Achieving Pareto-Optimality in Quantum Circuit Compilation via a Multi-Objective Heuristic Optimization Approach

Aleksandra Świerkowska*†, Jorge Echavarria*, Laura Schulz*, Martin Schulz†,
*Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities, Garching, Bavaria, Germany
{aleksandra.swierkowska, jorge.echavarria, laura.schulz}@lrz.de
†Technical University of Munich, Garching, Bavaria, Germany
{martin.w.j.schulz}@tum.de

***Abstract*—High Performance Computing-Quantum Computing (HPCQC) integration presents a promising yet challenging opportunity, particularly in the area of quantum circuit compilation and optimization, requiring further advancements in the field of Quantum Computing (QC). To address this, we introduce the Munich Quantum Compiler, a key component of the Munich Quantum Software Stack (MQSS). This compiler employs a heuristic-based approach to select a Pareto-optimal subset of optimizations in the form of LLVM passes for quantum circuits described in an LLVM-compliant Intermediate Representation (IR).**

***Index Terms*—Quantum Computing, Multi-objective Optimization, Quantum Compilation, LLVM, QIR, MOEA, Genetic Algorithm, NSGA-II**

I. Introduction

In recent years, Quantum Computing (QC) has demonstrated significant potential for achieving exponential performance improvements over classical algorithms for certain classes of computational problems [1]–[5]. In order to enable wider growth and development of the quantum potential, efforts towards High Performance Computing-Quantum Computing (HPCQC) integration have been initiated [6], [7]. Reaching their goal of providing seamless cooperation between classical and quantum parts of the system would allow to reach a new territory of research, mainly in the form of hybrid algorithms.

However, to be able to achieve a hybrid software stack, it is necessary to design highly sophisticated compilers capable of both: 1) providing effective optimizations for quantum circuits, such as reducing their size to allow execution before significant coherence degradation, and 2) adapting the quantum circuits to the unique capabilities and limitations of the available quantum accelerators. Furthermore, a quantum compiler should be able to support the common software stack for classical and quantum applications. For that reason, we decided to utilize Quantum Intermediate Representation (QIR) [8], an LLVM-compliant Intermediate Representation (IR) supporting interleaving quantum and classical instructions within a single program [9], as a mid-level quantum circuit representation in the Munich Quantum Compiler, a key component of the Munich Quantum Software Stack (MQSS) [10]–[12]. This approach allowed us to apply optimizations and other required transformations to the circuit through the application of custom LLVM passes. Nevertheless, employing compilation schemes akin to the classical world, such as iteratively applying LLVM optimization passes, introduces classical challenges into the quantum domain, as we will elaborate.

The challenges associated with finding Pareto-optimal optimization subsets [13] and the sequence in which to apply them [14], also known as phase ordering, have been extensively studied in the classical compilation field. These are well-known NP-Hard problems [15]. The proposed solutions range from utilizing Genetic Algorithms (GAs) [16]–[18], for years deemed as state-of-the-art, through more modern approaches based on, for example, Machine Learning (ML) [19] or Reinforcement Learning (RL) [15], [20], [21].

Similarly, ML and RL are usually utilized regarding quantum compilation [22]–[25], mainly due to their efficient execution times. However, most of the proposed approaches in the literature focus on optimizing a singular objective, usually a complex figure of merit consisting of a weighted sum of multiple different objectives, such as depth or number of gates. Although the importance of some quantum metrics is well-established, the development of effective quantum performance metrics remains an active area of research [26]. To provide a change of metrics in a figure of merit in model-based compilers, each time a tedious and time-consuming model retraining is necessary.

The novelty of the quantum optimization approach proposed in the Munich Quantum Compiler lies in providing a multi-objective optimization through the utilization of a GA, more specifically, a non-dominated sorting-based Multi-Objective Evolutionary Algorithm (MOEA) called Non-dominated Sorting Genetic Algorithm II (NSGA-II). This approach yields a set of solution candidates belonging to the Pareto frontier, none of which is fully dominated by any other solution found. While GAs have proven to be highly effective in the classical domain, to the best of our knowledge, they have not yet been applied to the quantum version of this problem.

