

Evaluating Mutation Techniques in Genetic Algorithm-Based Quantum Circuit Synthesis

Michael Kölle

LMU Munich

Munich, Germany

michael.koelle@ifi.lmu.de

Tom Bintener

LMU Munich

Munich, Germany

Maximilian Zorn

LMU Munich

Munich, Germany

Gerhard Stenzel

LMU Munich

Munich, Germany

Leo Sükel

LMU Munich

Munich, Germany

Thomas Gabor

LMU Munich

Munich, Germany

Claudia Linnhoff-Popien

LMU Munich

Munich, Germany

ABSTRACT

Quantum computing leverages the unique properties of qubits and quantum parallelism to solve problems intractable for classical systems, offering unparalleled computational potential. However, the optimization of quantum circuits remains critical, especially for noisy intermediate-scale quantum (NISQ) devices with limited qubits and high error rates. Genetic algorithms (GAs) provide a promising approach for efficient quantum circuit synthesis by automating optimization tasks. This work examines the impact of various mutation strategies within a GA framework for quantum circuit synthesis. By analyzing how different mutations transform circuits, it identifies strategies that enhance efficiency and performance. Experiments utilized a fitness function emphasizing fidelity, while accounting for circuit depth and T operations, to optimize circuits with four to six qubits. Comprehensive hyperparameter testing revealed that combining delete and swap strategies outperformed other approaches, demonstrating their effectiveness in developing robust GA-based quantum circuit optimizers.

CCS CONCEPTS

- Hardware → Quantum computation; • Computing methodologies → Genetic algorithms; • Theory of computation → Evolutionary algorithms.

KEYWORDS

Variational Quantum Circuits, Automated Circuit Design, Mutation

ACM Reference Format:

Michael Kölle, Tom Bintener, Maximilian Zorn, Gerhard Stenzel, Leo Sükel, Thomas Gabor, and Claudia Linnhoff-Popien. 2025. Evaluating Mutation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GECCO '25, July 14–18, 2025, Malaga, Spain

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1465-8/2025/07...\$15.00

<https://doi.org/10.1145/3712256.3726402>

Techniques in Genetic Algorithm-Based Quantum Circuit Synthesis. In *Genetic and Evolutionary Computation Conference (GECCO '25)*, July 14–18, 2025, Malaga, Spain. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3712256.3726402>

1 INTRODUCTION

Quantum computing offers a promising way to tackle complex problems that classical computers cannot solve [8, 9, 25]. Quantum circuits lie at the core of quantum computing and are essential for implementing quantum algorithms [3]. The efficient synthesis and optimization of these circuits is critical yet challenging, especially on NISQ devices, which have limited qubits and high error rates [17, 19]. As more performant quantum hardware continues to emerge, efficient and automated quantum circuit synthesis grows increasingly important, because manual circuit creation is not sustainable [1, 2, 30].

The synthesis of quantum circuits is difficult because of the complexity of quantum operations and the limits of current hardware. GAs offer a possible solution by using evolutionary strategies to improve quantum circuit designs [24]. GAs are especially effective at exploring large, complex solution spaces [4, 13], but their performance in quantum circuit synthesis and optimization remains an open area of research [15, 21]. In particular, examining how different mutation strategies interact with quantum circuits may provide new ways to enhance optimization efficiency.

To investigate these mutation strategies, we conducted a series of experiments. We developed a multifunctional quantum environment that can create, manipulate, and evaluate quantum circuits efficiently. It includes a circuit optimizer that works within the Clifford + T set, following standard protocols [23], and an unbiased circuit generator that produces diverse candidate circuits for testing. An autonomous system switches between serial and parallel processing modes based on available hardware and data, ensuring optimal resource use.

A GA framework was then implemented within this quantum environment. The fitness function in this framework is derived from fidelity, circuit depth, and the number of T operations [12, 26, 31]. The framework allows either a single-population or an island model, along with tournament selection for elites and offsprings,

immigrants, and single-point crossover [10, 14, 28]. The mutation strategies include change, delete, add, swap, and every possible combination of these, using either static or dynamic mutation rates. Hyperparameters are tuned autonomously after each repetition to improve performance. Six-qubit circuits form the dataset for these evaluations, and the same dataset is reused to maintain consistency in performance measurements.

Our contributions involve a comprehensive, modular quantum environment that serves as a framework for evaluating GA-based and other machine learning approaches in quantum computing. We also supply empirical data on how various mutation strategies perform in quantum synthesis and offer guidance on their efficient application.

The remainder of this work is structured as follows: In Section 2, we review related work on quantum state preparation, quantum circuit optimization, and the application of genetic algorithms in quantum computing. Section 3 introduces our quantum circuit environment. Section 4 describes the experimental setup. Section 5 presents the results, analyzing the performance of the genetic algorithm, the impact of mutation strategies, and the effects of mutation rate, population size, and adaptive mutation. Finally, Section 6 concludes the paper, summarizing key findings and discussing potential directions for future research.

2 RELATED WORK

This section provides an overview of the foundational and recent research that underpins this work. It focuses on quantum state preparation, quantum circuit optimization, quantum circuit synthesis, and the application of GAs in quantum computing.

