Model-Driven Optimization for Automated Quantum Program Synthesis

Patryk Tymoteusz Pilichowski

University of Antwerp,

Abstract

This report presents the synthesis and review of a model-driven optimization (MDO) approach to automated quantum program synthesis proposed in the reviewed paper. A quantum synthesis problem is implementing undefined circuit components (Oracles) in quantum programs. This requires both functional correctness and optimization of noisy intermediate scale quantum (NISQ) trade-offs between accuracy and computational cost. Model-driven optimization (MDO), specifically the MOMoT framework, can be applied to quantum program synthesis by representing circuits as explicit metamodels, applying rule-based transformations via Henshin, and optimizing through NSGA-III genetic algorithms with fitness evaluation delegated to quantum SDKs. The approach is shown in the reviewed paper on Grover's algorithm, and the authors report multiple Pareto-optimal Oracle implementations achieving up to perfect accuracy with varying computational costs. Compared to alternative methodologies, the MDO approach may provide better modularity, SDK-agnostic design, and separation of concerns between circuit structure and optimization logic, however scalability remains constrained by classical simulation limits to only tens of qubits. These results suggest that established model-driven engineering techniques can be effectively used to solve challenges in quantum software development.

Keywords: Model-driven Optimization, Quantum Software Engineering, Program Synthesis, Model-driven Engineering

Generative AI tools were used solely for language rephrasing and clarity.

Email address: patryk.pilichowski@student.uantwerpen.be (Patryk Tymoteusz Pilichowski)

Contents

1. Introduction

The problem of a quantum program with missing or abstract parts of a circuit that block execution, known as an Oracle, is addressed with automated quantum program synthesis. An implementation of the Oracle is produced such that the quantum program executable, and:

1. The resulting circuit realizes a desired quantum functionality, typically expressed as a target quantum state.
2. The circuit optimally balances objectives relevant to the NISQ-era, such as the non-trivial trade-off between accuracy and the computational cost of the produced solutions. Accuracy is defined as the similarity between produced output quantum state and expected target state, while computational cost reflects the structural complexity of the circuit, for example the length and depth of the gate sequence, where longer circuits typically incur higher costs and increased susceptibility to noise.

Automated quantum program synthesis can be relevant when it is difficult to manually design fitting quantum circuits to a corresponding problem. This is especially the case when non-trivial optimization constraints and objectives are present, and thus difficult to approach analytically.

Formally, the problem is to search the space of valid quantum circuits composed of elementary quantum gates and parameters under certain structural and semantic constraints, to find circuits that satisfy functional correctness while optimizing multiple cost metrics.

In the paper under study, the problem is instantiated as a model-driven optimization (MDO) problem:

- Quantum circuits are represented as model instances conforming to the corresponding meta-model.
- Circuit construction is performed via rule-based model transformations.
- Correctness and quality evaluation is delegated to a quantum SDK that integrated into the (MOMoT) framework

The MOMoT framework implements this pattern, integrating the Eclipse Modeling Framework (EMF) for model management, Henshin for model transformations (that are actually formulated as graph transformations), and the MOEA framework for multi-objective search. Once the meta-model and transformation rules are defined, the framework handles the optimization mechanics.

This report is a synthesis and critical review of the work presented in Gemeinhardt et al. (2023); all methods, demonstrations, and results discussed originate from the reviewed paper and related work and are not reproduced or extended here.

The remainder of this report is organized as follows. Section 2 discusses related work. Section 3 provides essential background on quantum computing, genetic algorithms, and model-driven optimization. Section 4 details prominent approaches to automated quantum program synthesis such as genetic programming, reinforcement learning, and the MDO-based method. Section 5 describes the demonstration of the MDO approach on Grover's algorithm. Section 6 analyzes and compares these methods. Finally, Section 7 concludes with possible implications and future directions.

2. Related Work

Automated quantum program synthesis has been approached through genetic programming, reinforcement learning, and model-driven optimization paradigms.

Băutu and Bautu (2007); Spector et al. (1998) established genetic programming for quantum circuits by evolving gate sequences through mutation and crossover. Stein and Färber (2025) extended this with improved configurability and quantum advantage considerations. Kuo et al. (2021) developed a deep RL framework treating circuit construction as sequential decision-making.

Burdusel and Zschaler (2019) addressed scalability in search-based model engineering through distributed workflows. The MOMoT framework builds on this, integrating EMF, Henshin transformations, and MOEA for multi-objective search.

