NovaQ: Improving Quantum Program Testing through Diversity-Guided Test Case Generation

Tiancheng Jin
Kyushu University, Japan
jintc1@f.ait.kyushu-u.ac.jp

Shangzhou Xia
Kyushu University, Japan
xia.shangzhou.218@s.kyushu-u.ac.jp

Jianjun Zhao*
Kyushu University, Japan
zhao@ait.kyushu-u.ac.jp

Abstract—Quantum programs are designed to run on quantum computers, leveraging quantum circuits to solve problems that are intractable for classical machines. As quantum computing advances, ensuring the reliability of quantum programs has become increasingly important. This paper introduces *NovaQ*, a diversity-guided testing framework for quantum programs. *NovaQ* combines a distribution-based test case generator with a novelty-driven evaluation module. The generator produces diverse quantum state inputs by mutating circuit parameters, while the evaluator quantifies behavioral novelty based on internal circuit state metrics, including magnitude, phase, and entanglement. By selecting inputs that map to infrequently covered regions in the metric space, *NovaQ* effectively explores under-tested program behaviors. We evaluate *NovaQ* on quantum programs of varying sizes and complexities. Experimental results show that *NovaQ* consistently achieves higher test input diversity and detects more bugs than existing baseline approaches.

Index Terms—quantum programs, test case, diversity, magnitude, phase, entanglement

I. INTRODUCTION

Quantum programs are designed to run on quantum computers, leveraging quantum circuits to solve problems that are intractable for classical machines [1]. They are widely used in domains such as cryptography [2], optimization [3], and quantum simulation [4]. As quantum hardware and software evolve, the increasing complexity of quantum circuits makes ensuring the correctness and robustness of quantum programs critical [5]–[7], particularly given the high cost of quantum computation [8] and the susceptibility to subtle errors [9].

Despite growing interest in quantum computing, systematic testing of quantum programs remains challenging and underdeveloped [10], [11]. Traditional testing techniques are often ineffective for quantum programs due to the probabilistic and non-deterministic nature of quantum behavior [12]. Existing approaches usually rely on randomly generated inputs [13], which lack diversity and make bug detection difficult. These challenges require new testing methodologies that can efficiently explore the quantum input space and identify faults.

This paper presents *NovaQ*, a testing framework designed specifically for quantum programs. NovaQ consists of two core components: a distribution-based test case generator and a novelty-driven test case evaluator. The generator creates test inputs by sampling gate parameters from Gaussian distribu-

tions [14], applying them to a parameterized initial quantum circuit, and executing the circuit in the all-zero quantum state to produce quantum state vectors. To maintain diversity, NovaQ employs a mutation-based strategy that perturbs the mean and variance of the parameter distributions throughout iterations.

To evaluate test cases, NovaQ analyzes internal circuit state metrics—such as magnitude, phase, and entanglement—to compute a novelty score that quantifies behavioral diversity. Test inputs that explore novel areas of the state space are prioritized in subsequent iterations. This feedback loop enables NovaQ to efficiently explore diverse circuit behaviors and uncover bugs in quantum programs.

We evaluate *NovaQ* on a set of quantum programs with varying circuit sizes and complexities. Specifically, we apply NovaQ to extend the test case generator proposed by Ye *et al.* in QuraTest [13], adjusting the three parameters of the U gate to generate diverse test cases. The number of qubits in the generated test cases ranges from 3 to 12. These test cases are used as inputs to benchmark programs to assess their bug-detection capability. Experimental results show that NovaQ consistently outperforms baseline testing approaches in terms of both test case diversity and fault detection. By combining guided test case generation with diversity metrics tailored to quantum behavior, NovaQ provides a promising approach for enhancing the reliability of quantum software systems.

II. BACKGROUND

This section introduces essential background concepts related to quantum programs and quantum circuits.

A. Quantum Bits

The fundamental elements in quantum programs are quantum bits, commonly referred to as *qubits*. A classical bit can take the value 0 or 1, while a qubit can exist in a linear superposition of these two basis states. The basis states $|0\rangle$ and $|1\rangle$ are represented as $|0\rangle = [1, 0]^\top$ and $|1\rangle = [0, 1]^\top$, and are called computational basis states. A general qubit state is written as $|q\rangle = \alpha |0\rangle + \beta |1\rangle$, where α and β are complex numbers satisfying the *normalization condition* $|\alpha|^2 + |\beta|^2 = 1$.

