Resilience of Quantum Optimization Algorithms

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Yanjun Ji

aus Shandong, China

Hauptberichter: Prof. Dr. Ilia Polian
Mitberichter: Prof. Dr. Peter Rabl

Tag der mündlichen Prüfung: 7. November 2024.

Institut für Technische Informatik der Universität Stuttgart
2024

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Prof. Dr. Ilia Polian, for his unwavering guidance, invaluable feedback, and constant encouragement throughout my PhD journey.

I would also like to express my appreciation to my adviser, Prof. Dr. Peter Rabl, whose constructive comments and insightful suggestions significantly improved the quality of my dissertation.

Additionally, I would like to thank my colleagues for their insightful discussions and helpful suggestions.

I acknowledge the use of IBM Quantum services for this work and to advanced services provided by the IBM Quantum Researchers Program. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

This work was funded in part by the Ministry of Economic Affairs, Labour and Tourism Baden Württemberg, under the project QORA in the frame of the Competence Center Quantum Computing Baden-Württemberg.

Stuttgart, May 2024
Yanjun Ji

Abstract

Quantum optimization algorithms (QOAs) show promise in surpassing classical methods for solving complex problems. However, their practical application is limited by the sensitivity of quantum systems to noise. This study addresses this challenge by investigating the resilience of QOAs and developing strategies to enhance their performance and robustness on noisy quantum computers. We begin by establishing an evaluation framework to assess the performance of QOAs under various conditions, including simulated noise-free and error-modeled environments, as well as real noisy hardware, providing a foundation for guiding the development of enhancement strategies. We then propose innovative techniques to improve the performance of algorithms on near-term quantum devices characterized by limited qubit connectivity and noisy operations. Our study introduces an effective compilation process that maximizes the utilization of classical and quantum resources. To overcome the restricted connectivity of hardware, we develop an algorithm-oriented qubit mapping approach that bridges the gap between heuristic and exact methods, providing scalable and optimal solutions. Additionally, we demonstrate, for the first time, selective optimization of quantum circuits on real hardware by optimizing only gates implemented with low-quality native gates, providing significant insights for large-scale quantum computing. We also investigate error mitigation strategies and their dependence on hardware features and algorithm implementation details, emphasizing the synergistic effects of error mitigation and circuit design. While error mitigation can suppress the effects of noise, hardware quality and circuit design are ultimately more critical for achieving high performance. Building upon these insights, we explore the cooptimization of algorithm design and hardware implementation to achieve optimal performance and resilience. By optimizing gate sequences and parameters at the algorithmic level and minimizing error-prone two-qubit gates during compilation, we demonstrate significant improvements in QOA performance. Finally, we explore the practical application of QOAs in real-world problems, emphasizing the importance of optimizing parameters in problem instances to identify optimal solutions. With extensive experiments conducted on real devices, this dissertation makes a substantial contribution to the field of quantum optimization, providing both theoretical foundations and practical strategies for addressing the challenges posed by near-term quantum hardware. Our findings pave the way for the realization of practical quantum computing applications and unlock the full potential of QOAs.

Zusammenfassung

Quantenoptimierungsalgorithmen (QOAs) zeigen vielversprechende Ergebnisse bei der Überlegenheit klassischer Methoden zur Lösung komplexer Probleme. Ihre praktische Anwendung wird jedoch durch die Empfindlichkeit von Quantensystemen gegenüber Rauschen begrenzt. Diese Studie geht dieser Herausforderung an, indem sie die Resilienz von QOAs untersucht und Strategien zur Verbesserung ihrer Leistung und Robustheit auf verrauschten Quantencomputern entwickelt. Wir beginnen mit der Entwicklung eines Bewertungsrahmens zur Beurteilung der Leistung von QOAs unter verschiedenen Bedingungen, einschließlich simulierter rauschfreier und fehlermodellierter Umgebungen sowie realer verrauschter Hardware, um eine Grundlage für die Entwicklung von Verbesserungsstrategien zu schaffen. Anschließend schlagen wir innovative Techniken zur Verbesserung der Leistung von Algorithmen auf kurzfristigen Quantenrechnern mit begrenzter Qubit-Konnektivität und verrauschten Operationen vor. Unsere Studie stellt einen effektiven Kompilierungsprozess vor, der die Nutzung von klassischen und Quantenressourcen maximiert. Um die begrenzte Konnektivität der Hardware zu überwinden, entwickeln wir einen algorithmusorientierten Qubit-Mapping Ansatz, der die Lücke zwischen heuristischen und exakten Methoden schließt und skalierbare und optimale Lösungen bietet. Darüber hinaus demonstrieren wir zum ersten Mal eine selektive Optimierung von Quantenschaltungen auf realer Hardware, indem wir nur Gatter optimieren, die mit niedrigqualitativen nativen Gattern implementiert wurden, was wichtige Erkenntnisse für das Quantencomputing im großen Maßstab liefert. Wir untersuchen auch Fehlermitigationsstrategien und ihre Abhängigkeit von Hardware-Eigenschaften und Algorithmus-Implementierungsdetails, wobei wir die synergistischen Effekte von Fehlermitigation und Schaltkreisdesign betonen. Während Fehlermitigation die Auswirkungen von Rauschen unterdrücken kann, sind Hardwarequalität und Schaltkreisdesign letztendlich entscheidender für das Erreichen hoher Leistung. Aufbauend auf diesen Erkenntnissen untersuchen wir die Co-Optimierung von Algorithmusdesign und Hardwareimplementierung, um optimale Leistung und Widerstandsfähigkeit zu erreichen. Durch die Optimierung von Gattersequenzen und Parametern auf algorithmischer Ebene und die Minimierung fehleranfälliger Zwei-Qubit-Gatter während der Kompilierung demonstrieren wir signifikante Verbesserungen der QOA-Leistung. Schließlich untersuchen wir die praktische Anwendung von QOAs in realen Problemen und betonen die Bedeutung der Optimierung von Parametern in Probleminstanzen, um optimale Lösungen zu identifizieren. Mit umfangreichen Experimenten, die an realen Geräten durchgeführt wurden, leistet diese Dissertation einen wesentlichen Beitrag zum Gebiet der Quantenoptimierung, indem sie sowohl theoretische Grundlagen als auch praktische Strategien zur Bewältigung der Herausforderungen von kurzfristiger Quantenhard-

ware bereitstellt. Unsere Ergebnisse ebnen den Weg für die Realisierung praktischer Quantencomputing-Anwendungen und erschließen das volle Potenzial von QOAs.

Contents

List of Figures

List of Tables

List of Abbreviations

AQC	Adiabatic Quantum Computing
AQQMAP	Algorithm-Oriented Qubit Mapping
AR	Approximation Ratio
CA	Calibration-Aware
CLOPS	Circuit Layer Operations per Second
COBYLA	Constrained Optimization BY Linear Approximation
CPMG	Carr-Purcell-Meiboom-Gill
CR	Cross Resonance
DC	Digitized Counterdiabatic
DD	Dynamical Decoupling
DO	Decomposition and Optimization
ECR	Echoed Cross Resonance
GHZ	Greenberger–Horne–Zeilinger
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno
MQT	Munich Quantum Toolkit
NAM	Noise-Aware Matching
NISQ	Noisy Intermediate-Scale Quantum
PEC	Probabilistic Error Cancellation
PortOpt	Portfolio Optimization
QA	Quantum Annealing
QAOA	Quantum Approximate Optimization Algorithm
QASM	Quantum Assembly Language
QEC	Quantum Error Correction
QEM	Quantum Error Mitigation
QOA	Quantum Optimization Algorithm
QPT	Quantum Process Tomography
QPU	Quantum Process Unit
QUBO	Quadratic Unconstrained Binary Optimization
QV	Quantum Volume
RB	Randomized Benchmarking
SEM	Standard Error of the Mean
SMT	Satisfiability Modulo Theories
SP	Success Probability
SWAPNK	Swap Network
TAPT	Topology-Aware Pre-Transpilation
UDD	Uhrig Dynamical Decoupling
VQA	Variational Quantum Algorithm
VQE	Variational Quantum Eigensolver
ZNE	Zero Noise Extrapolation

Chapter 1

Introduction

Quantum computing utilizes the principles of quantum mechanics, such as superposition and entanglement, to provide a significant advantage over classical computing. While classical bits are limited to the states 0 or 1, qubits can exist in a superposition of both states simultaneously. This ability to explore multiple states at once gives quantum computers exceptional computational capabilities. Quantum algorithms have already reached significant milestones, including Shor's efficient integer factoring algorithm [1] and Grover's fast search algorithm for unstructured databases [2]. One particularly promising application for near-term quantum devices is solving optimization problems. Classical methods face challenges when dealing with increasingly complex problems due to the large search spaces involved. Quantum optimization algorithms (QOAs) offer an alternative that could potentially surpass classical approaches [3, 4]. QOAs encode optimization problems into a quantum system and use quantum parallelism to evaluate numerous solutions concurrently. This approach has the potential for exponential speedup in tackling complex optimization tasks and has implications for diverse fields such as finance [5, 6, 7], material science [8, 9, 10, 11], and chemistry [12, 13, 14, 15, 16]. However, near-term quantum devices face significant challenges due to inherent noise and imperfections, such as decoherence and imperfect gate control. These factors contribute to high error rates, which hinder the performance of quantum algorithms [17, 18]. Therefore, it is crucial to mitigate these errors and enhance the resilience of QOAs in order to realize practical benefits before fully fault-tolerant quantum computers become available.

Several studies have extensively examined the negative impact of noise on the performance of QOAs. Farhi et al. [19] introduced the quantum approximate optimization algorithm (QAOA) for solving combinatorial optimization problems but pointed out that its performance is heavily dependent on circuit depth, which is limited by noise. Experimental and theoretical studies have confirmed that quantum noise can reduce the fidelity of the final output state, the target cost function value, and the crucial gradient obtained from the QAOA execution [20]. Similarly, Gilyen et al. [21] proposed an algorithm for calculating the gradient of cost functions in quantum optimization, but its performance is significantly affected by noise in the underlying quantum hardware. In addition to affecting algorithm performance, noise can also impact the decision to use a quantum approach. Moussa et al. [4] presented a machine learning framework for selecting between QOAs and classical heuristics in optimization tasks. The level of noise in the available quantum devices is a critical

factor in this selection process. Highlighting this dependence, Stilck et al. [18] compared classical and quantum algorithms for optimization problems. Their findings suggest that achieving significant quantum advantage depends on either reducing noise rates or tailoring optimization problems to the specific hardware topologies.

Improving the resilience of QOAs and effectively mitigating the detrimental effects of noise and errors are of paramount importance to fully exploit the potential of QOAs. Fontana et al. [22] simulated the effects of noise on variational quantum algorithms (VQAs) and found that circuits with redundant parameterized gates are more resilient to noise. Endo et al. [23] proposed a protocol for evaluating the effects of errors and designing efficient error mitigation circuits for quantum algorithms. In a practical implementation, Kandala et al. [24] successfully demonstrated an error mitigation protocol on a superconducting quantum processor, significantly improving computational capabilities without hardware modifications. To further enhance error mitigation techniques, Huggins et al. [25] introduced a near-term friendly strategy called virtual distillation. This strategy utilizes entanglement and measurement of copies of a noisy state to reduce errors. Maciejewski et al. [26] investigated the impact of readout noise on the performance of QAOA and showed that error mitigation techniques significantly improve optimization quality. Building on these advancements, LaRose et al. developed Mitiq [27], a Python package specifically designed for error mitigation on noisy quantum computers. Mitiq offers various error mitigation methods such as zero-noise extrapolation (ZNE) and probabilistic error cancellation (PEC). Despite these valuable contributions, there is still a need for a systematic methodology to comprehensively evaluate and enhance the resilience of QOAs. Additionally, research on effectively integrating error mitigation techniques into the design and implementation of QOAs is still in its early stages. Addressing these gaps is crucial for unlocking the true power of QOAs on noisy intermediate-scale quantum (NISQ) devices [17].

