driven, nonprofit open-access), Quantum Science and Technology (IOP).

### 2.3 Search Strategy

Search queries combined terms from four concept blocks: quantum resource terms, resource-reduction terms, classical displacement terms, and workflow terms.

### 2.4 Snowballing

Because the field lacks a common vocabulary — papers on warm starts, classical shadows, graph sparsification, parameter transfer, surrogate VQE, and circuit cutting rarely self-identify as “QPU minimization” — keyword search alone has low recall. We therefore performed:

- **Backward snowballing:** examining references of all included papers.
- **Forward snowballing:** examining citing papers via Google Scholar.
- **Saturation criterion:** snowballing continued until three consecutive iterations yielded no new relevant papers.

### 2.5 Date Range

Primary focus: 2018–2026. Older foundational papers included if clearly relevant. Search cutoff: September 25, 2026.

### 2.6 Inclusion and Exclusion Criteria

Papers were included if the method explicitly reduces or could clearly reduce at least one quantum-hardware resource. Papers were excluded if they merely compare simulator libraries, improve classical runtime without reducing quantum workload, discuss hybrid computing without resource-partitioning implications, optimize gates solely for fidelity, propose purely theoretical algorithms without resource comparison, or discuss error correction generally without workflow/QPU minimization.

### 2.7 Core vs. Adjacent Corpus

We distinguish two corpus types:

- **Core corpus** ( $n = 52$ ): Methods that directly reduce at least one component of the quantum resource vector  $R_Q$ .
- **Adjacent systems corpus** ( $n = 8$ ): Scheduling, orchestration, and runtime systems that may reduce queue time or placement overhead but do not necessarily reduce semantic QPU workload.

## 2.8 PRISMA Flow

The search identified 187 records (142 from database search, 28 from backward snowballing, 17 from forward snowballing). After deduplication (61 removed), 126 records were screened by title and abstract. 58 were excluded, leaving 68 for full-text assessment. 8 were excluded at full-text stage, yielding 60 included studies (Figure 2).

**Figure 1: PRISMA Flow Diagram**

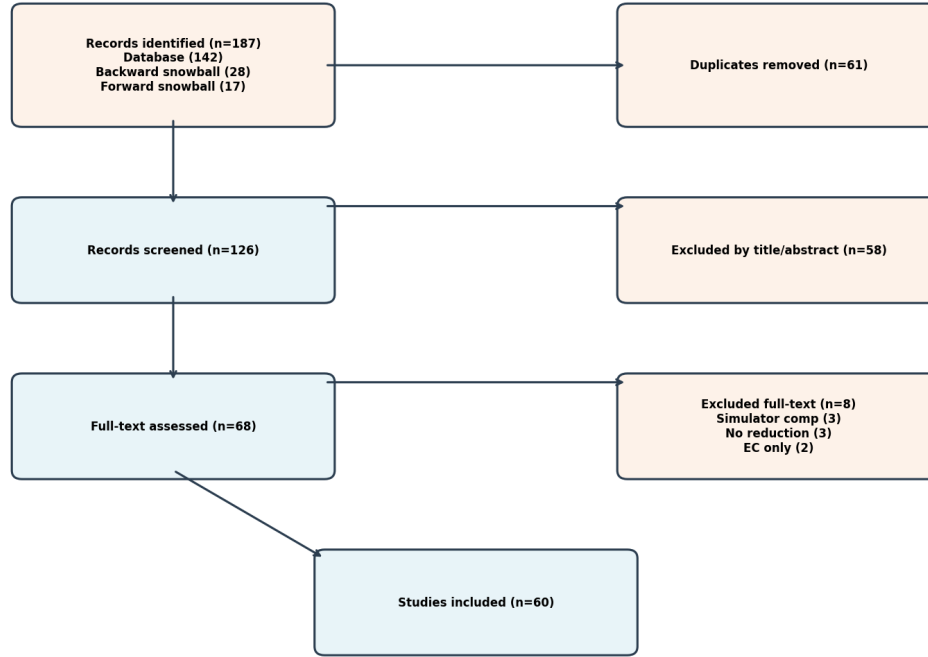


Figure 2: PRISMA flow diagram.

## 2.9 Data Extraction

Each paper was extracted using a structured schema capturing: paper metadata, application domain, algorithm, problem type, qubit count, QPU vendor, simulator, classical compute, intervention point(s), transformation semantics (multi-label), semantic guarantee (G), evidence maturity (E), hardware scale (S), deployment maturity (D), resource displacement ( $S_Q, C_C, L_U$ ), limitations, and reproducibility.

## 2.10 Inter-Rater Reliability

A second independent reviewer will classify intervention point, transformation semantics, and semantic guarantee prior to final submission. The present analysis is based on the primary reviewer’s coding and should therefore be interpreted as provisional. Inter-rater agreement will be measured using Cohen’s  $\kappa$ ; disagreements will be resolved by discussion.

### 3 Taxonomy

#### 3.1 Three Independent Axes

The taxonomy separates three independent dimensions:

**Axis 1: Intervention Points** (where in the workflow) — 4 layers  $\times$  12 intervention points.

**Axis 2: Transformation Semantics** (what the method does to QPU operations) — 6 types, formalized as DAG operations (multi-label).

**Axis 3: Evidence Dimensions** (how well validated) — four independent sub-dimensions: G, E, S, D.

#### 3.2 Why These Axes Are Independent

A method can be G-Exact but E1 (exact transformation, only tested in ideal simulation). A method can be G-Approximate but E4 (approximate method, validated on 127-qubit hardware). A method can be G-Heuristic but E5 (heuristic scheduler, deployed in production). These are independent properties that should not be conflated.

