Towards
Application-Aware Quantum Circuit Compilation

Nils Quetschlich*   Florian J. Kiwit†‡   Maximilian A. Wolf†‡   Carlos A. Riofrío†
Lukas Burgholzer*   Andre Luckow†‡   Robert Wille*§
*Chair for Design Automation, Technical University of Munich, Germany
†BMW Group, Munich, Germany
‡Ludwig Maximilian University, Munich, Germany
§Software Competence Center Hagenberg GmbH (SCCH), Austria
{nils.quetschlich, lukas.burgholzer, robert.wille}@tum.de
{florian.kiwit, maximilian.ww.wolf}@bmw.de   {carlos.riofrio, andre.luckow}@bmwgroup.com
https://www.cda.cit.tum.de/research/quantum

Abstract—**Quantum computing has made tremendous improvements in both software and hardware that have sparked interest in academia and industry to realize quantum computing applications. To this end, several steps are necessary: The underlying problem must be encoded in a quantum circuit, a suitable device must be selected to execute it, and it must be *compiled* accordingly. This compilation step has a significant influence on the quality of the resulting solution. However, current state-of-the-art compilation tools treat the quantum circuit as a sequence of instructions without considering the actual application it realizes—wasting a yet untapped potential to increase the solution quality. In this work, a different approach is explored that explicitly incorporates the application considered and aims to optimize its solution quality during compilation. Initial results show the benefits of this approach: For an industry-inspired application of a quantum generative model, the proposed approach outperformed Qiskit's most-optimized compilation scheme and led to better solution quality. Therefore, this work presents a first step towards *application-aware* compilation.**

I. INTRODUCTION

Quantum computing has made tremendous progress in both hardware and software development in recent years. This has sparked interest not only in academia but also in industry with an increasing number of quantum computing applications being explored [1]. Realizing such applications using quantum computing has gained significant attention in various domains such as finance [2], logistics [3], and chemistry [4]. Usually, these approaches follow a similar *workflow* consisting of multiple steps (such as, e.g., described in [5]).

Starting with the problem itself, a respective quantum algorithm generally capable of solving it must be selected together with either a quantum simulator or an actual device. Then, the problem must be *encoded* in a quantum circuit based on the selected algorithm. This circuit must then be *compiled* into an executable quantum circuit for the chosen device (if a noise-free simulator is selected, this step can be skipped). The compiled quantum circuit can be *executed* afterwards—resulting in a histogram of measurement results that must be *decoded* to extract the actual solution. Workflows like this help end users realize their applications and can also be extended to cover further aspects, e.g., resource estimation [6].

In the following, the focus is on the compilation step, which greatly affects the quality of the solution of the considered application and, therefore, can make the difference between a successful execution and obtaining completely random results. To achieve the best possible solution, current compilers usually offer different levels of optimization as a trade-off between the compilation time and *quality* of the compiled circuit. The quality of a circuit is typically defined based on certain *figures of merit* that act as a *proxy* for the true solution quality of the considered application. However, currently established figures of merit do not consider the actual application realized by the quantum circuit being compiled in their evaluation of the circuit's quality.

In this work, a different approach to compilation is explored. Instead of compiling for a figure of merit that acts as a proxy for the resulting solution quality of a considered application, the solution quality *itself* is used to guide the compilation. For that, two steps are required: Firstly, an *application-aware* figure of merit must be defined that, obviously, must depend on the application considered. It relates the compiled circuit to the quality of the solution of the application it realizes. Second, a compilation environment is necessary that allows one to consider this customizable figure of merit.

