A Systematic Literature Review on Optimization Techniques for Quantum Computing Compilers

Camilla Vitoria Bueno da Rocha, Ana Luiza Diniz Santos, Matheus Alcântara Souza

Computer Architecture and Parallel Processing Team (CArT)
Departamento de Ciencia da Computação – Pontifícia Universidade Católica de Minas Gerais (PUC Minas) Belo Horizonte - MG - Brazil

{camilla.bueno, aldsantos}@sga.pucminas.br, matheusalcantara@pucminas.br

Abstract. *The rapid development of Quantum Computing (QC) as a promising computing paradigm has garnered significant attention for its ability to harness quantum mechanical properties for computation. With classical computing facing limitations outlined by Moore's Law, QC emerges as a potential alternative for tackling complex computational problems. Yet, to unlock its full potential, robust and optimized compilers are pivotal, especially in addressing challenges posed by circuits with numerous qubits. In this systematic literature review, we analyze 18 articles to identify proposed optimizations for quantum compilers, exploring their applications, performance impacts, and emerging trends. Our primary goal is to offer valuable insights into the recent advancements and challenges in QC compiler optimizations. This will be achieved through the clustering of optimization groups, ultimately facilitating further progress in the development of highly optimized quantum algorithms and circuits.*

1. Introduction

For decades, Moore's Law proved to be correct, which ensured significant development for computing. Recently, researchers and engineers have raised concerns about the potential "Death of Moore's Law" [Williams 2017] due to the physical limitations of chips, as we are approaching the smallest possible size for a transistor in the near future.

As a result, new computing paradigms are gaining significant prominence, with QC being one of the most notable. QC uses quantum mechanical properties such as superposition and entanglement to perform computation, in contrast to classical computing, which operates based on classical physics principles. Research on this topic started in the early eighties, and there have been many advances in the field because of its potential to increase computation power and speed significantly in various computing fields since then such as artificial intelligence [Chauhan et al. 2022] and cryptography [Lella et al. 2022]. However, to fully realize the potential of QC and become a working paradigm to impact people's daily lives truly there is a need for robust studies and advancements in the field.

In classical computing, following extensive research efforts, compilers have advanced to a stage where they can execute a range of optimizations. This achievement has facilitated a higher level of abstraction and streamlined development processes, being a fundamental step in the advancement of computing as a whole. Acknowledging

Camilla and Ana are undergraduate students in Computer Science and Computer Engineering. The authors thank FAPEMIG, CNPq, and PUC Minas for their support in this research.

the demand for more robust and optimized compilers in the realm of QC, we propose a Systematic Literature Review (SLR), that aims to identify proposed optimizations for QC compilers in the existing literature. This paper explores how these optimizations are applied, their impacts on performance, and the emerging trends associated with them.

This paper is organized as follows. Section 2 presents the steps we set to conduct our SLR and some information regarding our search method. Section 3 presents the result of our research evaluation. In Section 4 we present our final considerations.

2. Methodology

Our SLR consists of the following methodology. We defined three research questions: (1) *What are the proposed optimizations for QC compilers?*, (2) *How have these optimizations been conducted?*, and (3) *What technical challenges are QC compilers facing, and how have they been addressed?*.

We included works published within the years 2018 to 2023 from renowned scientific databases, namely IEEE Xplore, IBM Quantum Network Papers, and Association for Computing Machinery (ACM). The query string used was: "quantum computing compilers." In addition, we refined the search for a specific look-up adding various filters available in each site [1].

Finally, only works written in the English language are considered. To ensure the focus of this study, duplicates of previously identified studies and articles that tangentially touch upon the topic will be excluded from the analysis. Table 1 summarizes the number of articles we selected.

Table 1. Primary studies selection

Scientific Database	Initial Keyword Search	First Exclusion	Total Inclusion
IEEE	88	12	4
IBM QNP	10	8	4
ACM	105.030	10	4

The selection process involved a dynamic reading of the articles, beginning with their titles, followed by a detailed review of the abstracts. Thus, the final selection was based on a complete analysis of the conclusion and future works sections. After selecting 12 articles from these databases, we further expanded our search by adding 6 articles from the references of these established works.