B. Quantum Gates and Circuits

Quantum gates manipulate the states of qubits and form the building blocks of quantum circuits. Similar to gates

in classical digital circuits, quantum gates are linear and reversible, operating on a fixed number of qubits with equal numbers of input and output lines.

This paper uses the U gate, a commonly used single-qubit gate, defined as:

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

Quantum circuits represent executable quantum programs. They consist of an ordered sequence of quantum gates applied to qubits, followed by measurement operations. The output of a quantum circuit is obtained by measuring the final state of the qubits.

C. Quantum Program Module

Similar to classical modules that encapsulate code for specific functionality, quantum program modules encapsulate reusable units composed of quantum gate sequences, subcircuits, measurement operations, or subalgorithm implementations. For instance, the Inverse Quantum Fourier Transform (IQFT) module [15] is a widely used module for converting phase information into magnitude information.

III. METHODOLOGY

A. Overview

NovaQ consists of two main components: a test case generator that produces qubit states based on quantum circuits equipped with parameters sampled from Gaussian distributions and a test case evaluator that quantifies diversity using internal quantum state metrics. The testing process iteratively mutates parameter distributions, generates test circuits, evaluates their behavioral novelty, and retains the most promising distributions for further mutation. The overall workflow is shown in Figure 1.
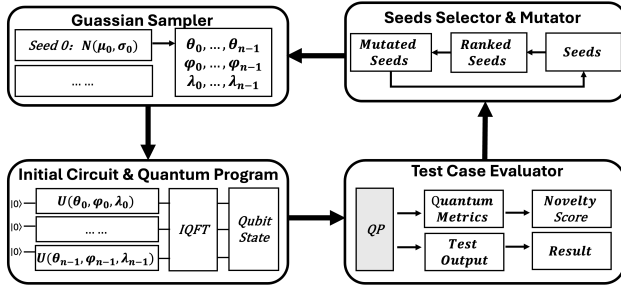

Fig. 1: Workflow of *NovaQ*. The module labeled with a shadow (i.e., QP) represents the quantum programs under test.

B. Distribution-Based Test Case Generation

The core of *NovaQ*'s test case generation lies in constructing parameterized initial quantum circuits. Each seed is defined as a pair of real numbers that represent the mean and variance of a normal distribution $N(\mu, \sigma)$. Samples drawn from these distributions are used to parameterize the U gates in the initial circuit. Each circuit consists of these parameterized

gates followed by a fixed Inverse Quantum Fourier Transform (IQFT) layer to introduce complex interference patterns.

By setting different values for the U gate parameters θ, ϕ, and λ, applying these gates to each qubit initialized in the $|0\rangle$ state results in qubit states with varied characteristics. The initial circuit takes the all-zero quantum state $|0\rangle^{\otimes n}$ as input and outputs a qubit state vector. This vector is then used as the input to the quantum program under test. Without the use of diverse initial circuits, the generated test inputs would lack variation, significantly reducing the likelihood of exposing bugs. We formally define a test case as follows:

Definition 1. *A test case refers to the output qubit state vector of the initial quantum circuit—comprising parameterized U gates and an IQFT layer—when executed on the all-zero input state* $|0\rangle^{\otimes n}$.

To ensure diversity in test inputs, *NovaQ* initializes multiple seed distributions and applies small mutations to them in each iteration. For each seed, a fixed number of test circuits is generated and evaluated for behavioral diversity and fault-detection potential.

Definition 2. *A bug in a quantum program refers to a violation of the expected input-output correspondence. In a correct quantum program, the output probability distribution should remain consistent for a given input; any deviation from this expected behavior may indicate the presence of a bug.*

C. Diversity-Based Evaluation and Bug Detection

To effectively identify faults in quantum programs, it is crucial to explore the qubit state input space thoroughly. For each test case, the execution result on the target quantum program is analyzed to extract internal quantum state features—specifically, phase, magnitude, and entanglement—using vector simulations. To prevent generated test cases from trapping in specific regions of the input space, we design the *novelty score*, which guides the exploration toward underrepresented behaviors in the input space, thereby enhancing test case diversity.

