Benchmarking Under Fragility in a Shifting Quantum Landscape

Nathan Kittichaikoonkij[1 [0009−0006−0266−6959]],
Nutthapat Pongtanyavichai[1 [0009−0008−2527−3730]],
Poopha Suwananek[1 [0009−0008−5973−9148]],
Prabhas Chongstitvatana[1 [0000−0003−0744−7801]], and
Kamonluk Suksen[1 [0009−0006−1452−6400]⋆]

Chulalongkorn University, Bangkok, Thailand
{6538047621, 6532068721, 6532141821}@student.chula.ac.th,
Prabhas.C@chula.ac.th, kamonluk@cp.eng.chula.ac.th

Abstract. Quantum computing holds promise speedups for many hard problems, especially in optimization, yet the practical evaluation of quantum algorithms is increasingly hindered by the volatility of cloud-based quantum services. In this study, we confront the challenge of reproducibility and infrastructure fragility through a case-based investigation comparing classical solvers (Gurobi, Fixstars) with D-Wave's quantum annealer. While our original intent was to benchmark solver performance on standard QUBO formulations, repeated disruptions—deprecations of IBM's Qiskit backends, inconsistencies in quantum API behavior, and unstable parameter mappings—reframed our effort into a study of methodological brittleness. We argue that this infrastructural instability is not an incidental inconvenience but a core research challenge in quantum benchmarking in a rapidly shifting computational landscape.

Keywords: Quantum Optimization · QUBO · Quantum Annealing.

1 Introduction

Quantum optimization via the Quadratic Unconstrained Binary Optimization (QUBO) has become more prevalent as its potential to outperform classcal solvers such as Gurobi and Fixstars on certain combinatorial optimization problems. Yet in practice, today's quantum benchmark landscape is increasingly fragile, cloud services evolve rapidly, APIs deprecate and software stacks changes which often renders previous reproducible experiments obsolete.

Despite rapid algorithmic progress, the practical landscape of quantum benchmarking remains fragile. Cloud-based quantum computing as a service (QCaaS) is inherently dynamic for instances: cloud services evolve rapidly, APIs are deprecated, and platform access models shift. These changes frequently compromise reproducibility, even for otherwise sound experimental designs.

⋆ Corresponding author

To address this challenge, the quantum benchmarking literature increasingly calls for standards that go beyond raw speed or accuracy. For instance, Hashim et al. [8] propose a three-tier framework encompassing quantum device characterization, verification of solution validity, and validation of application correctness. However, their focus remains primarily at the hardware level, without addressing software and platform volatility.

In our experiment, we attempted to benchmark QUBO solvers across three classes of problems—3SAT, Quadratic Assignment (QAP), and the Traveling Salesman Problem (TSP)—using classical (Gurobi, Fixstars) and quantum (D-Wave, Qiskit/QAOA) approaches. However, our efforts to deploy QAOA on IBM Qiskit were derailed by backend incompatibilities and API changes[1].

Despite our efforts to mitigate the problems by downgrading to earlier Qiskit versions, we continued to face backend incompatibilities and persistent failures in executing quantum algorithms. API updates frequently broke existing code, while backend services either became unavailable or exhibited new, undocumented behavior. These recurring disruptions not only made it difficult to maintain a consistent experimental environment but also undermined the reproducibility and reliability of our benchmarking efforts.

This paper argues that fragility constitutes not only an inconvenience, but a substantive research challenge in its own right. We reconceptualize our prior benchmark study within this broader context, demonstrating that the evaluation of QUBO solvers in an evolving quantum computing landscape reveals inherent vulnerabilities in reproducibility, fairness, and methodological integrity.

2 Related Work

2.1 Benchmarking Classical Solvers for QUBO Problems

Classical methods for solving QUBO problems rely on mature combinatorial optimization techniques. Gurobi is a widely used commercial solver that applies mixed-integer programming (MIP) with presolve strategies, cutting planes, and heuristics to handle binary quadratic forms [5]. Fixstars Amplify provides a GPU-accelerated simulated annealing engine tailored to QUBO problems, offering fast approximate solutions via thermal heuristics [4]. Other works have explored parallel tempering and metaheuristic hybrids for QUBO, although few studies benchmark these methods under shared problem formulations.