To explore the benefits of the proposed approach, it has been evaluated for an industry-inspired application of a *quantum generative model*. These generative learning applications are experiencing rapid growth and attracting widespread interest with applications extending from anomaly detection to text and image generation, as well as speech and video synthesis. However, generative learning still faces practical challenges, such as the need for large datasets, extensive hyperparameter tuning, long convergence times, and high computational resource requirements. *Quantum generative modeling* might offer an opportunity to overcome these challenges and improve its capabilities by providing faster training, increased expressive power, improved inference, and reduced energy footprint [7]–[9]. Furthermore, quantum systems are inherently probabilistic, which makes them a natural fit for modeling probability distributions.
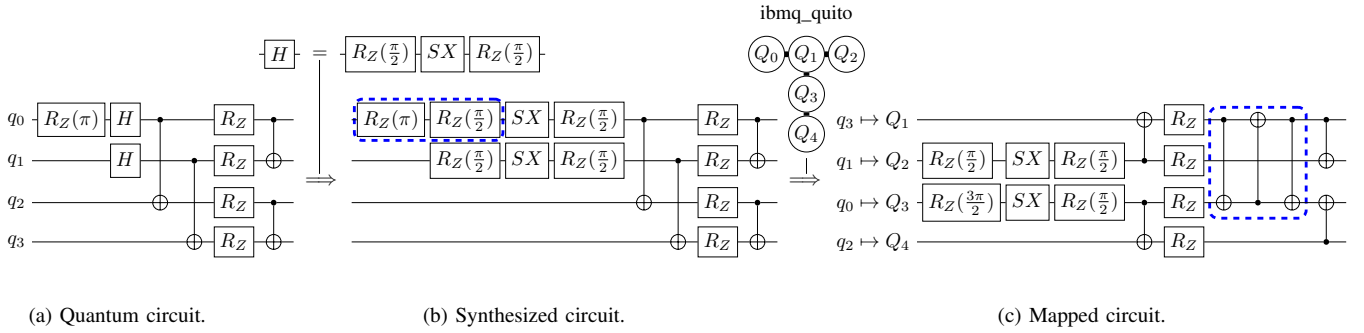
Fig. 1: Quantum circuit compilation flow.

(a) Quantum circuit. (b) Synthesized circuit. (c) Mapped circuit.

To realize the proposed compilation scheme, an application-aware figure of merit is derived and used during compilation based on the MQT Predictor framework [10]—an open-source compilation framework that utilizes artificial intelligence to determine how to best compile a given quantum circuit for a customizable figure of merit. Experimental evaluations for all considered application instances show that the proposed approach results in a higher solution quality compared to the state-of-the-art compiler Qiskit [11] using its most-optimized compilation scheme. Therefore, this work constitutes a first step towards *application-aware* compilation.

The remainder of this work is structured as follows: In Section II, the necessary background on quantum circuit compilation is provided along with a discussion of related work. Then, Section III motivates the proposed approach, which is described in more detail based on an exemplary application in Section IV and evaluated in Section V. Lastly, the proposed approach is discussed in Section VI and concluded in Section VII.

II. BACKGROUND

This section describes how a given quantum circuit encoding any kind of application is compiled to become executable on a chosen device and reviews the related work for quantum circuit compilation.

A. Quantum Circuit Compilation

Starting with the quantum circuit that could contain *any* quantum gate with no restrictions on the type and number of qubits used, an equivalent circuit shall be derived by *compiling* it such that it adheres to the constraints induced by the chosen device. To this end, different *compilation passes* are used, namely *synthesis*, *optimization*, and *mapping* as visualized in Fig. 1.

Example 1. *Assume that the application considered is encoded as the quantum circuit shown in Fig. 1a. This circuit shall now be compiled so that it becomes executable on the ibmq_quito device with five qubits whose qubit connectivity is visualized above the arrow between Fig. 1b and Fig. 1c.*

Synthesis passes are run to ensure that all the quantum gates present in the circuit are translated into executable gate types—constituting the so-called *native gateset* of a device.

Example 2. *For the chosen ibmq_quito device, the native gateset comprises the gates: $\{R_Z, SX, CX, X, ID\}$. However, the H gates present in the circuit shown in Fig. 1a are not included and therefore must be synthesized to a sequence of native gates—in this case as a sequence of RZ, SX, RZ gates shown in Fig. 1b.*

Optimization passes are typically run before and after synthesis passes to enable efficient synthesis and reduce the overhead introduced by translating to native gates.

Example 3. *The synthesized quantum circuit from Fig. 1b can be optimized by merging the subsequent R_Z gates (marked in dashed blue) into one gate by adding their angles.*

Lastly, *mapping* passes are run to assign each *logical* qubit of the quantum circuit to a *physical* qubit of the chosen device. Often it is not possible to find a *layout* such that the entire circuit adheres immediately to the device's connectivity. Typically, this is solved by inserting $SWAP$ gates into the circuit to dynamically permute the qubit arrangement on the device—a procedure called *routing* that can lead to significant overhead in terms of additional quantum gates. To counteract this potential increase in the number of gates, optimization passes are run again on the resulting circuit to reduce its size. This finally results in a fully executable quantum circuit for the chosen device that represents the same functionality as the initial circuit, but usually has a very different structure.