2.1 Quantum State Preparation

Quantum hardware typically initializes each qubit in the $|0\rangle$ state by default, yet quantum algorithms often require more complex states. Preparing these states manually is time-intensive, especially for multi-qubit algorithms, and does not guarantee an optimal solution. To automate this process, researchers have proposed several methods, including state decomposition [16], black-box approaches [22], and adiabatic methods [5].

Another prominent direction uses machine learning algorithms to train parameterized circuits that generate specific target states. Recent papers demonstrate that reinforcement learning agents can efficiently assemble state preparation sequences [11, 18], reinforcing the notion that goal-oriented methods are well-suited for automating state preparation.

2.2 Quantum Circuit Optimization and Synthesis

Quantum circuit optimization and synthesis are core techniques in state preparation. Optimization aims to reduce the number of operations in existing circuits by removing redundancies, reordering gates, or minimizing T-gates [7]. This step is crucial in NISQ-era hardware, where qubit counts are low and noise is high. Reducing circuit depth and gate count mitigates noise, improving overall reliability.

Synthesis centers on decomposing target states into smaller subcircuits or template-based solutions. Efficient decomposition

and template matching help researchers and automated tools build complex circuits more readily [7]. Despite ongoing progress, the search for better optimization and synthesis strategies continues, including contributions from the present work.

2.3 Genetic Algorithms in Quantum Computing

Inspired by the successful use of reinforcement learning in state preparation, researchers have also explored GAs for optimizing quantum circuits. GAs excel at navigating large search spaces, making them promising candidates for quantum state preparation tasks [6, 20, 29]. Miranda et al. [15] notably demonstrated the effectiveness of an island-model GA, in which the population splits into partially isolated subpopulations. This division helps avoid premature convergence by maintaining diversity across subgroups. Sunkel [27] introduced a comprehensive GA framework that offers standardized interfaces and extensibility, enabling future enhancements without altering core components. Meanwhile, Ge [7] presented guidelines for crafting fitness functions and prioritizing circuit properties in the NISQ era, where limited qubit counts and elevated error rates constrain feasible circuit sizes. This work also highlighted metrics for mitigating noise and error accumulation, and provided strategies for balancing gate fidelity against circuit depth to ensure viability on near-term hardware.

3 QUANTUM CIRCUIT ENVIRONMENT

This section introduces the quantum circuit environment developed for synthesizing quantum states with a GA. It describes the GA structure and outlines how optimization proceeds for a given problem. The goal is to explain the main components and their implementation. This environment uses flexible approaches to candidate generation, mutation, and evaluation to adapt to diverse quantum state synthesis tasks. Fig. 1 presents the streamlined optimization process used in this work, highlighting the steps for refining candidate circuits toward the target quantum state.

3.1 Candidate Representation

A candidate C in a population P is a potential solution for an optimization problem O . Each C_i with $0 < i < |P|$ is stored as a one-dimensional list of quantum operations:

- **Id:** A label denoting a quantum operation (e.g., Hadamard, CNOT).
- **Wires:** A number or list specifying which qubit(s) the operation targets.

Fig. 2 shows how a list of operations translates into a quantum circuit. This list-based format simplifies iteration, manipulation, and assessment when performing genetic operations such as mutation and crossover.

3.2 Target State

The target state is the desired quantum state for a specific algorithm or experiment, and it defines the objective in quantum circuit optimization. Each candidate's effectiveness is measured by its success in producing this state. A target state can be characterized by its statevector and density matrix.

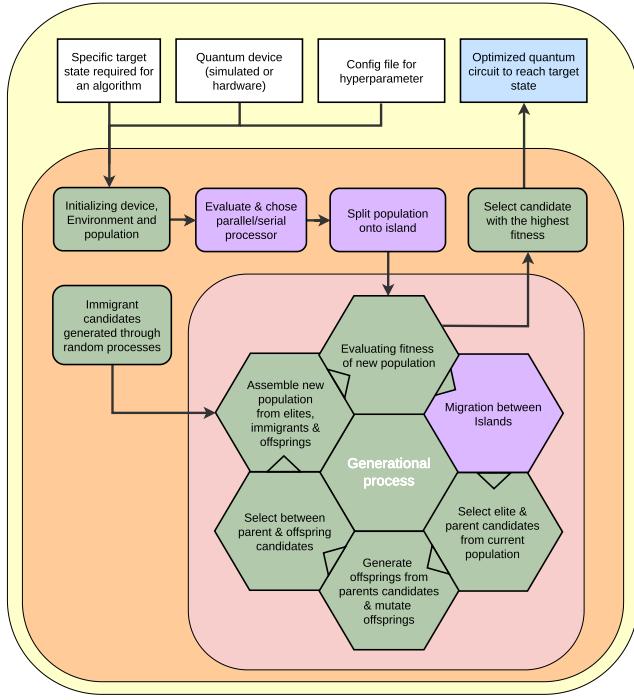


Figure 1: The GA optimization process. Yellow represents the outer layer, orange represents GA steps, and red represents the repeated optimization cycle. White boxes indicate inputs, blue boxes indicate outputs, green boxes indicate essential steps, and purple boxes indicate optional steps.