Codognet et al. compared digital annealing with D-Wave on QAP instances, showing that classical annealers can outperform quantum hardware depending on problem structure and embedding efficiency [1].

2.2 Quantum Annealing Benchmarks

Quantum annealing (QA) has been experimentally evaluated on problems like Max-Cut, 3SAT, and QAP [9]. The D-Wave Advantage series supports Ising

[1] https://quantum.cloud.ibm.com/docs/en/api/qiskit/release-notes

and QUBO formats using minor embedding to map logical variables to hardware qubits. However, embedding overhead and chain breakage remain significant limitations [2]. Villar-Rodriguez et al. conducted a large-scale sensitivity study, revealing that performance depends heavily on tuning parameters like `chain_strength`, `annealing_time`, and `schedule` [15].

These findings emphasize the need for controlled benchmarking methodologies that isolate performance factors across parameter sweeps.

2.3 Gate-Based Quantum Optimization

Gate-model solvers like QAOA (Quantum Approximate Optimization Algorithm) provide an alternative to annealing, encoding QUBO problems into parameterized quantum circuits [3]. While both implementations in Qiskit, and Cirq offer access to both simulators and real devices, CUDA-Q is focused on simulations and does not support real quantum devices. However, practical execution of QAOA remains constrained by circuit depth, noise, and calibration drift.

3 Background

3.1 QUBO and Ising Formulations

Combinatorial optimization problems often admit reformulations into the *Quadratic Unconstrained Binary Optimization* (QUBO) model, a standard mathematical structure in both classical and quantum computing [6]. A QUBO instance is defined by a real symmetric or upper triangular matrix $Q \in \mathbb{R}^{n \times n}$ and seeks a binary vector $x \in \{0,1\}^n$ minimizing the objective function:

$$E(x) = x^T Q x \tag{1}$$

This framework enables encoding of problems such as Max-Cut, 3SAT, QAP, and TSP by transforming constraints into penalty terms. For example, a constrained problem of the form $Ax = b$ can be absorbed into the QUBO cost using a quadratic penalty term:

$$E(x) = x^T Q x + \lambda \|Ax - b\|^2 \tag{2}$$

where $\lambda > 0$ controls the weight of the constraint penalty. The resulting function remains quadratic in x, allowing the entire problem to be expressed as a QUBO.

QUBO is equivalent to the Ising model, commonly used in quantum annealing. The transformation between the binary variable $x_i \in \{0,1\}$ and the Ising spin variable $s_i \in \{-1,+1\}$ is given by:

$$x_i = \frac{1 + s_i}{2} \tag{3}$$

Applying this transformation, the QUBO Hamiltonian becomes:

$$E(\mathbf{x}) = \sum_i Q_{ii} x_i + \sum_{i<j} Q_{ij} x_i x_j, \tag{4}$$

$$E(\mathbf{s}) = \sum_i Q_{ii} \left(\frac{1+s_i}{2}\right) + \sum_{i<j} Q_{ij} \left(\frac{1+s_i}{2}\right)\left(\frac{1+s_j}{2}\right) \tag{5}$$

$$= \sum_i \frac{Q_{ii}}{2}(1+s_i) + \sum_{i<j} \frac{Q_{ij}}{4}(1+s_i)(1+s_j) \tag{6}$$

$$E(\mathbf{s}) = \text{const} + \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j, \tag{7}$$

where h_i and J_{ij} are derived from Q, and the resulting energy landscape corresponds to the Ising Hamiltonian model [11].