Example 4. *For the circuit shown in Fig. 1b, there is no layout that immediately satisfies the connectivity of the ibmq_quito device which is shown above the arrow between Fig. 1b and Fig. 1c. Therefore, one $SWAP$ gate is added, which has to be synthesized into native gates again—resulting in the now fully executable quantum circuit shown in Fig. 1c with the synthesized $SWAP$ gate highlighted in dashed blue.*

B. Related Work

Since quantum circuit compilation is a very active field of research, various compilation methods have been proposed for synthesis [12]–[18], optimization [18]–[23], and mapping [24]–[36]. Over the years, these methods have been adopted and incorporated by many quantum *Software Development Kits* (SDKs) to create automated, powerful, and easily accessible compilation flows. Prominent representatives from

industry of such SDKs are IBM's Qiskit [11], Quantinuum's TKET [37], and Xanadu's PennyLane [38]. Furthermore, various academic compilation software tools have matured to become more and more comprehensive, such as the Berkeley Quantum Synthesis Toolkit [39], and the Munich Quantum Toolkit (MQT) [40] (e.g., MQT QMAP [36]).

All of the mentioned SDKs are capable of compiling a given quantum circuit for a chosen device and usually offer pre-configured sequences of compilation passes with an adjustable degree of optimization—constituting a trade-off between the time to create the compiled circuit and its quality. As a rule of thumb, the highest degree is used to derive the best possible compiled quantum circuit—however, usually no clear optimization criterion is given and it remains vague what the SDK is actually optimizing for. Additionally, these SDKs try to address the needs of many and therefore follow a "one-fits-all" approach—aiming for a good *average* performance across diverse quantum circuits and applications realized by them. However, when considered a *specific* application, this can lead to an untapped potential for the quality of its solution.

III. MOTIVATION

Due to the variety of different available compilers and proposed compilation schemes, the respectively compiled quantum circuits also significantly differ, e.g., in terms of the number of gates, the mapping of logical to physical qubits, and the induced overhead by adhering to a device's restricted connectivity. Although all compiled quantum circuits represent the same functionality and should lead to the same solutions to the application considered, this is not the case in reality due to the imperfection of the currently available *Noisy Intermediate Scale Quantum* (NISQ) computers [41].

This raises the question of how to evaluate the quality of a compilation scheme. To this end, different metrics acting as *figures of merit* have been proposed to compare compilation schemes. Those figures of merit try to resemble how well a compiled circuit can be executed on a quantum computer and, by that, act as a *proxy* for the solution quality of the considered application. These figures of merit can be grouped into three categories with an increasing effort to calculate but also higher resemblance of the actual execution:

1) Figures of merit based on characteristics of the compiled quantum circuit: This includes efficiently computable metrics such as the depth of the compiled circuit or its gate count—frequently limited to the number of two-qubit gates because, for most current architectures, their execution error rate is significantly higher than that of single-qubit gates.

2) Figures of merit based on the characteristics of the compiled quantum circuit *and* quantum device: Here, information such as specific gate error rates and qubit decoherence times is also considered, e.g., in the *expected fidelity* used in [10] and the *estimated success probability* described in [42].

3) Figures of merit based on the comparison of (noiseless) simulations and actual execution results of the compiled quantum circuit: When executed both on a noiseless simulator and on an actual quantum device, the overlap in the resulting histograms can be compared, such as, e.g., in [42], [43]. The compilation scheme leading to the largest overlap is selected as the most promising one.

Example 5. *To make those figures of merit more accessible, the compiled quantum circuit from Fig. 1c has been evaluated based on representatives of all three categories:*
1) *Two-Qubit Gate Count:* 7
2) *Expected Fidelity [10]:* 0.5814
3) *Histogram Intersection [43]:* 0.6655

Although these figures of merit cover different aspects of the compiled circuit and its execution, they are used for specific purposes and neglect the end users' perspective, which prioritizes the solution quality of the considered application above all else.

Plain gate counts are typically used to assess single compilation passes, such as mapping, where these numbers directly reflect the passes' efficacy. However, they do not incorporate hardware-induced errors such as gate execution, measurement errors, and decoherence times—making them inadequate for assessing entire compilation flows. Integrating hardware characteristics through noise models derived from calibration data provides a rough estimate of circuit execution quality. Yet, these models often inadequately represent the actual device due to outdated or incomplete data as shown, e.g., in [42]. Comparing the histogram of noiseless simulations with the ones from actual executions of compiled circuits works well for non-variational quantum algorithms but may not be optimal for variational algorithms like QAOA [44] (which is considered one of the most promising algorithms in the current NISQ era [45]), where noise could potentially aid optimization by preventing convergence to local minima, similar to *gradient descent optimization algorithms* [46] used in classical machine learning.

Furthermore, all figures of merit currently used in compilation treat the to-be-compiled quantum circuit as a sequence of instructions without considering the actual application it realizes—wasting a yet untapped potential to increase the solution quality, which is exactly what counts for the potential end users. This is especially unfavorable since the compilation itself can make the difference between a successful execution and obtaining completely random results. Thus, compiling a given quantum circuit in the most promising fashion constitutes an important challenge in the development of quantum computing applications.