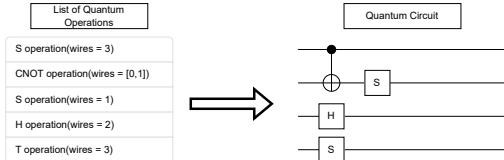


Figure 2: An example list of quantum operations and its corresponding circuit.

3.3 Environment Initialization

The genetic algorithm requires three key inputs to begin: a target density matrix, which specifies the desired quantum state and thus defines the optimization goal; a quantum device (either simulated or physical) to execute and evaluate the circuits; and a configuration file containing GA parameters such as generation count, stopping criteria, population size, and mutation rates. To simplify customization, these parameters are organized into distinct sections, including run, population, island, fitness, evolutionary, and parallel. For instance, the population section defines the number of candidates and the minimum and maximum circuit depths, while the island section governs the number of islands and the frequency of candidate migration between them. This structured approach makes it easier to adjust the GA for different experimental setups and performance objectives.

3.4 Population Initialization

Population initialization produces an initial set of candidates on which future generations build. Fig. 3 shows the process of generating random quantum circuits by choosing circuit depth, quantum operations, and qubit assignments. A short post-processing step removes trivial candidates lacking sufficient complexity. This approach provides a diverse initial population and lays the groundwork for effective optimization.

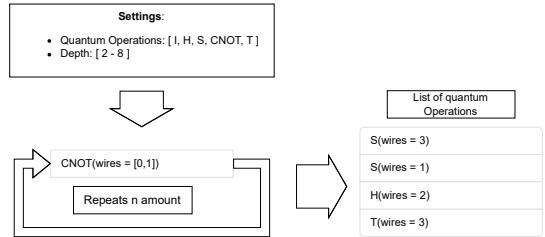


Figure 3: Using Python lists to define circuit depth and operations for each candidate during population initialization.

3.5 Evaluation Process

To evaluate candidates, the framework offers three main methods that target different circuit configurations: Parallel processing, which runs batches of candidates concurrently on multicore processors, is well-suited to circuits with four or more qubits. In contrast, serial single processing evaluates candidates individually and minimizes overhead, making it optimal for circuits with fewer than four qubits and shallow depth. For smaller circuits that have higher depth, serial batch processing assesses all candidates in a single batch, reducing redundancy. If the configuration does not specify a particular method, the framework runs a quick test on the initial population to determine which approach completes fastest and then applies that method for all remaining generations.

3.6 Fitness Evaluation

The fitness function incorporates fidelity, circuit depth, and the number of T operations to quantify how well a circuit meets hardware constraints while approximating the target state. The fidelity score, which measures how closely a candidate approximates the target state, is the primary metric for an optimal solution.

$$F(\rho, \sigma) = \left(\text{tr} \left(\sqrt{\rho \sigma \sqrt{\rho}} \right) \right)^2 \quad (1)$$

where ρ and σ are both density matrices. For practical use, a score between 0.90–0.99 is typically aimed for. Circuit depth measures sequential gate execution; shallower circuits reduce noise accumulation. The number of T gates is another key factor, as T gates are resource-intensive in the Clifford+T gate set. The overall fitness is computed as:

$$s_{\text{fitness}} = w_{\text{fidelity}} \cdot s_{\text{fidelity}} - w_d \cdot s_d - w_{\text{T-ops}} \cdot s_{\text{T-ops}}, \quad (2)$$

where w is the weight for each metric, d the circuit depth and s is the corresponding score.

3.7 Evolutionary Step

During each generation, the GA refines the population to improve solutions. Offspring form the bulk of the new population and are created through crossover, where randomly selected parents are split, recombined, and mutated. This process introduces new genetic material, promoting diversity while preserving successful traits. Meanwhile, elites—top-performing candidates—advance directly to the next generation without changes, ensuring that high-fidelity solutions remain intact. To maintain diversity and prevent early convergence, a subset of immigrants is randomly generated and included in the population, offering new genetic variations that might otherwise not emerge.

Mutations then refine candidates by applying minimal changes to their structure. In the *change* mutation, one quantum operation is substituted with another of the same length, ensuring that the circuit depth remains constant (Fig. 4). The *delete* mutation removes an operation to simplify the circuit (Fig. 5), while the *add* mutation introduces a new operation at a random position, increasing circuit complexity (Fig. 6). Finally, the *swap* mutation exchanges two operations without altering their type (Fig. 7). This combination of offspring creation, elite preservation, immigrant introduction, and incremental mutations enables the GA to explore the solution space thoroughly while retaining strong candidates for subsequent iterations.

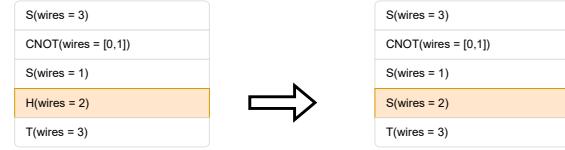


Figure 4: An example of the *change* strategy. The modified operation is highlighted in orange.

An experimental feature is adaptive mutation, which adjusts mutation parameters in response to average fitness, population diversity, and remaining generations. It encourages extensive exploration early on, then converges toward stronger solutions as the optimization nears completion.