3.2 Quantum Annealing Process

Quantum annealing (QA) is a metaheuristic that uses quantum tunneling to explore the solution space of discrete optimization problems. The annealing process begins with a driver Hamiltonian H_D whose ground state is easily prepared (often a transverse field). Over a time-dependent schedule, the system interpolates toward a problem Hamiltonian H_P encoding the QUBO (or Ising) cost function:

$$H(t) = A(t)H_D + B(t)H_P \tag{8}$$

The coefficients $A(t)$ and $B(t)$ define the annealing schedule, which generally satisfies $A(0) \gg B(0)$ and $A(T) \ll B(T)$, where T is the total annealing time. Under adiabatic conditions, the system remains in its instantaneous ground state throughout evolution [13].

Unlike classical simulated annealing, which relies on thermal noise to escape local minima, QA leverages quantum tunneling, enabling transitions across energy barriers that may trap classical solvers [14].

3.3 QAOA on Gate-based Quantum Computers

The Quantum Approximate Optimization Algorithm (QAOA) is a variational algorithm designed for gate-based quantum devices, inspired by adiabatic quantum computation and quantum annealing. QAOA approximates the solution to combinatorial optimization problems, such as QUBO or Ising models, by alternating between the application of a problem Hamiltonian H_P and a driver Hamiltonian H_D.

The QAOA circuit consists of p layers, each applying a unitary evolution under the Hamiltonians H_P and H_D with variational parameters γ_k and β_k:

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \prod_{k=1}^{p} e^{-i\beta_k H_D} e^{-i\gamma_k H_P} |+\rangle^{\otimes n} \tag{9}$$

Here, $|+\rangle^{\otimes n}$ is the uniform superposition over all computational basis states. The parameters $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ are optimized by a classical outer loop to minimize the expectation value of H_P [3].

In the limit $p \to \infty$, QAOA can theoretically reproduce the adiabatic trajectory of quantum annealing, but in practice, even shallow circuits often yield good approximate solutions. Unlike analog quantum annealing, QAOA is compatible with near-term devices, noisy gate-model quantum hardware and enables hybrid quantum-classical optimization [7].

4 Experiment

4.1 Solver Overview

We evaluated solver frameworks across combinatorial optimization problems: Fixstars Amplify, D-Wave Advantage, and IBM Qiskit QAOA. All solvers were accessed via official cloud APIs or SDKs using default configurations, unless otherwise stated.

While additional classical solvers, including Gurobi and brute-force baselines, were included in our full benchmark suite, we will not go into details for all of them. Full comparative results are reported in our prior work [10] and are excluded here for brevity and clarity of discussion.

Fixstars Amplify. A GPU-accelerated, quantum-inspired annealer that performs simulated annealing (SA) over QUBO-defined landscapes. It runs entirely on cloud hardware via REST API. Experiments used the default Simulated Annealing engine under the Basic Evaluation Plan.

D-Wave Quantum Annealing. Experiments were conducted using the Advantage System 6.4 quantum annealer through the D-Wave Leap cloud platform. We employed default parameters except for `num_reads`, which was set to 1000 to ensure sufficient sampling. Parameters such as `chain_strength` and `annealing_time` were left at default values in line with recommendations for initial benchmarking.

IBM Qiskit QAOA. Due to persistent execution failures, QAOA was excluded from the benchmark comparison and analyzed qualitatively in Section 5.

4.2 QUBO Model Formulations

Each combinatorial problem was encoded in QUBO form. For constraint-based problems (QAP and TSP), constraints were incorporated using quadratic penalty terms scaled by a fixed weight λ. The general QUBO structure takes the form:

$$H = H_O + \lambda g(x) \tag{10}$$

where H_O is the objective and $g(x)$ is the penalty function enforcing feasibility. Penalty weights were chosen based on empirical calibration to prevent solution distortion while preserving constraint enforcement.

4.3 Execution Environment and Protocols

All solvers were evaluated on identical problem instances generated using fixed seeds. D-Wave and Fixstars experiments were run via their respective cloud APIs. Solver performance was measured under the following protocols:

- Each solver was given a maximum runtime of 10 seconds per problem instance.
- Fixstars and D-Wave were used as-is through the Amplify SDK.
- D-Wave's `num_reads` parameter was set to 1000.
- Each reported result represents the average of 10 independent trials.