In this work, a different approach is explored: instead of defining a *proxy* for the expected solution quality of a considered application and using this as the figure of merit during compilation, the solution quality of the application *itself* is used—offering the potential to further increase it for a considered application.

IV. Proposed Approach

Realizing such an *application-aware* compilation scheme for a considered application requires two steps: An application-aware figure of merit must be defined and a compilation environment must be set up that supports customizable figures of merit. In the following, these steps are described in detail. To this end, a running example of an industry-inspired application is used for illustrative purposes, which is introduced first.

A. Quantum Generative Modeling

Generative learning is experiencing rapid growth and is attracting widespread interest across industries, with applications ranging from anomaly detection to text and image generation, as well as speech and video synthesis. Ultimately, the objective of training a generative model is to express the underlying distribution of a dataset using a machine learning model. In *quantum generative learning*, this model is represented by a parameterized quantum circuit [47].

Example 6. *A* Quantum Circuit Born Machine *(QCBM) [48]–[51] is a quantum generative model that is trained as depicted in Fig. 2. To learn the target distribution—in this case, the distribution of a two-dimensional dataset that resembles the letter X (indicated by the orange tiles)—the parameter values of the shown parameterized quantum circuit are adjusted so that the resulting model distribution is as close as possible to the target distribution.*

B. Application-Aware Figure of Merit

A figure of merit must be defined that takes into account the actual application. For MaxCut problems, it might be the size of the cut. For ground state estimation, it might be the current estimate for the ground state energy. For the traveling salesperson problem, it might be the cost of the determined route. Hence, such *application-aware* figures of merit might look quite different depending on the considered application.

Furthermore, variational algorithms are executed in a hybrid classical-quantum fashion in which the parameter values of the underlying parameterized quantum circuit are optimized using a classical optimizer. Those optimizers usually already optimize for an application-aware figure of merit that defines the solution quality of the considered application. Therefore, the same metric should also be used during the compilation—aiming for the best overall resulting solution quality.

Example 7. *During the QCBM training described in Example 6, the deviation of the model distribution Q to the target distribution P shall be minimized. This difference is determined using the Kullback-Leibler (KL) divergence,*

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

Whenever the KL divergence reaches its minimum, the training is completed and the lower it is, the better the model is trained. Therefore, it is a suitable candidate for the

application-aware figure of merit *in this exemplary application. To minimize the KL divergence, the model parameters are adapted via the covariance matrix adaption evolutionary strategy (CMA-ES) [52], a gradient-free optimizer.*

C. Compilation Environment

A compilation environment is necessary that actually supports customizable figures of merit. Although this is not the case for the currently frequently used SDKs such as Qiskit and TKET, other tools offer this freedom, such as, e.g., the *MQT Predictor* framework [10]. This compilation framework allows one to use a customizable figure of merit and learns, based on artificial intelligence, the optimal sequence of compilation passes for the considered application. To this end, *reinforcement learning* is utilized to determine the most promising sequence of synthesis, optimization, and mapping passes from a large number of available passes provided by multiple quantum SDKs, such as Qiskit [11] and TKET [37].

To realize the proposed approach, significant changes to the *MQT Predictor* framework were necessary. Previously, the framework would try to learn an optimized sequence of passes to compile arbitrary circuits to a particular device—following a "one-fits-all" approach similar to other SDKs. For the use case considered here, only a single input circuit (aiming to solve a particular application instance) is considered and the goal is to find the compiled circuit that is expected to achieve the best application result. Due to the variational nature of most applications currently being explored using quantum computing, the framework also had to be extended to handle the compilation of parameterized quantum circuits—a seemingly simple, yet daunting task in practice due to many subtleties across different quantum compilers and their handling of these circuits. Last, but certainly not least, the framework had to be extended with the possibility to execute/simulate the resulting circuits in order to evaluate the application-aware figure of merit as well as running the application.