3. Quantum Compiler Optimization Techniques

Optimization techniques play a fundamental role in enhancing the efficiency and performance of QC applications, especially as quantum systems grow in complexity and scale. In this section, we will explore a range of optimization approaches employed in QC compilers. The purpose is to comprehensively analyze proposed optimizations in the existing literature and their impacts on compilers. We will elucidate the methods by which they

[1]For the ACM database, we set that the first exclusion phase would include 10 studies. This ensured the manageability of our selection process, given the papers volume it returned.

were recognized and applied, aiming to identify discernible trends within the QC compiler research field. Specific focus areas among researchers, such as particular stages in the compilation process, will also be explored.

3.1. Qubit Routing

Qubit routing in QC involves mapping quantum operations to specific qubits based on the quantum machine's topology. Swap gates are inserted to relocate qubits, ensuring adjacent qubits can execute instructions with multiple qubits. However, introducing swap gates increases circuit depth, leading to higher error probabilities and longer execution times due to quantum decoherence [Knill et al. 2008]. Moreover, this affects the execution time which affects the stability and integrity of the system. Good compilers must achieve a minimum mapping, aiming to insert the fewest possible swaps.

The minimum mapping problem, aiming to insert the fewest swaps, is NP-Complete [Botea et al. 2018], requiring heuristic or approximate algorithms for near-optimal solutions. Researchers explored reinforcement learning in [Pozzi et al. 2022]. Despite showing promising results on widely used benchmarks for this problem, the method is significantly slower than established compilers such as Qiskit [Qiskit 2023] and Tket [Sivarajah et al. 2020]. Additionally, this method is difficult to be applied in systems with low connectivity and when multiple qubits need to interact with a specific one.

Another approach, SABRE [Li et al. 2019], offers an efficient heuristic for optimal mapping in linear time, delivering substantial improvements in execution time and gate reduction for various quantum systems. The algorithm's first focus is identifying a good initial mapping as it directly impacts the performance of the resulting circuit [Siraichi et al. 2018]. Once the initial mapping is obtained, the heuristic is then carried out, which includes a cost function capable of considering subsequent gates and controlling swap parallelism. The results demonstrate substantial improvements in execution time and gate reduction, applicable to both mathematical calculations that involve a few qubits and larger quantum systems.

In [Liu et al. 2022], the NASSC (Not All Swaps Have the Same Cost) algorithm connects routing and optimization stages. The key is that a circuit generated by routing with a higher number of swaps can yield the best possible solution with minimum depth when subjected to subsequent optimizations. The strategy reduced the circuit depth and CNOT gate count substantially, in comparison with SABRE. These algorithms demonstrate the potential for further optimizations in QC.

Addressing NISQ technology, [Zulehner and Wille 2019] proposes a compilation approach for SU(4) quantum circuits into IBM QX architectures, addressing the NISQ technology. The method involves decomposing operations into elementary gates and a mapping procedure to associate logical to physical qubits. It uses a mixed-integer programming approach to optimize the mapping process and minimize swap operations.

3.2. Intermediate Representation

Intermediate Representation (IR) optimization plays a fundamental role in improving QC programs. The source code is converted into an IR before being translated into machine-

executable code. The IR serves as an abstract form, preserving the semantic characteristics of the original program while enabling several optimization techniques.

The Paulihedral framework is a work in this field [Li et al. 2022]. It is designed to optimize the quantum simulation kernel [2], aiming to optimize the simulation of quantum physical systems. The key is the new Pauli IR, capable of preserving the semantics and high-level constraints of quantum simulation kernels. This enables large-scale optimizations that would be challenging to achieve at the low-level quantum gate level.

Another relevant article, proposed by [Nguyen and McCaskey 2022], presents a quantum-classical Multi-Level Intermediate Representation (MLIR). This approach enables the creation of an optimizable, retargetable, and forward-looking compiler for quantum programming languages. Leveraging the MLIR framework, quantum language expressions are mapped to LLVM IR. The article focuses on applying optimizations in IR that significantly reduce both "multi-qubit" operations and circuit depth. These optimizations involve a combination of classical techniques, such as constant propagation, loop unrolling, and dead-code elimination, along with quantum-specific optimizations like "gate permutation" and "gate sequence simplification." Notably, the researchers emphasize the importance of performing optimizations in a "flat" region, where loop unrolling and function inlining play vital roles, contributing to better results. The resulting compiler demonstrates significantly reduced compilation times compared to standalone quantum language approaches and comparative compilers. Additionally, the compiler provides quantum resource optimizations, significantly reducing entangled operations, a common source of noise in quantum programs.