Performance was measured in terms of success rate, solution accuracy, and time-to-first-optimum (see Section 4.4 for metric definitions).

4.4 Experimental Metrics

To evaluate solver behavior, we recorded:

- **Feasibility Rate:** The proportion of runs returning syntactically valid solutions.
- **Solution Accuracy:** The percentage of solutions matching known optima or baselines.
- **Execution Time:** Average solver runtime per instance (ms).

4.5 Experiment Result

The results indicate that Fixstars achieves high accuracy with relatively short runtimes, showing better scalability than traditional solvers. Its performance on sparse problems like 3SAT remains consistently strong, and even dense problems like QAP are solvable up to moderate sizes.

In contrast, the D-Wave quantum annealer currently performs worse overall due to limitations in processing larger problem sizes on its hardware, effectively handling sparse problems like 3SAT but struggling with dense ones such as QAP and TSP because of minor embedding challenges and topology constraints.

Despite these limitations, D-Wave exhibits promising signs of linear time growth, with only modest increases as problem sizes scale. This trend is evident across all three problem types (Figures 1,2,3), highlighting its potential for future advancements in tackling complex combinatorial optimization efficiently.

These findings underscore the evolving role of quantum annealing in the NISQ era, emphasizing the need for hybrid approaches to overcome current hardware barriers and achieve broader quantum advantages.

Table 1 is adapted from our previous work [10].

Table 1. Summary of D-Wave results for 3SAT, QAP, and TSP

3SAT			QAP			TSP		
size	acc(%)	time (ms)	size	acc(%)	time(ms)	size	acc(%)	time(ms)
20	100.000	95.803	4	100	94.444	4	100	94.443
40	99.000	106.503	5	70	99.603	5	100	99.603
70	97.091	126.205	6	0	106.165	6	20	106.164
111	97.033	190.226				7	0	124.323
268	97.523	193.467						
400	95.661	211.567						
530	95.721	227.981						

5 Benchmark Fragility

Reproducibility is a well-established concern in quantum algorithm evaluation, but its root causes increasingly stem from infrastructural—not algorithmic—fragility. Mauerer & Scherzinger (2022) explicitly highlight the importance of reproducibility engineering in quantum software experiments. Their approach advocates for packaging code and configuration to remain traceable even when common hardware or vendor platforms change [12].

Our own experience supports this diagnosis. After initial success using D-Wave's Advantage System 6.4 through the Leap cloud interface, it could not be accessed with the same free-tier account after D-Wave releases as new pricing model. In the case of IBM Qiskit, we attempted to follow the official QAOA MaxCut tutorial[2], which was functional on simulators. However, adapting the same circuit logic to more general QUBO problems like TSP failed to yield valid results. Even on simulation, QAOA produced nonsensical tours for instances less than 5 nodes. This contrasts sharply with classical solutions obtained using NumPy and Scipy-based optimizers, which matched the known optimum. Furthermore, when transpiled and submitted to real quantum backends, jobs consistently failed to complete due to interactivity timeouts in the Qiskit Runtime environment. These timeouts likely stemmed from session mismanagement or delays exceeding the platform's job queuing threshold, which disconnects users from backend access during extended idle periods between QAOA iterations.

6 Discussion

Our benchmarking results reveal not only the relative performance of quantum and classical QUBO solvers, but also the operational fragility that underlies current quantum computing infrastructure—both at the hardware and software level. This section reflects on the empirical findings, methodological decisions,

[2] https://quantum.cloud.ibm.com/docs/en/tutorials/quantum-approximate-optimization-algorithm

and broader reproducibility implications, particularly in light of evolving quantum cloud services.

6.1 QAOA and Workflow Limitations in General QUBO Problems

Building on the practical failures discussed in Section 5, we now reflect on the broader implications these issues have for solver behavior, quantum benchmarking methodology. Although the Qiskit version (`1.3.1`) supported QAOA under the `qiskit.algorithms` and `qiskit.optimization` modules, these modules were removed from official qiskit support at the time of experimentation. They remained widely indexed in online documentation and tutorials, leading us to adopt an outdated implementation path unknowingly.