II. Methods

Optimization is a crucial component of mostly every compiler, making the development of effective pass selection

979-8-3315-4137-8/24/$31.00 ©2024 IEEE
DOI 10.1109/QCE60285.2024.10297

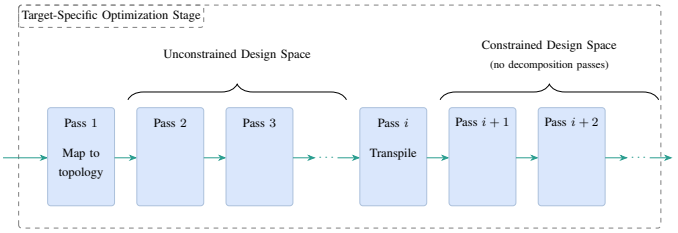


Fig. 1: Target-Specific Optimization Stage of the Munich Quantum Compiler.

strategies highly desirable. Although the number of strategies implemented over the years in various quantum programming frameworks, such as, for example, Qiskit [27], is constantly growing, the Munich Quantum Compiler distinguishes itself by utilizing a heuristic implemented in C++ to prioritize performance and efficiency. Furthermore, unlike tools like QIR Adaptor Tool (QAT) [28], which also apply a series of LLVM passes to QIR for optimizations, our compiler aims for seamless integration within HPCQC ecosystems.

The Munich Quantum Compiler aims to find a Pareto-optimal combination of LLVM passes to optimize multiple metrics simultaneously, such as the number of gates in the circuit or its depth. Given the varying characteristics of individual circuits and changing constraints throughout the pipeline execution, the effectiveness of pass subsets can differ. To address this problem, the Munich Quantum Compiler provides a subset tailored for each submitted quantum circuit by conducting a two-fold Design Space Exploration (DSE) to derive a Pareto-optimal solution. The two key stages of this process are the *Target-Agnostic Optimization Stage* and the *Target-Specific Optimization Stage*:

Target-Agnostic Optimization Stage: This stage is inherently simpler than the Target-Specific Optimization Stage, as it does not conduct an exhaustive DSE. Instead, it allows for a fully unconstrained exploration of the design space consisting of all the available target-agnostic optimization passes. These passes do not require any knowledge about the quantum accelerator intended for executing the optimized quantum circuit. Some examples include passes merging rotations, passes replacing specific sequences of gates, or passes removing pairs of inverse gates (UU^{-1}) as, for instance, two Pauli-X gates acting on the same qubit:

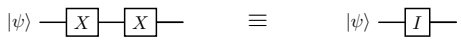


Fig. 2: Applying two consecutive Pauli-X gates to a qubit is equivalent to applying an identity operation [29].

The primary goal of this stage is to optimize the quantum circuit to reduce its execution time in the subsequent, more complex stages of the MQSS.

Target-Specific Optimization Stage: This stage is significantly more intricate than the Target-Agnostic Optimization Stage, as it needs to not only provide optimizations but also other types of transformations, enabling the quantum circuit to be executed on the target quantum accelerator. The configuration of passes we propose for this stage is presented in Fig. 1. This stage begins with the mapping of the quantum circuit qubits to the topology of the target accelerator. It is achieved by applying a pass utilizing, for example, the Munich Quantum Toolkit (MQT) QMAP [30] mapper, a tool capable of mapping the provided quantum circuit to the topology of the target architecture. Due to the expected potential overhead introduced by the mapper, this pass is followed by another DSE of optimization passes, including target-specific ones. Afterward, the quantum circuit is transpiled to the native gate set of the target quantum accelerator. Note that the above modifications should be sufficient to submit the circuit to the target quantum accelerator. However, we provide additional optimizations due to the critical importance of maximizing the circuit's quality for successful execution. To protect the transpiled circuit from introducing unsupported gates, these optimizations are constrained to exclude decomposition passes.

Fig. 3 presents a simplified example of the process occurring during the Target-Specific Optimization Stage. As shown, the additional application of optimizations following mapping and removal of unsupported gates allows us to maximally reduce gate overhead without compromising the mapping or gate set necessary for the successful execution of the circuit on the selected quantum device. Furthermore, utilizing a heuristic for DSE allows the application of a different decomposition, in this case, to each SWAP gate, depending on what is on each side for the maximal performance enhancement.