This dissertation centers on the challenge of enhancing the performance and robustness of QOAs for practical applications on near-term quantum devices. Figure 1.1 provides an overview of our investigation into resilience in quantum computing, which encompasses various interconnected components and processes. Resilience, a primary focus of this research, is significantly influenced by factors such as noise, errors, algorithm structure, compilation strategies, and hardware characteristics. To effectively assess resilience, the first part of this dissertation employs a multifaceted approach involving theoretical analysis, numerical simulations, and real hardware experiments, focusing on quantitatively analyzing the resilience properties of QOAs under different noise conditions, using established error models and resilience metrics to comprehensively assess performance and explore the impact of various influential factors. Based on this comprehensive evaluation, the second part of the dissertation implements tailored strategies to improve the noise resilience of QOAs. This includes optimizing the quantum compilation process, selectively optimizing quantum circuits, investigating factors affecting error mitigation efficiency, and cooptimizing the algorithm implementation with hardware constraints. These advancements will result in more resilient QOAs that can be applied in diverse fields such as finance, machine learning, and chemistry. The third part then demonstrates the performance of QOAs on real-world problem applications, highlighting the importance of parameter optimization and the trade-off between performance and noise impact in realistic

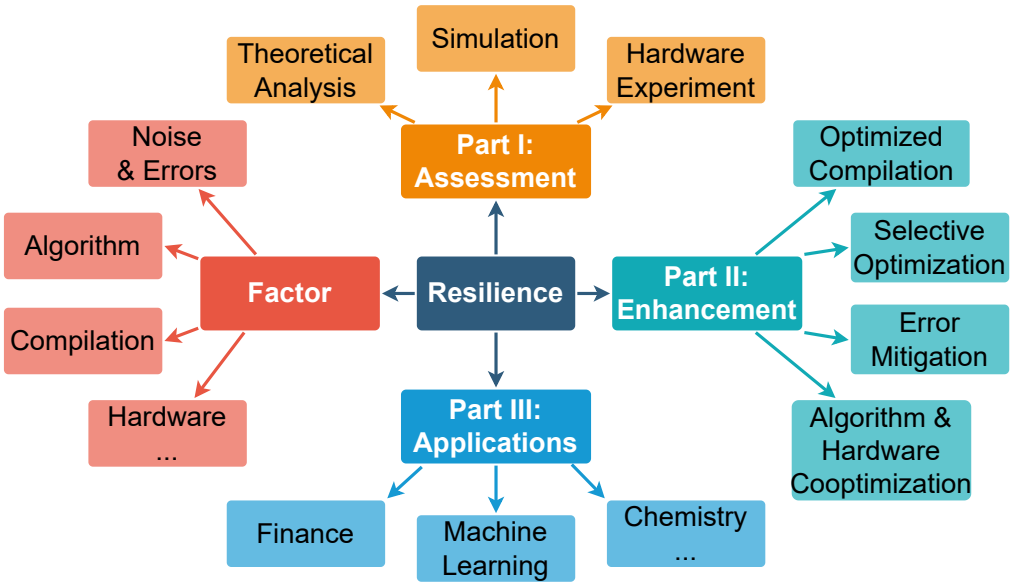

Figure 1.1: A schematic illustration of components involved in investigating and enhancing resilience in quantum computing, encompassing factors influencing resilience, assessment methods, enhancement strategies, and potential applications across diverse fields.

scenarios. The goal of this dissertation is to provide methodologies and techniques that enhance the performance, reliability, and robustness of QOAs on near-term quantum devices, enabling their potential in practical applications. By thoroughly investigating the resilience of QOAs, this research significantly contributes to the development of robust and valuable QOAs for real-world problems, showcasing their potential to bridge the gap between theoretical advancements and transformative impacts in various application domains.

The main contributions of this dissertation are summarized as follows.

- **Identification of Key Factors for Robust Quantum Algorithms**: We investigate the performance of QOAs under various noise conditions to identify the key factors that impact their resilience. By conducting simulations and real-device demonstrations, we gain valuable insights for developing robust algorithms that can function effectively even in noisy environments.
- **Reliable and Efficient Cross-Layer Compilation for VQAs**: We introduce a novel cross-layer circuit compilation framework specifically designed for near-term quantum devices. Our framework prioritizes minimizing noise and errors during compilation, enabling the reliable and efficient execution of VQAs. Comprehensive benchmarking experiments on real hardware demonstrate the effectiveness of our approach, contributing to the efficient implementation of algorithms on NISQ devices.
- **Scalable and Optimal Qubit Mapping Solutions for VQAs on Diverse Topologies**: A key contribution of this dissertation is the development of the algorithm-

oriented qubit mapping (AOQMAP) approach. This methodology provides scalable and optimal solutions for mapping VQAs with fully connected two-qubit interactions onto diverse topologies including linear, T-shaped, and H-shaped. Our approach is applicable to any number of qubits and VQA layers, offering significant flexibility for implementing VQA on various hardware configurations. Additionally, we propose solutions for VQAs with partially connected gates by optimizing the initial qubit order. This comprehensive approach significantly reduces the impact of errors inherent in executing VQAs on actual devices.

- **Selective Optimization of Quantum Circuits for Performance Improvement and Resource Reduction**: We introduce a novel approach to selective optimization for quantum algorithms, demonstrating its implementation on quantum hardware for the first time. Our technique strategically focuses on optimizing only the two-qubit gates within algorithms implemented by low-fidelity native gates. This targeted approach significantly reduces the possible resource consumption for gate calibration while improving performance. This advancement holds significant promise for the development of large-scale quantum computation, bringing us closer to achieving quantum advantage.

- **Synergistic Error Mitigation for High-Performance Quantum Algorithms**: We present a novel investigation into the potential synergies between dynamical decoupling (DD) error mitigation and optimized circuit design. Our investigation demonstrates that strategically incorporating DD sequences within alternative gate implementations and circuit optimization techniques significantly enhances algorithm performance and robustness. These findings open new avenues for advancing quantum computing and enabling the development of high-fidelity and robust quantum algorithms on near-term devices.

- **Cooptimization of Algorithm Design and Hardware Implementation**: We tackle the challenge of optimizing both the algorithm design and hardware implementation for digitized counterdiabatic quantum optimization algorithms on near-term quantum devices. Traditionally, these algorithms are optimized solely at the algorithmic level. Our approach integrates the previously developed AOQMAP strategy with parameter optimization and gate sequence optimization strategies, leading to substantial improvements in performance, resilience, and the ability to leverage diverse error mitigation techniques. This holistic approach, addressing challenges across all layers of the quantum software stack, unlocks the true potential of QOAs on NISQ devices.

- **Advancing QOAs for Real-World Impact**: We further advance the applicability of QOAs to real-world problems by investigating parameter settings in problem instances and resilience factors from both application and algorithmic perspectives. Through in-depth case studies, we demonstrate the crucial role of parameter optimization in problem instances. We highlight the trade-off between achieving a higher success probability with deeper circuit depths and the increased impact of noise in realistic scenarios, emphasizing the importance of the strategies we previously developed to overcome them.

This dissertation is organized into three main parts and ten chapters as outlined below.

Chapter 2 provides an overview of the theoretical foundations relevant to QOAs, including essential concepts in quantum computing, classical and quantum optimization algorithms, and resilience concepts in quantum algorithms.

Part I: Resilience Assessment

Chapter 3 introduces quantified resilience metrics and utilizes various error models to evaluate the noise resilience of QOAs. By assessing the algorithm's performance under various noise conditions, we analyze factors impacting its robustness, such as the number of measurement shots and compilation quality.

Part II: Resilience Enhancement Strategies

Chapter 4 focuses on proposing optimized compilation by leveraging the features of algorithms and properties of quantum devices. We develop an efficient compilation process and detail the development of optimal and scalable qubit mapping solutions for VQAs that can be transformed between different quantum devices, followed by a framework to optimize compilation for general algorithms.

Chapter 5 investigates selective optimization strategies with a specific type of quantum hardware known as bipotent architectures. These architectures are characterized by the ability to implement the same logical gate using two distinct native hardware implementations. By selectively optimizing only partial gates within the circuits, we improve the performance and robustness of algorithms. This approach has the potential to pave the way for solutions in larger-scale quantum computing, where efficient resource utilization becomes crucial.

Chapter 6 studies the synergistic effects of the dynamical decoupling (DD) error mitigation strategy and optimized circuit design for maximizing the performance and robustness of algorithms on near term quantum devices. By analyzing hardware and algorithmic factors, we highlight the significance of a holistic approach that leverages hardware features, algorithm design, and error mitigation strategies for high quality and reliable execution of near term quantum algorithms.

Chapter 7 explores performance and robustness enhancements through a cooptimization approach that considers both the algorithm design and hardware implementations. We achieve this by cooptimizing algorithm parameters, gate sequences, and incorporating realistic hardware constraints. Extensive experiments on real quantum devices demonstrate the significant advantages of our cooptimization approach compared to traditional methods.

Part III: Practical Applications

Chapter 8 delves into the practical applications of QOAs. We investigate the challenges associated with QOAs and strategies for adapting real-world problems into a format suitable for QOA implementation. Using portfolio optimization as a case study, we analyze the robustness of QOAs under varying problem parameters and algorithm configurations.

Chapter 9 concludes the dissertation by summarizing the key research findings, contributions, and implications. It also discusses promising future directions for building on this research to further advance resilient quantum algorithms.

Chapter 2

Theoretical Foundations

In this chapter, we review the theoretical concepts and technical background essential to analyzing and enhancing the resilience of QOAs. We first cover the basics of quantum computing. We then delve into the landscape of classical and quantum optimization algorithms, followed by a detailed introduction to QAOA. Finally, we explore the resilience in quantum computing, discuss its significance in the NISQ era, present features of NISQ devices, and review techniques to improve resilience.

2.1 Quantum computing basics

This section introduces concepts of qubits, quantum gates, superposition, and entanglement. We discuss the qubit representation using Dirac notation and common single and multi-qubit gates that perform defined operations on qubit states.

2.1.1 Qubits and quantum gates

A qubit is a fundamental unit of quantum information. In contrast to a classical bit, which can only be in one state at a time, the qubit can exist in a superposition of the two base states [28]. Qubits can be physically implemented in a variety of quantum systems, such as quantum dots [29, 30], superconducting circuits [31, 32, 33, 34], and trapped ions [35, 36, 37, 38]. The two qubit states $|0\rangle$ and $|1\rangle$ correspond to two distinguishable quantum states of the physical system, which can be chosen as two energy levels, such as electron spin states and photon polarizations. High control and readout fidelity of the qubit states are essential requirements for any candidate qubit implementation. A qubit state $|\psi\rangle$ can be represented on a Bloch sphere, as show in Figure 2.1. The north and south poles of the Bloch sphere correspond to the basis states $|0\rangle$ and $|1\rangle$, respectively. The point on sphere surface denotes a superposition state of the form

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle\,, \tag{2.1}$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$.

Qubit gates are unitary operations that act on one or more qubits to perform quantum logic operations, which can be categorized into single-qubit gates and multi-qubit gates. Single qubit gates operate on a single qubit, whereas multi-qubit gates act on two or more qubits. Sequences of single-qubit gates can be used to rotate a qubit

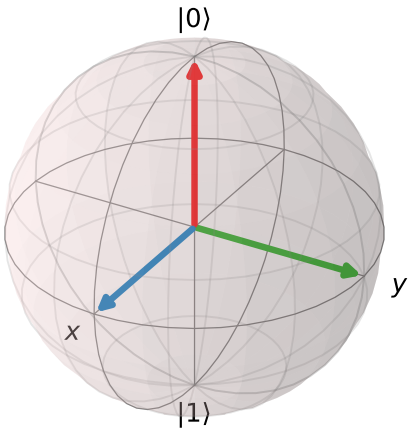

Figure 2.1: Bloch sphere representation of a qubit. The sphere depicts all possible states of a two-level quantum system, with the qubit assigning a point on the surface of the sphere.

to an arbitrary state on the Bloch sphere. The commonly used single-qubit gates are summarized in Table 2.1. The Pauli gates X, Y, and Z correspond to sigma matrices in quantum mechanics. The X gate flips a qubit from $|0\rangle$ to $|1\rangle$ and vice versa, while the Z gate leaves $|0\rangle$ unchanged and flips the phase of $|1\rangle$. The Y gate combines the effects of X and Z gates. The Hadamard gate creates an equal superposition of $|0\rangle$ and $|1\rangle$, while S and T gates induce phase shifts.

Multi-qubit gates are essential for entangling qubits and enabling quantum parallelism, which are both crucial for achieving universal quantum computation. Among these gates, the controlled-NOT (CNOT or CX) gate is a fundamental element for creating and manipulating entangled states. It achieves this by flipping the target qubit only when the control qubit is in the state $|1\rangle$. In contrast, the CZ gate is responsible for flipping the phase of the target qubit when the control qubit is in the state $|1\rangle$. Another significant gate is the CH gate, which applies a Hadamard gate to the target qubit when the control qubit is in the state $|1\rangle$. Additionally, the SWAP gate allows for the exchange of states between two qubits, while the Toffoli gate flips the target qubit if both control qubits are in the state $|1\rangle$.

A universal gate set is a small set of quantum gates that can be combined and repeated to enact arbitrary single-qubit rotations and multi-qubit controlled operations. It can be formed by single qubit gates along with the CX [28], such as {H, S, T, CX}. Universal gate sets are meaningful in quantum computing because they provide a way to approximate any arbitrary unitary operation on a quantum system.

2.1.2 Superposition and entanglement

Quantum mechanics differs fundamentally from classical physics because of two key principles: superposition and entanglement. Superposition allows quantum algorithms to evaluate functions on all inputs simultaneously, while entanglement enables higher information capacity through quantum teleportation and superdense coding. Leveraging superposition and entanglement, quantum information processing can principle outperform classical approaches.

Table 2.1: Common quantum logic gates.

Gate	Matrix	Function
Pauli-X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	Bit-flip: Swaps $\|0\rangle$ and $\|1\rangle$ states
Pauli-Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Bit and phase flip: Swaps $\|0\rangle$ and $\|1\rangle$ and adds a π phase shift
Pauli-Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	Phase flip: Adds a π phase shift to $\|1\rangle$ state
Hadamard	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	Superposition: Puts qubit into equal superposition of $\|0\rangle$ and $\|1\rangle$
Phase S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	Phase shift by $\pi/2$
Phase T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$	Phase shift by $\pi/4$
Controlled-NOT	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	Conditional bit flip: Flips target qubit if control is $\|1\rangle$
Controlled-Z	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$	Conditional phase flip: Applies Z to target if control is $\|1\rangle$
Controlled-H	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$	Applies Hadamard gate to target if control is $\|1\rangle$
SWAP	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	Swaps states of two qubits
Toffoli	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$	Conditional bit flip with two controls

The superposition principle states that quantum systems can exist in a linear combination of multiple states simultaneously. For instance, an electron can be in a superposition of spin up and spin down states, or a photon can be in a superposition of horizontal and vertical polarization. This principle allows quantum systems to exhibit phenomena impossible in classical physics, such as interference, tunneling, and uncertainty. However, a notable characteristic of quantum mechanics is that measurement collapses superpositions into a single eigenstate. Prior to measurement, a system exists in a coherent superposition of multiple states. Upon measurement, it collapses into one of the possible outcomes with a defined probability. This prevents direct observation of superpositions, but interference effects can definitively prove the existence of superpositions.