Example 8. *To compile the underlying quantum circuit of the QCBM application for the KL divergence as its application-aware figure of merit, the MQT Predictor framework [10] has been adapted accordingly. By providing the quantum circuit of the considered application, such as, e.g., an application instance with 4 qubits as shown in Fig. 2, the framework over time learns the most efficient combination of compilation passes that leads to the highest solution quality—while, during the training, the application itself is run for each determined sequence of compilation passes.*

V. Obtained Results

In this section, the application described in the running examples of Section IV is evaluated with three different problem sizes of 4, 6, and 8 qubits using the described setup based on the MQT Predictor framework $v2.0.0$ and Qiskit $v0.45.3$ in Python[1]. For circuit execution, noisy simulators are used

---

[1]The source code to create those results is publicly-available on GitHub (https://github.com/cda-tum/mqt-predictor/tree/quantum_generative_modeling).
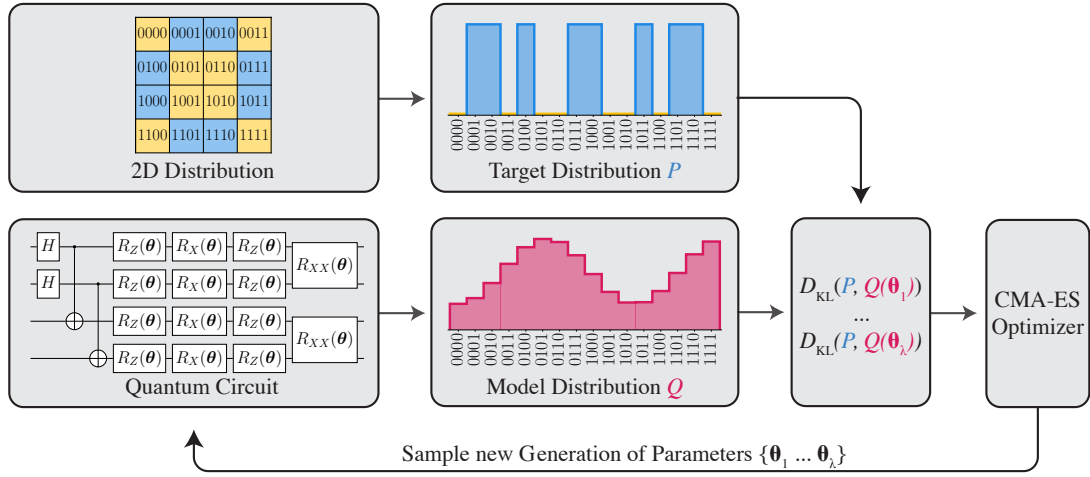
Fig. 2: The QCBM fits the model distribution Q, generated by a parameterized quantum circuit, to the target distribution P, given by the dataset. For a population of size λ, the fitness of each candidate solution $\boldsymbol{\theta}_i$ is evaluated. The CMA-ES optimizer samples a new population of candidate solutions, based on the best-performing candidate solutions of the previous generation.

instead of actual devices due to their limited access and associated costs, although the proposed methodology is not limited to classical simulations and also works on actual devices[2]. To that end, mock-ups of the *ibmq_quito*, *ibmq_nairobi*, and *ibmq_montreal* devices with 5, 7, and 27 qubits are used, and each problem instance has been run on the smallest fitting device. The results are shown in Fig. 3.

To evaluate the proposed application-aware compilation scheme, the solution quality of the considered application—in this case, the KL divergence for the QCBM application—is used as a figure of merit with lower values representing better solutions. Since the application itself runs for a certain number of epochs to optimize the parameter values of the underlying quantum circuit, the figure of merit generally decreases over the epochs, and the lowest KL divergence reached at *any* epoch constitutes the best solution. To this end, the same compiled and parameterized quantum circuit is used throughout all episodes—just with different parameter values. The proposed approach is compared to Qiskit's most-optimized compilation scheme (O3) as the baseline to beat, as this is what was originally used for the application considered. Furthermore, the results of Qiskit's default compilation scheme (O1) are also denoted to underline the effect compilation has on the solution quality. For both baselines, 25 compilation runs have been carried out to cover the spread of their compilation outcomes and, therefore, solution quality. To this end, the worst, median, and best run have been determined—again, measured by the lowest KL divergence that occurred during those runs.

The results for the 4 qubits application instance (whose underlying quantum circuit has been depicted as the exemplary circuit shown in Fig. 2) executed on the mock-up of the *ibmq_quito* device are shown in Fig. 3a. To visualize the solution quality of all 25 runs of both the most-optimized and

---

[2]In fact, running on real devices can be expected to yield the most accurate and optimized results as it does not involve any modeling.

the default baselines, their spread is visualized as a tube graph between the best and worst solution quality of all 25 runs at the respective epoch. Additionally, the median compilation run of the most-optimized leading to the median solution quality is denoted within the corresponding tube.