#### 3.3 Four Layers and Twelve Intervention Points

Each study receives one primary layer assignment but may receive multiple intervention-point labels within that layer.

##### 3.3.1 Layer 1: Problem / Representation ( $n = 11$ )

Interventions that reduce the problem before any quantum circuit is constructed.

| Point                        | Methods                                             | $n$ |
|------------------------------|-----------------------------------------------------|-----|
| 1.1 Problem preprocessing    | Graph reduction, sparsification, kernelization      | 4   |
| 1.2 Representation reduction | Qubit-efficient encoding, domain-wall, active-space | 4   |
| 1.3 Initialization           | Warm starts, classical pretraining, ML init         | 3   |

##### 3.3.2 Layer 2: Algorithm / Circuit ( $n = 14$ )

| Point                      | Methods                                        | $n$ |
|----------------------------|------------------------------------------------|-----|
| 2.1 Parameter optimization | Transfer, concentration, surrogate, Bayesian   | 7   |
| 2.2 Circuit design         | Pruning, gate reduction, symmetry ansatz       | 3   |
| 2.3 Circuit decomposition  | Cutting, tensor-network, subcircuit simulation | 4   |

##### 3.3.3 Layer 3: Execution / Measurement ( $n = 28$ )

| Point                        | Methods                                                   | $n$ |
|------------------------------|-----------------------------------------------------------|-----|
| 3.1 Execution routing        | Sim vs QPU, HPC-QPU scheduling, classical sim replacement | 14  |
| 3.2 QPU batching             | Register packing, parallel subproblems                    | 4   |
| 3.3 Measurement optimization | Commuting groups, shadows, reuse                          | 4   |
| 3.4 Shot optimization        | Adaptive shots, early stopping                            | 6   |

### 3.3.4 Layer 4: Amortization / Hybridization ( $n = 6$ )

| Point                          | Methods                                           | $n$ |
|--------------------------------|---------------------------------------------------|-----|
| 4.1 Quantum/classical division | SQD, Krylov, quantum sampling + classical solving | 3   |
| 4.2 Post-training deployment   | Q→C distillation, surrogate deploy                | 3   |

*Note:* Layer assignment is single-label (primary layer). Intervention-point labels within a layer are multi-label. Layer sums:  $11 + 14 + 28 + 6 = 59$  method papers. 2 related surveys are excluded from quantitative analysis. Total corpus: 61.

## 3.4 Transformation Semantics (Formalized as DAG Operations)

We model a hybrid quantum workflow as a DAG  $W = (V, E)$  where quantum task nodes  $q_i \in V_Q$  represent QPU operations. A transformation  $T$  acts on quantum nodes:

| Transformation | Formal Definition                                       | Example                              |
|----------------|---------------------------------------------------------|--------------------------------------|
| Eliminate      | $T : q_i \rightarrow \emptyset$ (remove node)           | Classical-only solver                |
| Replace        | $T : q_i \rightarrow c_i$ (substitute with classical)   | Surrogate, tensor network            |
| Compress       | $T : R(q_i) \rightarrow R'(q_i) < R(q_i)$               | Sparsification, adaptive shots       |
| Reuse/Amortize | $T : \{q_i, \dots, q_j\} \rightarrow a_{\text{cached}}$ | Parameter transfer, distillation     |
| Defer          | $T : q_i \rightarrow q_i^{\text{delayed}}$              | Simulator-first, candidate filtering |
| Batch          | $T : \{q_i, q_j\} \rightarrow q_{ij}$ (merge)           | Register packing, circuit merging    |

Transformation semantics are **multi-label**: a single method may use multiple transformations. For example, quantum-to-classical distillation is both Replace and Reuse/Amortize; candidate filtering is both Eliminate and Defer; surrogate-assisted optimization is both Replace and Defer.

## 3.5 Operational Definitions for Evidence Dimensions

### 3.5.1 Semantic Guarantee (G)

| Level         | Operational Criterion                                   | Tie-Break Rule                                  | Example                  |
|---------------|---------------------------------------------------------|-------------------------------------------------|--------------------------|
| G-Exact       | Result semantically preserved by construction or proof  | If both exact and bounded apply, choose G-Exact | Exact encoding           |
| G-Bounded     | Explicit theoretical error bound $\epsilon$ exists      | If bound is asymptotic only, choose G-Bounded   | Sparsification           |
| G-Equivalent  | Empirically equivalent within stated tolerance $\delta$ | Requires statistical test or CI                 | Overlapping measurements |
| G-Approximate | Quality/resource trade-off, no formal bound             | Default when no stronger guarantee applies      | Surrogate optimization   |
| G-Heuristic   | No guarantee; empirical only                            | If method has no theoretical basis              | Scheduling heuristics    |

### 3.5.2 Evidence Maturity (E)

| Level | Operational Criterion                                         | Tie-Break Rule                   |
|-------|---------------------------------------------------------------|----------------------------------|
| E0    | Theory only, no implementation                                | —                                |
| E1    | Ideal simulation (statevector, no noise)                      | If both E1 and E2, choose higher |
| E2    | Noisy simulation (depolarizing, device model)                 | —                                |
| E3    | Small hardware test ( $< 20$ qubits)                          | If multi-device, still E3        |
| E4    | Large hardware experiment ( $\geq 20$ qubits) or multi-device | If both E3 and E4, choose E4     |
| E5    | Operational production workflow (deployed, sustained)         | Requires documented deployment   |