3.8 Island Model

The island model is an optional feature that partitions the population into smaller, parallel subpopulations. Each island explores different regions of the solution space in isolation. During later generations, selective migration shares high-fidelity candidates between islands, boosting convergence toward globally optimal solutions. This parallel isolation also mitigates premature convergence by maintaining diverse subpopulations.

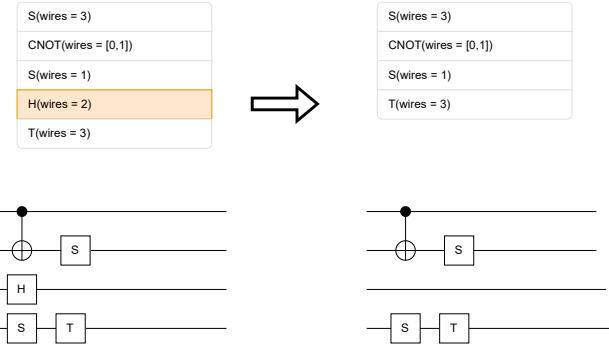


Figure 5: An example of the *delete* strategy. The removed operation is highlighted in orange.

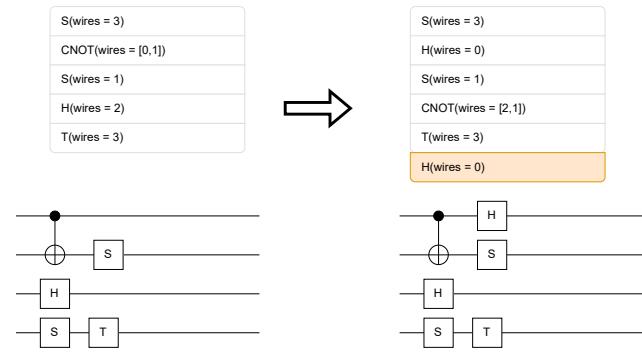


Figure 6: An example of the *add* strategy. The newly inserted operation is highlighted in orange.

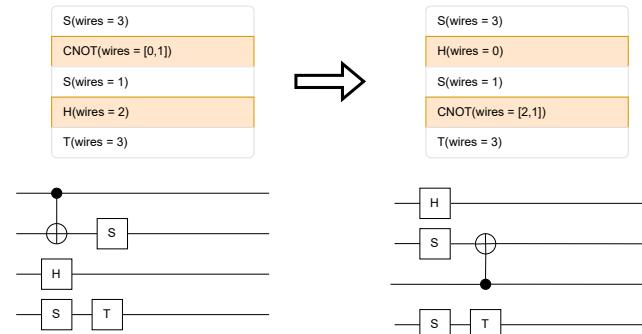


Figure 7: An example of the *swap* strategy. The exchanged operations are highlighted in orange.

3.9 Quantum Circuit Optimization

Fully random candidate creation can produce extraneous or canceling operations. To address this, the environment includes a selective optimizer. It merges or removes redundant gates in each candidate's circuit (e.g., converting two consecutive T gates to an S gate) during fitness evaluation. The best candidate also undergoes a final optimization step before deployment. This selective application retains

GA-driven improvements without incurring excessive computational overhead for every generation.

3.10 Tools and Libraries

This environment relies on several well-established libraries to maintain reliability and comparability. PennyLane, an open-source Python framework, enables seamless quantum programming on both simulated and real hardware, allowing the easy translation of operation lists into executable circuits. NumPy supports efficient handling of matrices and arrays, reducing generation times and enabling large-scale simulations. Optuna automates hyperparameter tuning for the GA, storing experiment results in a manageable database that facilitates performance analysis. Lastly, Slurm distributes computational jobs across multiple server nodes, enabling parallel processing and accelerating large-scale or repeated optimizations. Taken together, these tools create a robust and adaptable environment for generating, evaluating, and improving candidate circuits across various mutation settings.

4 EXPERIMENTAL SETUP

This section describes the decisions made to set up the GA evaluations for quantum circuit optimization. It covers how the evaluations were conducted, including computational resources, dataset generation, hyperparameter tuning, and performance metrics. Each element aims to create a robust, reproducible framework that can assess the GA's effectiveness across diverse scenarios.

4.1 Dataset

The datasets used in this work comprise randomly generated PennyLane circuits, subsequently optimized as noted in Section 3.9. Each dataset contains circuits with a fixed number of qubits yet varying circuit depths to resemble realistic quantum states. The qubit count ranges from five to eight, as fewer than five qubits often resulted in near-optimal solutions that showed little variation among mutation methods, and more than eight qubits demanded prohibitive computational resources. Circuit depths span from five to fifteen operations, providing enough complexity to challenge the GA while remaining computationally tractable.

4.2 Performance Metric

The performance metric measures the GA's effectiveness in optimizing quantum circuits. Specifically, it is defined as the average of the highest final fitness scores across all circuits in a dataset. This metric evaluates both solution quality and consistency under varying datasets and optimization conditions.

4.3 Hyperparameter Optimization

Hyperparameter optimization proceeded in two stages. The first stage focused on initialization settings, such as average, minimum, and maximum circuit depths, and tested different numbers of islands for a generic target circuit with four to eight qubits. Wide parameter bounds in Optuna enabled a broad search to establish a solid starting point for subsequent experiments.