To evaluate novelty, the continuous space defined by the three metrics is discretized into a finite set of grid cells. Each metric is first normalized to the range $[l_d, u_d]$, and then divided into N equal intervals. Thus, each qubit state is mapped to a unique grid cell defined by its discretized magnitude, phase, and entanglement values. The novelty score of a state is inversely related to the frequency with which its corresponding grid cell has been visited: the less frequently visited, the more novel. This computation is formalized as follows:

$$\eta_d = \left\lfloor \frac{u_d - l_d}{N} \cdot (\phi_d - l_d) \right\rfloor \tag{1}$$

Here, η_d is the index of the interval for metric d, and ϕ_d is the metric value (e.g., magnitude, phase, or entanglement). For each qubit state, we obtain the triplet (ϕ_m, ϕ_p, ϕ_e) representing its metrics, which is then mapped to the discrete grid cell (η_m, η_p, η_e).

Definition 3. *The novelty score of a qubit state is defined as the relative visitation frequency $\frac{\rho}{N^3}$ of its corresponding grid cell, where ρ is the number of previously recorded states mapped to that cell. A lower novelty score indicates that the state lies in a region of the metric space that is rarely explored.*

D. Iterative Selection and Optimization

The test generation process follows an iterative mutation-selection loop. After generating and evaluating the test circuits from the current seed set, each seed is scored based on the average novelty of its corresponding test cases. The top-performing seeds are selected to form the next generation, guiding the search toward distributions that yield diverse and potentially fault-revealing inputs. Specifically, the mean and variance are sampled within the ranges $[-15, 15]$ and $[0.1, 30]$, respectively. Each mutation perturbs the values of mean and variance as follows:

$$\mu' = \mu + \Delta\mu, \quad \Delta\mu \in [-0.5, 0.5]$$
$$\sigma'^2 = \sigma^2 + \Delta\sigma^2, \quad \Delta\sigma^2 \in [-0.5, 0.5]$$

This process continues until a user-defined budget (e.g., total number of test cases) is reached.

IV. EVALUATION

We evaluated the effectiveness of *NovaQ* by comparing it with a baseline method in terms of test case diversity and bug detection capability. The evaluation follows the testing methodology described in [13], and measures the fault-detection performance of test cases generated by both the baseline and *NovaQ*.

RQ1: Does *NovaQ* generate more diverse test cases compared to the baseline?

To answer this question, we compare the diversity of test cases generated by *NovaQ* with that of the baseline. Diversity is evaluated using a discretized $10 \times 10 \times 10$ grid on three quantum-specific metrics: magnitude, phase, and entanglement, as proposed in [13]. A higher number of occupied grid cells indicates greater diversity.

We apply both methods to the IQFT generator. While the baseline study in [13] reports results only for 5-qubit circuits, our evaluation includes circuits with 3, 5, 7, 10, and 12 qubits, for a broader comparison between different circuit sizes.

In *NovaQ*, the initial seed pool contains $n = 10$ randomly initialized seeds. For each iteration, we apply controlled mutations to the three parameters (θ, ϕ, λ) of all U gates in the selected seeds. The mutation range for both mean and variance of the parameter distributions is ± 0.5. To avoid premature convergence to local optima, each seed selected for the next generation has a 10% probability of undergoing random mutation instead of guided mutation.

After generating all $N = 1500$ test cases, we compute the diversity by counting the number of occupied grid cells in the three-dimensional space defined by the three metrics. The results are summarized in Table I. Across all qubit sizes, *NovaQ* consistently generates more diverse test cases than the
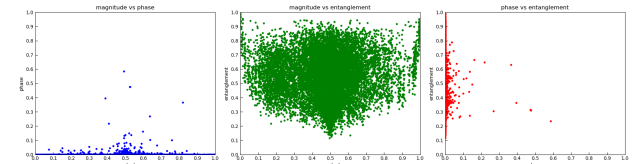
baseline. For qubit numbers of 3, 5, 7, 10, and 12, *NovaQ* achieves improvements of 10.57%, 13.07%, 39.40%, 55.73%, and 107.37%, respectively, in test case diversity compared to the baseline. Because the input space expands exponentially with the number of qubits, leading to an increased proportion of certain grid regions within the space. The baseline's random parameter generation often becomes trapped in high-proportion grids, *NovaQ* leverages a novelty-driven mechanism to explore uncovered regions, thereby attaining substantially higher coverage, especially for larger qubit systems.