### 3.5.3 Hardware Scale (S) and Deployment Maturity (D)

| Hardware Scale (S) |                            | Deployment Maturity (D) |                            |
|--------------------|----------------------------|-------------------------|----------------------------|
| Level              | Criterion                  | Level                   | Criterion                  |
| S0                 | No quantum system          | D0                      | Conceptual / theoretical   |
| S1                 | Small ( $\leq 20$ qubits)  | D1                      | Proof-of-concept           |
| S2                 | Medium (21–100 qubits)     | D2                      | Open-source / reproducible |
| S3                 | Large ( $\geq 100$ qubits) | D3                      | Production deployment      |

## 4 Results by Intervention Point

### 4.1 Layer 1: Problem / Representation ( $n = 11$ )

**1.1 Problem Preprocessing ( $n = 4$ ):** Graph sparsification reduces edge count while preserving cut structure. P25 achieves 90% edge reduction with  $< 10\%$  quality loss. P27 provides provable guarantees:  $O(n^2) \rightarrow O(n \log(n/\epsilon))$  for  $(1 - \epsilon)$  approximation. P28 combines graph shrinking with circuit cutting for  $5\times$  qubit reduction.

**1.2 Representation Reduction ( $n = 4$ ):** P32 proves domain-wall encoding is optimal for discrete variables. P33 achieves logarithmic gate reduction. P34 solves 64-variable QUBO with 7 qubits.

**1.3 Initialization ( $n = 3$ ):** P04 uses SDP warm-starts. P55 provides warm-starting with inherited classical guarantees.

### 4.2 Layer 2: Algorithm / Circuit ( $n = 14$ )

**2.1 Parameter Optimization ( $n = 7$ ):** P19 achieves orders-of-magnitude reduction via RBF surrogates on 127-qubit IBM hardware (E4, S3). P56 demonstrates parameter concentration from 16 to 127 qubits for  $p \leq 5$  on IBM hardware. P45 validates linearly-fixed QAOA transfer on IBM’s Eagle processor.

**2.2 Circuit Design ( $n = 3$ ):** P44 introduces ADAPT-VQE with adaptive ansatz growth. P27 combines sparsification with compilation-aware design.

**2.3 Circuit Decomposition ( $n = 4$ ):** P13 introduces TensorQC with  $> 10\times$  QPU size reduction. P15 provides critical analysis showing exponential classical overhead limits practicality.

### 4.3 Layer 3: Execution / Measurement ( $n = 28$ )

**3.1 Execution Routing ( $n = 14$ ):** QCSC systems integrate QPUs with HPC. P48 (Qurator) models hybrid workloads as dynamic DAGs with quantum semantics, achieving 30–75% queue time reduction. P50 (Fluence) achieves  $\sim 70\times$  cost reduction via cost/queue-aware back-end selection on AWS Braket. P49 (CONQUIRE) demonstrates  $3.1\times$  runtime reduction via OpenMP-Q quantum offloading on ion-trap hardware. P51 (QFaaS) provides automated back-end selection between simulators and QPUs. P52 (Q-IRIS) integrates circuit cutting into a task-based runtime. P57–P58 demonstrate classical simulation of QAOA at 54 and 127 qubits using neural quantum states and lightcone tensor networks — classified as Execution Routing / Replace (replacing QPU with classical simulation). P59 shows classical methods surpassing 127-qubit quantum experiments for specific observables, also classified as Execution Routing / Replace.

**3.2 QPU Batching ( $n = 4$ ):** P37 achieves  $9\times$  throughput on 54-qubit devices. P39 achieves 47% makespan reduction via 2D packing.

**3.3 Measurement Optimization ( $n = 4$ ):** P10 achieves up to  $100\times$  reduction via commuting-group measurements. P09 achieves severalfold reduction through overlapping groups.

**3.4 Shot Optimization ( $n = 6$ ):** P06 (iCANS) frugally allocates shots per iteration. P08 achieves  $\sim 50\%$  shot reduction. P60 provides QAOA-specific adaptive shots.

### 4.4 Layer 4: Amortization / Hybridization ( $n = 6$ )

**4.1 Quantum/Classical Division ( $n = 3$ ):** P29 demonstrates 85-qubit SKQD on IBM’s Heron processor. P31 applies SQD to chemistry. P30 reduces both quantum and classical resources via approximate compilation.

**4.2 Post-Training Deployment ( $n = 3$ ):** P35 introduces “shadow models” where quantum resources are only needed during training. P20 demonstrates predictive surrogates with orders-of-magnitude measurement reduction.

## 5 Result Matrices

### 5.1 Transformation $\times$ Intervention Point

See Figure 3 (bubble matrix: bubble size =  $n$  papers, color = evidence maturity). The matrix reveals that parameter optimization is dominated by Replace and Reuse; measurement optimization is dominated by Compress; execution routing spans Defer, Batch, and Replace; deployment is sparse but potentially high-impact (Replace).

