Model-Driven Optimization for Quantum Program Synthesis with MOMoT

Felix Gemeinhardt, Martin Eisenberg, Stefan Klikovits, Manuel Wimmer
Christian Doppler Laboratory for Model-Integrated Smart Production (CDL-MINT)
Institute of Business Informatics – Software Engineering
Johannes Kepler University Linz, Austria
{firstname.lastname}@jku.at

***Abstract*—In the realm of classical software engineering, model-driven optimization has been widely used for different problems such as (re)modularization of software systems. In this paper, we investigate how techniques from model-driven optimization can be applied in the context of quantum software engineering. In quantum computing, creating executable quantum programs is a highly non-trivial task which requires significant expert knowledge in quantum information theory and linear algebra. Although different approaches for automated quantum program synthesis exist—e.g., based on reinforcement learning and genetic programming—these approaches represent tailor-made solutions requiring dedicated encodings for quantum programs. This paper applies the existing model-driven optimization approach MOMoT to the problem of quantum program synthesis. We present the resulting platform for experimenting with quantum program synthesis and present a concrete demonstration for a well-known quantum algorithm.**

***Index Terms*—Quantum Circuit Synthesis, Model-Driven Optimization, Quantum Software Engineering**

I. Introduction

Recent years have shown significant and diverse progress in the application of AI for model transformations, concerning endeavours such as model translation [1], generation [2], or repair [3], [4]. In the subfield of *Model-Driven Optimization* (MDO), AI methods integrated with modeling environments were investigated for optimizing models given constraints and quantitative objectives [5]–[7]. In this context, synthesis of design models was investigated as well with the Class Responsibility Assignment case [8]. Thereby, the aim is to assign program features to classes while maintaining separation of concerns, in order to obtain a high-quality software model. A similar challenge arises in the development of quantum programs, which has gained increasing traction in research due to the complexity and required knowledge.

Automated Software Engineering for Quantum Computing. The field of Quantum Software Engineering has emerged at the intersection of classical software engineering to *Quantum Computing* (QC) with the overall goal of porting concepts obtained from decades of research in classical software engineering to the domain of QC [9], [10]. The idea of applying *Model-Driven Engineering* (MDE) [11] techniques in the domain of QC has been adopted in the literature, e.g., regarding a conceptual model of quantum programs [12] and software modernization [13], [14]. Similarly, methods from classical computing, such as reinforcement learning [15]–[17] and genetic programming [18]–[20], have been proposed for the automated discovery and synthesis of quantum programs. The latter is particularly relevant for a broad adoption of QC, because designing a suitable quantum program for a desired computational task requires extensive knowledge in quantum information theory and linear algebra [10], [21].

Problem Statement. Even though *(i)* MDO has been adopted for AI-driven model transformations, *(ii)* techniques from MDE have been used in the context of QC [12]–[14], and *(iii)* search approaches have already been investigated for the problem of automated quantum program synthesis [18], [22], [23], to the best of our knowledge, there exist no MDO approaches for the automated synthesis of quantum programs. The latter not only entails capturing relevant entities of quantum programs, but also the possibility to harness the comfort of existing model transformation tools (e.g., Henshin [24]) and MDO frameworks, such as MOMoT [7]. Thus, we open the MDO toolbox for quantum programs to envision a holistic approach that supports the definition but also to facilitate development, in parts or as a whole, by leveraging AI methods.

Contributions. Our contributions can be summarized as follows. First, we present a method to conduct automated quantum program synthesis using the existing MDO engine MOMoT [7]. We integrate quantum-specific functionalities of dedicated *Quantum Software Development Kits* (Q-SDKs) into the MOMoT framework, in order to analyze generated programs in terms of their validity and utility in relevant criteria. Third, we demonstrate the feasibility of our proposed approach by applying it to a well-known quantum algorithm. Thereby, we foster the amalgamation of QC and AI [25] by investigating the application of MDO to quantum programs. Based on our initial results, we hope to stimulate further MDO research for Quantum Software Engineering. Drawing from dedicated models for representing quantum programs and configurable MDO approaches, it would allow to further analyse the application of different AI and MDE approaches for quantum program synthesis in the future.

Structure. The paper is structured as follows. Section II provides background information before the proposed quantum program synthesis approach is outlined in Section III. The demonstration of the method is presented in Section IV. We discuss related work in Section V and conclude in Section VI.

II. Background

This work incorporates several domains and disciplines that are introduced next, including the application of MDO tools, fundamentals of QC, and synthesis of quantum programs.

A. The MOMoT Framework

In general, MDO tools [5]–[7] leverage the abstraction principle of MDE to define and solve optimization problems on the model level. To that effect, optimization techniques are adopted and lifted to operate on the same abstraction level. The exploration of the search space takes place as changes are applied to the model by means of model transformations. Model transformations constitute a key technology in MDE [11], [26]. They derive a new model from an existing one (out-place) or modify a model directly (in-place) [27], with MDO mostly making use of the latter.

The MOMoT framework [7] denotes an integration of technologies to provide a problem-agnostic environment for model optimization. It bridges a modeling and transformation toolset (based on the EMF [28]) with the MOEA framework[1] as it integrates the latter's search algorithms to guide the exploration. The design space is induced by a provided set of transformation rules, which are formulated and executed as graph transformations using Henshin [24]. The latter rely on a graph matching mechanism to locate patterns to be transformed and manipulate the graph representation. The underlying models emerging from these transformations are evaluated on domain-related conditions and fitness functions.