The proposed approach surpasses both baselines, yielding the best KL divergence, and hence, solution quality. Meanwhile, both baselines show qualitatively similar convergence behavior and solution quality. Measured by the minimal KL divergence across all epochs and compared to the minimal KL divergence of the best baseline, the proposed approach results in an improvement of

$$1 - \frac{min_{proposed}}{min_{best\ baseline}} \approx 6.21\%.$$

This performance gain is achieved *without any* adjustments of the encoding of the generative model as a quantum circuit and is achieved solely taking into account the actual application during the compilation. Compared to the median/worst run of the most-optimized baseline, the improvement is even greater with $9.59\%/32.54\%$. This performance improvement, even over the most-optimized baseline, highlights the potential of application-aware compilation.

For the application instance with 6 qubits executed on the mock-up of the *ibmq_nairobi* device, the solution quality is shown in Fig. 3b. Although the spread of the default baseline is significantly larger than that of the most-optimized one, its solution quality is comparable in the respective best cases. Similarly to before, the proposed approach again results in the best solution quality with an improvement of $8.23\%/13.30\%/22.94\%$ compared to the best/median/worst run of the most-optimized baseline. The 8 qubit application instance that is executed on the *ibmq_montreal* device has led to qualitatively similar results, as shown in Fig. 3c. Here, the proposed approach resulted in an improvement of $2.03\%/16.45\%/32.47\%$ compared to the best/median/worst
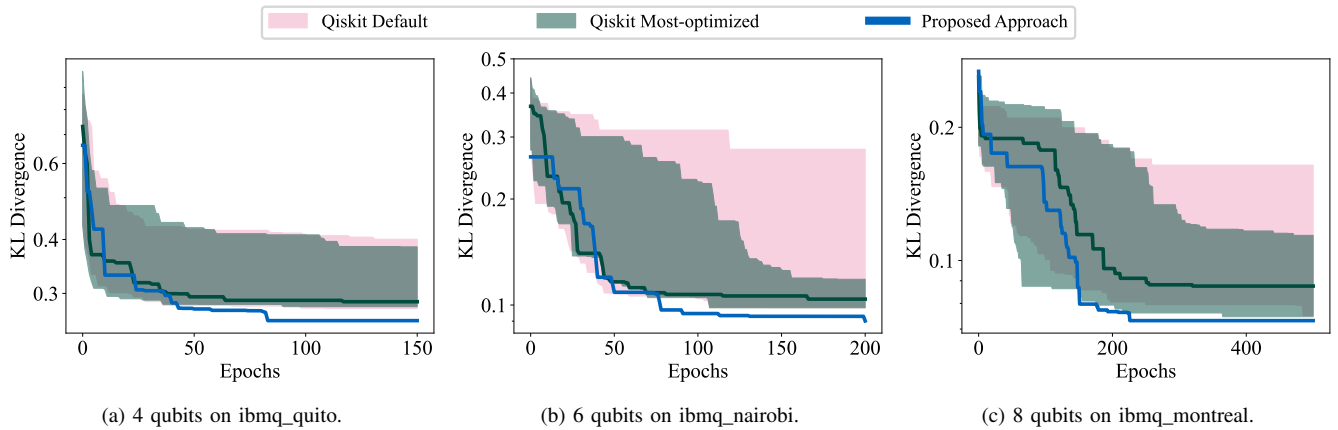
Fig. 3: Results of the training process across three different circuit sizes. Lower values indicate a higher solution quality.

run of the most-optimized baseline. For the training itself, the number of necessary epochs to reach convergence increases with the application size due to the larger number of quantum circuit parameters.

Although the evaluations indicate a general trend of improving solution quality with larger application instances, this is mainly caused by the formulation of the underlying 2D dataset. The percentage of tiles that are used to represent the X letter (orange tiles in Fig. 2) decreases with the application size. For the shown 4 qubit instance it is 8 out of 16 tiles, while for the 6 qubit instance it is significantly lower with 12 out of 36 tiles. To show how this relation affects the KL divergence values, consider the case where the randomly initialized circuits yield the superposition—leading to an initial KL divergence value of 0.69 for the 4 qubit instance and 0.29 for the 6 qubit instance. Therefore, the relative differences in solution quality yielded by different compilation schemes for the same application instance are more meaningful than the absolute values reached across different application sizes.

Furthermore, it should be noted that the spread of both baselines varies significantly between the application instances considered and, respectively, used mock-ups of the actual devices during noisy simulation. This is due to a combination of reasons: the calibration data of the devices being considered are rather different, the circuit width and depth are also different, and, lastly, there is randomness in the classical CMA-ES optimizer, too. Therefore, more studies are needed to clearly determine the influence of each of these factors.

VI. Discussion

Determining useful solutions for the application is a challenge in and of itself in the current NISQ era. Therefore, no opportunity should be left unexplored to further increase the solution quality and application-aware compilation could be a further step towards meaningful use of quantum computing.