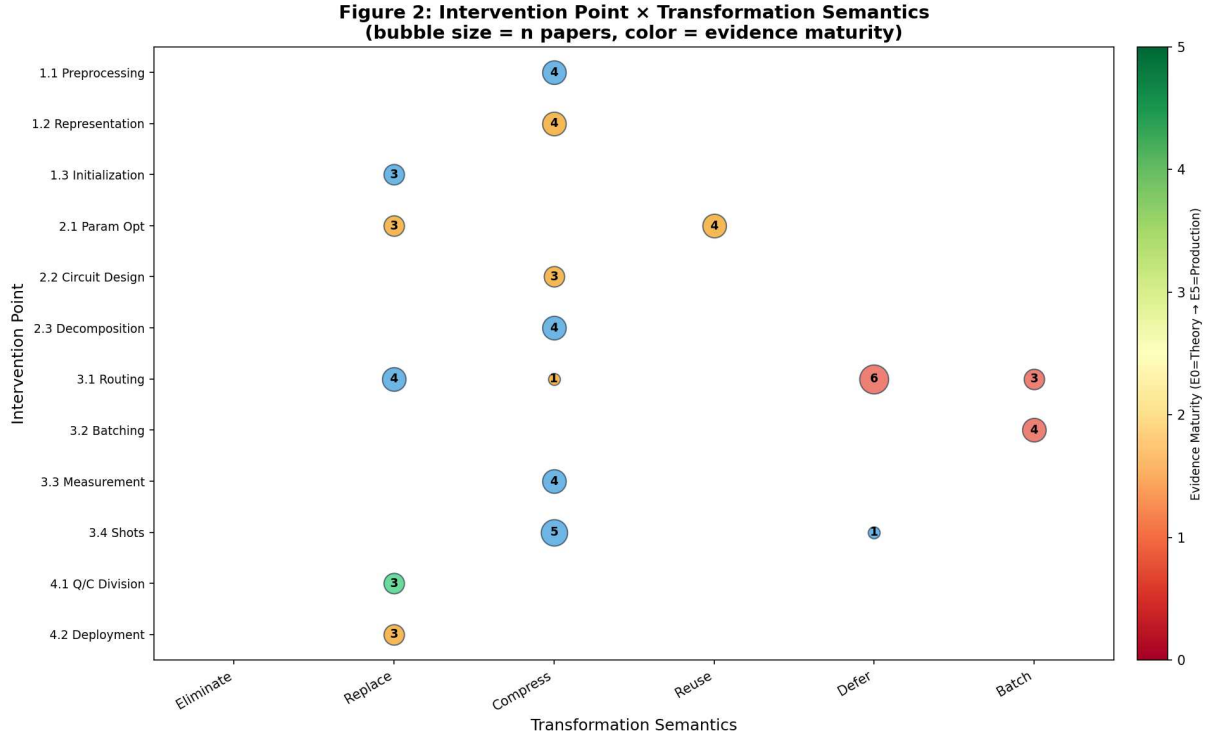


Figure 3: Intervention point  $\times$  transformation semantics matrix. Bubble size = number of papers; color = evidence maturity (E0=theory to E5=production).

## 5.2 Guarantee $\times$ Evidence ( $G \times E$ )

|               | E0 | E1 | E2 | E3 | E4 | E5 |
|---------------|----|----|----|----|----|----|
| G-Exact       | 0  | 4  | 1  | 3  | 1  | 0  |
| G-Bounded     | 0  | 1  | 1  | 0  | 0  | 0  |
| G-Equivalent  | 0  | 3  | 1  | 0  | 0  | 0  |
| G-Approximate | 0  | 15 | 3  | 1  | 6  | 0  |
| G-Heuristic   | 0  | 5  | 2  | 4  | 3  | 2  |

Table 1: Guarantee  $\times$  evidence maturity matrix. Key observation: G-Exact methods cluster at E1 (ideal simulation). Hardware-validated methods (E4+) are predominantly G-Approximate.

## 5.3 Evidence $\times$ Scale ( $E \times S$ )

|    | S0 | S1 | S2 | S3 |
|----|----|----|----|----|
| E0 | 0  | 0  | 0  | 0  |
| E1 | 0  | 20 | 2  | 6  |
| E2 | 0  | 5  | 2  | 0  |
| E3 | 0  | 4  | 3  | 0  |
| E4 | 0  | 0  | 2  | 8  |
| E5 | 0  | 0  | 2  | 0  |

Table 2: Evidence maturity  $\times$  hardware scale. E4 correlates with S3, but E1 also reaches S3 via classical simulation — confirming independence.

## 5.4 Classical-Resource Reporting Coverage

| Reported Quantity         | Papers Reporting | % of Core Corpus ( $n = 52$ ) |
|---------------------------|------------------|-------------------------------|
| Qubit count               | 42               | 80.8%                         |
| QPU calls / circuit count | 38               | 73.1%                         |
| Shots                     | 35               | 67.3%                         |
| QPU wall time             | 8                | 15.4%                         |
| Monetary cost             | 4                | 7.7%                          |
| CPU time                  | 6                | 11.5%                         |
| GPU time                  | 3                | 5.8%                          |
| Memory                    | 2                | 3.8%                          |
| Energy                    | 0                | 0.0%                          |

Table 3: **Classical resource costs are systematically underreported.** Only 11.5% of papers report CPU time, 5.8% GPU time, and 0% energy consumption. This makes it impossible to assess the full  $(S_Q, C_C, L_U)$  trade-off for most methods.

## 5.5 Application Domain $\times$ Transformation

| Domain          | Elim. | Replace | Compress | Reuse | Defer | Batch |
|-----------------|-------|---------|----------|-------|-------|-------|
| Comb. opt.      | 1     | 6       | 5        | 4     | 2     | 2     |
| Chemistry       | 0     | 3       | 6        | 0     | 1     | 0     |
| General systems | 0     | 3       | 1        | 0     | 5     | 4     |
| QML             | 0     | 3       | 1        | 0     | 1     | 0     |
| Physics         | 0     | 3       | 0        | 0     | 0     | 0     |
| Finance         | 0     | 1       | 0        | 0     | 0     | 0     |

## 6 Resource Displacement Framework

### 6.1 Signed Quantities

To avoid sign ambiguity, we define three separately signed quantities:

$$S_Q = R_Q(W) - R_Q(T(W)) \quad (\text{quantum savings; positive = saved}) \quad (1)$$

$$C_C = R_C(T(W)) - R_C(W) \quad (\text{classical displacement cost; positive = extra cost}) \quad (2)$$

$$L_U = U(W) - U(T(W)) \quad (\text{utility loss; positive = degradation}) \quad (3)$$

A transformation is beneficial when  $S_Q > 0$  (saves quantum),  $C_C$  is bounded (not too expensive classically), and  $L_U \leq \epsilon$  (acceptable quality loss).