A. Multi-Objective Evolutionary Algorithm (MOEA)

To achieve high flexibility in objective selection, we opted to use the MOEA known as NSGA-II. The parameters and operators for NSGA-II detailed in the following were empirically chosen to best suit the design space.

a) Encoding: Binary encoding proved ineffective, as it can only encode whether a given pass should be applied, without specifying the sequence or the number of applications to the quantum circuit under optimization. In contrast, integer encoding allowed for both the removal and repetition of passes.

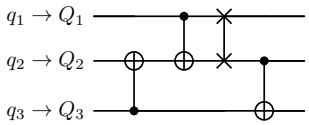
b) Operators: We decided on using a swap mutation [31] and two-point crossover [32] operators as, compared to other operators we tested, they demonstrated better effectiveness in improving the quality of the solutions.

c) Chromosome size: The most promising results were observed with a chromosome size three times the number of available passes.

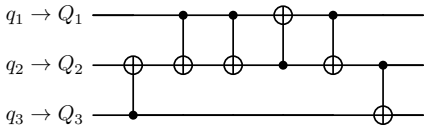
B. Design Space

To introduce transformations to quantum circuits described in QIR, a collection of custom LLVM passes was created. The passes consist of both target-specific and target-agnostic transformations, which can be further divided into *commutation*, *decomposition*, *reduction*, and *structural* types. Each pass introduces a specific modification, such as the inverse gates removal shown in Fig. 2, contributing to enhancing the

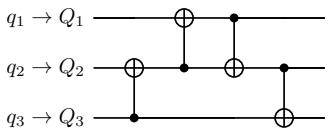
(a) The exemplary quantum circuit and the linear topology to which it needs to be mapped.



(b) Mapping the quantum circuit to the given topology results in the lack of a coupling mapping between q_1 and q_3, because of which it is necessary to insert an additional SWAP gate in order to execute it on the selected quantum accelerator.



(c) Assuming that SWAP gates are not part of the native gate set of the target device, a target-specific optimization would apply a decomposition.



(d) Target-agnostic optimization would then remove the consecutive CNOT gates analogically as in Fig. 2 [29], bringing us to the optimized quantum circuit that can be executed on the target accelerator with minimized overhead.

Fig. 3: Circuit modifications occurring during the Target-Specific Optimization Stage for an exemplary fragment of a quantum circuit mapped to a linear topology $Q_1 - Q_2 - Q_3$ of a quantum accelerator.

quantum circuit's performance. To provide a possibly comprehensive optimization coverage, passes can overlap in the sets of gates they act on or operate counterproductively, with their actions negating each other. These mutually exclusive passes necessitate addressing the NP-Hard subset selection problem.

III. Empirical Findings

A. Experimental Setup

a) Design Space: The design space for our experimentation consisted of 36 passes selected from those currently available for the Munich Quantum Compiler. For an optimization pass to be considered during the DSE, it had to meet the following criteria: 1) the pass must be capable of functioning autonomously, 2) it must be able to occupy any position in the optimization pipeline, and 3) there must be no sequence of passes where the application of this pass compromises the integrity of the quantum circuit.

b) Benchmark Suite: To evaluate the effectiveness of our approach, we used a benchmark suite of 175 quantum circuits with up to 20 qubits, sourced from [33] and parsed into QIR.

c) Objectives: The performance of the Munich Quantum Compiler was evaluated based on multiple objectives: 1) the number of gates, 2) circuit depth, 3) entanglement ratio, defined as the ratio of two-qubit interactions to the number of all gate operations, 4) critical depth, defined as the ratio of two-qubit interactions on the longest path that sets the circuit depth to the number of all two-qubit interactions in the circuit, and 5) parallelism, defined as:

$$\left(\frac{n_g}{d} - 1\right) \frac{1}{n-1} \tag{1}$$

where n is the number of qubits, n_g the number of gates, and d is the circuit depth [26]. All of these objectives were minimized.

As research into metrics for quantifying quantum circuits evolves, the objectives may change. Our approach, however, can adapt to such changes instantly, without the need for retraining, unlike, for example, ML-based solutions. Moreover, in the future, we plan to extend the set of objectives to include circuit fidelity and *liveness* [26].

B. Results

Table I presents the results of each objective averaged over the benchmark suite. The number of gates and circuit depth are expressed as ratios relative to the corresponding measurements of the non-optimized circuits.