The second stage refined the mutation process. It used narrower parameter bounds to optimize mutation rate, the number of mutations per candidate, and related parameters. This step aimed to

strike a balance between exploration and exploitation, enhancing the GA's capacity to discover high-quality solutions without converging prematurely.

All experiments ran on Linux-based high-performance machines in the CIP-Pool at Ludwig Maximilian University, using SLURM for job scheduling. SLURM's built-in seeding ensured reproducibility, with each experiment executed on seeds one through four in random order.

5 RESULTS

This work examines how different mutation strategies affect quantum state preparation using a GA. The flexible environment introduced in earlier sections generated extensive data through numerous experiments, which were then evaluated for insights into the effectiveness of each strategy. A detailed description of the experimental setup, including computational resources and dataset generation, can be found in Section 4.

Rather than comparing the GA's performance to other methods, we focus on how each mutation strategy performs within the same GA framework. We infer effectiveness through well-known metrics, with fidelity as a primary measure of optimization success.

5.1 Genetic Algorithm Performance

We first tested the GA with target states using different qubit counts to explore how complexity influences algorithmic performance. Four-qubit and six-qubit target states emerged as especially informative. Four-qubit systems, while consistently solved by the GA, did not present significant challenges to distinguish among mutation strategies. Fig. 8 and Fig. 9 show that most strategies performed near optimality for four qubits, offering limited insights into their comparative strengths.

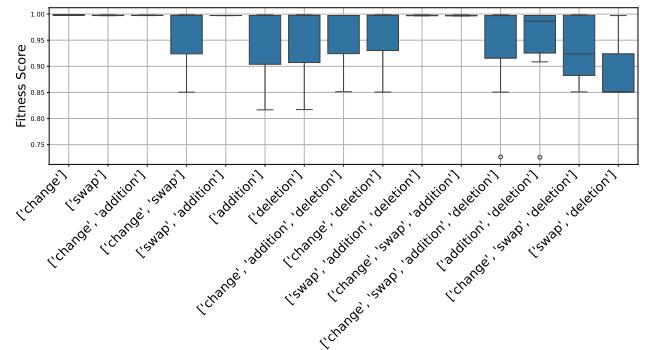


Figure 8: Histogram of average final fitness scores for various mutation strategy combinations on four qubits.

Because four qubits proved too simple for a meaningful test, we shifted to six-qubit target states, which provided higher complexity and revealed clearer differences in the performance of various mutation strategies.

5.2 Impact of Mutation Strategies

We evaluated different mutation strategies for six-qubit target states using a dataset of 150 distinct, optimized circuits. Each target state

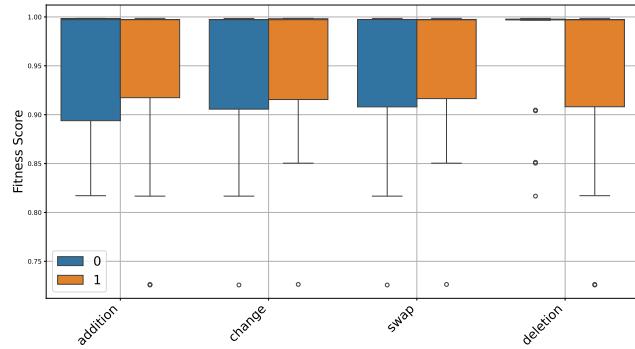


Figure 9: Fitness score comparison on four qubits, showing whether a strategy was present in a trial. Most results approach optimal values.

underwent 150 GA generations. This experiment aimed to identify whether specific strategies or strategy combinations yield superior fitness outcomes. We used diverse hyperparameter values to minimize biases in parameter settings. Table 1 summarizes the results of 300 trials under different conditions.

Table 1: Results from 300 trials with different parameter settings chosen by Optuna. The highest mean and 75th percentile appear in the swap, addition combination, while swap, addition, deletion yields the highest median and 25th percentile.

| Strategy | Mean | Median | 25th Pctl | 75th Pctl |
|---------------------|---------------|---------------|---------------|---------------|
| deletion (del) | 0.3432 | 0.2803 | 0.2452 | 0.3488 |
| addition (add) | 0.2939 | 0.2725 | 0.2614 | 0.3435 |
| change (ch) | 0.2660 | 0.2557 | 0.2215 | 0.3156 |
| swap (sw) | 0.3070 | 0.2480 | 0.2302 | 0.2973 |
| ch, add | 0.3209 | 0.2895 | 0.2414 | 0.3168 |
| sw, add | 0.3598 | 0.2864 | 0.2387 | 0.4243 |
| sw, del | 0.3531 | 0.2825 | 0.2495 | 0.3626 |
| ch, del | 0.3298 | 0.2704 | 0.2607 | 0.3088 |
| ch, sw | 0.3281 | 0.2684 | 0.2483 | 0.3096 |
| add, del | 0.2845 | 0.2628 | 0.2587 | 0.2844 |
| sw, add, del | 0.3424 | 0.2962 | 0.2630 | 0.3554 |
| ch, sw, add | 0.3207 | 0.2833 | 0.2506 | 0.2980 |
| ch, add, del | 0.2740 | 0.2746 | 0.2416 | 0.2938 |
| ch, sw, del | 0.3289 | 0.2574 | 0.2260 | 0.3674 |
| ch, sw, add, del | 0.2827 | 0.2650 | 0.2420 | 0.2828 |

Individual Strategies. Fig. 11 shows that including a mutation strategy typically boosts performance relative to not using it, except for *change*, which lowers overall fitness. *Swap* increases variability the most, while *change* reduces it—at the cost of lower mean fitness. Both *deletion* and *addition* have minimal influence on standard deviation.