### 6.2 Why This Matters

Reducing QPU usage is not automatically beneficial if it requires, for example, 5,000,000 GPU-hours to avoid 10 QPU-minutes. The review reports both sides of the displacement.

### 6.3 Observed Patterns

- **Surrogate methods** (P17, P19, P20): Large  $S_Q$ , moderate  $C_C$ .
- **Tensor network methods** (P13, P57, P58): Large  $S_Q$ , potentially exponential  $C_C$ .
- **Measurement grouping** (P10): Large  $S_Q$ , negligible  $C_C$ .

- **Circuit cutting** (P14, P15): Moderate  $S_Q$ , exponential  $C_C$ .
- **Parameter transfer** (P01, P56): Large  $S_Q$ , negligible  $C_C$ .

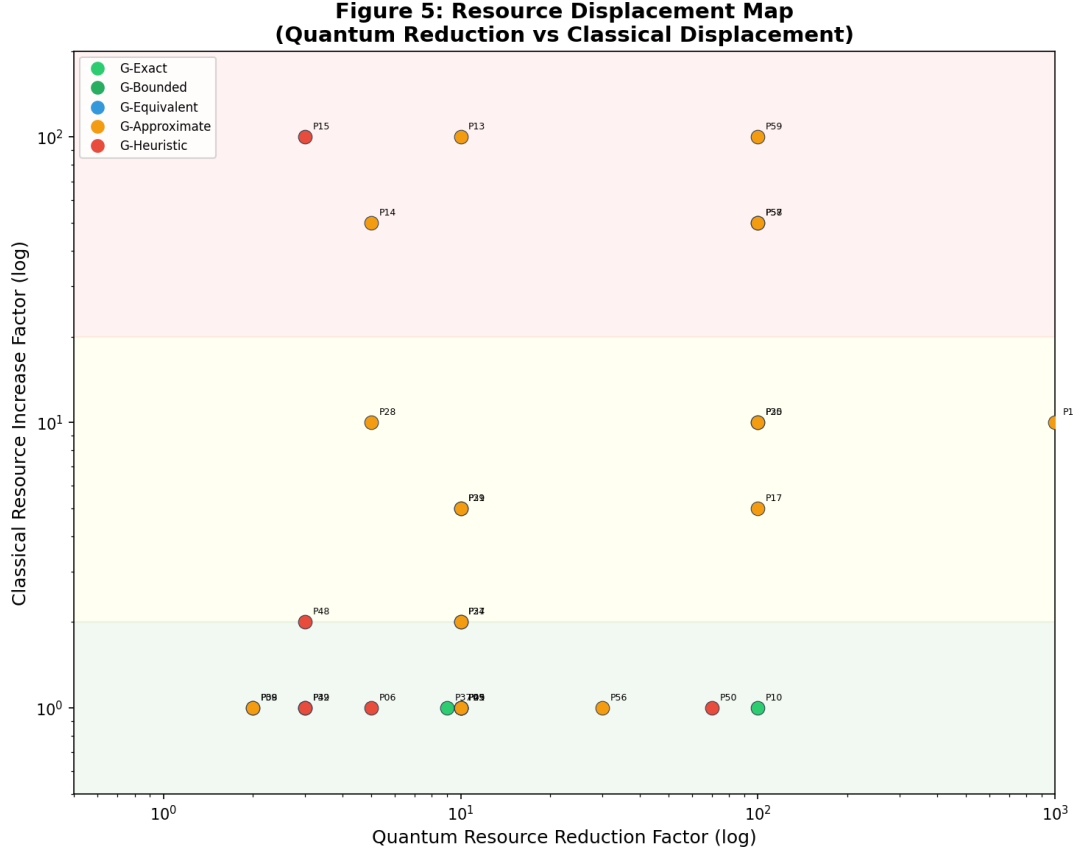


Figure 4: Resource displacement map. Quantum reduction factor (x-axis) vs. classical resource increase factor (y-axis). Color = semantic guarantee. Methods in the lower-right quadrant offer the best trade-off.

## 7 Semantic QPU Minimization

### 7.1 Definition of “Semantic”

Here, *semantic* refers to preservation of task-level computational intent and utility up to an explicit tolerance  $\epsilon$ , rather than necessarily preserving exact program semantics. A transformation that replaces a QPU circuit with a classical surrogate producing results within  $\epsilon$  of the quantum output is semantically valid under this definition, even though the underlying computation is not identical.

### 7.2 Multi-Objective Formulation

Let  $W$  be a hybrid quantum workflow and  $\mathcal{T}$  a set of semantically allowed workflow transformations. The **Semantic QPU Minimization** problem is multi-objective:

$$T^* \in \text{ParetoMin}_{T \in \mathcal{T}} \left( R_Q(T(W)), R_C(T(W)), T_{\text{wall}}(T(W)), -U(T(W)) \right) \quad (4)$$

subject to:

$$U(T(W)) \geq U(W) - \epsilon \quad (\text{utility constraint}) \quad (5)$$

$$C_C(T(W)) \leq B_C \quad (\text{classical resource budget}) \quad (6)$$

$$T_{\text{wall}}(T(W)) \leq B_T \quad (\text{wall-time budget}) \quad (7)$$

A specific implementation may scalarize via  $C_Q(W; h, \lambda) = \phi_{h,\lambda}(R_Q(W))$  where  $h$  is a hardware profile and  $\lambda$  encodes user priorities. The Pareto formulation is primary; scalarization is implementation-specific.