A third prominent article, also authored by [McCaskey and Nguyen 2021], explores the utility of the MLIR in QC. The authors extend MLIR with a new quantum dialect, allowing for the expression and compilation of common quantum assembly languages. This quantum-classical IR can be reduced to LLVM IR, adhering to the specification of the Quantum Intermediate Representation (QIR) [3]. The compiler developed from this approach also exhibits significantly reduced compilation times and provides quantum resource optimizations, enabling rapid prototyping of quantum compilers and integration with classical compilation approaches.

3.3. Peephole Optimization

Peephole optimization is a technique in which a set of instructions is substituted with another equivalent set that is more efficient. This method often relies on commutativity, which is a property ensuring that the order of operands does not impact the final result. The Quantum Approximate Optimization Algorithm capitalizes on the commutativity property [Alam et al. 2020]. The authors presented a compilation flow tailored to the characteristics of this problem. By defining an optimal gate order, they effectively minimize the number of inserted swaps, reducing circuit depth and gate count. Despite the problem's inherent complexity (O(n!)), which results in increased execution time, the authors adeptly apply efficient methods within this flow, demonstrating that specialized compilers can provide notable advantages and optimizations for QC tasks.

Another study that considers the commutation of operations is

[2]Essential subroutine that appears as an extensive sequence of gates in various quantum programs.

[3]QIR: https://learn.microsoft.com/en-us/azure/quantum/concepts-qir

[Itoko and Imamichi 2020]. In this research, the authors concentrate on the quantum operation scheduling (QOS) compilation stage and address it as a job-shop problem. The study demonstrates that by incorporating commutativity considerations into this specific compilation stage, a notable improvement in execution time, up to 7%, can be achieved, even without accounting for optimizations in earlier stages, on average.

Also, in [Shi et al. 2019], the authors identified the QOS stage as a crucial area with significant opportunities for optimization. The article stands out by handling blocks of up to 10 qubits, which came at the cost of increased compilation time and diverged from the more common approach of using one or 2-qubit blocks typically found in the literature. The authors introduced innovations by implementing steps to detect commutativity and aggregate instructions, enabling the creation of more efficient schedules and customized, optimized control pulses. Their efforts led to a remarkable average reduction of circuit latency by five times.

In contrast to the previously mentioned articles, the work presented in [Liu et al. 2021] diverges by relying on something other than the commutativity of operations. Instead, the authors introduced a novel technique called "Relaxed Peephole Optimization." This method involves replacing operations on qubits in known bases or pure states with simpler operations, utilizing two new passes named Quantum Basis-State Optimization and Quantum Pure-State Optimization. When implemented in Qiskit [Qiskit 2023], the method demonstrated outstanding results, on average reducing 11% of CNOT gates in the final circuit. This significant improvement demonstrates that research in this area can produce meaningful results.

3.4. Noise-adaptive

The noise-adaptive approach in quantum compilers refers to the compiler's ability to adjust its optimization strategies based on the specific noise characteristics of a quantum device. This adaptation considers hardware noise statistics, such as qubit error rates and fidelity, to generate optimized quantum programs for the target machine. NISQ is a term coined by John Preskill in his article [Preskill 2018] discussing the challenges and opportunities associated with QC in medium-scale devices subject to noise. The presence of noise in quantum hardware is one of the main limitations faced, making noise-adaptive adaptation in compilers an essential area of study to enhance the efficiency and reliability of quantum programs executed on these machines.

In response to this challenge, [Murali et al. 2019] propose and evaluate back-end compiler techniques to map and optimize high-level QC programs, considering the diverse hardware characteristics of NISQ systems. These techniques employ an LLVM IR of the quantum program and generate quantum executables on the IBM QC public machine. The work implements and evaluates various mapping methods, including optimal and heuristic approaches, regarding the availability of dynamic calibration data from the machine, noise parameters, routing strategies, and the trade-off between compilation time scalability and execution time success. The experimental results demonstrate gains of up to 18x in the program success rate compared to the Qiskit compiler [Qiskit 2023].

In another study, the authors [Ferrari and Amoretti 2022] present a novel noise-adaptive quantum compilation strategy demonstrating computational efficiency. This approach considers a heavy hexagon lattice for the coupling map of the quantum device.