These failures were not due to quantum noise, or algorithmic instability, but to volatility at the software and platform layers. Such infrastructural disruptions pose a methodological threat to the integrity of longitudinal quantum experiments and must be addressed as a first-class concern in quantum benchmarking research.

6.2 Reproducibility and Structural Fragility in QCaaS Workflows

The broader insight from our benchmarking effort is the fragility of quantum workflows under the QCaaS paradigm. Access to D-Wave's cloud platform was inconsistent: after initial experiments using Advantage 6.4, subsequent team members were unable to access the same solver endpoints under the free-tier account. Similarly, software-layer volatility in Qiskit made it difficult to execute and generalize otherwise functional circuits. These issues were not due to algorithmic shortcomings, but rather to unstable access models, poorly maintained legacy APIs, and weak documentation pathways.

This supports the view that reproducibility in quantum computing must be treated as a systems-level concern. Fragility arises not only from quantum noise or hardware constraints but from the broader ecosystem—SDK evolution, cloud API policies, and the discoverability of supported workflows. Until these structural issues are addressed, benchmarking results must be interpreted within the context of their platform dependencies and temporal validity.

6.3 Toward Robust Benchmarking Standards

The fragility observed in quantum benchmarking is not an incidental artifact—it is a reproducible phenomenon that demands formal attention. As quantum computing transitions into an infrastructure-intensive discipline, benchmarking methodologies must evolve to address both algorithmic performance and systems-level resilience.

To that end, we propose a four-pronged framework for designing robust quantum benchmarks that are resilient to evolving APIs, deprecations, and cloud-access variability. This framework aims to standardize practices across research teams and improve reproducibility in future studies.

1. **Standardized Baselines:** Adopt shared, publicly available QUBO formulations with open-source reference implementations and solution sets. These should include canonical problems like Max-Cut, 3SAT, QAP, and TSP across defined sizes (e.g., 4–20 variables). Publishing not just problem definitions but also embedding logic and parameter settings ensures consistent testing.
2. **Snapshot-Aware Experimentation:** Every benchmark run should log the full software and platform stack, including:
   - Backend name and version (e.g., `advantage_6.4`, `ibmq_brisbane`)
   - SDK version (e.g., `qiskit==1.3.1`)
   - API changes or warnings during execution
   - Execution date, user tier (free, pay as you go, premium, etc), and region (when relevant)

   These metadata form a "benchmark snapshot" that, while not guaranteeing full reproducibility on real quantum hardware—due to ongoing backend evolution and calibration drift—enable more reliable re-execution in simulators and simplify migration to updated software stacks.
3. **Resilient Methodology:** Benchmarks should emphasize robustness over narrow tuning. Parameter sensitivity (e.g., D-Wave's `chain_strength`, annealing time) should be reported as distributions or sweeps, not single values. Similarly, use QUBO encodings that are platform-agnostic—avoiding reliance on proprietary transpilation steps when possible.
4. **Benchmarking the Benchmarks:** Core benchmark suites should be re-executed periodically (e.g., quarterly or annually) to measure longitudinal drift in performance or compatibility. This meta-benchmarking helps detect when infrastructure evolution introduces silent errors, regressions, or improvements.

Together, these components shift benchmarking from a one-off evaluation to a reproducible and portable protocol. In future work, we envision a federated benchmarking registry—akin to MLPerf in machine learning—that tracks quantum benchmark scores, version metadata, and known failure modes.

Finally, we recommend that platform-layer failure modes (e.g., runtime errors, silent crashes, or API deprecations) be explicitly reported in benchmark papers, not discarded as outliers. Treating infrastructure behavior as part of the benchmark result will accelerate progress toward both trustworthy evaluations and robust software-hardware co-design.