B. Quantum Computing: The Circuit Model

In QC, information is processed by harnessing quantum mechanical phenomena. Different models exist for the information processing, whereby the quantum circuit model [29] is the most commonly used realistic model. In analogy to logic gates for classical computation, a quantum circuit comprises the application of quantum gates in an ordered manner [21]. Quantum gates are sequentially applied to the quantum information, which is stored on so-called *qubits*. The current era of QC is referred to as the *Noisy Intermediate-Scale Quantum* (NISQ) era, where limitations of the immature quantum hardware are mitigated by significant means of auxiliary classical computation [30]. In this era, the quantum gates are frequently parameterized, which allows to cope with the imprecise quantum information processing of current quantum hardware [31]. Parameters represent real-valued rotation angles ranging between 0 and 2π, specifying the concrete action of the gate. They are optimized by classical numerical quantum-aware parameter optimizers, which constitute a central element of NISQ-era quantum algorithms [31]–[34].

A simple example of a parameterized quantum circuit is depicted in Fig. 1. As illustrated, any arrangement of elementary quantum gates realizes a certain functionality and produces a certain quantum state. A quantum state is generally denoted as a complex-valued vector, where the according vector space

[1] http://moeaframework.org/

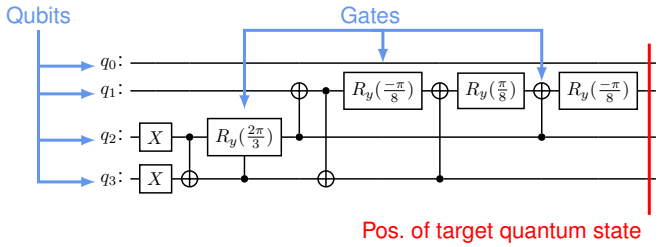


Fig. 1: Example parameterized quantum circuit; red line: position of produced quantum state

is spanned by the computational basis states. In quantum algorithms, circuits often comprise specific elements without a defined implementation of elementary gates, referred to as *Oracles* (cf. Fig. 3). Lacking the information on the implementation of Oracles prohibits the execution of a quantum circuit [35].

C. Automated Quantum Program Synthesis

Finding an implementation, i.e., the required elementary quantum gates to realize a certain functionality, is known to be a highly non-trivial task which is required for the quantum program to be executable. Furthermore, the prevailing limitations given in the NISQ-era result in trade-offs, where the *accuracy* and the *computational cost* of the solutions are arguably the most important metrics in this regard. Accuracy measures the functionality of a quantum program by assessing the similarity between the produced output quantum state and an expected target state. However, long sequences of gates—an indicator for the computational cost—may lead to cumulative errors in the computation which renders possible advantages in terms of theoretical accuracy obsolete.

Several approaches have been explored to automate the discovery and synthesis of quantum programs, where especially reinforcement learning [15]–[17], [36]–[38] and genetic programming approaches [39]–[41] have been studied.

D. Synopsis

In the next section, we will describe how the MOMoT framework can be applied to automatically synthesize quantum programs in terms of circuit models. To this end, a genetic algorithm is used to search for implementations that meet the trade-off between accuracy and computational cost present in the NISQ-era.

III. Approach

A. Overview

The goal of our proposed approach is to find an elementary gate implementation of a quantum program (i.e., the Oracle in Fig. 3), such that the overall quantum circuit is executable. As illustrated in Fig. 3, the position of the Oracle in the circuit as well as the position of the target state can be freely chosen by the user.

The overall process of our proposed approach is illustrated in Fig. 2. One central ambition is the neutrality with respect to

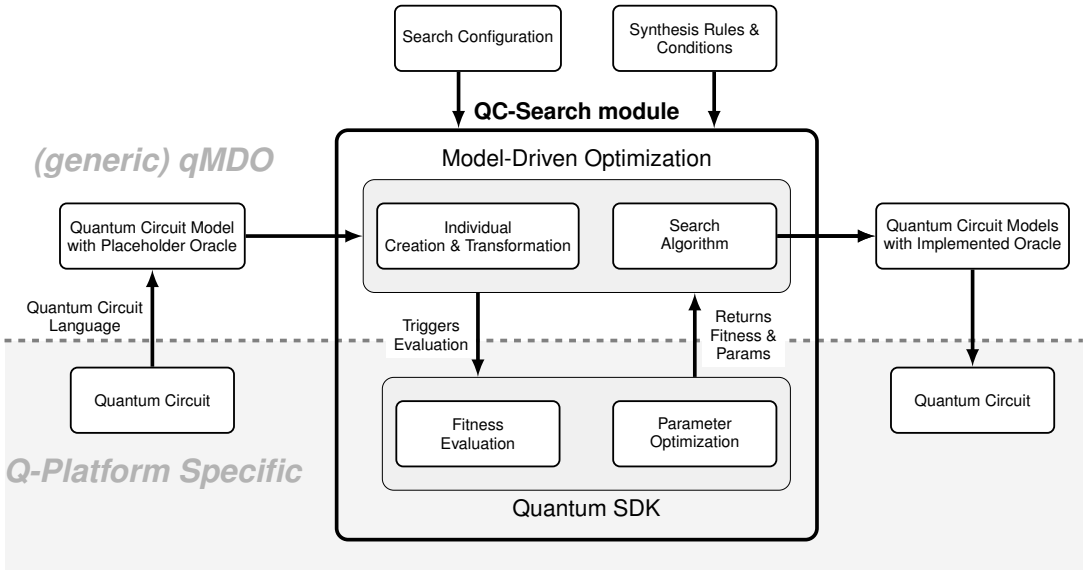


Fig. 2: Components and modules of the overall workflow of the quantum program search approach (Quantum circuit transformation to circuit model is left as future work)

quantum circuit definitions of different Q-SDKs. The quantum circuit designer can thus declare an overall quantum circuit with an arbitrary-sized Oracle as illustrated in Fig. 3 by using familiar Q-SDKs (*Q-Platform Specific*). In a next step, the circuit is transformed to a model instance of a quantum circuit which conforms to the according *Quantum Circuit Language*. Thus, the optimization is conducted on the generic level (*qMDO*). Note, that we have not implemented model injectors yet but rather start directly at the model-level and keep the transformation from the Q-SDK representation as future work. The resulting *Quantum Circuit Model with Placeholder Oracle* serves as input for the search process.