In a setup with all of the passes applied, the specific order was determined through a series of trials, during which we made strategic arrangements aiming to improve the performance. However, as presented in ("All passes" in Table I), this approach led to an increase in both the number of gates and depth compared to the original circuits. This increase is due to the introduction of unnecessary decompositions, which negatively affect the overall circuit quality.

In another approach, only a subset of these passes is selected ("Handpicked" in Table I). In this setup, the subset and the order of passes were again determined through iterative tries and based on prior knowledge, with each pass applied at most once. Although this approach mitigates the issues seen in the "All passes" setup, the improvements over the non-optimized circuit are negligible.

In order to compare the effectiveness of the MOEA-based approaches, from each set of solution candidates, a non-dominated candidate resulting in the lowest number of gates was chosen, as the number of gates is included in the calculations of most of the other objectives.

As one can observe, the application of NSGA-II with binary encoding, which involves a fixed order of a subset of passes, did not improve the average result, as shown in the row "NSGA-II bin.". Since the order of passes was fixed, the

TABLE I: The averages of the results obtained for each type of experimental setup. The numbers in bold highlight the lowest value achieved for each objective.

Setup	Gates ratio	Depth ratio	Ent. ratio	Critical depth	Parallelism
Non-optimized	1	1	0.5821	0.2083	0.5419
All passes	2.9805	2.4356	0.2147	0.1972	0.6074
Handpicked	0.9195	0.9697	0.5011	0.3535	0.5524
NSGA-II bin.	0.9195	0.9697	0.5011	0.3535	0.5524
NSGA-II int.	**0.055**	**0.1378**	**0.0038**	**0.0114**	**0.3368**

TABLE II: Comparison of average execution times of different NSGA-II setups for a single quantum circuit.

Setup	Avg. execution time [s]
NSGA-II bin.	4.514
NSGA-II int.	9.903

results quickly converged to those of the "Handpicked" setup. However, using integer encoding to determine the order of optimization passes and allowing for repetitions led to a notable performance enhancement even in the simplest setup with a population size of 8 individuals evolved over 8 generations. Utilizing a more sophisticated setup with a population size of 128 individuals evolved over 128 generations resulted in a significant improvement shown in the row "NSGA-II int." in Table I.

Table II presents average execution times comparison for a single quantum circuit of NSGA-II setups utilizing different encodings. The "NSGA-II bin." row presents the result for a setup with binary encoding with a population size of 32 individuals evolved over 32 generations. Analogically, the "NSGA-II int." row shows the execution time of a setup with integer encoding ran with the same parameters.

IV. Conclusions

We propose the first multi-objective solution for quantum circuit optimization which employs NSGA-II. Using an optimized setup with integer encoding, allowing for repetitions of optimization passes, significantly improves the average fitness of the candidate solutions in a benchmarking suite compared to manually chosen optimization sets, with the number of gates improved by 94%, depth by 85.8%, entanglement ratio by 99.2%, critical depth by 96.8%, and parallelism by 39%. The compiler is easily adaptable to different requirements, allowing for the introduction of new objectives without the need to modify the optimization algorithm.

A. Future Work

The main downside of using MOEAs is the long execution time. As the future work on the Munich Quantum Compiler, we plan to explore various approaches with the potential of mitigating this issue:

a) Predictor: To prevent unnecessary execution time on converged candidate solutions, we propose training a *predictor model*, similar to the one introduced in [34], using the highly optimized dataset generated by NSGA-II. This model can then predict the optimized objective values for a circuit, and reaching these values can serve as an additional stopping condition in the heuristic.

b) Distributed Genetic Algorithm: One way to increase efficiency would be to parallelize appropriate parts of the code, such as the application of passes to solution candidates and fitness evaluation. However, only certain parts of the code can be parallelized, as candidate selections must still be performed sequentially to ensure individuals are chosen from the entire undivided pool.

c) Machine Learning: On the one hand, a MOEA-based approach can generate Pareto-optimal candidate solutions but likely requiring higher execution times than, for example, ML-based techniques. On the other hand, ML-based approaches can yield results significantly quicker than MOEAs, but they tend to demand a substantial dataset of already highly optimized circuits. Access to the diverse training datasets generated from MOEA solutions can significantly improve the efficiency of ML-based methodologies, leading to superior results. That is, the results generated by the NSGA-II setup applied across a comprehensive set of quantum circuits could be utilized as a training dataset for a ML-based solutions. Successful application of ML models to the phase-ordering problem is well-documented in the literature [22], and it could potentially enhance the time-efficiency of the Munich Quantum Compiler.