The proposed approach allows end users to further increase the quality of their solution without adjusting how they derive their quantum circuit. However, it comes with a large effort in both setting up the proposed compilation scheme and

running it—while the latter can be significantly reduced when executing on actual devices instead of using a noisy simulator. Consequently, end users should determine whether this effort outweighs the gains in solution quality.

This work constitutes a first step towards exploring the direction of application-aware quantum compilation offering many possible paths to further explore this idea, e.g., when choosing between different quantum devices to run the considered application or even before selecting the most promising algorithm and encoding the problem, respectively. Additionally, the influence of other factors such as the underlying noisy simulator or actual devices as well as the classical optimizers and their influence should be further investigated to fully explore the potential the proposed approach offers.

VII. Conclusion

Quantum circuit compilation greatly affects the quality of the solution when using quantum computers to realize a considered application and often makes the difference between a successful execution and obtaining completely random results. Current state-of-the-art compilers offer a selectable optimization level as a trade-off between the compilation time and the *quality* of the compiled circuit—based on *figures of merit* aiming at resembling how successful a compiled circuit can be executed on a quantum computer and, by that, acting as a *proxy* for the solution quality of the considered application. However, they treat the to-be-compiled quantum circuit as a sequence of instructions without considering the actual application it realizes—wasting a yet untapped potential to increase the solution quality. In this work, a different compilation approach has been proposed that optimizes directly for the solution quality *itself* instead for any proxy of it. To showcase the effectiveness over the current compilation schemes, the experimental evaluations for an industry-inspired quantum generative model showed an increased solution quality for all evaluated application instances compared to Qiskit's most-optimized compilation scheme. By this, this work presents a first step towards *application-aware* compilation.

REFERENCES

[1] A. Bayerstadler *et al.*, "Industry quantum computing applications," *EPJ Quantum Technology*, 2021.

[2] N. Stamatopoulos *et al.*, "Option pricing using quantum computers," *Quantum*, 2020.

[3] S. Harwood *et al.*, "Formulating and Solving Routing Problems on Quantum Computers," *IEEE Transactions on Quantum Engineering*, 2021.

[4] S. McArdle *et al.*, "Quantum computational chemistry," *Rev. Mod. Phys.*, 1 2020.

[5] N. Quetschlich, L. Burgholzer, and R. Wille, "Towards an Automated Framework for Realizing Quantum Computing Solutions," in *Int'l Symp. on Multi-Valued Logic*, 2023.

[6] N. Quetschlich, M. Soeken, P. Murali, and R. Wille, "Utilizing resource estimation for the development of quantum computing applications," 2024. arXiv: 2402.12434.

[7] 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*, 2019.

[8] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, "Expressive power of parametrized quantum circuits," *Physical Review Research*, 2020.

[9] B. Villalonga *et al.*, "Establishing the quantum supremacy frontier with a 281 pflop/s simulation," *Quantum Science and Technology*, 2020.

[10] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Predictor: Automatic Device Selection with Device-Specific Circuit Compilation for Quantum Computing," 2023. arXiv: 2310.06889.

[11] A. Javadi-Abhari *et al.*, "Quantum computing with Qiskit," 2024. arXiv: 2405.08810.

[12] B. Giles and P. Selinger, "Exact synthesis of multiqubit Clifford+T circuits," *Physical Review A*, 2013.

[13] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2013.

[14] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control Toffoli gates," in *Int'l Symp. on Multi-Valued Logic*, 2011.

[15] A. Zulehner and R. Wille, "One-pass design of reversible circuits: Combining embedding and synthesis for reversible logic," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2018.

[16] P. Niemann, R. Wille, and R. Drechsler, "Advanced exact synthesis of Clifford+T circuits," *Quantum Information Processing*, 2020.

[17] T. Peham *et al.*, "Depth-optimal synthesis of Clifford circuits with SAT solvers," in *Int'l Conf. on Quantum Computing and Engineering*, 2023. arXiv: 2305.01674.

[18] E. Younis and C. Iancu, "Quantum circuit optimization and transpilation via parameterized circuit instantiation," in *Int'l Conf. on Quantum Computing and Engineering*, 2022.

[19] W. Hattori and S. Yamashita, "Quantum circuit optimization by changing the gate order for 2D nearest neighbor architectures," in *Int'l Conf. of Reversible Computation*, 2018.

[20] G. Vidal and C. M. Dawson, "Universal quantum circuit for two-qubit transformations with three controlled-NOT gates," *Physical Review A*, 2004.