7 Conclusion

This study revisits the problem of benchmarking quantum and classical solvers on combinatorial optimization tasks encoded as QUBO models. Our original goal was to evaluate D-Wave's quantum annealing system alongside classical baselines and gate-based quantum algorithms. While our experiments did succeed

in comparing D-Wave, Fixstars, Gurobi, and Brute Force on standard benchmarks (3SAT, QAP, TSP), a deeper insight emerged: the fragility of quantum benchmarking pipelines is now an intrinsic feature of working in this domain.

D-Wave's annealer performed reliably within its embedding limits, showcasing low-latency performance and high solution accuracy. Fixstars Amplify, a quantum-inspired classical solver, consistently delivered the fastest results across problem types and sizes. Gurobi remained a strong general-purpose baseline, particularly for small-to-medium instances. Yet despite our preparation, efforts to run QAOA on IBM Quantum failed due to systemic shifts in the software ecosystem: deprecated primitives, incomplete V2 support, and mismatched transpilation workflows rendered the platform unusable for our purposes. These failures were not due to algorithmic flaws or user error but stemmed from structural volatility in the tooling stack.

The implications of these results extend beyond raw performance. Benchmarking in quantum computing is no longer only a matter of evaluating speed or scalability — it now requires explicit awareness of platform evolution, software versioning, and service deprecation. As Hashim et al. note [8], reproducibility in quantum computing must grapple with a moving target: API instability, hardware calibrations, and cloud service transitions all confound longitudinal analysis.

We emphasize that benchmarking in quantum computing is no longer a purely algorithmic task. It is an exercise in infrastructure navigation, reproducibility engineering, and version-aware experimentation. Platform instability—whether through API changes, job failures, or embedding collapse—must be recognized as a core challenge, not a peripheral concern. To move the field forward, we recommend:

- Treating solver configurations and backend versions as first-class experimental parameters.
- Recognizing and reporting platform-level failure modes as benchmark outcomes.

As quantum computing transitions from theoretical promise to practical evaluation, methodological resilience will be as critical as raw performance. Our findings are not a verdict on solver supremacy, but a call for mature, reproducible, and infrastructure-aware benchmarking standards in the next phase of quantum algorithm research.

Acknowledgments. The first three authors contributed equally to this work.

References

1. Codognet, P., Diaz, D., Abreu, S.: Quantum and digital annealing for the quadratic assignment problem. In: 2022 IEEE International Conference on Quantum Software (QSW). pp. 1–8 (2022). https://doi.org/10.1109/QSW55613.2022.00016