From the MDO perspective, inputs consist of *Synthesis Rules & Conditions* as the scope for exploration of the Oracle, and the *Search Configuration*, which concerns the algorithms that drive the exploration as well as quantum specific specifications such as the target quantum state and its position in the circuit. Bundled with the aforementioned input circuit using the configuration language of MOMoT [7], they form a fully-fledged and executable setup for the *QC-Search Module*.

The QC-Search Module comprises two parts. First, the *Model-Driven Optimization* employs a search engine to set and apply sequences of transformation rules on the input *Quantum Circuit Model*. The generation procedure is contingent upon the used method's explorative and exploitative capabilities. The fitness evaluation, an integral part of the search process, is outsourced to the *Quantum SDK*, since the calculations of the fitness targets should be natively conducted, as they require the information from simulated quantum processes. For this purpose, we provide automated code generation facilities that target the given *Quantum SDK* and build the respective quantum circuit object at this level. Beside calculations of the fitness targets, also the optimal parameters for the elementary gates comprising the Oracle are computed within the *Quantum SDK* and exported to the *Quantum Circuit Model*. The Pareto-set of executable *Quantum Circuit Models with Implemented Oracle* constitutes the output of the search process. This set represents the found trade-off solutions of the multi-objective search and the *Quantum Circuit Models* are transformed back to the specific Q-SDK representation.

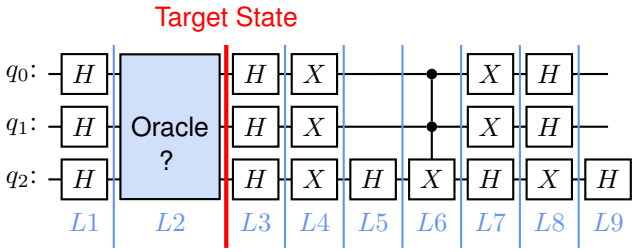


Fig. 3: Quantum circuit for Grover Search; Oracle: searched quantum program, red line: position of target quantum state, L1-L9: Layers

B. Model-Driven Optimization

As mentioned earlier, we apply the MOMoT framework to search for quantum circuits with preferable characteristics. From the brief introduction in Section II, a problem definition in MOMoT comprises *(i)* the domain meta-model, *(ii)* a conforming domain model instance as the starting point of the search, *(iii)* the transformation model, and *(iv)* the objective functions and potential constraints.

a) Domain model: The currently used modeling language for the quantum circuit synthesis problems and solutions (Fig. 4) is based on the modeling approach proposed in [42]. We select those language elements specific to the context of quantum program synthesis, where an elementary gate

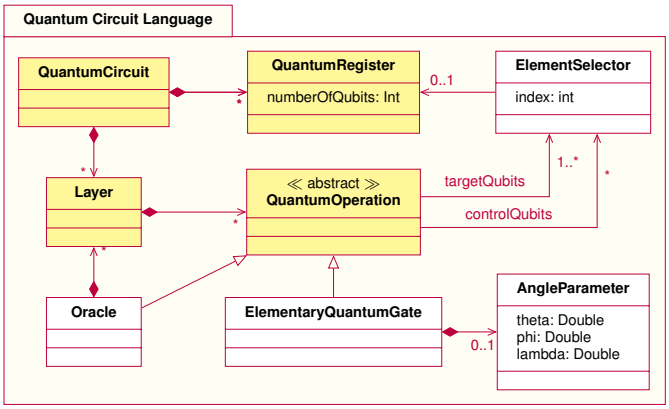


Fig. 4: Meta-Model of the Quantum Circuit Language for Circuit Synthesis

implementation of an undefined Oracle is searched. Note, that by using the EMF to define the language, it remains extensible for further concepts required in general quantum circuit design as well as allows to import other formats to quantum circuit models. A *QuantumOperation*, i.e., an *ElementaryQuantumGate* (H gate, X gate,...) or the *Oracle*, are applied to certain qubits via the *ElementSelector* class. The elementary quantum gates may be parameterized with *AngleParameters*. In this regard, *targetQubits* are the qubits that the gate actually acts on, while optional *controlQubits* control whether the gate is applied on the target qubit. We also added the concepts of *Layers* in our language. These ensure that gates are not simultaneously applied to the same qubit, i.e., within one layer, each qubit can only be addressed once as a target or control qubit. For example, consider the `H` gate located in layer 5 of Fig. 3. The subsequent multi-controlled `X` gate acts on three qubits, hence can only be placed in the following layer as qubit q_2 is already addressed by the `H` gate. The recursive structure allows to place gates into layers of the Oracle, but prohibits that the Oracle contains the same layer in which it is placed.

b) Initial model instance: The circuit under study (Fig. 3) accommodates three qubits, each of which is subject to a H gate (layer L1), which is followed by the yet undefined Oracle (layer L2). The remaining elementary quantum gates after the position of the target quantum state realize subsequent functionalities of the overall circuit. However, they do not influence the effects of preceding gates and are, therefore, not relevant in the context of the investigated problem.

c) Transformation model: The composition of the Oracle is done by gradual insertion of gates from the gate library. This includes: 1) the selection of the elementary quantum gate (e.g., H gate), 2) picking the layer to which the gate is to be added, 3) selection of the target qubit and potential control qubit, and 4) if the selected gate is subject to angle parameters, setting of those parameters.