Combined Strategies. Fig. 10 highlights several promising combinations, especially *swap, addition, swap, addition, deletion*, and

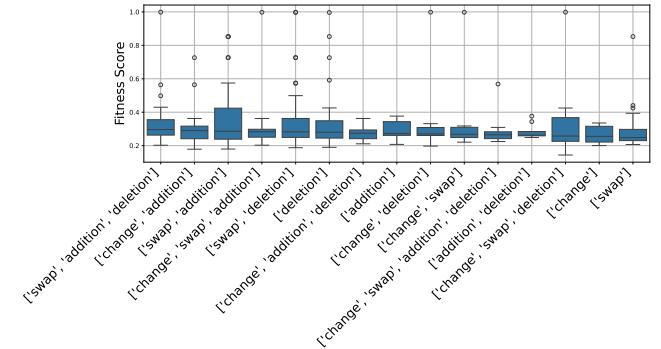


Figure 10: Histogram of average final fitness scores for various strategy combinations on six qubits.

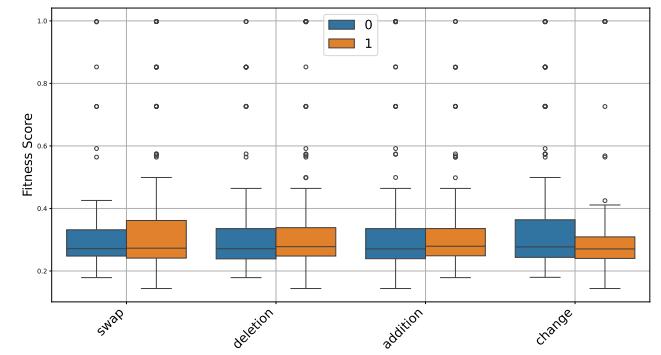


Figure 11: Effect on fitness scores if a strategy was present in a trial for six qubits. Outliers likely reflect the inherent randomness of the GA.

swap, deletion. Notably, these do not include *change*, reinforcing that *change* generally underperforms. The strong impact of *swap* is intuitive, because swapping operations can introduce significant restructuring. The use of *addition* or *deletion* helps maintain or adjust circuit complexity.

Summary. *Swap, deletion* emerges as a particularly robust strategy, consistently achieving near-top results and reducing circuit depth over time. *Addition* can be useful but constantly increases depth, making *swap, deletion* an efficient choice for retaining strong performance while limiting circuit growth.

5.3 Impact of Mutation Rate, Population, and Adaptive Mutation

We also investigated how hyperparameters—mutation rate, population size, and adaptive mutation—affect mutation strategy performance.

Mutation Rate. Fig. 12 and Fig. 13 show that, within the tested ranges, further adjusting mutation rates after narrowing their bounds had little effect on final fitness. This indicates prior tuning already identified rates near local optima.

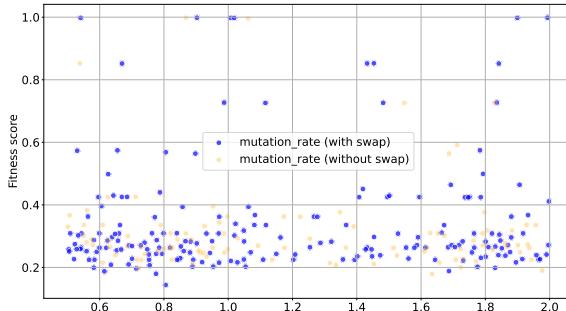


Figure 12: Influence of different mutation rates on *swap*. Rates were already near optimal, resulting in minimal performance changes.

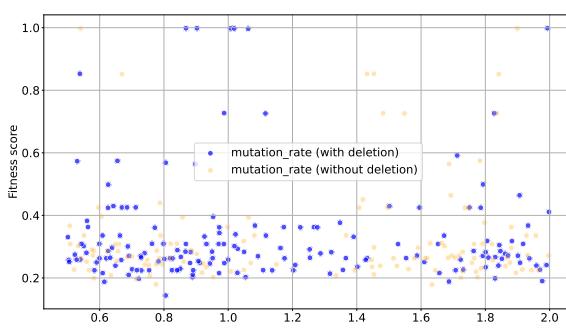


Figure 13: Influence of different mutation rates on *deletion*. Minimal effects are visible.

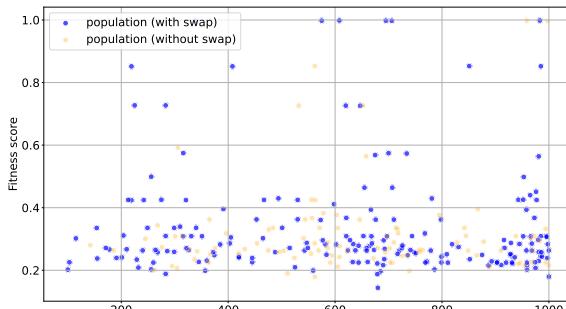


Figure 14: Effect of population size on *swap*. Larger populations slightly increase performance but demand more computational resources.