Gemeinhardt et al. (2023) applied MDO to quantum program synthesis using explicit meta-modeling and rule-based transformations with no need for direct encoding.

3. Background

3.1. Quantum Computing

3.1.1. Quantum State

The qubit, also known as a quantum state, is the basic unit of quantum information. It is a 2-dimensional complex vector typically denoted as $|\psi\rangle$, using the Dirac notation. It can be described as a linear combination of complex column vectors: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, with probability amplitudes $\alpha, \beta \in \mathbb{C}$ and $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$.

The equation represents superposition. Unlike the classical bit, it can exist in both state 0 and 1 simultaneously. Once the qubit is measured however, it must collapse to one of the two, with respective probabilities $|\alpha|^2$ and $|\beta|^2$. It thus lastly follows that the probabilities must be constrained to the second axiom of probability theory: $|\alpha|^2 + |\beta|^2 = 1$.

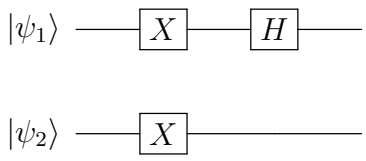
3.1.2. Quantum Gates

Operations on qubits can be described by quantum gates represented by unitary matrices. The action of, for instance, unitary operation $X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$ applied on a 1-qubit state $|\psi_1\rangle$ is found from their multiplication: $U|\psi_1\rangle = |\psi_2\rangle$. A composition of quantum gate(s) is called a quantum circuit, whose initial quantum state is typically $|0\rangle$. A quantum circuit of the last shown example can also be represented visually using a diagram:

$$|\psi_1\rangle \text{ ——} \boxed{X} \text{——}$$

In practice, a quantum system has more than a single qubit input. An n-qubit input $|\psi\rangle \in \mathbb{C}^{2^n}$ is equal to the following general equation: $|\psi\rangle = \alpha_0|0...00\rangle + \alpha_1|0...01\rangle + ... + \alpha_{2^n-1}|1...11\rangle$, also written more compactly: $\alpha_0|0\rangle + \alpha_1|1\rangle + ... + \alpha_{2^n-1}|2^n-1\rangle = \sum_{i=0}^{2^n-1}\alpha_i|i\rangle$ with probability amplitudes $\alpha_i \in \mathbb{C}$. Multi-qubit circuit can be described using the tensor product.

For instance, the following is a 2-qubit circuit acting on (unentangled) input $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$: $(H \otimes I)(X \otimes X)\,|\psi\rangle = (HX \otimes X)\,|\psi\rangle$, which corresponds to the following diagram:



3.2. Genetic Algorithm

A genetic algorithm is a search and optimization technique inspired by natural evolution. We typically start with a randomly generated population of candidate solutions, and then:

1. Candidate solutions are evaluated using fitness functions.
2. Selection: "fitter" candidates are selected to be parents using some selection operator.
3. Crossover: Pairs of parents are "combined" to create their offspring using some crossover operator.
4. Mutation: Offsprings are randomly changed using some mutation operator.
5. The new generation are now the candidate solutions. Repeat the process or stop.

3.3. Model-driven Optimization

3.3.1. Description

Model-driven optimization (MDO) is a specific application of search-based model engineering (SBME) where the goal is to optimize models to quantitative objectives while satisfying constraints. An MDO problem consists of:

- Domain meta-model: Definition of the problem domain which captures entities, relationships, constraints, etc...
- Initial model instance: A conforming instance that serves as a starting point.

- Transformation model: A set of rules that can modify the model instance into a new conforming instance.
- Objective functions: the mathematical function(s) to optimize (minimize or maximize).
- Constraints: Conditions that the model must satisfy.
- Fitness functions: Functions that measure the solution quality.
- Search algorithm: Guides the exploration of the search space by iteratively applying transformation rules, evaluating candidates against objective functions to find Pareto-optimal solutions: the set of all best possible trade-offs to a multi-objective problem. Genetic algorithms (e.g NSGA-II or NSGA-III) are typically used for this purpose.

The process of this approach can be generally described as:

1. Start with an initial conforming instance.
2. Apply transformation rules to generate new conforming model variants.
3. Evaluate each variant using fitness functions.
4. Use the search algorithm to evolve the candidates toward optimality.
5. Go back to step 2 unless termination.
6. Output a set of Pareto-optimal models, also called the Pareto front.