These steps partly demand conditional logic and loops in the semantics, e.g., to select a qubit not already occupied or

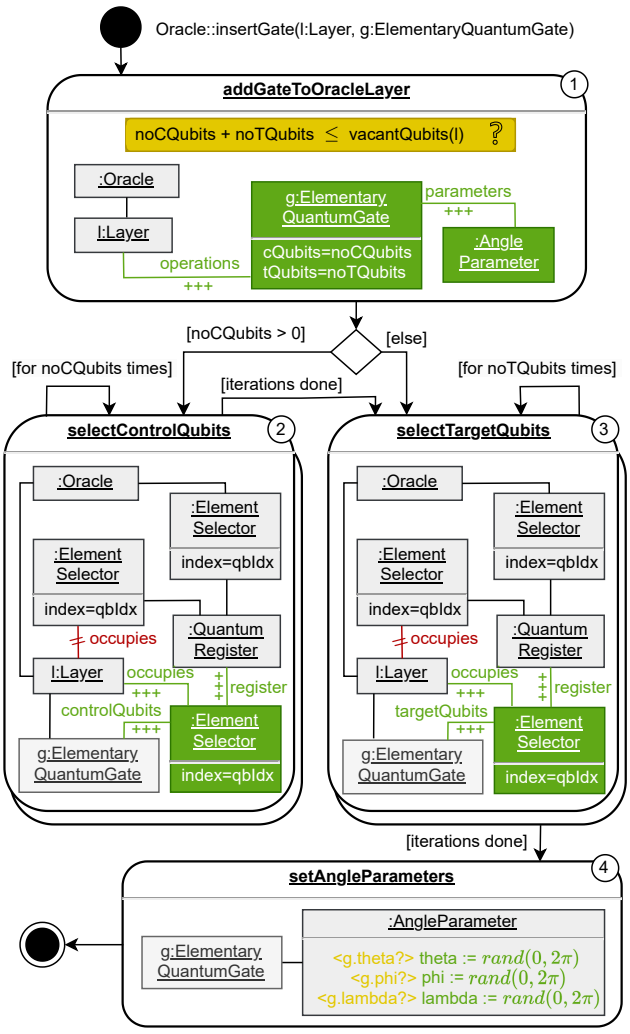


Fig. 5: Story diagram for the insertion of a gate into the Oracle

to realize multiple qubit selections depending on the gate. Such control structures are defined with Henshin units[2], which in turn call other units and/or rules.

The units and rules to fulfil the individual steps when adding a gate are embedded in an overlying unit and sequentially connected. The schematic procedure is depicted in Fig. 5. Based on the language elements in Fig. 4, a randomly selected *ElementaryQuantumGate* `g` is added to one of the *Layers* `l` of the *Oracle*. Depending on the gate type, the *controlQubit(s)* are designated first (2), if required, and then the *targetQubit(s)* (3). A negative application condition prevents the multiple selection of qubits at the current layer. Finally, again depending on the gate, the *AngleParameters* are initialized (4). Conditions are used to ensure semantically correct programs, e.g., in the first step to only select and proceed with a layer having sufficient vacant qubits to implement the chosen gate, or for setting the relevant angle parameters. In addition to adding gates to the Oracle, two further rules each reassign a target

[2]https://wiki.eclipse.org/Henshin/Units

qubit or a control qubit of a gate within the respective layer.

d) Fitness evaluation: The evaluation is performed using the *Quantum SDK*. A corresponding program is provided for this purpose, which is executed in advance by MOMoT in a separate process. Therein the Python environment is initialized with the libraries and functions required for the evaluation of a quantum circuit. These functions are later invoked from MOMoT after the code that defines the circuit, as reflected by a model instance, is generated and passed via a process interface. Upon termination of the circuit evaluation, the process sends back the fitness assessment, the objectives of which are elaborated in the following section.

C. Fitness Objectives and Parameter Optimization

Recently, we proposed a multi-objective genetic programming approach for automated quantum program synthesis, which allows for an explicit consideration of NISQ-era limitations and trade-offs [23]. Here, we reuse the fitness evaluation implemented for our previous search approach. The fitness objectives are defined as 1) the *overlap*—a commonly known measure in quantum physics [21]—evaluates the accuracy of a quantum program. It measures the similarity between the obtained output quantum state and the user-defined target quantum state at the user-defined position in the circuit; 2) The *number of gates* denotes the size of the individual. 3) The *depth* denotes the number of gates to be applied sequentially in the quantum circuit, i.e., the longest path between data input and output. This definition equals the lowest possible number of layers to realize a quantum program [35]. 4) The *number of non-local gates* denotes the number of multi-qubit gates. 5) A low *number of parameters* is supposed to speed-up the fitness evaluation and is taken as a rough measure for the complexity of the numerical optimization. Therefore, the trade-off between accuracy (1) and computational cost (2)-(5) [43] is explicitly considered. In each fitness evaluation, a numerical optimizer finds the parameters of the individual that maximize the overlap. These optimized parameters are returned together with the computed fitness values and updated in the quantum circuit model.