Population Size. Fig. 14 and Fig. 15 show a minor performance boost when increasing the population size, but the gains do not justify the substantial rise in computational costs. Generally, allocating resources to run more generations may be more beneficial than increasing population size.

Adaptive Mutation. Fig. 16 and Fig. 17 illustrate that adaptive mutation reduces variance but tends to yield lower overall fitness. Although it balances exploration and exploitation by altering rates

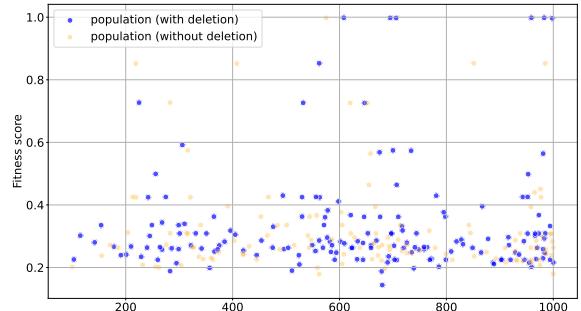


Figure 15: Effect of population size on *deletion*, showing a slight improvement at higher population sizes.

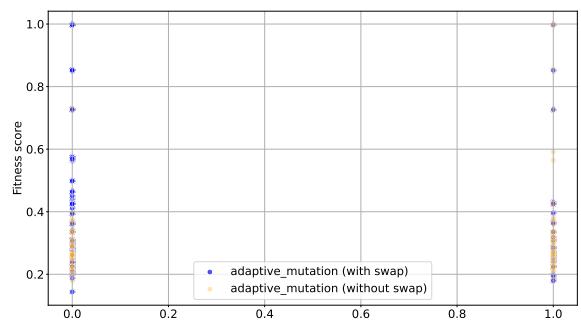


Figure 16: Impact of adaptive mutation on *swap*. Although variability decreases, final fitness also trends lower.

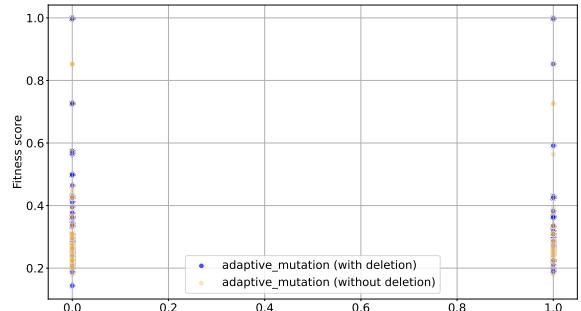


Figure 17: Impact of adaptive mutation on *deletion*. Similar patterns of reduced variability coincide with reduced fitness.

in response to population diversity and convergence, those adjustments appear to curb performance. A well-tuned static rate often suffices and may simplify implementation while producing stronger final results.

6 CONCLUSION

This work investigated how different mutation strategies affect GA performance in quantum state preparation. We conducted extensive experiments in a highly parameterized quantum circuit environment, concluding that the *swap*, *deletion* strategy generates optimized quantum circuits with the greatest efficiency.

The rising demand for automated circuit synthesis and optimization in NISQ hardware motivated this research. By examining various mutation strategies and their interactions with quantum circuits, we provided empirical data that highlight both the strengths and limitations of GA-based approaches. Our flexible environment and comprehensive dataset represent a main contribution of this work, enabling systematic GA evaluations and offering a resource for further improvements in quantum circuit optimization.

Hardware constraints limited the number of experiments we could conduct. Although we aimed for thorough testing within these constraints, longer trials and larger populations may provide deeper insights. While the findings may not have immediate large-scale effects, identifying potential improvements in the rapidly evolving quantum computing field could lead to significant long-term impact.

Future investigations could explore larger quantum systems, bigger populations, and direct comparisons with alternative optimization methods. Further refining adaptive mutation schemes, perhaps by integrating dynamic heuristics, may also improve convergence speed and performance. We hope that these results encourage ongoing research into evolutionary algorithms for quantum computing, thereby advancing more efficient and fully automated quantum algorithms.