4. Automated Quantum Program Synthesis

4.1. Genetic Programming Approach

Genetic Programming (GP) represents quantum circuits as syntax trees or linear sequences of gate operations. Each circuit is a "chromosome" that evolves through crossover (swapping subcircuits) and mutation (inserting, deleting, modifying gates). Fitness evaluation simulates the circuit and compares output to target state.

This method works directly on circuit encodings and there's no "model-driven" layer. Circuit structure is thus implicit in the representation. Even the basic GP techniques like Băutu and Bautu (2007); Spector et al. (1998) are adaptable to new problems which makes them reusable. There are modern GP tools for quantum synthesis, such as Stein and Färber (2025), that are even more configurable.

4.2. Reinforcement Learning Approach

Reinforcement learning (RL) treats circuit synthesis as a sequential decision problem. An agent learns a policy that constructs circuits gate-by-gate, receiving rewards based on circuit quality. Deep RL can discover sophisticated patterns through trial-and-error over many training iterations.

The difference is that RL learns how to search rather than directly searching. The policy network becomes a learned heuristic for circuit construction. Generally this tends to be skewedly beneficial to a set of similar synthesis problems rather than a set of varying ones. When circuit requirements change frequently, the method can become less reusable and the training cost may not be justified. There are however, reusable deep RL frameworks like Kuo et al. (2021) that are modular and compatible with different RL architectures.

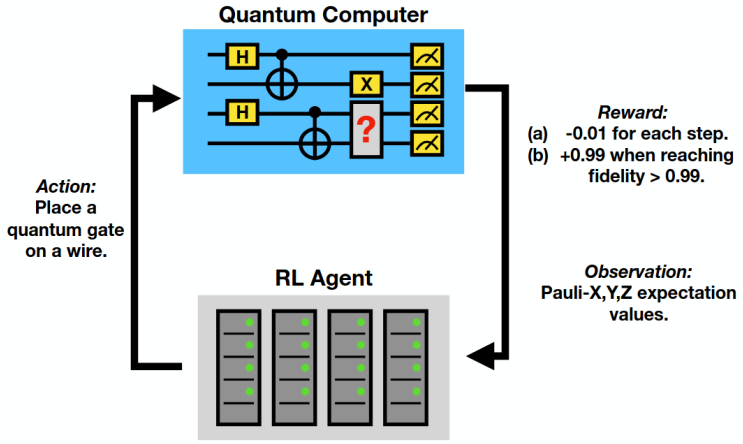


Figure 1: Workflow of the DRL-based approach Kuo et al. (2021)

4.3. MDO-based Approach

Gemeinhardt et al. (2023) lays out the method as following:

1. Define the quantum circuit with a placeholder Oracle and the target quantum state using a Q-SDK.
2. Circuit is transformed to a Quantum Circuit Model instance conforming to the Quantum Circuit Language meta-model.

3. Provide rules & conditions of synthesis (e.g transformation rules), the search configuration (genetic algorithm NSGA-III and its configuration).
4. Candidate Oracle implementations are generated by applying transformation rules defined in EMF transformation tool Henshin.
5. Quantum Circuit Model with Implemented Oracle is translated to a quantum program, executable in Q-SDK.
6. Q-SDK simulates the circuit, computes fitness values (e.g accuracy, #gates, depth), and optimizes gate parameters.
7. New candidates are produced using the search algorithm, by typically making use of selection, crossover, and mutation operators in a genetic algorithm. Go back to step 4 unless termination condition is true.
8. Output is a Pareto-optimal set of Quantum Circuit Models transformed to SDK-specific executable quantum programs.