The second distinctive feature of quantum mechanics is entanglement. Entangled systems share quantum properties, such as spin, momentum, or polarization, in such a way that the measurement of one system affects the outcome of the measurement of another system, even if they are spatially separated. For instance, two electrons can be prepared in one of the four maximally entangled Bell states

$$\left|\Psi^{\pm}\right\rangle = \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle \pm |\downarrow\uparrow\rangle), \tag{2.2}$$

$$\left|\Phi^{\pm}\right\rangle = \frac{1}{\sqrt{2}}(|\uparrow\uparrow\rangle \pm |\downarrow\downarrow\rangle). \tag{2.3}$$

Here the $|\uparrow\downarrow\rangle$ denotes one electron in spin up state and the other in spin down. Measuring one electron's spin instantaneously determines the measurement of the other, regardless of separation. This nonlocal correlation cannot be explained by any classical mechanism. The Bell states form an orthonormal basis for two qubits and exhibit maximum entanglement. They play a vital role in quantum information, underlying quantum teleportation, superdense coding, and quantum cryptography.

Entanglement can also exist between more than two particles. For instance, Greenberger–Horne–Zeilinger (GHZ) states [39, 40] involve three or more particles and are a special type of entangled state where the state of each particle is completely determined by the state of other particles. The simplest 3-qubit GHZ state can be

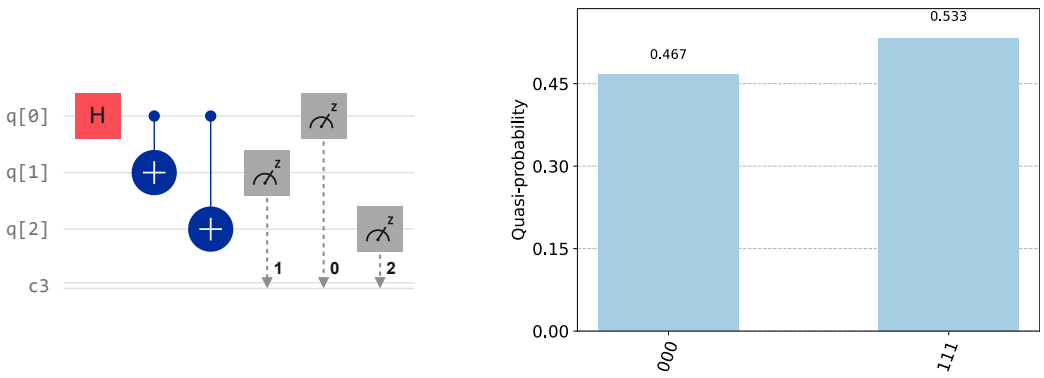

Figure 2.2: Circuit representation of the GHZ state and its simulation using Qiskit's Qasm simulator with 1024 circuit repetitions (shots).

expressed as

$$|\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|\uparrow\uparrow\uparrow\rangle + |\downarrow\downarrow\downarrow\rangle). \tag{2.4}$$

The generation of GHZ state using a quantum circuit is presented in Figure 2.2. Three qubits are first prepared in ground state $|000\rangle$, corresponding to $|\uparrow\uparrow\uparrow\rangle$. Then, a Hardamard gate (H) is applied to the first qubit, followed by two CX gates acting on qubit pairs (0, 1) and (0, 2). Finally, the qubits are measured. We perform simulations using Qiskit, a popular open-source quantum computing software development kit that offers a suite of tools for quantum circuit simulation, optimization, and compilation. Simulating the circuit using Qiskit's Qasm simulator with 1024 repetitions (shots), we obtain the possibility of measuring a $|000\rangle$ state of 0.467 and a probability of measuring the $|111\rangle$ state of 0.533. We observe that the simulation result does not exactly correspond to the theoretical result of 0.5 for each state. This is due to the probabilistic nature of the measurement, also known as shot noise.

2.2 Classical and quantum optimization algorithms

We start by establishing a solid understanding of optimization problems and then delve into well-established classical optimization techniques. Next, we introduce quantum optimization algorithms. Finally, taking portfolio optimization as an example, we introduce the formulation of QAOA for solving it.

2.2.1 Introduction to optimization

Optimization involves finding the best solution from a set of possible solutions, while satisfying constraints [41, 42, 43, 44, 45]. The set of all feasible solutions satisfying the constraints is called the feasible region. The goal of optimization is to find a solution in the feasible region that minimizes or maximizes an objective function, which represents the measure of quality or performance of the solution. Mathematically, an optimization problem can be defined as

$$\begin{aligned} &\text{minimize :} && f(\mathbf{x}), && (2.5)\\ &\text{subject to :} && g(\mathbf{x}) \leq \mathbf{b}, && (2.6)\\ & && h(\mathbf{x}) = \mathbf{0}, && (2.7) \end{aligned}$$

where $f(\mathbf{x})$ is the objective function, which is a scalar function that maps a vector of variables $\mathbf{x}$ to a real number. $\mathbf{x}$ is a vector of variables that must be chosen from a set of feasible solutions. $g(\mathbf{x})$ is a vector of m inequality constraints, where each component $g_i(\mathbf{x})$ must be less than or equal to a constant b_i with $i = 1, ..., m$. $h(\mathbf{x})$ is a vector of k equality constraints, where each component $h_j(\mathbf{x})$ with $j = 1, ..., k$ must be equal to a constant 0. The objective of an optimization problem is to find a vector $\mathbf{x}^*$ that minimizes $f(\mathbf{x})$, subject to the constraints $g(\mathbf{x}) \leq \mathbf{b}$ and $h(\mathbf{x}) = \mathbf{0}$. It is sufficient to focus only on minimization problems when designing optimization algorithms. This is because any maximization problem can be converted into a minimization problem by simply negating the objective function, and then solved using minimization techniques.

Many real-world problems can be formulated as optimization problems, where the goal is to identify the best solution among various possibilities [44]. Optimization problems come in a wide variety of forms, such as finding the shortest path [46, 47], minimal cost flow [48, 49], and optimal resource allocation [50, 51, 52, 53, 54, 55]. Both classical and quantum algorithms aim to solve these problems efficiently.

2.2.2 Classical optimization algorithms

Classical optimization algorithms have been extensively studied and remain fundamental tools across various scientific disciplines [56, 57]. These algorithms can be broadly categorized into two main classes: exact methods and heuristics. Exact methods guarantee finding the global optimum through mathematical proofs, eliminating suboptimal solutions. Examples include dynamic programming [58] and linear programming [59]. However, their applicability is limited to problems with continuous, differentiable, and convex functions. Additionally, most of them have exponential time complexity, which makes them impractical for problems involving nonlinearities or differential equations.

Heuristic algorithms [57] prioritize efficiency by employing techniques such as greedy construction [60], local search [61], and random sampling [62, 63] to find reasonable-quality feasible solutions. While these methods can handle large-scale complex problems, they lack optimality guarantees and can become trapped in local optima, leading to suboptimal solutions. Metaheuristics [64, 65, 66, 67] address this limitation by incorporating higher-level strategies such as randomization to escape local optima and enhance simple heuristics. Examples include simulated annealing [68, 69, 70], tabu search [71, 72, 73], and genetic algorithms [74]. Unlike exact methods, metaheuristics balance diversification and intensification in the search process. While they often discover high-quality solutions, their performance is heavily influenced by the problem structure. Moreover, parameter tuning remains a challenge [75]. In practice, selecting the most suitable optimization technique requires careful consideration of problem characteristics and practical constraints. Real-world applications often necessitate combining different approaches to achieve optimal results.

Optimizers are specialized algorithms designed to find the best solution within given constraints, often through parameter fine-tuning. They can be categorized as gradient-based and gradient-free. Gradient-based optimizers [76, 77] leverage the objective function's gradient to guide the search towards the optimal solution. They iteratively update parameters by following the direction of steepest descent or ascent in the parameter space, utilizing the gradient of the cost function. These methods are prevalent in machine learning due to their efficiency in handling large-scale optimization problems [78]. Common examples include stochastic gradient descent [79], conjugate gradient [80], and limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [81] algorithms. Gradient-free optimizers [82], on the other hand, are employed when the objective function is non-differentiable or its gradient computation is expensive. These methods explore the parameter space by evaluating the objective function at different configurations and adjusting parameters accordingly. Popular examples include genetic algorithms [83], simulated annealing [70], Bayesian opti-

mization [84], constrained optimization by linear approximation (COBYLA) [85], and Nelder-Mead method [86].

2.2.3 Quantum optimization algorithms (QOAs)

QOAs have recently emerged as promising approaches for solving complex optimization problems using principles of quantum mechanics. These problems often involve many variables and constraints, making them difficult to solve using conventional computational methods. QOAs encode optimization problems into a quantum Hamiltonian or circuit with the goal of leveraging quantum effects to find optimal solutions more efficiently. The objective function of a quantum optimization problem is typically a quadratic function, with constraints that can be linear or nonlinear.

Several QOAs have been proposed. Two prominent QOA paradigms are quantum annealing (QA) [87] and variational quantum algorithms (VQAs) [88]. QA leverages quantum effects such as tunneling and superposition to search for the global minimum of an objective function encoding a computational problem. It is based on the principle of adiabatic quantum computing (AQC) [89], where solutions are encoded to the ground states of a problem Hamiltonian H_P. The system starts in the easily initialized ground state of an initial Hamiltonian H_I, which is slowly transformed into H_P, driving the system adiabatically to the ground state of H_P that encodes the solution. The runtime in AQC scales with the inverse minimum energy gap between ground and excited states. QA is similar to AQC, but it allows for transitions out of the ground state by introducing transverse field driving terms in the system Hamiltonian $H(t) = \lambda(t)H_I + [1 - \lambda(t)]H_P$ [90, 91, 89]. Here, $\lambda(t)$ is the annealing schedule that satisfies $\lambda(0) = 1$ and $\lambda(T) = 0$, with T being the final time. This can help the system escape local minima and find the global optimum. Quantum annealing, in contrast to adiabatic optimization, allows for non-adiabatic evolution, and at high temperatures, it exhibits similarities to classical simulated annealing. However, careful tuning of the annealing schedule $\lambda(t)$ is critical.

VQAs, on the other hand, leverage a parameterized quantum circuit $U(\theta)$ acting on an initial state to explore a space of possible solutions to a problem. By measuring outputs and feeding into a classical optimizer adjusting θ, the parameters can be tuned to find the solution. The hybrid quantum-classical loop enables optimization using shallow circuits. Prominent VQAs include the QAOA [19] and the variational quantum eigensolver (VQE) [92]. QAOA applies alternating quantum gates to gradually shape a superposition state towards an approximate solution [19]. Meanwhile, VQE finds the minimum eigenvalue of a problem Hamiltonian using a parameterized circuit tuned by classical optimization [92, 93]. These hybrid algorithms are designed for near-term quantum devices by optimizing shallow circuits in a feedback loop.

While QOAs have the potential to revolutionize the way we solve optimization problems, they are still in the development stage and face challenges such as limited access to quantum computers and sensitivity to noise, which can degrade their performance. Despite these challenges, QOAs have the potential to provide quantum speedups, which could lead to different scaling behavior on certain problems. It is important to note, however, that no asymptotic speedup is currently expected from QOAs. Further research is needed to fully characterize when quantum speedups are

achievable and to compare the strengths and limitations of classical and quantum optimization approaches.

2.2.4 Quantum approximate optimization algorithm (QAOA)

We employ portfolio optimization as a representative problem instance to illustrate the application of QAOA in solving such optimization challenges. This section is based on the paper [94].

Portfolio optimization

Portfolio optimization is the process of selecting a set of assets to invest in order to maximize the expected return while minimizing the risk. This is a challenging problem, as it involves finding the optimal trade-off between risk and return. The classical approach to portfolio optimization is based on Markowitz's [95] mean-variance optimization framework and has been widely used for decades. Markowitz mean-variance optimization uses a linear programming algorithm to optimize portfolio performance. It takes into account the expected returns and covariance matrix of assets to identify the optimal portfolio that minimizes risk while maximizing return. However, this approach has several limitations, such as its reliance on historical data and its inability to handle complex, non-linear relationships between assets. The QAOA has the potential to address this challenge [96, 94, 6]. The objective function in portfolio optimization is typically the expected return of the portfolio, subject to a constraint on the risk.

Problem defination

We define first the portfolio optimization problem. Consider the cost function

$$F(z_1, z_2, \ldots, z_n) = q \sum_{i,j=1}^{n} z_i z_j \sigma_{ij} - (1-q) \sum_{i=1}^{n} z_i \mu_i, \;\; z_i = 0, 1, \tag{2.8}$$

which consists of the portfolio weights $z_i = 0$ or 1 of stock i, the covariance matrix of the stock returns σ_{ij} and the expected return μ_i. Furthermore, n is the number of available assets and the factor q sets the risk preference of the investor: $q = 1$ would be used if the investor is fully risk-averse, i.e., he wants to choose the portfolio with the lowest risk, irrespective of the return. $q = 0$ would be used in case of an aggressive investor who takes only the return of the stocks into account. In real-world applications, however, q is set between 0 and 1.