2. D-Wave: Programming the d-wave qpu: Parameters for beginners (2020), https://www.dwavequantum.com/media/qvbjrzgg/guide-2.pdf, accessed: 2025-04-17
3. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm (2014), https://arxiv.org/abs/1411.4028
4. Fixstars: Fixstars amplify annealing engine (2024), https://amplify.fixstars.com/en/docs/amplify/v1/clients/fixstars.html, accessed: 2025-04-17
5. Fixstars: Gurobi optimizer - fixstars amplify sdk documentation (2024), https://amplify.fixstars.com/en/docs/amplify/v1/clients/gurobi.html, accessed: 2025-01-05
6. Glover, F., Kochenberger, G., Du, Y.: A tutorial on formulating and using qubo models (2019)
7. Harrigan, M.P., Sung, K.J., Neeley, M., Satzinger, K.J., Arute, F., Arya, K., Atalaya, J., Bardin, J.C., Barends, R., Boixo, S., Broughton, M., Buckley, B.B., Buell, D.A., Burkett, B., Bushnell, N., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Demura, S., Dunsworth, A., Eppens, D., Fowler, A., Foxen, B., Gidney, C., Giustina, M., Graff, R., Habegger, S., Ho, A., Hong, S., Huang, T., Ioffe, L.B., Isakov, S.V., Jeffrey, E., Jiang, Z., Jones, C., Kafri, D., Kechedzhi, K., Kelly, J., Kim, S., Klimov, P.V., Korotkov, A.N., Kostritsa, F., Landhuis, D., Laptev, P., Lindmark, M., Leib, M., Martin, O., Martinis, J.M., McClean, J.R., McEwen, M., Megrant, A., Mi, X., Mohseni, M., Mruczkiewicz, W., Mutus, J., Naaman, O., Neill, C., Neukart, F., Niu, M.Y., O'Brien, T.E., O'Gorman, B., Ostby, E., Petukhov, A., Putterman, H., Quintana, C., Roushan, P., Rubin, N.C., Sank, D., Skolik, A., Smelyanskiy, V., Strain, D., Streif, M., Szalay, M., Vainsencher, A., White, T., Yao, Z.J., Yeh, P., Zalcman, A., Zhou, L., Neven, H., Bacon, D., Lucero, E., Farhi, E., Babbush, R.: Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. Nature Physics **17**(3), 332–336 (Feb 2021). https://doi.org/10.1038/s41567-020-01105-y, http://dx.doi.org/10.1038/s41567-020-01105-y
8. Hashim, A., Nguyen, L.B., Goss, N., Marinelli, B., Naik, R.K., Chistolini, T., Hines, J., Marceaux, J.P., Kim, Y., Gokhale, P., Tomesh, T., Chen, S., Jiang, L., Ferracin, S., Rudinger, K., Proctor, T., Young, K.C., Blume-Kohout, R., Siddiqi, I.: A practical introduction to benchmarking and characterization of quantum computers (2024), https://arxiv.org/abs/2408.12064
9. Jain, S.: Solving the traveling salesman problem on the d-wave quantum computer. Frontiers in Physics **9** (2021). https://doi.org/10.3389/fphy.2021.760783, https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2021.760783
10. Kittichaikoonkij, N., Pongtanyavichai, N., Suwananek, P., Chongstitvatana, P., Suksen, K.: Benchmarking quantum computing for combinatorial optimization. In: Int. Conf. on Electrical Engineering/Electronics, Computer, Telecommunication, and Information Technology. Bangkok, Thailand (May 2025), available at https://www.cp.eng.chula.ac.th/ prabhas/paper/2025/m12118-kittichaikoonkij%20final.pdf
11. Mandal, A., Roy, A., Upadhyay, S., Ushijima-Mwesigwa, H.: Compressed quadratization of higher order binary optimization problems (2020), https://arxiv.org/abs/2001.00658
12. Mauerer, W., Scherzinger, S.: 1-2-3 reproducibility for quantum software experiments (2022), https://arxiv.org/abs/2201.12031

13. Rajak, A., Suzuki, S., Dutta, A., Chakrabarti, B.K.: Quantum annealing: an overview. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **381**(2241) (dec 2022). https://doi.org/10.1098/rsta.2021.0417, http://dx.doi.org/10.1098/rsta.2021.0417
14. Silevitch, D., Rosenbaum, T., Aeppli, G.: Quantum tunneling vs. thermal effects in experiments on adiabatic quantum computing. The European Physical Journal Special Topics **224**, 25–34 (2015). https://doi.org/10.1140/epjst/e2015-02340-6, https://doi.org/10.1140/epjst/e2015-02340-6
15. Villar-Rodriguez, E., Osaba, E., Oregi, I.: Analyzing the behaviour of d'wave quantum annealer: fine-tuning parameterization and tests with restrictive hamiltonian formulations (2022), https://arxiv.org/abs/2207.00253