D. Tool Support

The modification of quantum programs in the QC Search module is based on rule transformations and the technologies integrated for their orchestration. Besides the demonstrated assembly of the Oracle, the scope of modifications is extensible within the Henshin Ecosystem. The configuration language in MOMoT provides high reusability, e.g. of the transformation model set out in Section III-B, for any programs of the proposed language. Furthermore, it represents a common source for all adjustments concerning search scope, search algorithms as well as quantum-related settings like the allowed number of gates or layers. The Quantum Circuit Models are realized using the EMF as an Eclipse plug-in[3]. The proposed approach is transparent concerning the lower-level quantum programming language. However, for demonstration, the Python-based [44] Qiskit Q-SDK [45] was chosen for the fitness evaluation and parameter optimization. Note, that the agnosticism and modularity regarding the lower-level quantum programming language remains because *(i)* the quantum program synthesis facilities are by design independent from the underlying quantum programming language, and *(ii)* the required functionalities for the fitness evaluation and parameter optimization are available also in other Q-SDKs (e.g., [46]).

[3]https://github.com/cdl-mint/automated-quantum-program-search

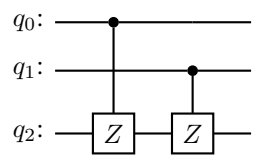


Fig. 6: Analytical solution for *Oracle* implementation comprising two controlled-Z gates

IV. Demonstration

In the following, we demonstrate how our proposed approach can be applied to find implementations for quantum programs in order to arrive at an executable form of the designed quantum circuit.

A. Use case description: Grover Search

The Grover Search algorithm [21] allows to perform searches with a quadratic speedup in the time complexity compared to classical approaches ($\mathcal{O}(N)$ to $\mathcal{O}(\sqrt{N})$). Furthermore, it is applied as a subroutine in several other quantum algorithms [47], [48]. The specific use case has been taken from [49] (cf. Fig. 3).

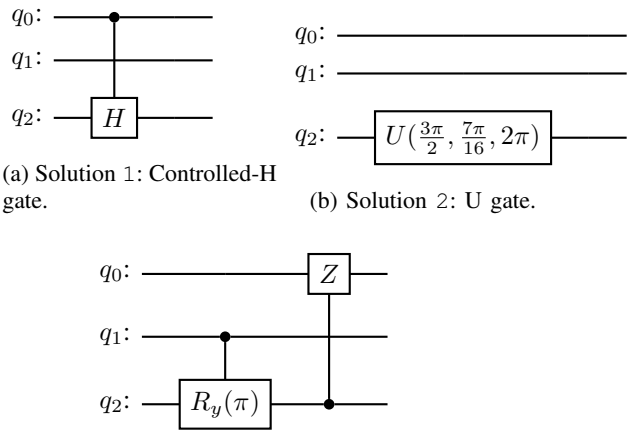
When applying the Grover Search algorithm, the Oracle operator has to be specified for each individual use case as it marks the computational basis states of interest to the user. Therefore, there is no general implementation of elementary quantum gates available. Based on our example, the states of interest are defined to be the sixth and seventh computational basis state. Thus, the Oracle is specified by its conducted quantum state transformation (up to normalization)

$$[1,1,1,1,1,1,1,1] \rightarrow [1,1,1,1,1,-1,-1,1] \qquad (1)$$

which has the analytical solution of two controlled-Z gates as the implementation of the Oracle (Fig. 6). The search algorithm should try to find this or equivalent implementations that realize the functionality of producing the target quantum state given in Eq. 1.

B. Experimental setup and user inputs

a) Configuration of MOMoT: For the validation of our approach—and considering a many-objective problem—we chose a common setting for a genetic algorithm from the adopted MDO tool, namely: The NSGA-III algorithm [50] with one-point crossover ($p = .8$) and two mutation operators to remove ($p = .1$) and/or replace ($p = .2$) a randomly selected operation from the operation sequence. Infeasible operations



(a) Solution 1: Controlled-H gate.

(b) Solution 2: U gate.

(c) Solution 3: Controlled-RY gate and controlled-Z gate.

Fig. 7: Optimal oracle solutions obtained through NSGA-III

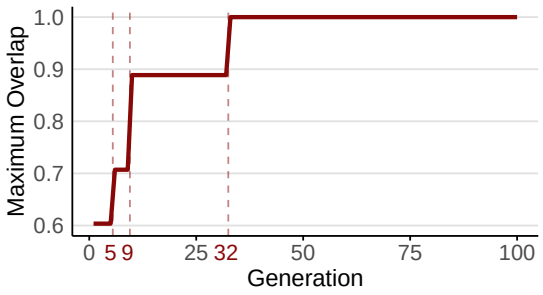


Fig. 8: Maximum overlap observed across generations

in sequences emerging from recombination become subject to a repair mechanism and are removed. The population is initialized with 25 individuals and the search terminates after 100 generations.

The quantum circuit specific user-inputs have been set as follows: 1) the target state as given in Eq. 1; 2) the position of the target state is after Layer 2; 3) the maximum length of a chromosome (i.e., gates and qubit reassignments) is 4.

b) Gate set definition: The gate set comprises the available elementary quantum gates that can realize the implementation of the Oracle. Combined with the range of addressable qubits, it defines the search space for the synthesis problem.

From all single- and two-qubit gates provided in Qiskit [45], we exclude 1) gates which can be expressed by a parameterized gate which is included in the used gate set; 2) gates that are specifically applicable to a certain type of quantum hardware (e.g., superconductor-based quantum devices).

C. Demonstration results

The previously described configuration is executed five times, and the best trade-offs in the resulting solutions in terms of accuracy and cost metrics are depicted in Fig. 7.