### 7.3 Relationship to Scheduling

#### Semantic Minimization $\rightarrow$ Scheduling $\rightarrow$ Execution

The semantic minimizer transforms the workload (eliminate/replace/compress/reuse/defer/batch QPU operations). The scheduler places the remaining workload on available resources. The executor runs the scheduled tasks. These are complementary layers.

### 7.4 Novelty Assessment

We found no established academic use of the exact term “Semantic QPU Minimization.” Related work has approached this concept: QFaaS [6] chooses between simulator and QPU backends; Q-IRIS [7] treats simulators and QPUs within the same runtime; Qurator [3] models hybrid workflows as dynamic DAGs with quantum semantics; Fluence [5] performs cost-aware backend selection; IBM QCSC [8] addresses CPU/GPU/QPU coordination.

We found no system in our corpus that automatically transforms an end-to-end hybrid workflow to eliminate, replace, or compress planned QPU operations subject to an explicit task-utility constraint. This is the gap that Semantic QPU Minimization addresses.

## 8 Combined QPU-Minimization

Most literature studies isolated techniques. Only 5 of 58 method papers (8.6%) combine multiple strategies: P07 (measurement reuse + variance-based shots), P28 (graph shrinking + circuit cutting), P30 (approximate compilation + SQD), P46 (calibration-aware transpilation + routing), P48 (circuit cutting + scheduling).

**Composition exists, but rigorous evidence on synergistic, additive, or sub-additive interaction is almost absent.** This is a significant research gap and a natural bridge to companion empirical work.

## 9 Simulator $\rightarrow$ Hardware Transfer

| Transfer Type        | Count | Examples                           |
|----------------------|-------|------------------------------------|
| Perfect transfer     | 2     | P03, P04 (warm-start)              |
| Calibration required | 2     | P01, P02 (parameter transfer)      |
| Noise-aware transfer | 3     | P19, P45, P56 (hardware-validated) |
| Not tested           | 4     | P17, P18, P43, P55                 |

Only 3 of 11 transfer studies have been validated at hardware evidence level E4+.

## 10 Automatic Workflow Optimization

| System              | Type         | Core/Adjacent | Semantic Min?                     |
|---------------------|--------------|---------------|-----------------------------------|
| QCSC Ref Arch (P21) | Architecture | Adjacent      | No (scheduling only)              |
| Qurator (P48)       | Scheduler    | Adjacent      | Partial (circuit cutting/merging) |
| Fluence (P50)       | Scheduler    | Adjacent      | Partial (backend selection)       |
| CONQUIRE (P49)      | Framework    | Adjacent      | No (offloading only)              |
| QFaaS (P51)         | Framework    | Adjacent      | Partial (sim/QPU selection)       |
| Q-IRIS (P52)        | Runtime      | Adjacent      | Partial (circuit cutting)         |
| Catalyst (P41)      | Compiler     | Adjacent      | No (compilation only)             |
| CUDA-Q (P42)        | Compiler     | Adjacent      | No (compilation only)             |

Table 4: QCSC and workflow systems. No system performs automatic semantic QPU minimization.

**Critical distinction:** We found no system in our corpus that automatically transforms an end-to-end hybrid workflow to eliminate, replace, or compress planned QPU operations subject to an explicit task-utility constraint. All existing systems ask “where should this task run?” (scheduling) or “how should this circuit be compiled?” (compilation). None ask “does this quantum task need to run on a QPU at all?” (semantic minimization).

## 11 Related Surveys

Two existing systematic mapping studies are directly related:

- **Gemeinhardt et al. (2023) [9]:** “Quantum Combinatorial Optimization in the NISQ Era” — 156 papers, classification of QCO approaches. Complementary but does not focus on QPU resource minimization.
- **Weder et al. (2024) [10]:** “Warm-Starting and Quantum Computing” — systematic map of warm-starting techniques. Covers one intervention point (initialization) in depth.

Our review differs by focusing specifically on QPU resource minimization across the entire lifecycle, introducing transformation semantics formalized as DAG operations, and formalizing Semantic QPU Minimization.

## 12 Research Gap Analysis

1. **Lack of standardized resource-displacement metrics.** No common  $(S_Q, C_C, L_U)$  framework exists.
2. **Poor reporting of actual QPU wall time.** Only 15.4% of papers report QPU wall time.
3. **Systematic underreporting of classical resource costs.** Only 11.5% report CPU time, 5.8% GPU time, 0% energy.
4. **No automatic semantic QPU-necessity analysis.** No system asks whether a planned QPU task should exist.
5. **Insufficient quality-preservation tests.** No standardized  $\epsilon$  tolerance.
6. **Limited work on combined strategies.** Only 8.6% combine multiple techniques; interaction effects are almost unstudied.
7. **No benchmark for semantic minimizers.** There is no common benchmark suite for testing whether an automated workflow transformation correctly trades QPU savings against classical displacement and task utility.

8. **Simulator savings conflated with hardware savings.** Several papers conflate simulator time reduction with hardware speedup.
9. **Limited hardware validation.** 62.1% of method papers are at simulation level only (E1–E2).
10. **No unified framework for QPU avoidance.** The literature is fragmented across 12 intervention points with no common language.