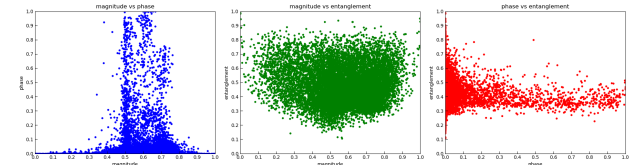TABLE I: Grid Coverage in 1,500 tests of Baseline vs. *NovaQ*

Qubit Number	Baseline		NovaQ	
	Grids	Coverage Rate	Grids	Coverage Rate
3	634	63.4%	701	70.1%
5	574	57.4%	649	64.9%
7	434	43.4%	605	60.5%
10	192	19.2%	299	29.9%
12	95	9.5%	197	19.7%

RQ2: In what way does *NovaQ* outperform the baseline?

To further analyze the results, we use the 12-qubit case from RQ1 as a representative example. In Figure 2, we use three three-view diagrams to show the experimental results, which were originally three-dimensional diagrams. Figures 2a and 2b show the distribution of test cases across the three diversity metrics. For magnitude and entanglement scores, both methods yield similar results. However, in the phase dimension, *NovaQ* generates more diverse test cases, leading to higher overall diversity. This demonstrates that *NovaQ*'s novelty-driven mechanism is effective in guiding test generation toward under-explored areas of the state space. Furthermore, the results indicate that *NovaQ*'s effectiveness in **RQ1** mainly comes from enhanced grid coverage of phase scores.

(a) Scores of Baseline

(b) Scores of *NovaQ*

Fig. 2: Diversity difference between Baseline and *NovaQ* of 12-qubit. The blue, green, and red parts represent the results based on the dimensions of (magnitude, phase), (magnitude, entanglement), and (phase, entanglement), respectively.

RQ3: Are the test cases generated by *NovaQ* more effective in detecting bugs in quantum programs compared to those generated by the baseline?

To answer this question, we evaluate the fault-detection capability of test cases produced by both *NovaQ* and the baseline. Following the methodology in [13], we randomly replace certain quantum gates in the common quantum algorithms with arbitrary quantum gates and filter out buggy programs that affect the original functionality according to *Definition 2* as the benchmark. We then measure the detection rate for each test suite generated by *NovaQ* and the baseline. A higher detection rate indicates stronger fault sensitivity and better testing effectiveness.

TABLE II: Number of bugs found in 1,500 tests of Baseline vs. *NovaQ*

Program	Baseline		NovaQ	
	Bugs Found	Accuracy	Bugs Found	Accuracy
Grover-03	1270	84.7%	1367	91.1%
Grover-05	1258	83.9%	1405	93.7%
Grover-07	1249	83.3%	1391	92.7%
Grover-10	1280	85.3%	1426	95.1%
Grover-12	1280	85.5%	1412	92.1%
PE-05	1251	83.4%	1387	92.5%
QFT-05	1277	85.1%	1341	89.4%

We conducted experiments using a faulty implementation of Grover's algorithm as a benchmark. In each run, we generate 1,500 test cases and count the number of faulty programs correctly identified. The results are summarized in Table II. Across all tested circuit sizes, *NovaQ* detects significantly more bugs than the baseline. For example, in the 12-qubit setting, *NovaQ* detects 1412 faults (92.1% accuracy), compared to 1280 faults (85.5% accuracy) by the baseline. This is because a quantum program may involve many possible execution branches, and faults can occur in any of the branches. By increasing the diversity of test cases, a wider range of branch combinations can be exercised, thereby improving the chances of exposing hidden bugs.

RQ4: How do the growth rates of Grid compare under the two methods?

To answer this question, we select the results of 15,000 test cases generated when the number of qubits is 3 for baseline and *NovaQ*, and plot a graph showing the relationship between the number of grids and the number of test cases. As Figure 3 shows, *NovaQ* has a higher growth rate than the baseline in generating more types of test cases. Furthermore, when generating the same number of test cases, *NovaQ* consistently generates more diverse test cases than the baseline.

Figure 3 also shows that when generating a particularly large number of test cases, although it is difficult for the new test cases generated by baseline and *NovaQ* to become more diverse, the upper limit of diversity of the test cases generated by *NovaQ* is higher than that of the baseline. This indicates that the mutation parameter method in *NovaQ* can generate types of test cases that cannot be generated by the pure random method in the baseline.