Of the three solutions, both 1 and 2 employ only one gate in the Oracle and show an overlap of 0.604 and 0.707, respectively. The U gate in 2 uses a single qubit only but carries three parameters whereas the non-local controlled-H gate has no parameters. Solution 3 consists of 2 consecutive non-local gates, where the controlled-RY gate carries a parameter, and induces as the only found alternative the maximum value of 1.0 for the overlap fitness. The optimum derived from analytical considerations (cf. Fig. 6) was not reproduced in these runs, but was confirmed as an obtainable solution after narrowing the search space by reducing the pool of available gates.

As the preferences of quantum circuit designers is usually directed towards quantum operators of high accuracy, we consider the feasibility of the proposed approach by examining the maximum observable overlap across generations as shown for one of the performed runs in Fig. 8. Based on the observed convergence to perfectly accurate solutions (overlap: 1.0), it is evident that the approach is capable of evolving programs that produce quantum states of decreasing deviation to the desired target state.

D. Discussion and limitations

This paper demonstrates the feasibility of using existing model-driven search approaches for automated quantum program synthesis. Therefore, our studies comprise only a minimum model that serves this purpose. For example, we implemented only two operations as to add gates and reassign qubits. Although the former covers the whole search space, the exploration phase could be improved by additional procedures or alternative implementations that specifically synergize with the crossover operator. Similarly, we propose a simple modeling language for quantum circuits. However, the language has been designed to be extensible to cover more sophisticated concepts required for quantum circuit design in general, as can be derived from the replication package. Also, in our current model, the target state input is realized by explicitly stating the complex valued vector. Methods for convenient target state definition will be integrated in the future.

We do not provide a performance evaluation of our proposed approach, which would comprise, among others, hyperparameter tuning, comparison of different search algorithms, and effects of additional transformation rules. Furthermore, we cannot provide generalizing statements, which would require the investigation of multiple quantum circuit use cases. We leave the according study of performance and scalability of our proposed approach as future work. However, as the proposed approach foresees the use of quantum simulators for direct access of the quantum state, the current scope is for small quantum circuits. Although the use of real quantum devices is possible in principle, reading an arbitrary quantum state by means of state tomography comes with exponential cost and, thus, also limits the scalability of the approach [51].

Finally, the investigated use case represents a very small quantum program example, where only a few trade-off solutions are produced. Furthermore, during the search process MOMoT would keep only one individual when the case of

equal fitness values is encountered. This further shrinks the final solution set and limits the diversity study of obtained individuals. We keep investigations on larger use cases with more evaluations and larger solution sets (e.g., as provided in [23]) as future work.

V. Related Work

MDO applications. Latest advancements concern the search methods and operators used, and frameworks that support the modeling and optimization workflow [5], [7], [52]–[56]. Application areas include production lines, modularization tasks, and refactoring [56], [57]. Furthermore, a model optimization engine was used as part of a digital twin in runtime simulations [58]. In substitution of search methods, learning-based techniques for rule transformations have been adapted and applied in several case studies [59].

MDE for QC. Modeling approaches for the design of quantum software have been suggested, where a UML [60] extension for the addition of basic quantum elements is proposed [61]. Furthermore, the use of UML-profiles [62], a conceptual model of quantum programs [12], and software modernization towards quantum software using MDE methods [13], [14] have been suggested. However, we are not aware of any approach which utilizes techniques from MDE for automated quantum program synthesis.

Evolutionary quantum circuit synthesis. Evolutionary approaches have been applied to automatically synthesize quantum programs [18], [63], [64]. To consider the trade-off between accuracy and computational cost present in the NISQ-era, multi-objective genetic programming approaches have been proposed, which are applicable to parameterized as well as non-parameterized quantum circuits [22], [23]. The approach proposed in [23] served as a basis for the fitness evaluation and parameter optimization conducted for this study. All mentioned approaches represent tailor-made solutions, whereas the approach proposed in this paper comes with the integration into MDO frameworks and no need for a direct encoding.

VI. Conclusion

This paper presented how MDO approaches, in particular MOMoT, can be applied to the problem of automated quantum program synthesis. We show how MOMoT can interact with quantum SDKs in order to provide the quantum-specific functionalities. Compared to tailor-made solutions for quantum program synthesis, this enables to use the features of existing MDO frameworks. Finally, we demonstrate the use of the overall framework for the Grover Search use case.

Future work. In the future, we plan to conduct evaluations of our proposed approach analogously as conducted previously using tailor-made solutions [23]. Second, we plan to extend the quantum circuit modeling language to capture more concepts for quantum circuit design. Partially based on such new concepts, we will also study the impact of further transformation rules for the search space exploration. Third, we will experiment with different search algorithms provided in MOMoT. Forth, we will complete the envisioned workflow depicted in Fig. 2 by defining the transformations from quantum circuits declared in specific quantum SDKs to the generic model representation.

Acknowledgement

Financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development and by the Austrian Science Fund (P 30525-N31) is gratefully acknowledged.

Data availability

All code and data is available at: https://github.com/cdl-mint/automated-quantum-program-search

References

[1] L. Burgueño, J. Cabot, S. Li, and S. Gérard, "A generic LSTM neural network architecture to infer heterogeneous model transformations," *Softw. Syst. Model.*, vol. 21, no. 1, pp. 139–156, 2022.

[2] J. A. H. López and J. S. Cuadrado, "Generating structurally realistic models with deep autoregressive networks," *IEEE Trans. Software Eng.*, vol. 49, no. 4, pp. 2661–2676, 2023.

[3] A. Barriga, L. Mandow, J. Pérez-de-la-Cruz, A. Rutle, R. Heldal, and L. Iovino, "A comparative study of reinforcement learning techniques to repair models," in *MODELS'20 Companion Proceedings*. ACM, 2020, pp. 47:1–47:9.