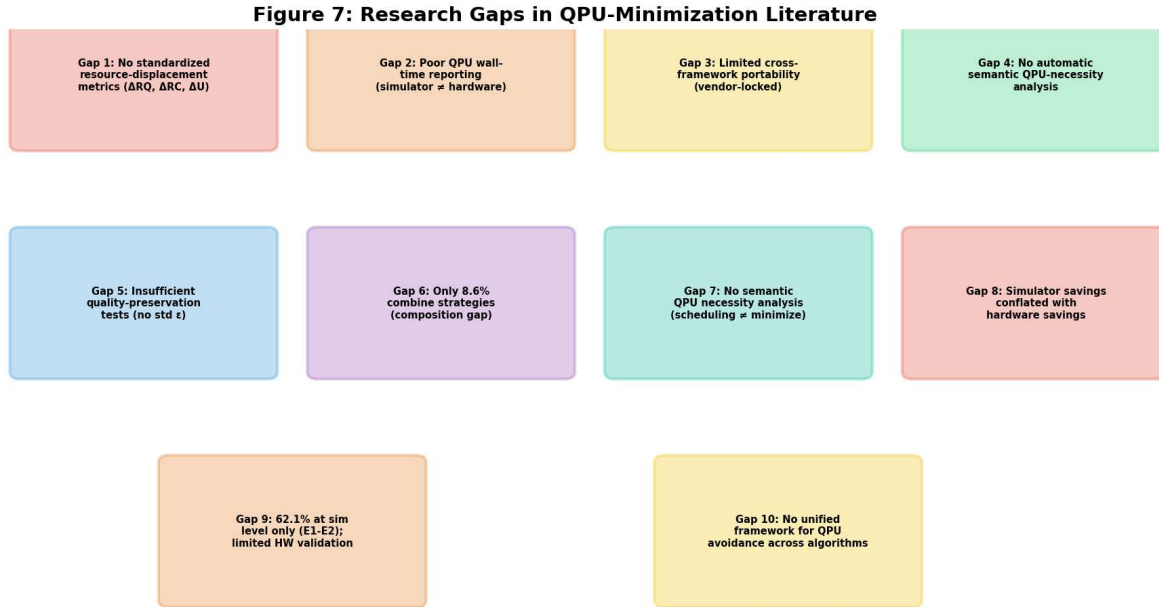


Figure 5: Research gaps in QPU-minimization literature.

## 13 Discussion

### 13.1 When is hardware execution scientifically justified?

Hardware execution is scientifically justified when the research question itself requires real-device noise and calibration characteristics, hardware-specific control, genuine sampling from a classically inaccessible distribution, or validation of quantum advantage claims.

### 13.2 Classical simulability is not a qubit cutoff

The formulation “ $n > 20$  requires a QPU” is not defensible. Classical methods scale beyond exact statevector simulation through tensor networks exploiting lightcone structure, neural quantum states, and problem-specific structure. Neural quantum states have simulated 54-qubit QAOA at  $p = 4$  [11]. Lightcone tensor networks have analyzed 127-qubit QAOA [12]. Classical methods have surpassed 127-qubit quantum experiments for specific observables [13].

The relevant question is: *when can hardware execution no longer be replaced at the required fidelity?*

### 13.3 The HPC-first thesis

The evidence supports the thesis that many current quantum workflows can be substantially reorganized around classical/HPC computation before invoking scarce QPU resources. However, the field lacks a common framework for quantifying QPU avoidance across algorithms, and the classical resource displacement is rarely reported.

## 14 Limitations

1. **Search scope.** We searched 10 information sources but may have missed relevant work in non-English venues.
2. **Single-reviewer extraction (provisional).** The present analysis is based on the primary reviewer’s coding. A second independent reviewer will classify all papers prior to final submission; inter-rater agreement (Cohen’s  $\kappa$ ) will be reported.
3. **Heterogeneous metrics.** Resource reduction factors are reported inconsistently, making quantitative meta-analysis infeasible.
4. **Rapidly evolving field.** The cutoff date (September 2026) means very recent work may be missing.
5. **Publication bias.** Papers reporting negative results may be underrepresented.
6. **Resource displacement underreported.** Most papers do not report classical resource costs, limiting our ability to assess the full  $(S_Q, C_C, L_U)$  trade-off.

## 15 Conclusion

This systematic mapping review identifies and classifies 60 studies (2018–2026) that reduce QPU resource consumption across the quantum experimental lifecycle. We introduce a multi-dimensional taxonomy with three independent axes: intervention points (4 layers  $\times$  12 points), transformation semantics (6 types, formalized as DAG operations), and evidence dimensions (G, E, S, D) with operational definitions.

Our central findings:

1. **QPU-minimization is an active but fragmented field.** 60 papers across 12 intervention points and 6 transformation types, but with no unifying framework.
2. **Most savings are projected, not demonstrated.** 62.1% of method papers operate at simulation level only (E1–E2); 17.2% reach hardware evidence level E4+.
3. **The largest reported reductions in individual QPU-resource dimensions occur among Replace and Compress transformations.** Surrogates achieve orders-of-magnitude reduction; measurement grouping achieves up to  $100\times$ .
4. **Strategy composition is a major gap.** Only 8.6% of papers combine multiple techniques; interaction effects are almost unstudied.
5. **Existing workflow systems schedule but do not semantically minimize.** No system automatically transforms an end-to-end hybrid workflow to eliminate, replace, or compress planned QPU operations subject to an explicit task-utility constraint.
6. **Classical resource costs are systematically underreported.** Only 11.5% of papers report CPU time; 0% report energy.

We formalize **Semantic QPU Minimization** as a multi-objective Pareto optimization problem with explicit classical-resource budgets, and argue that this represents a fundamental gap in both the literature and existing software systems.