[21] T. Itoko *et al.*, "Quantum circuit compilers using gate commutation rules," in *Asia and South Pacific Design Automation Conf.*, 2019.

[22] D. Maslov, G. Dueck, D. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2008.

[23] S. Niu *et al.*, "Powerful quantum circuit resizing with resource efficient synthesis," 2023. arXiv: 2311.13107.

[24] W.-H. Lin *et al.*, "Scalable optimal layout synthesis for NISQ quantum processors," in *Design Automation Conf.*, 2023.

[25] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2019.

[26] A. Matsuo and S. Yamashita, "An efficient method for quantum circuit placement problem on a 2-D grid," in *Int'l Conf. of Reversible Computation*, 2019.

[27] B. Tan and J. Cong, "Optimal qubit mapping with simultaneous gate absorption," in *Int'l Conf. on CAD*, 2021.

[28] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations," in *Design Automation Conf.*, 2019.

[29] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," in *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, 2019.

[30] T. Peham, L. Burgholzer, and R. Wille, "On Optimal Subarchitectures for Quantum Circuit Mapping," *ACM Transactions on Quantum Computing*, 2023.

[31] S. Hillmich, A. Zulehner, and R. Wille, "Exploiting Quantum Teleportation in Quantum Circuit Mapping," in *Asia and South Pacific Design Automation Conf.*, 2021.

[32] A. Zulehner and R. Wille, "Compiling SU(4) quantum circuits to IBM QX architectures," in *Asia and South Pacific Design Automation Conf.*, 2019.

[33] J. Liu *et al.*, "Tackling the qubit mapping problem with permutation-aware synthesis," in *Int'l Conf. on Quantum Computing and Engineering*, 2023.

[34] A. Cowtan *et al.*, "On the Qubit Routing Problem," in *Conf. on the Theory of Quantum Computation, Communication and Cryptography (TQC)*, 2019.

[35] L. Schmid, S. Park, and R. Wille, "Hybrid Circuit Mapping: Leveraging the Full Spectrum of Computational Capabilities of Neutral Atom Quantum Computers," in *Design Automation Conf.*, 2024.

[36] R. Wille and L. Burgholzer, "MQT QMAP: Efficient quantum circuit mapping," in *Int'l Symp. on Physical Design*, 2023.

[37] S. Sivarajah *et al.*, "t|ket⟩: A retargetable compiler for NISQ devices," *Quantum Science and Technology*, 2020.

[38] "Pennylane: Automatic differentiation of hybrid quantum-classical computations," 2018. arXiv: 1811.04968.

[39] E. Younis *et al.*, "Berkeley Quantum Synthesis Toolkit (BQSKit) v1," https://www.osti.gov/biblio/1785933, 2021.

[40] R. Wille *et al.*, "The MQT Handbook: A Summary of Design Automation Tools and Software for Quantum Computing," in *Int'l Conf. on Quantum Software*, 2024. arXiv: 2405.17543, A live version of this document is available at https://mqt.readthedocs.io.

[41] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, 2018.

[42] S. Dangwal, G. S. Ravi, L. M. Seifert, and F. T. Chong, "Clifford assisted optimal pass selection for quantum transpilation," 2023. arXiv: 2306.15020.

[43] M. Salm, J. Barzen, F. Leymann, and B. Weder, "Prioritization of compiled quantum circuits for different quantum computers," in *Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER)*, 2022.

[44] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014. arXiv: 1411.4028.

[45] R. Shaydulin and Y. Alexeev, "Evaluating quantum approximate optimization algorithm: A case study," in *Int'l Green and Sustainable Computing Conf. (IGSC)*, 2019.

[46] S. Ruder, "An overview of gradient descent optimization algorithms," 2017. arXiv: 1609.04747.

[47] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, 2019.

[48] S. Cheng, J. Chen, and L. Wang, "Information perspective to probabilistic modeling: Boltzmann machines versus born machines," *Entropy*, 2018.

[49] M. Benedetti *et al.*, "A generative modeling approach for benchmarking and training shallow quantum circuits," *npj Quantum Information*, 2019.

[50] F. J. Kiwit *et al.*, "Application-oriented benchmarking of quantum generative learning using QUARK," *Int'l Conf. on Quantum Computing and Engineering*, 2023.

[51] F. J. Kiwit *et al.*, "Benchmarking quantum generative learning: A study on scalability and noise resilience using QUARK," 2024. arXiv: 2403.18662.

[52] N. Hansen, Y. Akimoto, and P. Baudis, "CMA-ES/pycma on Github," Zenodo, https://doi.org/10.5281/zenodo.2559634, 2019.