Usually, an investment strategy consists in investing a fixed budget of money into a portfolio of securities. This budget constraint can be formulated in the binary

Material from: Brandhofer, Sebastian, Daniel Braun, Vanessa Dehn, Gerhard Hellstern, Matthias Hüls, Yanjun Ji, Ilia Polian, Amandeep Singh Bhatia, and Thomas Wellens, Benchmarking the performance of portfolio optimization with QAOA. Quantum Inf Process 22, 25, (2023), ©Springer.

optimization problem as follows:

$$\sum_{i=1}^{n} z_i = B, \tag{2.9}$$

The parameter B defines the number of assets to be chosen for the portfolio and thereby constrains the money invested. In the following, we will refer to those portfolios $z_1, z_2, \ldots, z_n$ which fulfill the constraint as 'feasible' portfolios, whereas the remaining ones will be called 'unfeasible' [97].

In order to solve the portfolio optimization problem, the covariance matrix and the return vector must be provided. Here, we use the following prescription to calculate these values: If the portfolio consists of n assets, we use $m+1$ historical daily prices $p_k^{(i)}$ where $i = 1, 2, \ldots, n$ and $k = 0, 1, \ldots, m$. Out of these values we compute the daily percentage price change via

$$r_k^{(i)} = \left(p_k^{(i)} - p_{k-1}^{(i)}\right) / p_k^{(i)}, 1 \leq k \leq m, \tag{2.10}$$

and the annualized return of the portfolio:

$$\mu_i = \left[\prod_{k=1}^{m} \left(1 + r_k^{(i)}\right)\right]^{\frac{252}{m}}. \tag{2.11}$$

The annualized covariance matrix is calculated with the equation

$$\sigma_{ij} = \frac{252}{m} \sum_{k=1}^{m} \left(r_k^{(i)} - \overline{r^{(i)}}\right)\left(r_k^{(j)} - \overline{r^{(j)}}\right), \tag{2.12}$$

where $\overline{r^{(i)}} = \frac{1}{m}\sum_{k=1}^{m} r_k^{(i)}$ is the mean of the daily price changes of asset i.

In order to fulfill the budget constraint, Eq. 2.9, a penalty term can be added to the function F as follows:

$$F^{(A)}(z_1, \ldots, z_n) = F(z_1, \ldots, z_n) + A\left(\sum_{i=1}^{n} z_i - B\right)^2. \tag{2.13}$$

The prefactor A should be chosen large enough such that, on the one hand, all unfeasible states (i.e., those which do not fulfill the constraint) yield $F^{(A)} > F_{\min}$. On the other hand, if A is chosen too large, the performance of QAOA is expected to deteriorate. It has been suggested to choose A such that all unfeasible states yield $F^{(A)} > F_{\max}$ [97, 96].

QAOA formulation

To solve the above optimization problem on a quantum computer, we convert the function $F^{(A)}$ into a quantum operator, also known as the problem Hamiltonian or

cost Hamiltonian

$$\hat{H}_c = \lambda \, F^{(A)} \left(\frac{\hat{I}_1 + \hat{Z}_1}{2}, \ldots, \frac{\hat{I}_n + \hat{Z}_n}{2} \right), \tag{2.14}$$

where $\hat{I}_j$ and $\hat{Z}_j$ denote the identity and the Pauli-$\hat{Z}$ operator, respectively, acting on qubit j. Additionally, we scale the function by a constant factor $\lambda > 0$, the significance of which will be explained below. Taking into account Eqs. 2.8 and 2.13, we can write $\hat{H}_c$ as a sum of two-qubit and single-qubit operators (plus an irrelevant constant term c)

$$\hat{H}_c = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} W_{ij} \hat{Z}_i \hat{Z}_j - \sum_{i=1}^{n} w_i \hat{Z}_i + c, \tag{2.15}$$

where $W_{ij} = \frac{\lambda}{2}(q\sigma_{ij} + A)$ and $w_i = \frac{\lambda}{2}\left[(1-q)\mu_i + A(2B-n) - q\sum_{j=1}^{n}\sigma_{ij}\right]$.

The QAOA variational quantum circuit then generates the following quantum state $|\psi_{\vec{\gamma},\vec{\beta}}\rangle$ depending on the parameters $\vec{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\vec{\beta} = (\beta_1, \ldots, \beta_p)$ (with number of iterations p, also referred to as 'QAOA depth' in the following):

$$|\psi_{\vec{\gamma},\vec{\beta}}\rangle_M = \hat{U}_M(\beta_p) e^{-i\gamma_p \hat{H}_c} \ldots \hat{U}_M(\beta_2) e^{-i\gamma_2 \hat{H}_c} \hat{U}_M(\beta_1) e^{-i\gamma_1 \hat{H}_c} |\psi_0\rangle_M, \tag{2.16}$$

where the initial state $|\psi_0\rangle_M$ and the operator $\hat{U}_M(\beta)$ depend on the choice of the mixer M, see below. Finally, all qubits are measured with respect to the standard basis in order to determine the mean value

$$E = \langle \psi_{\vec{\gamma},\vec{\beta}} | \hat{H}_c | \psi_{\vec{\gamma},\vec{\beta}} \rangle. \tag{2.17}$$

This information is then passed to a classical optimization routine, which suggests new values for the parameters $\vec{\gamma}$ and $\vec{\beta}$ in order to minimize the expectation value E. Similar approaches, where optimized parameters of a quantum circuit are found in a hybrid quantum-classical process, are also used in other fields such as quantum circuit learning [98] or Hamiltonian learning [99].

Different mixers

The standard mixer is defined by single-qubit rotations applied to each qubit:

$$\hat{U}_{\text{standard}}(\beta) = e^{i\beta \sum_{i=1}^{n} \hat{X}_i}. \tag{2.18}$$

The corresponding initial state is $|\psi_0\rangle_{\text{standard}} = |+\cdots+\rangle$, where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. This state is the eigenstate of the mixing operator $\hat{H}_m = -\sum_i \hat{X}_i$, also referred to as mixer Hamiltonian, associated to its smallest eigenvalue.

Alternatively, the initial state and the mixer can be adapted such that only states $|\boldsymbol{z}\rangle = |z_1, \ldots, z_n\rangle$ fulfilling the budget constraint, see Eq. 2.9, are populated during the algorithm [100, 101]. Correspondingly, the initial state is chosen as a uniform

superposition (Dicke state [102])

$$|\psi_0\rangle_{M_{XY}} = |D_B^n\rangle = \frac{1}{\sqrt{\binom{n}{B}}} \sum_{\substack{i_1,\ldots,i_n=0,1 \\ i_1+\cdots+i_n=B}} |i_1 \ldots i_n\rangle. \tag{2.19}$$

This state is an eigenstate of the operator $(\hat{X}_i\hat{X}_j + \hat{Y}_i\hat{Y}_j)$ (for all pairs of qubits i and j), which essentially swaps two qubits populating different states (i.e., $|01\rangle \leftrightarrow |10\rangle$), thereby keeping the total budget (or number of excitations) constant. Therefore, the basic ingredient of the XY-mixers is given by the following two-qubit operation:

$$\hat{R}_{i,j}^{(XY)}(\beta) = e^{i\beta(\hat{X}_i\hat{X}_j + \hat{Y}_i\hat{Y}_j)}. \tag{2.20}$$

Different versions M_{XY} of the XY-mixer are now obtained by applying this operation to different sets $S_{M_{XY}}$ of qubit pairs:

$$\hat{U}_{M_{XY}}(\beta) = \prod_{(i,j)\in S_{M_{XY}}} \hat{R}_{i,j}^{(XY)}(\beta). \tag{2.21}$$

We consider four versions of XY-mixers as follows:

(i) **Ring mixer** ($M_{XY} = \text{ring}$)

In the *ring mixer*, the operation $\hat{R}_{i,j}^{(XY)}(\beta)$ is applied only to neighbouring pairs of qubits, i.e., $S_{\text{ring}} = \{(1,2),(2,3),\ldots,(n-1,n),(n,n+1)\}$, where qubit $n+1$ is identified with qubit 1. Note that, since $[\hat{R}_{i,j}^{(XY)}, \hat{R}_{j,k}^{(XY)}] \neq 0$ for mutually distinct qubits i, j and k, the ordering in which the individual operations $\hat{R}_{i,j}^{(XY)}$ are applied in Eq. 2.21 is relevant. It is given by the order in which the corresponding qubit pairs appear in the above set S_{ring}, i.e., first (1,2), then (2,3) etc.

(ii) **Parity ring mixer** ($M_{XY} = \text{par\_ring}$)

In the *parity ring mixer* , this ordering is changed such as to obtain two subsets of commuting operations $\hat{R}_{i,i+1}^{(XY)}$ with odd or even i, respectively, i.e., $S_{\text{par\_ring}} = \{(1,2),(3,4),\ldots,(n_o,n_o+1),(2,3),(4,5),\ldots,(n_e,n_e+1)\}$, where n_e (or n_o) is the largest even (or odd) number $n_e \leq n$ (or $n_o \leq n$) (and, again, qubit $n+1$ is identified with qubit 1).

(iii) **Full mixer** ($M_{XY} = \text{full}$)

In the *full XY-mixer* ($M_{XY} = \text{full}$), the set S_{full} contains *all* pairs of qubits, where the ordering is chosen such that as many gates as possible can be performed in parallel, thus minimizing the depth of the circuit. If n is odd, the $\hat{R}_{i,j}^{(XY)}$'s are arranged into n subsets of $(n-1)/2$ commuting operations, where $i+j \bmod n = k$ in subset k. (E.g., for $n=5$: $S_{\text{full}} = \{(1,5),(2,4),(2,5),(3,4),(2,1),(3,5),(3,1),(4,5),(3,2),(4,1)\}$ with subsets $\{(1,5),(2,4)\}$, $\{(2,5),(3,4)\}$ etc.) If n is even, we first generate the subsets for $n-1$ as described above, and add the missing pair of qubits to each subset. (For example, for $n=6$, we add $(3,6)$ to the first subset $\{(1,5),(2,4)\}$, $(1,6)$ to the second, etc., i.e., $S_{\text{full}} = \{(1,5),(2,4),(3,6),(2,5),(3,4),(1,6),\ldots\}$.)

Together with the phase separation operator $e^{-i\gamma\hat{F}}$, a single iteration step in the full XY-mixer QAOA algorithm, see Eq. 2.16, reads as follows:

$$\hat{U}_{\text{full}}(\beta)e^{-i\gamma\hat{F}} = \prod_{(i,j)\in S_{\text{full}}} \hat{R}_{i,j}^{(XY)}(\beta) \prod_{(i,j)\in S_{\text{full}}} e^{-i\gamma W_{ij}\hat{Z}_i\hat{Z}_j} \prod_{i=1}^{n} e^{-i\gamma w_i \hat{Z}_i}. \tag{2.22}$$

(iv) **Quantum alternate mixer-phase ansatz** (M_{XY} = QAMPA)

Finally, we modify the full mixer according to the recently proposed quantum alternate mixer-phase ansatz (QAMPA) [103]. Instead of applying the operator $e^{-i\gamma\hat{F}}$ to all qubits before applying the mixer, we reorder the terms such that, for each pair of qubits, the gates corresponding to the mixer and the phase separation are contracted as follows:

$$\hat{U}_{\text{MP}}(\beta,\gamma) = \prod_{(i,j)\in S_{\text{full}}} e^{i\beta(\hat{X}_i\hat{X}_j+\hat{Y}_i\hat{Y}_j)-i\gamma W_{ij}\hat{Z}_i\hat{Z}_j} \prod_{i=1}^{n} e^{-i\gamma w_i\hat{Z}_i}. \tag{2.23}$$

Thereby, the QAOA circuit, see Eq. 2.16 is modified as follows:

$$|\psi_{\vec{\gamma},\vec{\beta}}\rangle_M = \hat{U}_{MP}(\beta_p,\gamma_p)\ldots\hat{U}_{MP}(\beta_2,\gamma_2)\hat{U}_{MP}(\beta_1,\gamma_1)|\psi_0\rangle_{M_{XY}}. \tag{2.24}$$

This reduces the number of CX gates by a factor 3/4. Since the CX gates constitute the most important source of errors in present NISQ devices, we may expect that this version is more resilient against errors than the full XY mixer.

While the standard mixer $\hat{H}_m$ can be implemented using single-qubit gates and the corresponding initial state $|\psi_0\rangle_{\text{standard}}$ is easily prepared with Hadamard gates, XY-mixers necessitate the implementation of two-qubit gates, and the corresponding initial state, the Dick state, requires numerous two-qubit gates to prepare. These additional two-qubit gates introduced by XY-mixers and the Dick state significantly increase noise, as two-qubit gates are the primary noise source in current quantum hardware. Therefore, in Part II, we focus on the standard mixer of QAOA and explore the effects of different mixers in Part III.

2.3 Resilience in quantum computing

In the previous section, we established the fundamentals of quantum computing and introduced optimization algorithms. We now discuss the significance of resilience, introduce the features of near term quantum devices, and present an overview of techniques for improving resilience in quantum computing.

2.3.1 State-of-the-art

Resilience refers to the ability of quantum systems to operate reliably in the presence of noise and control imperfections, which is a crucial aspect in realizing the full potential of quantum computation, communication, and other applications. The foundational work conducted in this field established the theoretical feasibility of

fault-tolerant quantum computation [104, 105, 106, 107, 108]. The threshold theorem states that a quantum computer with a physical error rate below a certain threshold (10^{-6}) can achieve arbitrarily low logical error rates by employing quantum error correction (QEC) schemes [109, 110]. This fundamental idea has motivated extensive research on optimized QEC codes and fault-tolerant techniques for achieving accurate quantum computation in the presence of noise.