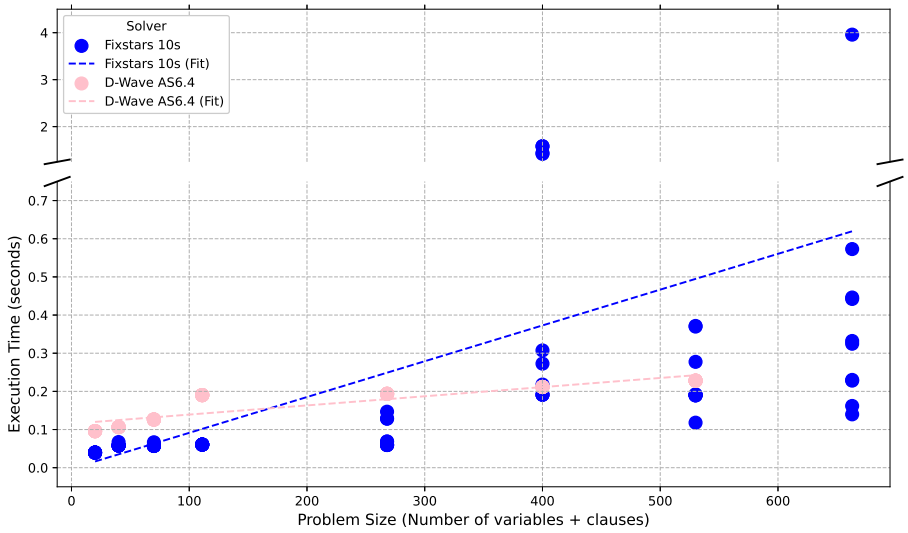

Fig. 1. 3SAT: Execution Time for Fixstars and D-Wave with Linear Regression Lines. Adapted from [10].

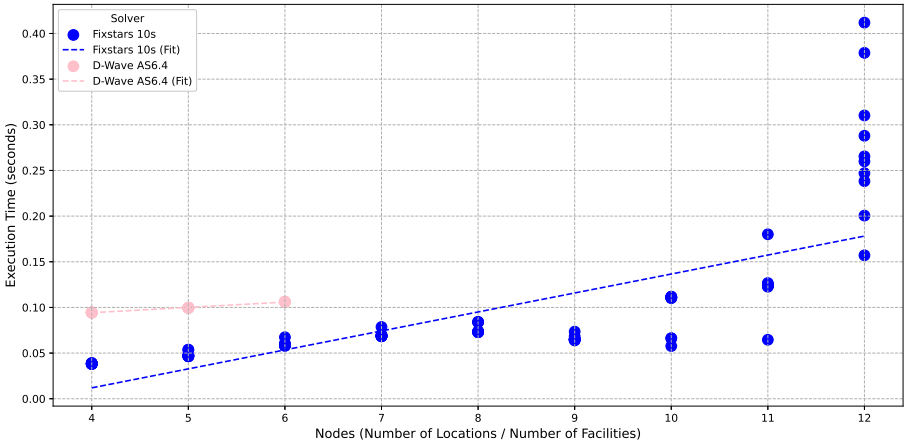

Fig. 2. QAP: Execution Time for Fixstars and D-Wave with Linear Regression Lines. Adapted from [10].

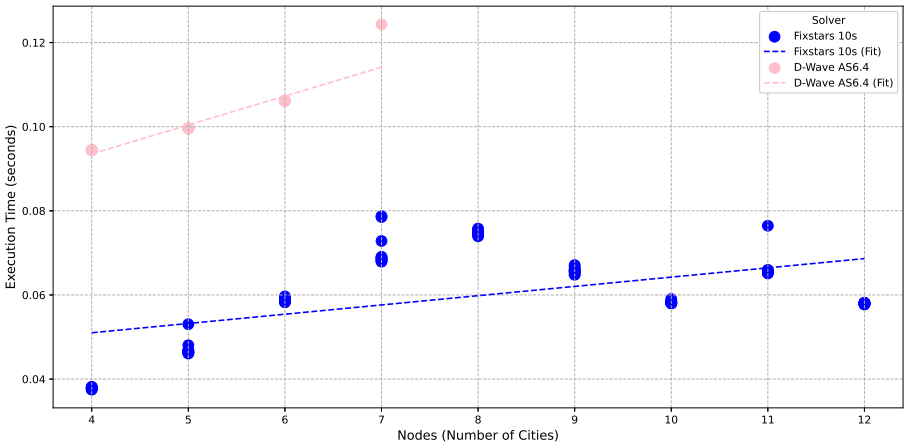

Fig. 3. TSP: Execution Time for Fixstars and D-Wave with Linear Regression Lines. Adapted from [10].