V. Related Work

Quantum software testing faces fundamental challenges due to the non-deterministic nature of quantum programs. Several studies [10], [12], [16] have surveyed the current landscape
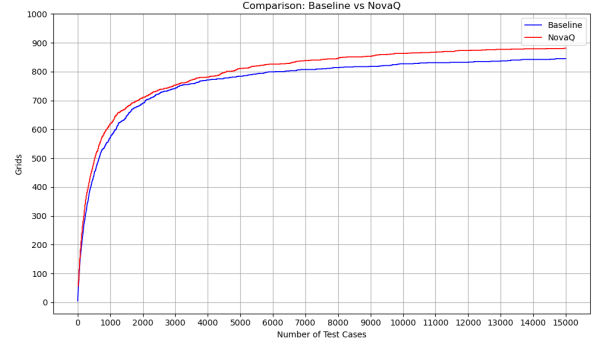

Fig. 3: 15,000 Test Case Results of 3-qubit: Baseline vs *NovaQ*

of testing approaches and challenges. To support systematic investigation, benchmark datasets have been developed. Bugs4Q [17], [18] collects 36 validated bugs from Qiskit and extends to 42 from GitHub and community platforms, while QBugs [19] provides a framework for organizing and reproducing quantum software bugs in a controlled setting. Building on these resources, various testing techniques have been explored, including search-based testing [20], combinatorial testing [21], metamorphic testing [22], concolic testing [23], mutation testing [24], and black-box testing [25], [26].

Beyond these methods, test case generation, a crucial step in classical software testing, has also been adapted for quantum programs. Coverage-based approaches have been proposed to generate effective test cases [20], [21], [27], and QuraTest [13] leverages quantum properties to guide generation, though existing methods have not fully exploited such properties to enhance diversity. NovaQ addresses this limitation by adopting a diversity-guided strategy to select more informative test cases. Complementary to test generation, several quantum-specific coverage criteria [28]–[31] have been introduced to support systematic evaluation, and tools and frameworks [30], [32]–[34] have been developed to facilitate test execution and assertion checking, together forming a growing ecosystem of quantum testing support.

VI. Conclusion

This paper presents *NovaQ*, a testing framework tailored for quantum programs. Unlike existing approaches that rely on random parameter generation, *NovaQ* employs a mutation-based strategy that perturbs the mean and variance of parameter distributions during test case generation to improve diversity. Experimental results show that across various qubit numbers, *NovaQ* outperforms the baseline method in terms of test case diversity and fault detection. Moreover, *NovaQ* achieves more effective exploration in the phase property.

Future work will explore applying *NovaQ* to additional test case generators beyond the UCNOT and IQFT methods, and evaluate its effectiveness across a broader range of quantum programs to further validate the impact of test case diversity on fault detection.

REFERENCES

[1] A. Y. Kitaev, A. Shen, M. N. Vyalyi, and M. N. Vyalyi, *Classical and quantum computation*. American Mathematical Soc., 2002, no. 47.

[2] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Reviews of modern physics*, vol. 74, no. 1, p. 145, 2002.

[3] A. Abbas, A. Ambainis, B. Augustino, A. Bärtschi, H. Buhrman, C. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen *et al.*, "Challenges and opportunities in quantum optimization," *Nature Reviews Physics*, pp. 1–18, 2024.

[4] I. M. Georgescu, S. Ashhab, and F. Nori, "Quantum simulation," *Reviews of Modern Physics*, vol. 86, no. 1, pp. 153–185, 2014.

[5] M. Piani, M. Cianciaruso, T. R. Bromley, C. Napoli, N. Johnston, and G. Adesso, "Robustness of asymmetry and coherence of quantum states," *Physical Review A*, vol. 93, no. 4, p. 042107, 2016.

[6] A. W. Harrow and M. A. Nielsen, "Robustness of quantum gates in the presence of noise," *Physical Review A*, vol. 68, no. 1, p. 012308, 2003.

[7] T. Jin and J. Zhao, "Scaffml: A quantum behavioral interface specification language for scaffold," in *2023 IEEE International Conference on Quantum Software (QSW)*. IEEE, 2023, pp. 128–137.

[8] A. Streltsov, H. Kampermann, and D. Bruß, "Quantum cost for sending entanglement," *Physical review letters*, vol. 108, no. 25, p. 250501, 2012.

[9] D. Gottesman, "An introduction to quantum error correction," in *Proceedings of Symposia in Applied Mathematics*, vol. 58, 2002, pp. 221–236.

[10] N. C. Leite Ramalho, H. Amario de Souza, and M. Lordello Chaim, "Testing and debugging quantum programs: The road to 2030," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 5, pp. 1–46, 2025.

[11] A. Miranskyy and L. Zhang, "On testing quantum programs," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2019, pp. 57–60.