REFERENCES

- [1] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. 2019. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology* 4, 4 (2019), 043001.
- [2] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (2017), 195–202.
- [3] Giulio Chiribella, G Mauro D'Ariano, and Paolo Perinotti. 2008. Quantum circuit architecture. *Physical review letters* 101, 6 (2008), 060401.
- [4] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. 2006. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag, Berlin, Heidelberg.
- [5] Eduardo A Coello Pérez, Joey Bonitati, Dean Lee, Sofia Quaglionni, and Kyle A Wendt. 2022. Quantum state preparation by adiabatic evolution with custom gates. *Physical Review A* 105, 3 (2022), 032403.
- [6] Floyd M Creevey, Charles D Hill, and Lloyd CL Hollenberg. 2023. GASP: a genetic algorithm for state preparation on quantum computers. *Scientific reports* 13, 1 (2023), 11956.
- [7] Yan Ge, Wu Wenjie, Chen Yuheng, Pan Kaisen, Lu Xudong, Zhou Zixiang, Wang Yuhan, Wang Ruocheng, and Yan Junchi. 2024. Quantum Circuit Synthesis and Compilation Optimization: Overview and Prospects. arXiv:2407.00736 [quant-ph] <https://arxiv.org/abs/2407.00736>
- [8] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, Pennsylvania, USA) (STOC '96). Association for Computing Machinery, New York, NY, USA, 212–219. <https://doi.org/10.1145/237814.237866>
- [9] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. 2002. *Classical and Quantum Computation*. American Mathematical Society, USA.
- [10] Padmavathi Kora and Priyanka Yadlapalli. 2017. Crossover operators in genetic algorithms: A review. *International Journal of Computer Applications* 162, 10 (2017).
- [11] Michael Kölle, Tom Schubert, Philipp Altmann, Maximilian Zorn, Jonas Stein, and Claudia Linnhoff-Popien. 2024. A Reinforcement Learning Environment for Directed Quantum Circuit Synthesis. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART. INSTICC, SciTePress*, 83–94. <https://doi.org/10.5220/0012383200003636>
- [12] Yeong-Cherng Liang, Yu-Hao Yeh, Paula EMF Mendonça, Run Yan Teh, Margaret D Reid, and Peter D Drummond. 2019. Quantum fidelity measures for mixed states. *Reports on Progress in Physics* 82, 7 (2019), 076001.
- [13] Tom V Mathew. 2012. Genetic algorithm. *Report submitted at IIT Bombay* 53 (2012).
- [14] Brad L Miller, David E Goldberg, et al. 1995. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems* 9, 3 (1995), 193–212.
- [15] Fernando T. Miranda, Pedro Paulo Balbi, and Pedro C. S. Costa. 2023. Synthesis of Quantum Circuits with an Island Genetic Algorithm. arXiv:2106.03115 [quant-ph] <https://arxiv.org/abs/2106.03115>
- [16] Mikko Möttönen and Juha Vartiainen. 2005. Decompositions of general quantum gates. *Frontiers in Artificial Intelligence and Applications* (05 2005).
- [17] Elijah Pelofske, Andreas Bärtschi, and Stephan Eidenbenz. 2022. Quantum volume in practice: What users can expect from nisq devices. *IEEE Transactions on Quantum Engineering* 3 (2022), 1–19.
- [18] Riccardo Porotti, Antoine Essig, Benjamin Huard, and Florian Marquardt. 2022. Deep reinforcement learning for quantum state preparation with weak nonlinear measurements. *Quantum* 6 (2022), 747.
- [19] John Preskill. 2018. Quantum computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.
- [20] Tom Rindell, Berat Yenilen, Niklas Halonen, Arttu Pönni, Ilkka Tittonen, and Matti Raasakka. 2023. Exploring the optimality of approximate state preparation quantum circuits with a genetic algorithm. *Physics Letters A, T<3T* 475 (2023), 128860.
- [21] Cristian Ruican, Mihai Udrescu, Lucian Prodan, and Mircea Vladutiu. 2008. A genetic algorithm framework applied to quantum circuit synthesis. *Nature Inspired Cooperative Strategies for Optimization (NISCO 2007)* (2008), 419–429.
- [22] Yuval R Sanders, Guang Hao Low, Artur Scherer, and Dominic W Berry. 2019. Black-box quantum state preparation without arithmetic. *Physical review letters* 122, 2 (2019), 020502.
- [23] Peter Selinger. 2015. Efficient Clifford+T approximation of single-qubit operators. *Quantum Info. Comput.* 15, 1–2 (Jan. 2015), 159–180.
- [24] V.V. Shende, S.S. Bullock, and I.L. Markov. 2006. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 6 (2006), 1000–1010. <https://doi.org/10.1109/TCAD.2005.855930>
- [25] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.
- [26] Andrew M Steane. 1999. Efficient fault-tolerant quantum computing. *Nature* 399, 6732 (1999), 124–126.
- [27] Leo Süntel, Darya Martyniuk, Denny Mattern, Johannes Jung, and Adrian Paschke. 2023. GA4QCO: genetic algorithm for quantum circuit optimization. *arXiv preprint arXiv:2302.01303* (2023).
- [28] Darrell Whitley, Soraya Rana, and Robert B Heckendorf. 1999. The island model genetic algorithm: On separability, population size and convergence. *Journal of computing and information technology* 7, 1 (1999), 33–47.
- [29] Andrew Wright, Marco Lewis, Paolo Zuliani, and Sadegh Soudjani. 2024. T-Count Optimizing Genetic Algorithm for Quantum State Preparation. *arXiv preprint arXiv:2406.04004* (2024).
- [30] Xin-Chuan Wu, Marc Grau Davis, Frederic T Chong, and Costin Iancu. 2020. QGo: Scalable quantum circuit optimization using automated synthesis. *arXiv preprint arXiv:2012.09835* (2020).
- [31] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. 2022. Quantum state preparation with optimal circuit depth: Implementations and applications. *Physical Review Letters* 129, 23 (2022), 230504.

Received 20 January 2025