Significant progress has been made in reducing gate error rates across various quantum computing platforms, with superconducting and trapped-ion qubits emerging as leading contenders. Superconducting qubits have achieved state-of-the-art two-qubit gate fidelities exceeding 99.9% [111, 112], corresponding to an error probability of approximately 0.1% per operation. Single-qubit fidelities have even reached 99.99% [113]. Notably, a recent research by Google Quantum AI demonstrated error rates below the critical threshold required for effective quantum error correction [114], a crucial milestone on the path to building scalable quantum computers. On the other hand, trapped-ion qubits offer advantages such as long coherence times and high-fidelity gates. Recent demonstrations have showcased fidelities exceeding 99.9% for two-qubit gates and 99.99% for single-qubit logic gates driven by lasers [115].

While fault-tolerant quantum computation remains the ultimate objective, achieving these low error rates presents a significant challenge. In the NISQ era, error mitigation techniques can play a crucial role in enhancing the resilience of quantum algorithms. A recent experiment involving a 127-qubit IBM quantum computer successfully tackled a complex model system beyond the capabilities of classical computation by leveraging error mitigation [116]. This achievement underscores the importance of error mitigation strategies alongside hardware development in unlocking the potential of near term quantum algorithms.

2.3.2 Noisy intermediate-scale quantum (NISQ) devices

Near-term quantum devices [17] are prone to errors that can significantly impact their accuracy and reliability. These errors can stem from various factors, including environmental noise, imperfect control of quantum gates, and interactions between qubits. Furthermore, NISQ devices are characterized by limited connectivity and a restricted number of available quantum gates (basis gates). In this section, we discuss different types of errors, techniques used to characterize them, and the key features of NISQ devices.

Error sources

Quantum computations are susceptible to various error sources that can compromise the fidelity of encoded information [117]. These errors can be broadly categorized into coherent and incoherent errors based on their impact on the quantum state [118, 119]. Coherent errors, stemming from systematic control imperfections or crosstalk between qubits, can be described by unitary rotations leading to deviations from the intended quantum operations [118, 119, 120, 121]. Examples include over- or under-rotations due to inaccurate microwave pulses and erroneous gate operations from imprecise pulse sequencing. These can be mitigated through optimization of pulse shapes [122, 123] and meticulous timing calibrations [124, 125]. Careful

engineering and calibration of qubit couplings are also crucial for high-fidelity two-qubit gates and crosstalk suppression [126, 127]. In contrast to coherent errors, incoherent errors arise from stochastic processes that introduce random fluctuations in the qubit state. Examples of incoherent errors include depolarizing noise, bit-flip error, and phase-flip error, which can be mathematically described using Pauli matrices. Finally, quantum computations are inherently susceptible to measurement error due to the probabilistic nature of quantum mechanics [128], which is unavoidable even with a perfect quantum computer operating under ideal conditions. Various mitigation strategies have been developed to mitigate errors. An overview of error mitigation can be found in [129].

Noise modeling and error characterization

Accurate noise modeling and error characterization are fundamental for developing efficient error mitigation techniques in quantum computation. Noise modeling is the process of building an error model that can mathematically describe the detrimental effects of various noise sources on qubit states and quantum gates. Error models can be categorized on the Markovian property. Markovian models [130] assume noise processes are memoryless, simplifying analysis but potentially neglecting non-Markovian effects observed in real hardware [131]. Conversely, non-Markovian models [132] capture correlations and memory effects in noise, providing a more accurate description. Advanced models employing the post-Markovian master equation [133, 134, 135] can further enhance model fidelity by better matching the observed experimental behavior of quantum devices [132]. In Section 3.1.2, we will present several commonly used error models that describe specific types of noise, which are then used to assess the performance of quantum algorithms in the presence of noise.

Error characterization involves identifying and quantifying errors through various techniques. Randomized benchmarking (RB) is a particularly effective tool for characterizing coherent noise affecting quantum gates by providing an average gate fidelity metric [118, 136, 137]. However, RB offers an incomplete description of noise dynamics. Conversely, quantum process tomography (QPT) reconstructs a complete picture of noise dynamics but suffers from scalability limitations for large-scale systems [138]. Several error characterization protocols have been developed. Wood et al. [139] introduced common noise sources in superconducting qubits and described techniques for characterizing and mitigating associated errors. Magesan et al. [136] introduced a scalable RB protocol for benchmarking quantum gates and estimating noise dependence. Magesan et al. (2012) [137] further developed an efficient protocol for measuring quantum gate error through interleaved randomized benchmarking. Sud et al. [140] presented a dual-map framework for error characterization, enabling the derivation of expectation values of observables in noisy circuits. Gupta et al. [141] introduced an autonomous learning framework for adaptive scheduling of sensor-qubit measurements and efficient spatial noise mapping, demonstrating advantages over brute-force approaches. Additionally, a general framework for QPT in the presence of time-correlated noise has been proposed to enhance existing characterization protocols [142]. Finally, innovative methods based on tensor networks offer a promising alternative approach to traditional protocols for characterizing errors on near-term quantum computers [143]. These characterization protocols play a crucial

role in validating hardware performance, identifying dominant error mechanisms, benchmarking improvements in noise mitigation strategies, and ultimately guiding approaches for enhancing qubit fidelities. However, error characterization remains a significant challenge for large-scale quantum systems, where current methods such as process tomography become impractical.

Features of NISQ devices

Superconducting quantum devices, a prominent type of NISQ computers operating with superconducting qubits [32, 144], exhibit unique characteristics that influence algorithm design and execution. This section explores their features using IBM quantum devices.

Figure 2.3 showcases the topologies of representative IBM quantum devices with varying qubit counts (27, 127, and 133). Each circle represents a qubit, and connecting lines depict possible interactions. Notably, these devices possess limited qubit connectivity, implying not all qubits can interact directly. The color scheme indicates gate error rates. Note that they fluctuate over time. These limitations in connectivity and fluctuating error rates necessitate careful consideration when developing quantum algorithms for NISQ devices.

Figure 2.4 is a screenshot displaying the specific properties of ibmq_ehningen. This 27-qubit device boasts a quantum volume (QV) [145] of 64 and executes circuits at a rate of 1900 circuit layer operations per second (CLOPS) [146]. QV quantifies a quantum device's capabilities by measuring the largest square (width equals depth) quantum circuit that can be executed successfully [145]. CLOPS assesses the speed of quantum devices by determining the number of quantum volume circuits it can execute per unit of time [146]. The supported gates include single-qubit gates (ID, R_Z, SX, and X) and the CX (controlled-NOT) two-qubit gate. Here, ID is identity gate, R_Z performs a single qubit rotation around the z-axis, X is the NOT gate, and SX is the square root of X. Notably, two-qubit gate errors are significantly higher (around 10^{-3}) compared to single-qubit gate errors (around 10^{-4}). Moreover, the readout error is around 10^{-2}. This screenshot also presents the decoherence times (T_1 and T_2), which characterize how long a qubit retains its quantum information.

Figure 2.5 offers a granular view of error rates for various single-qubit gates and measurements in ibmq_ehningen. These errors exhibit qubit-specific variations. For instance, qubit 7 experiences the highest single-qubit gate error (Figure 2.5(a)), while qubit 17 suffers from the most significant readout error (Figure 2.5(b)) and has the highest probability of misreading a state $|1\rangle$ (Figure 2.5(c)). Conversely, qubit 9 has the highest chance of being misread when prepared in $|0\rangle$ (Figure 2.5(d)).

Decoherence times, characterized by relaxation time (T_1) and dephasing time (T_2), are critical parameters for NISQ devices. These times represent the duration for which a qubit can maintain its quantum state before succumbing to environmental noise. For successful quantum information processing, T_1 and T_2 should be significantly longer than the gate operation time (the time it takes to perform a single quantum logic gate) on the qubits. Figure 2.6 illustrates T_1 and T_2 values for each qubit in ibmq_ehningen. Different qubits exhibit distinct decoherence times. In this case, qubit 7 experiences the shortest T_1 and T_2, while qubit 25 demonstrates the longest T_1 and T_2. It's important to note that these values fluctuate over time.

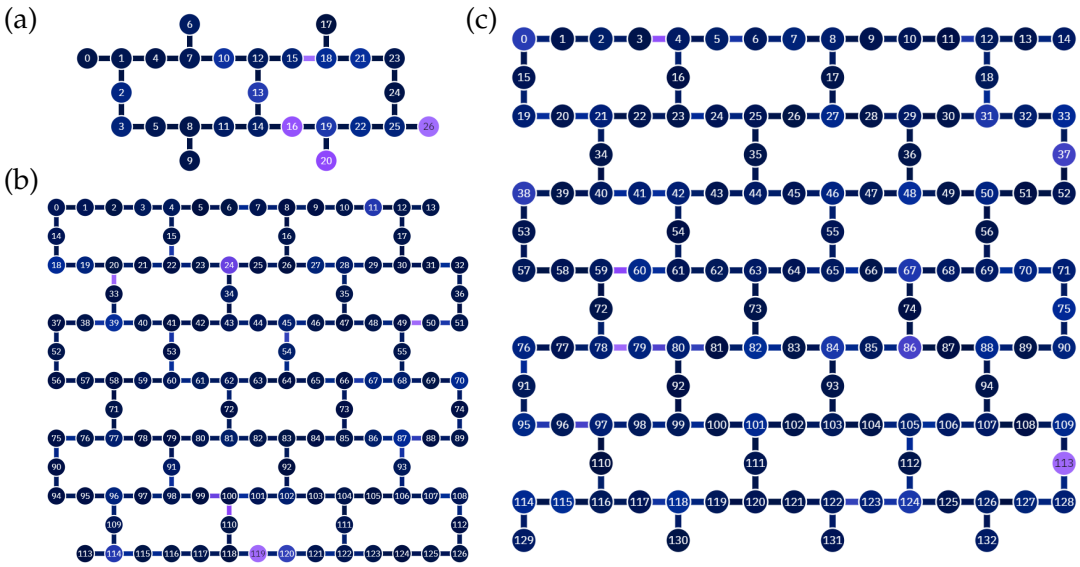

Figure 2.3: Topologies of IBM quantum devices with (a) 27 qubits, (b) 127 qubits, and (c) 133 qubits. Color indicates the error rate (see Figure 2.4).

Two-qubit gate errors are a dominant source of errors in NISQ devices, posing a significant challenge for achieving accurate quantum computations. As shown in Figure 2.7, the CX gate error rate on ibmq_ehningen can be around 10^{-3}, comparable to readout error rates. However, quantum algorithms typically involve a significantly higher number of CX gates compared to readouts, which are usually measured only once at the end. Additionally, SWAP gates, which are commonly used to address limited qubit connectivity in NISQ devices, introduce further noise themselves, as each SWAP gate needs to be implemented with three CX gates. This significantly increases the overall number of two-qubit operations in the circuit, further exacerbating the impact of gate errors. Moreover, variations in error rates and gate execution times exist across different qubits within the same device. Therefore, optimizing the use of qubits based on their individual properties and the specific structure of the quantum algorithm becomes crucial for improving performance on NISQ devices.

2.3.3 Key strategies for improving resilience

This section discusses strategies to improve the performance of quantum algorithms on NISQ devices without fully-fledged quantum error correction. While achieving fault-tolerant quantum computation with robust error correction is a long-term goal, practical applications can benefit from alternative approaches. We focus on error mitigation and resilience-optimized compilation strategies.