It involves placement, routing, and optimization tasks, considering the noise statistics of the device for some or all passes obtained from calibration data. Placement is performed at the beginning of the compilation process, defining a mapping between the input quantum circuit's virtual qubits and the device's physical qubits. Routing involves modifying the circuit to conform to the device's constraints. Benchmarking results reveal that the noise-adaptive compilation strategy is particularly effective for deep and square circuits.

In this study scenario, [Tannu and Qureshi 2019] examines the challenges associated with qubit allocation and movement in NISQ machines. The researchers emphasize the significance of considering the variability in error rates among different qubits and connections, as this can significantly impact the performance of the quantum system. To tackle this issue, they propose policies termed "Variability-Aware Qubit Movement" (VQM) and "Variability-Aware Qubit Allocation" (VQA). The VQM policy selects the path with the lowest error rate between two qubits, while the VQA policy chooses qubits with the lowest error rates to execute quantum operations. Additionally, the scientists present a hybrid policy, VQM+VQA, which combines the two preceding strategies, further enhancing the system's reliability and effectiveness. Experimental results demonstrate that Variability-Aware policies can increase the system's reliability by up to 1.7 times, offering a promising outlook for advancing QC in NISQ environments.

To complement the approach in the context of noise adaptation in quantum compilers, [Nishio et al. 2020] introduces a compilation tool that tackles the diversity in individual qubits' error rates in NISQ systems, to maximize the probability of success for real-world subroutines, such as adder circuits. Employing an established metric for selecting between potential paths and circuit alternatives, the research was evaluated on 20-qubit IBM systems like Tokyo and Poughkeepsie. The tool employs a heuristic based on beam search to efficiently explore the solution space, achieving the optimal qubit allocation and routing. The compilation assessment was carried out on adder circuits, revealing that the process increases the estimated probability of success and decreases the KL divergence in comparison to error-agnostic placement, thus making a significant contribution to enhancing the performance and dependability of quantum programs in NISQ systems.

3.5. Other optimizations

There are works unrelated to the previously mentioned significant groups, yet they showcase intriguing optimization methods and outcomes. One previous work explores an optimized compilation approach specifically focusing on distributed computing [Cuomo et al. 2023]. The efforts focused on optimizing telegates (remote operations) to achieve the shortest execution time while utilizing minimal resources. Their optimization strategy is based on a property they have introduced and referred to as quasi-parallelism. According to this property, even if an operation x requires another operation y to be executed before it, it does not necessarily mean that both operations cannot be executed simultaneously at the same time step.

Moreover, to enable one-way quantum computation on realistic photonic quantum architectures, [Zhang et al. 2023] propose the OneQ compilation framework. This framework takes high-level quantum algorithms as input and transforms them into low-level circuit descriptions optimized for execution on photonic quantum architectures. The methodology includes stages such as circuit optimization, resource allocation, and

scheduling while addressing the limited connectivity of photonic architectures through a novel routing algorithm.

Another work introduces a direct control at the microarchitectural level of quantum computers to enhance the performance of programs [Gokhale et al. 2020]. The compiler takes high-level algorithms as input and generates a low-level circuit description, enabling performance optimization. The applied techniques have been validated through millions of experimental shots on IBM quantum computers controlled via the OpenPulse control interface. This approach has demonstrated reduced error rates and faster execution times compared to standard gate-based compilations, showing its potential for significant advancements in near-term QC algorithms.

4. Conclusion

In this work, we have presented and explained four significant types of compiler optimization techniques for QC, along with prominent articles in the field and commonly used tools for testing these methods. Through our analysis, it becomes evident that for the advancement of QC, particularly in the NISQ era, substantial research and development are necessary to create methods that can generate optimized circuits, ensuring the efficiency and reliability of quantum programs. As a result, optimizations in QC compilers hold significant importance and demand continuous research and improvement.

References

Alam, M., Saki, A. A., and Ghosh, S. (2020). An efficient circuit compilation flow for quantum approximate optimization algorithm. In *Design Aut. Conf.*, pages 1–6.

Botea, A., Kishimoto, A., and Marinescu, R. (2018). On the complexity of quantum circuit compilation. In *Symp. on Combinatorial Search*.

Chauhan, V. et al. (2022). Quantum computers: A review on how quantum computing can boom AI. In *Int. Conf. on Adv. Comp. and Innov. Tech. in Eng.*, pages 559–563.