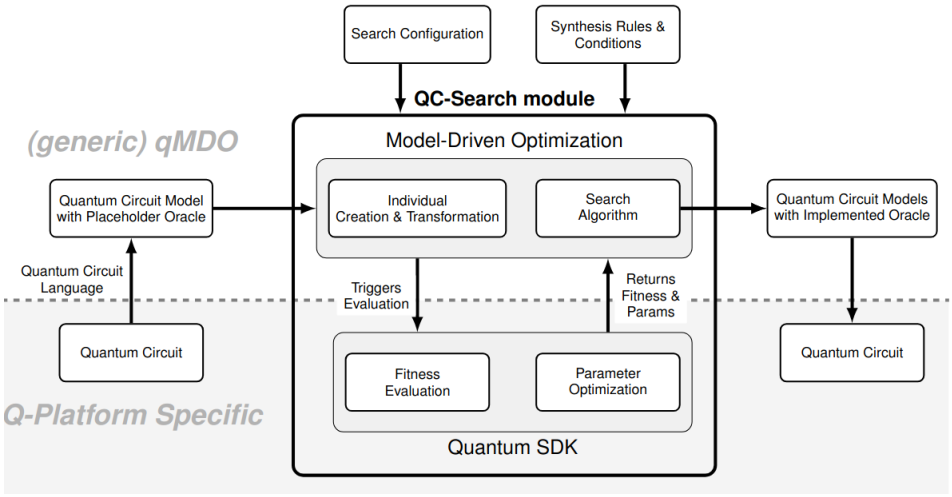


Figure 2: Workflow of the MDO-based approach Gemeinhardt et al. (2023)

5. Demonstration

The approach is demonstrated by the authors on a 3-qubit Grover Search algorithm, a quantum algorithm that provides quadratic speedup compared to classical approaches. This use case allows validation of the approach against a known analytical solution.

The Grover Search algorithm requires an Oracle operator that marks the computational basis states of interest. In this example, the Oracle must mark the sixth and seventh computational basis states out of eight possible states for a 3-qubit system by flipping the sign of their probability amplitudes while leaving others unchanged. Up to normalization:

$$[1, 1, 1, 1, 1, 1, 1, 1] \rightarrow [1, 1, 1, 1, 1, -1, -1, 1] \tag{1}$$

The analytical solution uses two controlled-Z gates, as shown in Figure 4. The synthesis problem is to have the search algorithm discover this or equivalent implementations automatically. The framework must explore the space of possible gate combinations to find implementations that achieve the required transformation while optimizing for multiple objectives including accuracy and circuit complexity.

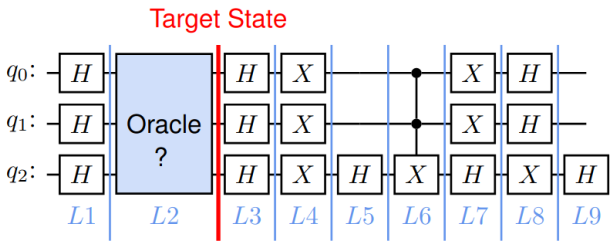


Figure 3: Circuit with Oracle.

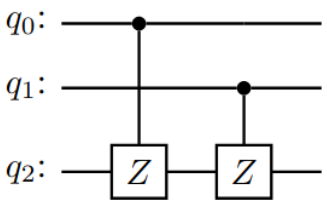


Figure 4: Analytical solution.

The MDO approach produced multiple Oracle implementations with different trade-offs between accuracy and computational cost. Specifically, three solutions were produced for the Pareto front: a minimal single-gate implementation with moderate accuracy, a parameterized single-gate solution with improved accuracy, and a 2-gate implementation achieving perfect accuracy. The convergence behavior showed the search algorithm evolved increasingly accurate quantum programs across generations, eventually reaching the target quantum state transformation.

The known analytical solution was not reproduced in initial runs, however it was found as a possible solution after narrowing the search space by reducing the set of usable gates. The paper's results confirm the feasibility of the MDO framework, that it can synthesize functionally correct quantum programs and balance the corresponding trade-offs.

6. Discussion

The MDO approach has certain advantages and limitations over genetic programming and reinforcement learning alternatives.

GP and RL typically require custom encodings tailored to each problem class. The MDO approach instead separates concerns through explicit meta-models and transformation rules. The components can be adapted to different synthesis problems within that domain (in this case quantum computing). The separation between model structure and optimization logic makes the solution more maintainable than encodings which are prone to change and where circuit representation is implicit. When requirements change, modifications can be localized to specific transformation rules rather than modification of the entire encoding scheme.

Fitness evaluation is delegated to external quantum simulators, which makes the framework portable and modular. This works with different SDKs by changing only the code generation target.

The multi-objective formulation using NSGA-III reflects NISQ issues well. Accuracy, circuit depth, and hardware cost must be traded off rather than optimized in isolation. Multi-objective GP approaches exist, but they may require implementing the optimization logic. The MDO approach gains this through MOEA framework integration, which provides tested implementations of various algorithms.

Building on MOMoT and EMF/Henshin provides mature tooling for model management, transformation, and optimization. This includes model validation, transformation debugging, visualization, and Eclipse integration.