[4] A. Barriga, A. Rutle, and R. Heldal, "AI-powered model repair: an experience report - lessons learned, challenges, and opportunities," *Softw. Syst. Model.*, vol. 21, no. 3, pp. 1135–1157, 2022.

[5] A. Burdusel and S. Zschaler, "Towards scalable search-based model engineering with mdeoptimiser scale," in *MODELS'19 Companion Proceedings*. IEEE, 2019, pp. 189–195.

[6] H. Abdeen, D. Varró, H. A. Sahraoui, A. S. Nagy, C. Debreceni, Á. Hegedüs, and Á. Horváth, "Multi-objective optimization in rule-based design space exploration," in *ASE*. ACM, 2014, pp. 289–300.

[7] R. Bill, M. Fleck, J. Troya, T. Mayerhofer, and M. Wimmer, "A local and global tour on momot," *Softw. Syst. Model.*, vol. 18, no. 2, pp. 1017–1046, 2019.

[8] F. Krikava, "Solving the TTC'16 class responsibility assignment case study with SIGMA and multi-objective genetic algorithms," in *Proceedings of the 9th Transformation Tool Contest*, ser. CEUR Workshop Proceedings, vol. 1758. CEUR-WS.org, 2016, pp. 55–60.

[9] J. Zhao, "Quantum software engineering: Landscapes and horizons," *arXiv preprint arXiv:2007.07047*, 2020.

[10] M. De Stefano, F. Pecorelli, D. Di Nucci, F. Palomba, and A. De Lucia, "Software engineering for quantum programming: How far are we?" *Journal of Systems and Software*, vol. 190, 2022.

[11] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice, Second Edition*, ser. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2017.

[12] S. Ali and T. Yue, "Modeling quantum programs: challenges, initial results, and research directions," in *Proc. of the 1st ACM SIGSOFT Int. Workshop on Architectures and Paradigms for Engineering Quantum Soft.*, 2020, pp. 14–21.

[13] L. Jiménez-Navajas, R. Pérez-Castillo, and M. Piattini, "KDM to UML Model Transformation for Quantum Soft. Modernization," in *Int. Conf. on the Quality of Information and Communications Technology*. Springer, 2021, pp. 211–224.

[14] R. Pérez-Castillo, M. A. Serrano, and M. Piattini, "Soft. modernization to embrace quantum technology," *Advances in Engineering Soft.*, vol. 151, p. 102933, 2021.

[15] E.-J. Kuo, Y.-L. L. Fang, and S. Y.-C. Chen, "Quantum architecture search via deep reinforcement learning," *arXiv preprint arXiv:2104.07715*, 2021.

[16] K. A. McKiernan, E. Davis, M. S. Alam, and C. Rigetti, "Automated quantum programming via reinforcement learning for combinatorial optimization," *arXiv preprint arXiv:1908.08054*, 2019.

[17] K. Murakami and J. Zhao, "Autoqc: Automated synthesis of quantum circuits using neural network," *arXiv preprint arXiv:2210.02766*, 2022.

[18] A. Gepp and P. Stocks, "A review of procedures to evolve quantum algorithms," *Genetic programming and evolvable machines*, vol. 10, no. 2, pp. 181–228, 2009.

[19] L. Ding and L. Spector, "Multi-objective evolutionary architecture search for parameterized quantum circuits," *Entropy*, vol. 25, no. 1, 2023.

[20] F. Gemeinhardt, A. Garmendia, and M. Wimmer, "Towards model-driven quantum soft. engineering," in *Second Int. Workshop on Quantum Soft. Engineering (Q-SE 2021) co-located with ICSE 2021*, 2021.

[21] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," *Phys. Today*, vol. 54, no. 2, p. 60, 2001.

[22] V. Potoček, A. P. Reynolds, A. Fedrizzi, and D. W. Corne, "Multi-objective evolutionary algorithms for quantum circuit discovery," *arXiv preprint arXiv:1812.04458*, 2018.

[23] F. Gemeinhardt, S. Klikovits, and M. Wimmer, "Hybrid multi-objective genetic programming for parameterized quantum operator discovery," in *Companion Proc. Genetic and Evolutionary Computation Conference (GECCO)*, 2023.

[24] T. Arendt, E. Biermann, S. Jurack, C. Krause, and G. Taentzer, "Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations," in *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Springer, 2010, pp. 121–135.

[25] Y. Zhu and K. Yu, "Artificial intelligence (ai) for quantum and quantum for ai," *Optical and Quantum Electronics*, vol. 55, no. 8, p. 697, 2023.

[26] D. C. Schmidt, "Model-driven engineering," *Computer*, vol. 39, no. 2, p. 25, 2006.

[27] K. Czarnecki and S. Helsen, "Feature-based survey of model transformation approaches," *IBM Syst. J.*, vol. 45, no. 3, pp. 621–646, 2006.

[28] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: Eclipse Modeling Framework*. Addison Wesley, 2008.

[29] G. Chen, D. Church, B. Englert, M. Zubairy *et al.*, "Mathematical models of contemporary elementary quantum computing devices," *Quantum Control: Mathematical and Numerical Challenges*, vol. 33, pp. 79–117, 2003.

[30] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[31] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke *et al.*, "Noisy intermediate-scale quantum (NISQ) algorithms," *arXiv preprint*, 2021.

[32] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[33] W. Lavrijsen, A. Tudor, J. Müller, C. Iancu, and W. De Jong, "Classical optimizers for noisy intermediate-scale quantum devices," in *Int. Conf. on quantum computing and engineering (QCE)*. IEEE, 2020.