Cuomo, D. et al. (2023). Optimized compiler for distributed quantum computing. *ACM Transactions on Quantum Computing*, 4(2).

Ferrari, D. and Amoretti, M. (2022). Noise-adaptive quantum compilation strategies evaluated with application-motivated benchmarks. In *Int. Conf. on Comp. Frontiers*, page 237–243, New York. ACM.

Gokhale, P. et al. (2020). Optimized quantum compilation for near-term algorithms with openpulse. In *Int. Symp. on Microarchitecture*, volume 2020-October, pages 186–200.

Itoko, T. and Imamichi, T. (2020). Scheduling of operations in quantum compiler. In *Int. Conf. on Quantum Comp. and Eng.*, pages 337–344, Los Alamitos.

Knill, E. et al. (2008). Randomized benchmarking of quantum gates. *Phys. Rev. A*, 77:12307.

Lella, E. et al. (2022). Cryptography in the quantum era. In *Workshop on Low Temperature Electronics*, pages 1–4.

Li, G., Ding, Y., and Xie, Y. (2019). Tackling the qubit mapping problem for nisq-era quantum devices. In *Int. Conf. on Arch. Support for Prog. Lang. and Operating Systems*, pages 1001–1014. ACM.

Li, G. et al. (2022). Paulihedral: A generalized block-wise compiler optimization framework for quantum simulation kernels. In *Int. Conf. on Arch. Support for Prog. Lang. and Operating Systems*, page 554–569, New York, NY, USA. ACM.

Liu, J., Bello, L., and Zhou, H. (2021). Relaxed peephole optimization: A novel compiler optimization for quantum circuits. In *Int. Symp. on Code Generation and Optimization*, page 301–314. IEEE Press.

Liu, J., Li, P., and Zhou, H. (2022). Not all swaps have the same cost: A case for optimization-aware qubit routing. In *Int. Symp. on High-Performance Comp. Arch.*, pages 709–725, Los Alamitos. IEEE Comp. Soc.

McCaskey, A. and Nguyen, T. (2021). A mlir dialect for quantum assembly languages. In *Int. Conf. on Quantum Comp. and Eng.*, pages 255–264, Los Alamitos. IEEE Comp. Soc.

Murali, P. et al. (2019). Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Int. Conf. on Arch. Support for Prog. Lang. and Operating Systems*, page 1015–1029, New York. ACM.

Nguyen, T. and McCaskey, A. (2022). Retargetable optimizing compilers for quantum accelerators via a multilevel intermediate representation. *IEEE Micro*, 42(05):17–33.

Nishio, S. et al. (2020). Extracting success from ibm's 20-qubit machines using error-aware compilation. *J. Emerg. Technol. Comput. Syst.*, 16(3).

Pozzi, M. G. et al. (2022). Using reinforcement learning to perform qubit routing in quantum compilers. *ACM Transactions on Quantum Computing*, 3(2).

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79.

Qiskit (2023). Qiskit: An open-source framework for quantum computing. Available in: https://doi.org.10.5281/zenodo.2573505.

Shi, Y. et al. (2019). Optimized compilation of aggregated instructions for realistic quantum computers. In *Int. Conf. on Arch. Support for Prog. Languages and Operating Systems*, page 1031–1044, New York. ACM.

Siraichi, M. Y. et al. (2018). Qubit allocation. In *Int. Symp. on Code Generation and Optimization*, page 113–125, New York, NY, USA. ACM.

Sivarajah, S. et al. (2020). Tket: a retargetable compiler for NISQ devices. *Quantum Science and Technology*, 6(1):014003.

Tannu, S. S. and Qureshi, M. K. (2019). Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers. In *Int. Conf. on Arch. Support for Prog. Lang. and Operating Systems*, page 987–999, New York. ACM.

Williams, R. S. (2017). What's next? the end of moore's law. *Comp. in Science & Eng.*, 19(2):7–13.

Zhang, H. et al. (2023). Oneq: A compilation framework for photonic one-way quantum computation. In *Int. Symp. on Comp. Arch.*, New York, NY, USA. ACM.

Zulehner, A. and Wille, R. (2019). Compiling su(4) quantum circuits to ibm qx architectures. In *Asia and South Pacific Des. Autom. Conf.*, page 185–190, New York. ACM.