The limitations are also notable. Scalability is the fundamental constraint, but also applicable to most QC-oriented solutions. Full quantum-state simulation limits the approach to small qubit counts, typically to tens of qubits, since classical simulation requires exponential resources. GP and RL face this same limitation with simulators. However, RL can potentially adapt to real quantum hardware using noisy fitness estimates, allowing larger applications.

Additionally, search space complexity is a challenge. With parameterized

gates and bigger set of gates for instance, the search space will grow rapidly with qubit count and circuit depth. GP approaches struggle with large search spaces as well, but RL can perhaps learn to navigate them more efficiently through heuristics acquired from training. The MDO approach relies on genetic algorithm exploration typically without problem specific guidance, which may be less efficient albeit broadly applicable.

Furthermore, solution quality is sensitive to population size, mutation rates, crossover probability, gate sets, repair rules, optimizer choice, etc... These sensitivities are only partially explored in the paper, making it hard to provide robust guidance. GP has decades of parameter tuning research with established heuristics. RL suffers from even greater sensitivity, often requiring extensive hyperparameter search.

The current evaluation lacks systematic performance comparison with specialized methods. Without benchmarking against GP and RL on identical problems with standardized metrics, it's difficult to assess whether MDO's structural benefits translate to competitive solution quality or efficiency. Modern GP tools and deep RL frameworks are highly optimized and may find better solutions faster despite custom encodings. Thorough empirical comparison is needed for definitive claims.

Finally, there seems to be an inherent trade off between generality and performance. The MDO approach prioritizes reusability and maintainability across varying problems, while specialized GP and RL approaches may achieve superior performance for a specific problem set. A custom GP with its own encoding and operators might outperform the generic MDO approach on particular circuits. Likewise, an RL agent trained extensively on a specific Oracle type might develop construction strategies that a generalized algorithm cannot match. Thus, the MDO approach chooses generality and software engineering benefits over performance on any single problem type.

7. Conclusion

The application of model-driven optimization to quantum program synthesis demonstrates a potential direction for bridging model-driven engineering with the new field of quantum computing. By representing quantum circuits as models and using established MDO frameworks like MOMoT, the

approach addresses the Oracle implementation problem while maintaining a separation of concerns for maintainability and reusability across various problems. The demonstration on Grover's algorithm confirms the feasibility of this method.

The MDO-based approach has advantages in terms of software engineering principles. Its explicit meta-modeling and rule-based transformations provide transparency compared to the implicit encodings found in genetic programming. The support for multi-objective optimization through the MOEA framework addresses the practical constraints of current quantum hardware by producing Pareto-optimal solution sets. These benefits come with acknowledged limitations however. Scalability remains fundamentally constrained by the exponential resource requirements of classical quantum simulation, which limits practical application to small qubit counts. The approach's generality may result in less efficient search compared to problem-specific genetic programming or reinforcement learning methods.

The work opens several directions for future investigation. Performance comparisons against established GP and RL approaches using standardized benchmarks would clarify the tradeoffs between generality and efficiency. Extensions to the quantum circuit modeling language could lead to producing more sophisticated solutions.

This research contributes to the field of quantum software engineering by showing how mature methodologies like model-driven engineering (specifically MDO), can be used to solve problems in a more nascent field like quantum computing. The inherent advantage of MDE is that everything can be modeled, and thus optimized within computational constraints.

References

Burdusel, A., Zschaler, S., 2019. Towards scalable search-based model engineering with mdeoptimiser scale, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 189–195. doi:10.1109/MODELS-C.2019.00032.

Băutu, A., Bautu, E., 2007. Quantum circuit design by means of genetic programming .

Gemeinhardt, F., Eisenberg, M., Klikovits, S., Wimmer, M., 2023. Model-driven optimization for quantum program synthesis with momot, in: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 614–621. doi:10.1109/MODELS-C59198.2023.00100.

Kuo, E.J., Fang, Y.L.L., Chen, S.Y.C., 2021. Quantum architecture search via deep reinforcement learning. URL: https://arxiv.org/abs/2104.07715, arXiv:2104.07715.

Spector, L., Barnum, H., Bernstein, H., 1998. Genetic programming for quantum computers .

Stein, C., Färber, M., 2025. Incorporating quantum advantage in quantum circuit generation through genetic programming. URL: https://arxiv.org/abs/2501.09682, arXiv:2501.09682.