[12] M. Paltenghi and M. Pradel, "A survey on testing and analysis of quantum software," *arXiv preprint arXiv:2410.00650*, 2024.

[13] J. Ye, S. Xia, F. Zhang, P. Arcaini, L. Ma, J. Zhao, and F. Ishikawa, "Quratest: Integrating quantum specific features in quantum program testing," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 1149–1161.

[14] X. Zhang, "Gaussian distribution," in *Encyclopedia of machine learning and data mining*. Springer, 2016, pp. 1–5.

[15] Y. S. Weinstein, M. Pravia, E. Fortunato, S. Lloyd, and D. G. Cory, "Implementation of the quantum fourier transform," *Physical review letters*, vol. 86, no. 9, p. 1889, 2001.

[16] A. García de la Barrera, I. García-Rodríguez de Guzmán, M. Polo, and M. Piattini, "Quantum software testing: State of the art," *Journal of Software: Evolution and Process*, vol. 35, no. 4, p. e2419, 2023.

[17] P. Zhao, J. Zhao, Z. Miao, and S. Lan, "Bugs4Q: A benchmark of real bugs for quantum programs," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 1373–1376.

[18] P. Zhao, Z. Miao, S. Lan, and J. Zhao, "Bugs4Q: A benchmark of existing bugs to enable controlled testing and debugging studies for quantum programs," *Journal of Systems and Software*, vol. 205, p. 111805, 2023.

[19] J. Campos and A. Souto, "Qbugs: A collection of reproducible bugs in quantum algorithms and a supporting infrastructure to enable controlled quantum software testing and debugging experiments," in *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*. IEEE, 2021, pp. 28–32.

[20] X. Wang, P. Arcaini, T. Yue, and S. Ali, "Qusbt: search-based testing of quantum programs," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 173–177. [Online]. Available: https://doi.org/10.1145/3510454.3516839

[21] ——, "Application of combinatorial testing to quantum programs," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, 2021, pp. 179–188.

[22] R. Abreu, J. P. Fernandes, L. Llana, and G. Tavares, "Metamorphic testing of oracle quantum programs," in *Proceedings of the 3rd International Workshop on Quantum Software Engineering*, 2022, pp. 16–23.

[23] S. Xia, J. Zhao, F. Zhang, and X. Guo, "Quantum concolic testing," *Proceedings of the ACM on Software Engineering*, vol. 2, no. ISSTA, pp. 1146–1166, 2025.

[24] D. Fortunato, J. Campos, and R. Abreu, "Qmutpy: A mutation testing tool for quantum algorithms and applications in qiskit," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2022, pp. 797–800.

[25] P. Long and J. Zhao, "A black-box testing framework for oracle quantum programs," *arXiv preprint arXiv:2505.07243*, 2025.

[26] ——, "Equivalence, identity, and unitarity checking in black-box testing of quantum programs," *Journal of Systems and Software*, vol. 211, p. 112000, 2024.

[27] X. Wang, P. Arcaini, T. Yue, and S. Ali, "Quito: a coverage-guided test generator for quantum programs," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 1237–1241.

[28] S. Ali, P. Arcaini, X. Wang, and T. Yue, "Assessing the effectiveness of input and output coverage criteria for testing quantum programs," in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2021, pp. 13–23.

[29] D. Fortunato, J. Campos, and R. Abreu, "Gate branch coverage: A metric for quantum software testing," in *Proceedings of the 1st ACM International Workshop on Quantum Software Engineering: The Next Evolution*, 2024, pp. 15–18.

[30] P. Long and J. Zhao, "Testing multi-subroutine quantum programs: From unit testing to integration testing," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 6, pp. 1–61, 2024.

[31] M. Shao and J. Zhao, "A coverage-guided testing framework for quantum neural networks," *arXiv preprint arXiv:2411.02450*, 2024.

[32] D. Fortunato, J. Campos, and R. Abreu, "Mutation testing of quantum programs: A case study with qiskit," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–17, 2022.

[33] G. J. Pontolillo and M. R. Mousavi, "Delta debugging for property-based regression testing of quantum programs," in *Proceedings of the 5th ACM/IEEE International Workshop on Quantum Software Engineering*, 2024, pp. 1–8.

[34] G. Li, L. Zhou, N. Yu, Y. Ding, M. Ying, and Y. Xie, "Projection-based runtime assertions for testing and debugging quantum programs," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, pp. 1–29, 2020.