References

[1] H. Liu, G. H. Low, D. S. Steiger, T. Häner, M. Reiher, and M. Troyer, "Prospects of quantum computing for molecular sciences," *Materials Theory*, vol. 6, no. 1, p. 11, 2022. [Online]. Available: https://doi.org/10.1186/s41313-021-00039-z

[2] A. Ajagekar and F. You, "New frontiers of quantum computing in chemical engineering," *Korean Journal of Chemical Engineering*, vol. 39, no. 4, pp. 811–820, 2022. [Online]. Available: https://doi.org/10.1007/s11814-021-1027-6

[3] P. S. Emani, J. Warrell, A. Anticevic, S. Bekiranov, M. Gandal, M. J. McConnell, G. Sapiro, A. Aspuru-Guzik, J. T. Baker, M. Bastiani, J. D. Murray, S. N. Sotiropoulos, J. Taylor, G. Senthil, T. Lehner, M. B. Gerstein, and A. W. Harrow, "Quantum computing at the frontiers of biological sciences," *Nature Methods*, vol. 18, no. 7, pp. 701–709, 2021. [Online]. Available: https://doi.org/10.1038/s41592-020-01004-3

[4] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," *IBM Journal of Research and Development*, vol. 62, no. 6, pp. 6:1–6:20, 2018.

[5] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, "Quantum computing for finance: State-of-the-art and future prospects," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–24, 2020.

[6] M. Schulz, M. Ruefenacht, D. Kranzlmüller, and L. B. Schulz, "Accelerating HPC With Quantum Computing: It Is a Software Challenge Too," *Comput. Sci. Eng.*, vol. 24, no. 4, pp. 60–64, 2022. [Online]. Available: https://doi.org/10.1109/MCSE.2022.3221845

[7] A. Elsharkawy, X. M. To, P. Seitz, Y. Chen, Y. Stade, M. Geiger, Q. Huang, X. Guo, M. A. Ansari, C. B. Mendl, D. Kranzlmüller, and M. Schulz, "Integration of Quantum Accelerators with High Performance Computing - A Review of Quantum Programming Tools," *CoRR*, vol. abs/2309.06167, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2309.06167

[8] QIR Alliance. (2021) QIR Specification. Also see https://qir-alliance.org. [Online]. Available: https://github.com/qir-alliance/qir-spec

[9] T. Lubinski, C. Granade, A. Anderson, A. Geller, M. Roetteler, A. Petrenko, and B. Heim, "Advancing Hybrid Quantum-Classical Computation with Real-Time Execution," Jun. 2022, arXiv:2206.12950 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2206.12950

[10] Munich Quantum Software Stack. (2024) Munich Quantum Software Stack. [Online]. Available: https://github.com/Munich-Quantum-Software-Stack

[11] M. Schulz, H. Ahmed, X. Deng, J. Echavarria, M. Gammelmark, S. Karlsson, E. Kaya, M. Reznak, L. B. Schulz, and M. Tovey, "From the physics lab to the computer lab: Towards flexible and comprehensive devops for quantum computing," in *Proceedings of the 21st ACM International Conference on Computing Frontiers: Workshops and Special Sessions*, ser. CF '24 Companion. New York, NY, USA: Association for Computing Machinery, 2024, p. 139–143. [Online]. Available: https://doi.org/10.1145/3637543.3653432

[12] M. Schulz, L. Schulz, M. Ruefenacht, and R. Wille, "Towards the munich quantum software stack: Enabling efficient access and tool support for quantum computers," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 02, 2023, pp. 399–400.

[13] P. A. Kulkarni, "Fast and Effective Solutions to the Phase Ordering Problem," 2007.