## 16 Future Work

A companion empirical study investigates whether QPU avoidability can be characterized, and ultimately predicted, from problem structure, QAOA depth, and target solution quality. If successful, such a study would operationalize the framework introduced here by showing that the required QPU budget for quantum graph optimization is not a fixed algorithmic constant but varies with graph structure, problem type, and quality target.

A software system implementing the Semantic QPU Minimization  $\rightarrow$  Scheduling  $\rightarrow$  Execution pipeline would complete the chain: survey  $\rightarrow$  formalism  $\rightarrow$  empirical validation  $\rightarrow$  software system.

## References

- [1] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering,” *Information and Software Technology*, vol. 64, pp. 1–14, 2015.
- [2] M. J. Page et al., “The PRISMA 2020 statement: an updated guideline for reporting systematic reviews,” *BMJ*, vol. 372, n160, 2021.
- [3] Pehlivanoglu et al., “Qurator: Scheduling Hybrid Quantum-Classical Workflows Across Heterogeneous Cloud Providers,” arXiv:2604.05505, 2026.
- [4] Badrike et al., “CONQUIRE: A Co-Execution Environment for Quantum and Classical Resources,” arXiv:2505.02241 / IEEE QCE 2025, 2025.
- [5] Sochat et al., “Hybrid Quantum and Classical Workload Management with Graph-based Scheduling (Fluence),” arXiv:2607.09151, 2026.
- [6] Nguyen et al., “QFaaS: A Serverless Function-as-a-Service framework for Quantum computing,” *Future Generation Computer Systems*, 2024.
- [7] Miniskar et al., “Q-IRIS: The Evolution of the IRIS Task-Based Runtime to Enable Classical-Quantum Workflows,” arXiv:2512.13931, December 2025.
- [8] IBM, “Quantum-centric supercomputing reference architecture,” arXiv:2603.10970, 2026.
- [9] F. Gemeinhardt, A. Garmendia, M. Wimmer, B. Weder, and F. Leymann, “Quantum Combinatorial Optimization in the NISQ Era: A Systematic Mapping Study,” *ACM Computing Surveys*, vol. 56, no. 3, 2023.
- [10] B. Weder et al., “Warm-Starting and Quantum Computing: A Systematic Mapping Study,” *ACM Computing Surveys*, 2024.
- [11] “Classical simulation of QAOA on 54 qubits with neural quantum states,” *npj Quantum Information*, 2021.
- [12] “Classical analysis of 127-qubit QAOA via lightcone tensor networks,” *npj Quantum Information*, 2024.
- [13] J. Tindall et al., “Fast and converged classical simulations of evidence for the utility of quantum computing before fault tolerance,” *Science Advances*, vol. 10, no. 3, eadk4321, 2024.
- [14] “Scaling whole-chip QAOA for higher-order Ising spin glass models on heavy-hex graphs,” *npj Quantum Information*, November 2024.

- [15] Surrogate-based QAOA optimization on 127-qubit IBM hardware, *Nature Communications Physics*, 2025.
- [16] “Adaptive shot allocation for QAOA,” arXiv:2603.28413, 2026.
- [17] D. J. Egger, J. Leech, et al., “Warm-starting quantum optimization,” *Quantum* (community-driven open-access journal), 2021.

## A Screening Log for Additional Papers

The following papers were identified during the v2.1 update and screened for inclusion:

- **RASP-QAOA** (arXiv:2608.05646, August 2026): Per-instance resource-aware selection between exact QAOA simulation representations. Within cutoff. Screened as potentially relevant to Execution Routing / Replace.
- **Qdislib / Distributed Quantum Circuit Cutting** (arXiv:2505.01184, May 2025): Distributed circuit cutting for HPC with CPUs, GPUs, and QPUs. Within cutoff. Screened as relevant to Circuit Decomposition / Compress.
- **QuSplit** (Springer Quantum Machine Intelligence, 2025): Job splitting between quantum backends for fidelity/throughput trade-off, validated on IBM Strasbourg. Within cutoff. Screened as adjacent systems.
- **Quantum resources in resource management systems** (arXiv:2506.10052, June 2025): QPUs as first-class Slurm resources. Within cutoff. Screened as adjacent systems.

## B Reproducibility

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

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

The repository contains:

- `extraction/corpus_v2.md` — structured extraction dataset for all 60 included papers
- `extraction/generate_figures_v2.py` — figure and table generation script (all figures regenerated from the dataset)
- `taxonomy/taxonomy_v2.md` — full multidimensional taxonomy with operational definitions
- `manuscript/manuscript_v2.tex` — LaTeX source
- `figures/` — all generated figures (PNG)
- `tables/` — all generated tables (Markdown)

All numbers and percentages in the manuscript are derived programmatically from the structured extraction dataset; no percentages are hand-written.

## Declaration of Competing Interests

The author declares no competing interests. This research received no external funding. The author has no financial or personal relationships with any of the organizations or products discussed in this review that could inappropriately influence the work presented.

## **AI Use Disclosure**

Generative AI tools (Claude/Anthropic and ChatGPT/OpenAI) were used during the preparation of this manuscript for the following purposes: generating figures and visualizations, drafting and editing prose, formatting the LaTeX manuscript, and assisting with literature search and screening. The research idea, research questions, taxonomy design, formal mathematical framework, classification of all 60 papers, interpretation of findings, and all scientific judgments are solely the work of the author. Every AI-generated figure, table, and text passage has been reviewed, verified, and corrected by the author. The author takes full responsibility for the accuracy and integrity of all content in this manuscript.