Error mitigation

The errors in NISQ devices can significantly impact the accuracy and reliability of computations, limiting their practical applications. To address this challenge, many error mitigation techniques have been proposed. Here, we introduce several error

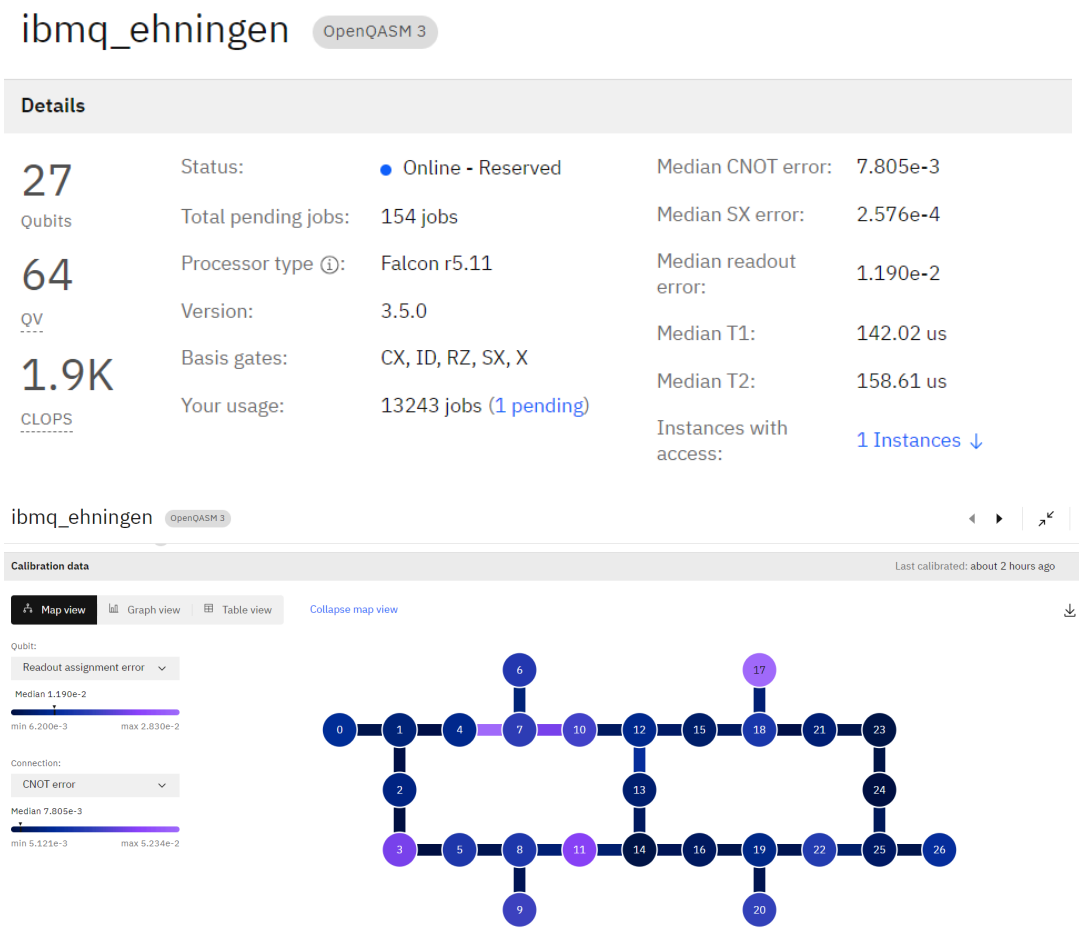

Figure 2.4: Properties of ibmq_ehningen (screenshot). The system comprises 27 qubits and has a quantum volume (QV) [145] of 64. It is capable of performing 1900 circuit layer operations per second (CLOPS) [146]. The basis gate set is {CX, ID R_Z, SX, X}. Notably, the two-qubit gate (CX) exhibits an error rate on the order of 10^{-3}, which is an order of magnitude greater than that of the single-qubit gates {ID, SX, X}. The R_Z gate is a virtual gate, implying that it can be implemented in hardware through frame changes, thereby resulting in zero error and duration. The decoherence times, T_1 and T_2, are approximately 142µs and 158µs, respectively. The topology's color scheme represents varying error rates. Specifically, the readout error ranges from 6.2×10^{-3} to 2.83×10^{-2}, while the CX gate error rate varies from 5.121×10^{-3} to 5.234×10^{-2}.

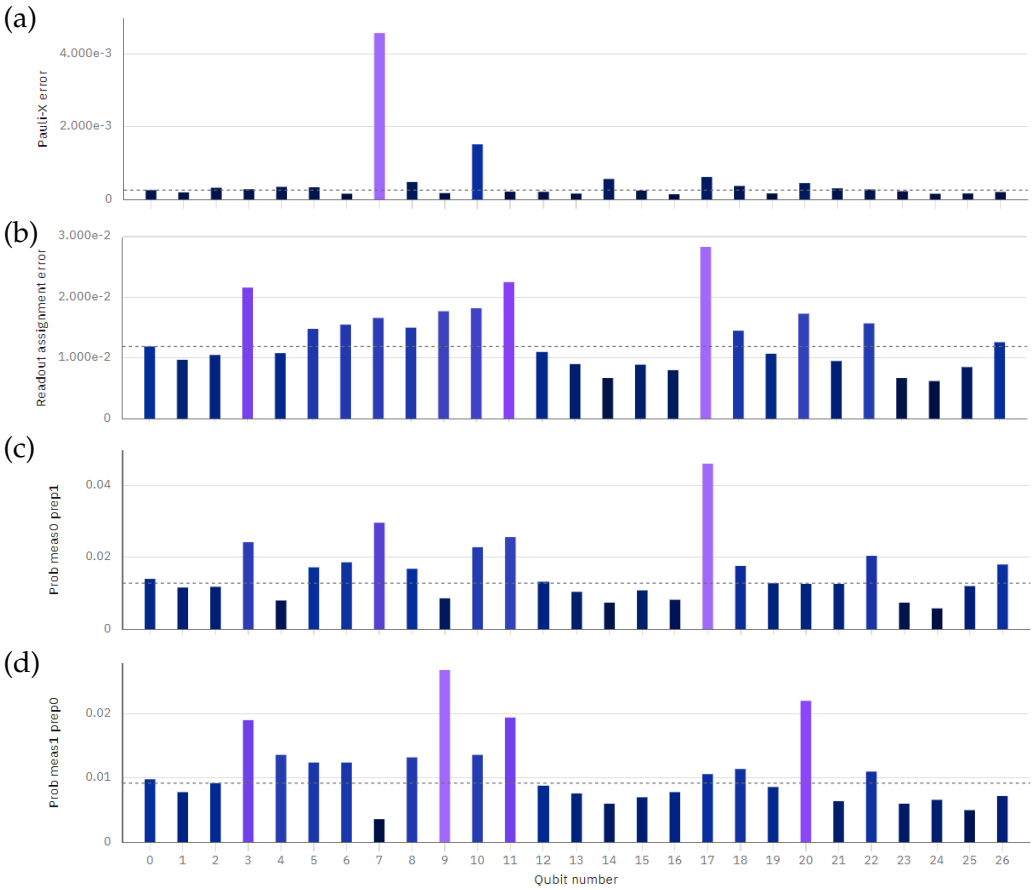

Figure 2.5: Error rates of single qubit gates of ibmq_ehningen: (a) Gate error, (b) readout error, (c) prob meas0 prep1, the probability of measuring state $|0\rangle$ when the qubit was actually prepared in state $|1\rangle$, and (d) prob meas1 prep0, the probability of measuring state $|1\rangle$ when the qubit was initially prepared in state $|0\rangle$. We observe that there is a higher probability of misreading a qubit prepared in the state $|1\rangle$ compared to one prepared in the state $|0\rangle$.

mitigation techniques [147], such as readout error mitigation (REM), dynamical decoupling (DD), and zero-noise extrapolation (ZNE).

Readout errors pose a significant challenge in current quantum systems, with error rates typically higher than single qubit gate errors by approximately one order of magnitude. These errors stem from various sources, including decoherence during the measurement process and the overlapping of measured physical quantities associated with $|0\rangle$ and $|1\rangle$ states. A range of mitigation techniques have been proposed [148, 149, 150, 26, 151, 152, 153, 154, 155, 156, 157, 158], aiming to enhance the precision of quantum state measurements, which typically involve constructing noise models or applying classical postprocessing to measured outcomes.

DD [159, 160, 161, 162] is a widely used error mitigation technique in quantum systems. Its primary objective is to suppress decoherence errors during qubit idle times by applying a sequence of pulses that implement the identity operation. DD is effective in protecting quantum states against noise and can also be utilized to mitigate crosstalk [163, 164] and coherent errors [165]. DD sequences typically consist

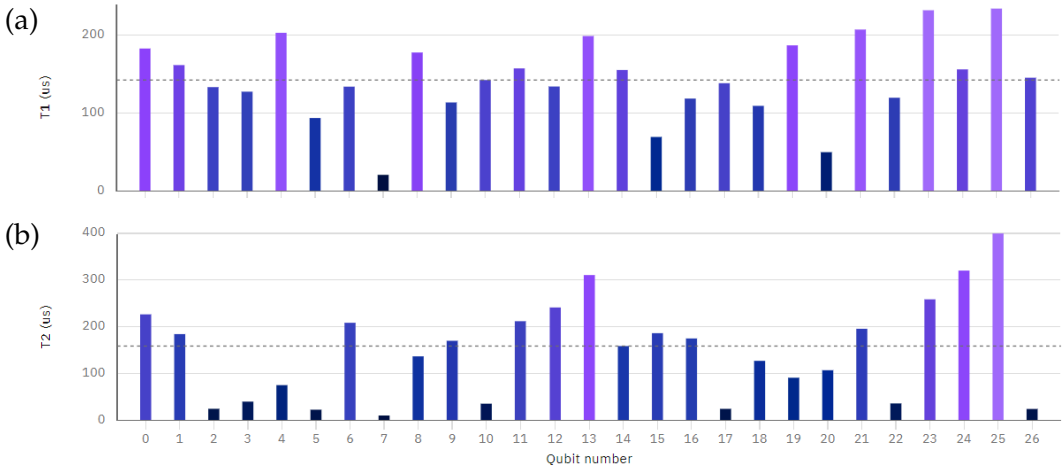

Figure 2.6: Decoherence times of ibmq_ehningen: (a) T_1, and (b) T_2. The relaxation time T_1 refers to the time it takes for an excited quantum state to decay towards the ground state. The dephasing time T_2 describes how long the phase of a qubit's superposition state remains intact.

of repetitive patterns of pulses, such as π pulses, which counteract the detrimental effects of system and environment coupling. The relative phases and delays between pulses play a crucial role in improving the robustness of DD sequences against pulse imperfections and unwanted environmental interactions. A variety of DD sequences, such as Carr-Purcell-Meiboom-Gill (CPMG) [166, 167], XY4 [168, 169, 170, 171], KDD [172], and Uhrig dynamical decoupling (UDD) [173], have been investigated and demonstrated promising results in various quantum systems, such as spin systems [174, 175, 176, 177, 178, 179], superconducting qubits [161, 163, 180], trapped ions [181], and quantum memories [182]. Moreover, DD techniques have been applied in simulations of quantum many-body systems [183] and quantum machine learning [184].

ZNE estimates ideal expectation values from noisy data by leveraging measurements taken at different controlled noise levels [185, 186, 23, 24]. It comprises two primary steps: noise scaling and extrapolation. Noise scaling is the process of modifying the quantum circuit to run at different noise levels. This can be achieved through pulse stretching [24] or by applying digital techniques such as gate folding [187]. Extrapolation is the process of fitting a mathematical model to noisy data and estimating the zero-noise value using different methods such as linear, polynomial, exponential, or adaptive extrapolation. A similar approach to ZNE is the probabilistic error cancellation (PEC) [186, 23, 188, 189], which also aims to reduce the impact of noise by sampling from a collection of circuits that emulate a noise inverting channel, effectively canceling out the noise.

Machine learning (ML) techniques are increasingly being used to mitigate errors in quantum computations [190, 191, 192, 193, 194]. These techniques have the potential to reduce overhead and maintain, or even surpass, the accuracy of conventional methods. For example, Kim et al. [190] introduced a neural network-based error mitigation technique that adjusts estimated measurement probabilities to improve

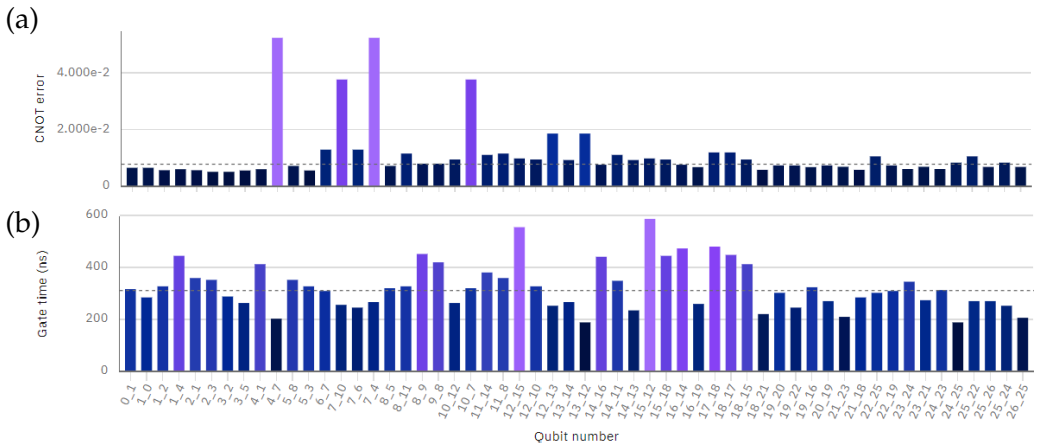

Figure 2.7: CX gate properties of ibmq_ehningen: (a) Error rate, and (b) execution time . The execution time refers to the duration it takes for a quantum gate operation to be executed.

algorithm accuracy without extensive error characterization. Similarly, Strikis et al. [191] proposed a learning-based method that uses automated training to learn the optimal error compensation strategy. Bennewitz et al. [195] introduced neural error mitigation, a method that applies quantum many-body machine learning techniques to improve estimates of ground states and observables obtained using VQE on noisy devices. Haug et al. [196] demonstrated the benefits of randomized measurements for quantum machine learning on noisy quantum computers, enabling the classification of large datasets. In a recent development, Sack et al. [197] successfully implemented a QAOA on non-planar random regular graphs with up to 40 nodes, made possible by an ML-based error mitigation technique.

Quantum compilation

In addition to error mitigation, quantum compilation techniques [198, 199, 200, 201, 202, 203, 204, 205, 206, 207] have been developed to minimize errors during the implementation of algorithms. Quantum compilation bridges the gap between high-level algorithms and the low-level control capabilities of the physics-based hardware by transforming a high-level quantum algorithm into an optimized sequence of gate operations that can be executed on a specific hardware platform. This process involves three key steps: decomposing the quantum algorithm into a sequence of native gate operations supported by the device, inserting SWAP gates to address the connectivity constraints (also referred to as qubit routing), optimizing the quantum circuit to minimize errors, and scheduling gates to minimize the waiting time between them. The primary goals of compilation are reducing the circuit depth and gate count, thereby mitigating the impact of errors and maximizing performance within the physical hardware limits of the device. The compilation process can be iterative, with the optimization stage being repeated until an optimal circuit is achieved.