[14] S.-A.-A. Touati and D. Barthou, "On the decidability of phase ordering problem in optimizing compilation," in *Proceedings of the 3rd conference on Computing frontiers*. Ischia Italy: ACM, May 2006, pp. 147–156. [Online]. Available: https://dl.acm.org/doi/10.1145/1128022.1128042

[15] Q. Huang, A. Haj-Ali, W. Moses, J. Xiang, I. Stoica, K. Asanovic, and J. Wawrzynek, "AutoPhase: Juggling HLS Phase Orderings in Random Forests with Deep Reinforcement Learning," Mar. 2020, arXiv:2003.00671 [cs]. [Online]. Available: http://arxiv.org/abs/2003.00671

[16] K. D. Cooper, P. J. Schielke, and D. Subramanian, "Optimizing for reduced code space using genetic algorithms," *ACM SIGPLAN Notices*, vol. 34, no. 7, pp. 1–9, May 1999. [Online]. Available: https://dl.acm.org/doi/10.1145/315253.314414

[17] P. Kulkarni, W. Zhao, H. Moon, K. Cho, D. Whalley, J. Davidson, M. Bailey, Y. Paek, and K. Gallivan, "Finding effective optimization phase sequences," *SIGPLAN Not.*, vol. 38, no. 7, p. 12–23, jun 2003. [Online]. Available: https://doi.org/10.1145/780731.780735

[18] L. Almagor, K. D. Cooper, A. Grosul, T. J. Harvey, S. W. Reeves, D. Subramanian, L. Torczon, and T. Waterman, "Finding effective compilation sequences," *ACM SIGPLAN Notices*, vol. 39, no. 7, pp. 231–239, Jun. 2004. [Online]. Available: https://dl.acm.org/doi/10.1145/998300.997196

[19] S. Kulkarni, "Improving compiler optimizations using Machine Learning," 2014.

[20] M. Trofin, Y. Qian, E. Brevdo, Z. Lin, K. Choromanski, and D. Li, "MLGO: a Machine Learning Guided Compiler Optimizations Framework," Jan. 2021, arXiv:2101.04808 [cs]. [Online]. Available: http://arxiv.org/abs/2101.04808

[21] M. Almakki, A. Izzeldin, Q. Huang, A. H. Ali, and C. Cummins, "Autophase V2: Towards Function Level Phase Ordering Optimization," May 2022.

[22] N. Quetschlich, L. Burgholzer, and R. Wille, "Predicting Good Quantum Circuit Compilation Options," May 2023, arXiv:2210.08027 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2210.08027

[23] N. Quetschlich, L. Burgholzer, and R. Wille, "Compiler Optimization for Quantum Computing Using Reinforcement Learning," Apr. 2023, arXiv:2212.04508 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2212.04508

[24] T. Fösel, M. Y. Niu, F. Marquardt, and L. Li, "Quantum circuit optimization with deep reinforcement learning," 2021.

[25] D. Kremer, V. Villar, H. Paik, I. Duran, I. Faro, and J. Cruz-Benito, "Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning," 2024.

[26] T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai, X.-C. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong, "SupermarQ: A Scalable Quantum Benchmark Suite," 2022.

[27] IBM Quantum. (2024) Transpiler Passes. [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/transpiler_passes

[28] QIR Alliance. (2024) QIR Adaptor Tool. [Online]. Available: https://www.qir-alliance.org/qat/

[29] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[30] R. Wille and L. Burgholzer, "MQT QMAP: Efficient Quantum Circuit Mapping," in *Proceedings of the 2023 International Symposium on Physical Design*, ser. ISPD '23. ACM, Mar. 2023. [Online]. Available: http://dx.doi.org/10.1145/3569052.3578928

[31] A. E. Eiben and J. E. Smith, "Genetic Algorithms," in *Introduction to Evolutionary Computing*, A. E. Eiben and J. E. Smith, Eds. Berlin, Heidelberg: Springer, 2003, pp. 37–69. [Online]. Available: https://doi.org/10.1007/978-3-662-05094-1_3

[32] A. C. Nearchou, "The effect of various operators on the genetic search for large scheduling problems," *International Journal of Production Economics*, vol. 88, no. 2, pp. 191–203, 2004, mathematical Methods and their Applications to Inventory Control and Related Topics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925527303001841

[33] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Bench: Benchmarking software and design automation tools for quantum computing," *Quantum*, 2023, MQT Bench is available at https://www.cda.cit.tum.de/mqtbench/.

[34] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Predictor: Automatic Device Selection with Device-Specific Circuit Compilation for Quantum Computing," 2023.