[34] X. Bonet-Monroig, H. Wang, D. Vermetten, B. Senjean, C. Moussa, T. Bäck, V. Dunjko, and T. E. O'Brien, "Performance comparison of optimization methods on variational quantum algorithms," *arXiv preprint arXiv:2111.13454*, 2021.

[35] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the NISQ era," *Quantum Science and Technology*, vol. 5, no. 4, p. 044007, 2020.

[36] K. Guy and G. Perdue, "Using reinforcement learning to optimize quantum circuits in the presence of noise," Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States), Tech. Rep., 2020.

[37] M. Pirhooshyaran and T. Terlaky, "Quantum circuit design search," *Quantum Machine Intelligence*, vol. 3, no. 2, pp. 1–14, 2021.

[38] E. Ye and S. Y.-C. Chen, "Quantum architecture search via continual reinforcement learning," *arXiv preprint arXiv:2112.05779*, 2021.

[39] L. Spector, H. Barnum, H. J. Bernstein, and N. Swamy, "Genetic programming for quantum computers," *Genetic Programming*, pp. 365–373, 1998.

[40] ——, "Quantum computing applications of genetic programming," *Advances in genetic programming*, vol. 3, pp. 135–160, 1999.

[41] A. Bautu and E. Bautu, "Quantum circuit design by means of genetic programming," *Romanian Physics*, vol. 52, no. 5-7, pp. 697–704, 2007.

[42] F. Gemeinhardt, A. Garmendia, M. Wimmer, and R. Wille, "A model-driven framework for composition-based quantum circuit design," https://se.jku.at/felix-gemeinhardt/?se_variable=2, 2022.

[43] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019.

[44] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[45] M. S. ANIS *et al.*, "Qiskit: An open-source framework for quantum computing," 2021.

[46] C. Developers, "Cirq," Apr 2022, see full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors.

[47] A. Gilliam, S. Woerner, and C. Gonciulea, "Grover adaptive search for constrained polynomial binary optimization," *Quantum*, vol. 5, p. 428, 2021.

[48] C. Durr and P. Hoyer, "A quantum algorithm for finding the minimum," *arXiv preprint*, 1996.

[49] C. Figgatt, D. Maslov, K. A. Landsman, N. M. Linke, S. Debnath, and C. Monroe, "Complete 3-qubit grover search on a programmable quantum computer," *Nature communications*, vol. 8, no. 1, pp. 1–9, 2017.

[50] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577–601, 2013.

[51] M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin, and Y.-K. Liu, "Efficient quantum state tomography," *Nature communications*, vol. 1, no. 1, p. 149, 2010.

[52] A. Burdusel, S. Zschaler, and D. Strüber, "MDEoptimiser: a search based model engineering tool," in *Companion Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS)*. ACM, 2018, pp. 12–16.

[53] A. Burdusel, S. Zschaler, and S. John, "Automatic Generation of Atomic Consistency Preserving Search Operators for Search-Based Model Engineering," in *Proceedings of the 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2019, pp. 106–116.

[54] S. John, A. Burdusel, R. Bill, D. Strüber, G. Taentzer, S. Zschaler, and M. Wimmer, "Searching for Optimal Models: Comparing Two Encoding Approaches," *J. Object Technol.*, vol. 18, no. 3, pp. 6:1–22, 2019.

[55] J. Martinez, D. Strüber, J. M. Horcas, A. Burdusel, and S. Zschaler, "Acapulco: an extensible tool for identifying optimal and consistent feature model configurations," in *26th ACM International Systems and Software Product Line Conference (SPLC)*. ACM, 2022, pp. 50–53.

[56] J. M. Horcas, D. Strüber, A. Burdusel, J. Martinez, and S. Zschaler, "We're not gonna break it! consistency-preserving operators for efficient product line configuration," *IEEE Trans. Software Eng.*, vol. 49, no. 3, pp. 1102–1117, 2023.

[57] M. Fleck, J. Troya, and M. Wimmer, "Towards generic modularization transformations," in *Companion Proceedings of the 15th International Conference on Modularity*. ACM, 2016, pp. 190–195.

[58] M. Eisenberg, D. Lehner, R. Sindelár, and M. Wimmer, "Towards reactive planning with digital twins and model-driven optimization," in *ISoLA*, ser. LNCS, vol. 13704. Springer, 2022, pp. 54–70.

[59] M. Eisenberg, H. Pichler, A. Garmendia, and M. Wimmer, "Towards reinforcement learning for in-place model transformations," in *24th International Conference on Model Driven Engineering Languages and Systems*. IEEE, 2021, pp. 82–88.

[60] OMG, "UML," https://www.omg.org/spec/UML/, 2017.

[61] C. A. Pérez-Delgado and H. G. Perez-Gonzalez, "Towards a quantum soft. modeling language," in *Proc. of the IEEE/ACM 42nd Int. Conf. on Soft. Eng. Workshops*, 2020, pp. 442–444.

[62] R. Pérez-Castillo, L. Jiménez-Navajas, and M. Piattini, "Modelling quantum circuits with UML," in *2nd IEEE/ACM International Workshop on Quantum Software Engineering, Q-SE@ICSE 2021, Madrid, Spain, June 1-2, 2021*. IEEE, 2021, pp. 7–12.

[63] U. Ahsan *et al.*, "AutoQP: Genetic Programming for Quantum Programming," in *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. IEEE, 2020, pp. 378–382.

[64] K. M. Barnes and M. B. Gale, "Meta-genetic programming for static quantum circuits," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 2016–2019.