Ensuring reliability, scalability, and flexibility is important in compilation process. The algorithm should be correctly rewritten and resistant to noise and errors, while also being scalable to large quantum circuits and flexible enough to handle a variety

of quantum algorithms and hardware platforms. Obtaining high-quality compilation solutions, such as those with minimal circuit depth, is crucial for minimizing errors. The circuit depth refers to the number of time steps or layers required to apply the sequential gates. The longer circuit depth, the more time there is for noise and errors to accumulate between operations, which can significantly reduce the fidelity of quantum circuit. In addition, resource utilization is a critical consideration for quantum circuit compilation. Near-term quantum devices have finite resources such as qubit counts, gate fidelities, and connectivity. Therefore, the compiler should aim to optimize resource utilization by minimizing the number of qubits, depth, and classical memory overhead required to execute the quantum circuit. Lastly, integrating error mitigation and correction techniques during circuit compilation is beneficial for enhancing resilience against noise.

Compilation strategies can be broadly classified into resilience-based and non-resilience-based approaches. Non-resilience-based compilation primarily focuses on minimizing the number of gates and circuit depth. Although this approach may unintentionally enhance resilience by reducing errors, it does not ensure optimal resilience. On the other hand, resilience-based compilation explicitly takes into account error rates during the compilation process, thus guaranteeing a more resilient resulting circuit. Several quantum circuit compilation frameworks exist, such as Qiskit [208], Cirq [209], Tket [210], and QPanda [211]. While these frameworks offer functionalities such as circuit simplification, optimization, and mapping of algorithms onto hardware, they often focus on fast compilation time, potentially leading to poor quality. To improve the efficiency of compilation, Quetschlich et al. proposed a reinforcement learning framework that leverages classical compiler optimization techniques to optimize quantum circuit compilation flow [212]. Moreover, the Munich quantum toolkit (MQT) [213] offers a comprehensive suite of design automation tools to address challenges across the entire quantum software stack.

Part I
Resilience Assessment

Preface

This initial part focuses on resilience assessment, a critical aspect of quantum algorithms. We establish methodologies to quantify the algorithm's robustness against noise and errors using error models and dedicated metrics. The QAOA for problem instances including portfolio optimization and MaxCut serves as a practical example for resilience analysis. We then dissect key factors influencing resilience, including measurement shots, optimizers, optimal parameters, and compilation quality. Understanding these factors is fundamental for designing and optimizing robust quantum algorithms. This initial exploration lays the groundwork for subsequent chapters that will explore strategies and avenues for enhancing resilience.

Chapter 3

Resilience analysis and evaluation

This chapter delves into the resilience of QOAs by evaluating their performance under different noise conditions. We begin by establishing a methodology and metrics to quantify resilience effectively. We then explore the implementation of QAOA, followed by analyzing factors that influence algorithm resilience. Sections 3.3.2 and 3.3.3 are based on the paper published in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)* under the title *Calibration-Aware Transpilation for Variational Quantum Optimization* [6].

3.1 Quantifying algorithm resilience

In the NISQ ear, the performance of algorithms depends not only on their theoretical correctness but also on their ability to function effectively despite inherent hardware errors. This ability of algorithms against noise and errors, also referred to as resilience, is significant for achieving high performance on NISQ devices.

3.1.1 Evaluation methodology

The resilience of quantum algorithms can be assessed by comparing results from simulation and experimental execution. As illustrated in Figure 3.1, the process begins by conducting noise-free simulations on classical computers to establish a theoretical benchmark for optimal algorithm performance. The simulator models the ideal operation of quantum algorithms and serves as a reference point for subsequent analyses. To bridge the gap between theory and practice, as the realistic hardware exhibits noise and errors, the algorithm can be simulated with error models that may incorporate different types of errors, such as bit-flip, phase-flip, and readout error, allowing for a more realistic assessment of the algorithm's resilience. However, mathematical models and assumptions may not fully capture the complexities of real quantum hardware. Therefore, the ultimate evaluation of an algorithm's resilience is achieved by executing it on actual quantum devices. Running the algorithm on physical hardware exposes it to the full range of noise and errors inherent to the specific quantum computing platform. The evaluation under different conditions provides valuable insights into the algorithm's susceptibility to specific error types and its performance degradation, as well as the extent to which hardware noise affects its performance.

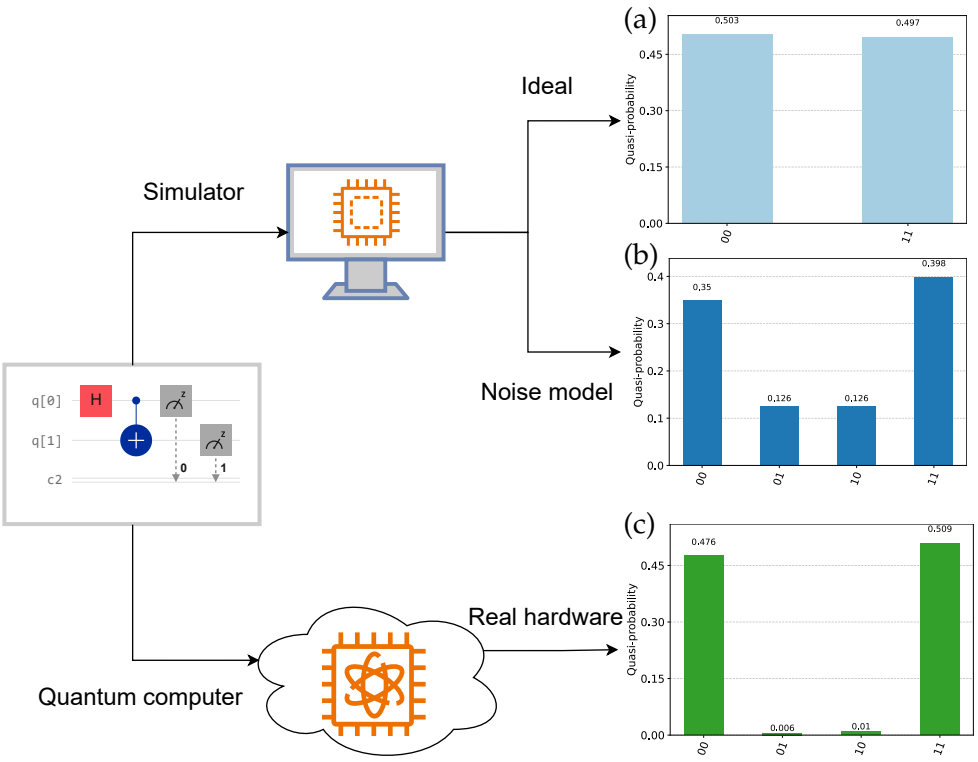

Figure 3.1: Execution modes of quantum algorithms illustrating the impact of noise on output distributions. The probability distributions of output bit strings are obtained from 1024 circuit repetitions (shots) under three conditions: (a) noise-free scenario using the Qiskit Qasm simulator, (b) Qiskit Qasm simulator with a customized noise model, and (c) real quantum hardware, the ibm_osaka is used for this demonstration. While simulations can predict ideal outcomes or incorporate noise models, real quantum devices exhibit inherent noise, resulting in deviations from simulated behavior.

Such analyses bridging the gap between theoretical predictions and practical performance are crucial for developing effective strategies to enhance algorithm resilience, which may involve error mitigation techniques, designing algorithms that are aware of noise, or improving the hardware itself.

3.1.2 Error models

Error models are important for accessing algorithm resilience. Different error models exhibit depending on various noise sources in quantum computers such as environmental noise, imperfect control of gates, and statistical errors. Here, we introduce some commonly used error models.

Pauli channels

The bit-flip, bit-phase flip and phase flip channels indicated by $\mathcal{E}_X$, $\mathcal{E}_Y$, and $\mathcal{E}_Z$ acting on qubits described by density matrix ρ with error rate p_ϵ are defined as [28]

$$\mathcal{E}_X(\rho) = p_\epsilon X\rho X + (1 - p_\epsilon)\rho, \tag{3.1}$$
$$\mathcal{E}_Y(\rho) = p_\epsilon Y\rho Y + (1 - p_\epsilon)\rho, \tag{3.2}$$
$$\mathcal{E}_Z(\rho) = p_\epsilon Z\rho Z + (1 - p_\epsilon)\rho, \tag{3.3}$$

where X, Y and Z denote the Pauli matrices.

Depolarizing channel

The depolarizing channel $\mathcal{E}_D$ acting on an n-qubit system describing by the density matrix ρ is given by [28]

$$\mathcal{E}_D(\rho) = (1 - p_\epsilon)\rho + p_\epsilon \frac{\mathcal{I}}{2^n}, \tag{3.4}$$

where p_ϵ is the probabilistic error rate and $\mathcal{I}$ is the identity matrix. For a single qubit, the depolarizing channel is

$$\mathcal{E}_D(\rho) = (1 - \frac{3p_\epsilon}{4})\rho + \frac{p_\epsilon}{4}\left(X\rho X + Y\rho Y + Z\rho Z\right), \tag{3.5}$$

which can be parametrized as

$$\mathcal{E}_D(\rho) = (1 - \lambda_\epsilon)\rho + \frac{\lambda_\epsilon}{3}\left(X\rho X + Y\rho Y + Z\rho Z\right), \tag{3.6}$$

where $\lambda_\epsilon = \frac{3p_\epsilon}{4}$.

Decoherence channels

Decoherence errors [214] including relaxation (T_1) and dephasing (T_2) arise from unintended qubit-environment interactions causing exponential decay of quantum superposition. Relaxation causes the excited state to decay to the ground state over time, while dephasing randomizes the relative phases between basis states. Both processes degrade the qubit's ability to maintain a superposition, limiting the effectiveness of quantum algorithms. Decoherence can be modeled mathematically using functions that describe this exponential decay of coherence based on characteristic relaxation times.

Readout error model

Readout errors occur during the process of measuring a qubit's state. These errors can arise from imperfections in the measurement apparatus and cause the measured state to differ from the actual qubit state. A common model for readout errors utilizes

the Pauli-X error channel. This model assumes the readout process flips the qubit state from $|0\rangle$ to $|1\rangle$ (or vice versa) with a certain probability p_ϵ.

Backend noise model

To simulate the behavior of quantum devices, IBM provides the backend noise model which includes readout error, gate error, and decoherence error based on the device properties. We note that this noise model does not include the crosstalk error.

3.1.3 Quantification metrics

Quantifying resilience in quantum algorithms critically depends on selecting appropriate metrics that accurately capture performance under various noise models. We present several metrics that facilitate a multidimensional assessment of quantum algorithm resilience.

Quantum state fidelity

Quantum state fidelity serves as a fundamental measure of similarity between ideal and noisy quantum states. Mathematically, it is defined as

$$F(\rho_1\rho_2) = \mathrm{Tr}\left[\sqrt{\sqrt{\rho_1}\rho_2\sqrt{\rho_1}}\right]^2, \tag{3.7}$$

where ρ_1 is the ideal noiseless quantum state and ρ_2 denotes the state influenced by noise. Fidelity provides a quantitative assessment of the stability of quantum states in the presence of errors.

Hellinger distance

The Hellinger distance quantifies the divergence between two probability distributions. Consider two probability measures, P and Q, defined on a measure space χ with probability densities $p(x)$ and $q(x)$, respectively. The Hellinger distance is expressed as

$$H(P,Q) = \sqrt{\frac{1}{2}\int_\chi \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 dx}. \tag{3.8}$$

For discrete probability distributions, the Hellinger distance simplifies to

$$H(P,Q) = \sqrt{(1 - \sum_i \sqrt{p_i q_i})}, \tag{3.9}$$

where p_i and q_i represent the probabilities of outcome i in distributions P and Q, respectively. The Hellinger distance is particularly valuable for assessing the distribution differences between ideal and noisy outcomes in quantum algorithms, enabling both simulations and real hardware applications.

Algorithm performance metrics

A common method for evaluating an algorithm's robustness to noise is directly comparing its performance under different conditions. We define the algorithm's performance as a parameter f, where higher values indicate better performance. In an ideal scenario without noise, the algorithm's performance is denoted by f_i, whereas under realistic conditions with noise, the performance is represented by f_n. The algorithm's resilience to noise can be assessed by analyzing the difference between f_n and f_i. A smaller difference signifies greater resilience, as the performance is less affected by noise. Conversely, a larger difference between f_n and f_i indicates higher susceptibility to noise, resulting in lower resilience. To quantify the effectiveness of an error mitigation strategy, we introduce the metric $\Delta f = f_m - f_n$, where f_m is the algorithm's performance with error mitigation and f_n is its performance without error mitigation. A positive Δf indicates that the error mitigation strategy has successfully improved the algorithm's performance, enhancing its resilience to errors. These performance metrics enable direct comparisons of an algorithm's resilience across various conditions, including quantifying the impact of noise and assessing the effectiveness of error mitigation techniques in improving the algorithm's robustness to noise and other factors. We will delve deeper into these metrics and their implications in Chapter 6.

3.2 Implementation of QAOA

The QAOA is a prominent VQA used for solving combinatorial optimization problems on NISQ devices [19]. In this section, we present the detailed implementation of QAOA for two specific problem instances: portfolio optimization and MaxCut, which serve as examples to illustrate how QAOA can be tailored to address different optimization tasks. The performance of QAOA can be evaluated using two metrics: approximation ratio and success probability. The approximation ratio measures the quality of the solution found by QAOA compared to the theoretical optimal value. The success probability, or ground state probability, reflects the probability of finding the optimal solution for a given problem instance.

3.2.1 QAOA for portfolio optimization

We first briefly summarize the QAOA approach for solving portfolio optimization, as described in Section 2.2.4. The cost Hamiltonian, which describes the portfolio optimization problem for n available assets, can be expressed as

$$\hat{H}_c = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{i,j} \hat{Z}_i \hat{Z}_j - \sum_{i=1}^{n} k_i \hat{Z}_i, \tag{3.10}$$

where parameters $c_{i,j}$ and k_i depend on factors specific to the portfolio optimization problem. In particular, $c_{i,j} = \frac{\lambda}{2}(q\sigma_{ij} + A)$ and $k_i = \frac{\lambda}{2}\left[(1-q)\mu_i + A(2B-n) - q\sum_{j=1}^{n}\sigma_{ij}\right]$, where λ is the global scaling factor, q is risk preference, σ_{ij} is the covariance between assets i and j, A is the penalty factor, B is the number of assets to be chosen, and μ_i is

the expected return of asset i. The terms $\hat{Z}_i\hat{Z}_j$ and $\hat{Z}_i$ correspond to the ZZ interaction on qubits (i, j) and Pauli Z operator acting on qubit i, respectively. The standard mixer Hamiltonian is given by [19]

$$\hat{H}_m = \sum_{i=1}^{n} \hat{X}_i, \tag{3.11}$$

where $\hat{X}_i$ is the Pauli X operator acting on qubit i. After a depth of p, the total system evolves to

$$|\psi_{\vec{\gamma},\vec{\beta}}\rangle = \prod_{k=1}^{p} e^{-i\beta_k H_m} e^{-i\gamma_k H_c} |\psi_0\rangle\,, \tag{3.12}$$

where $|\psi_0\rangle$ is the eigenstate of the mixer Hamiltonian. The aim of the QAOA is to find $2p$ parameters $(\beta_1, ..., \beta_p, \gamma_1, ..., \gamma_p)$ that minimize the expectation value of the cost Hamiltonian $E = \langle\psi_{\vec{\gamma},\vec{\beta}}|\hat{H}_c|\psi_{\vec{\gamma},\vec{\beta}}\rangle$. To quantify the solution founded by QAOA, we define the approximation ratio as

$$r = \frac{E - E_{\max}}{E_0 - E_{\max}}, \tag{3.13}$$

where E_0 represents the optimal value and $E_{\max}$ signifies the worst-case value. A larger value of r indicates higher performance, with $r = 1$ signifying that the QAOA achieves the ground state energy, corresponding to finding the optimal solution.

3.2.2 QAOA for MaxCut

Another application of QAOA is to find the maximum cut of a graph, i.e., to partition the nodes of the graph into two sets such that the number of edges between the sets is maximized, which is known as the MaxCut problem [19, 215]. The cost Hamiltonian describing the MaxCut problem $\hat{H}_c$ is given by [19]

$$\hat{H}_c = \frac{1}{2}\sum_{i,j} \omega_{i,j}(1 - \hat{Z}_i\hat{Z}_j), \tag{3.14}$$

where vertices i, j share an edge and $\omega_{i,j} = \omega_{j,i}$ is the weight corresponding to the edge. Unlike the Hamiltonian for portfolio optimization (Eq. 3.10), which includes

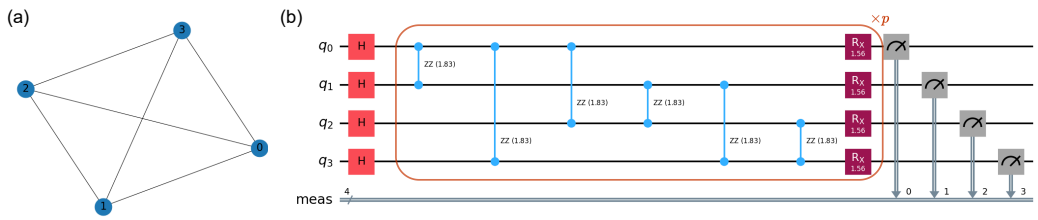

Figure 3.2: (a) Graph of the MaxCut problem and (b) Circuit implementation of QAOA at depth $p = 1$ for MaxCut on the graph shown in (a). For a depth of p, the gates in the box are repeated p times with distinct parameters. A total of $2p$ parameters are required to be determined using a classical optimizer.

$\hat{Z}_i$ terms, the Hamiltonian for MaxCut problems (Eq. 3.14) does not contain these terms. This reflects the absence of R_Z gates in the QAOA circuit for MaxCut problems. Similar to QAOA for portfolio optimization, a mixer Hamiltonian is applied to drive the system towards the desired state (Eq. 3.12) that encodes the solution. An example of implementing the QAOA for MaxCut is shown in Figure 3.2. For a QAOA depth p, the subcircuit in the box presented in Figure 3.2(b) is repeated p times. $2p$ parameters are determined by minimizing the expectation value of the problem Hamiltonian $\hat{H}_c$. The approximation ratio of QAOA for MaxCut is defined by

$$r = E/E_0, \tag{3.15}$$

where E is the mean value found by QAOA and E_0 is the optimal value.

3.3 Factors influencing algorithm resilience

The resilience of QOAs to noise and errors depends on both hardware properties and algorithmic considerations. Hardware factors, such as the number of qubits and their connectivity, significantly impact QOA performance. More qubits enable larger, more complex problems, while better connectivity facilitates efficient algorithm implementation. High-quality qubits with high gate fidelities and long coherence times are essential for achieving high-performance QOAs. Algorithmic design also plays a crucial role in QOA performance. The structure of the circuit, settings of parameters, and strategies used for optimization all influence the algorithm's ability to navigate the optimization landscape. Deeper circuits have the potential to yield better solutions, but they are also more susceptible to noise accumulation. In the following, we explore several factors that affect the performance of QAOA.

3.3.1 Measurement shots and optimizers

We investigate the impact of measurement shots and optimizers on the performance of QAOA applied to the 3-regular MaxCut problem. We compare the performance of the gradient-free COBYLA with the gradient-based SPSA optimizer for classical parameter optimization within the QAOA circuit.

Noise-free case

We begin by examining the noise-free scenario. Both optimizers are initialized with the same random values and set to 1000 iterations. Figure 3.3 displays the simulated probabilities of achieving various objective values for different measurement shot numbers using COBYLA and SPSA. For the 6-qubit case (Figures 3.3(a) and 3.3(b)), all configurations achieve a high probability of finding the optimal solution with a minimum objective value of -7. Additionally, COBYLA exhibits a generally increasing success probability with more shots, suggesting improved performance with an increasing number of shots. SPSA achieves better performance but shows less variation in success probability across different shot numbers. For the 10-qubit case (Figures 3.3(c) and 3.3(d)), only COBYLA with 8192 shots achieves the optimal solution with an objective value of -13 most of the times. In comparison, 16384 shots

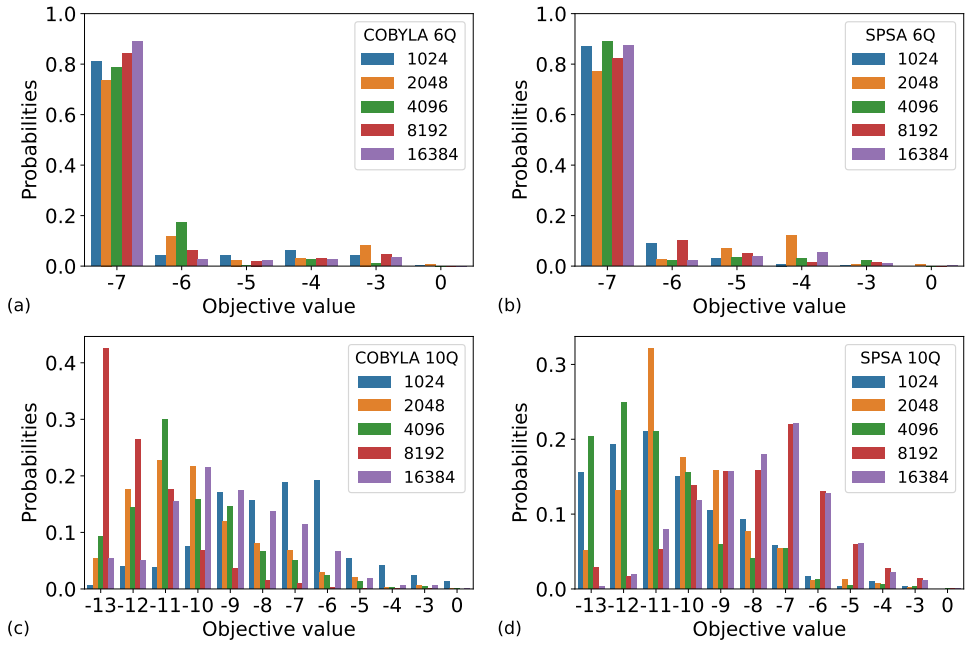

Figure 3.3: Simulated probabilities of obtaining different objective values for QAOA applied to randomly generated 3-regular MaxCut problems using Qiskit's Qasm simulator with varying shot numbers. (a) and (b) depict 6-qubit (6Q) QAOA at $p = 6$, while (c) and (d) represent 10-qubit (10Q) QAOA at $p = 6$. (a) and (c) employ COBYLA, whereas (c) and (d) utilize SPSA for parameter optimization.

have a lower performance than 8192 shots, meaning that selecting a larger number of shots may not always yield optimal results for optimizing algorithm parameters. Moreover, optimizing the number of shots presents a possible way to achieve high performance. Compared to COBYLA, SPSA exhibits low performance, especially for a large number of shots. The reason could be that as the problem size grows, the optimization becomes more complex, potentially causing the gradient-based SPSA to get stuck in local minima and miss the optimal solution.

These results indicate that gradient-free optimizers might be advantageous for identifying optimal solutions in large-scale problems. Moreover, the number of measurement shots significantly impacts optimization outcomes and, consequently, algorithm performance. We also observe a trend where performance increases with shots, reaches a peak at 8192 shots, and then decreases. Therefore, finding the optimal number of shots for a desired performance guarantee is crucial for balancing computational efficiency and solution quality, as large numbers of shots increase computational cost and also the required quantum resources for execution. Additionally, factors beyond the number of shots, such as initial parameter values and QAOA depth, can limit algorithm effectiveness. In such cases, even a very high number of shots cannot significantly improve the probability of finding the optimal solution.

Optimization under noise

We now examine the influence of noise on the optimization process for algorithms applied to the 3-regular MaxCut problem. Here, we analyze the performance of

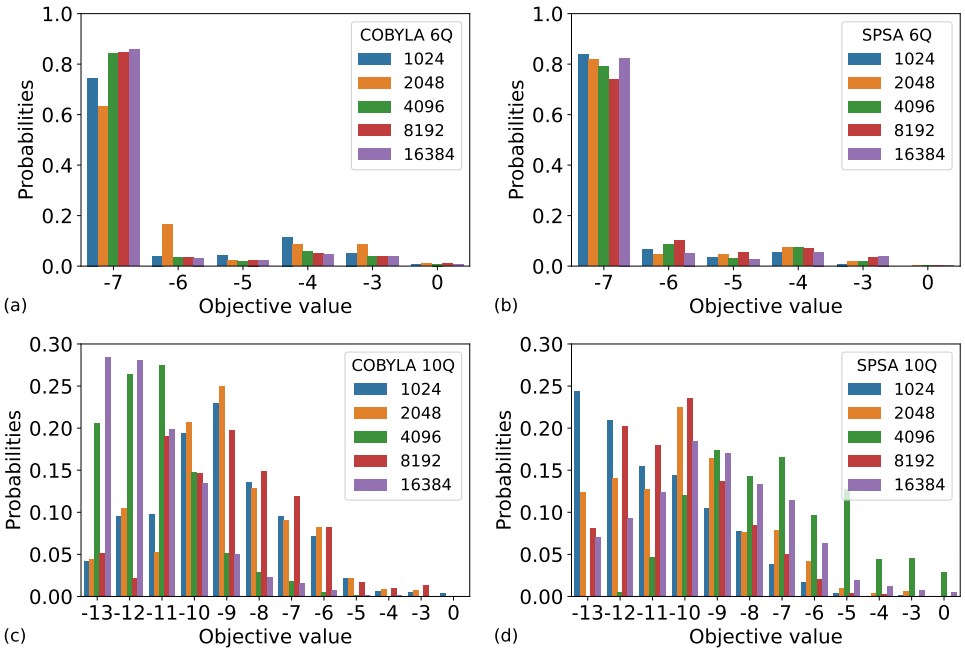

Figure 3.4: Simulated results of optimizing algorithm parameters under noise. Labels are the same as in Figure 3.3

COBYLA and SPSA optimizers when optimizing parameters in the presence of a well-defined noise model. We employ the Pauli-X error channel (Eq. 3.1), introducing bit-flip errors with a probability of 0.0005 to all single- and two-qubit gates. Our focus is on how noise impacts the optimization process and its effect on the success probability of finding the optimal solution for both 6- and 10-qubit QAOA instances.

For the 6-qubit QAOA case (Figures 3.4(a) and 3.4(b)), all configurations achieve the optimal solution, exhibiting similar behavior to the noise-free scenario across different shot numbers. This suggests that for smaller problem sizes, noise does not significantly affect the optimization process. For the 10-qubit QAOA case (Figures 3.4(c) and 3.4(d)), COBYLA with the highest number of shots (16384) achieves the highest success probability for the optimal solution, corresponding to the lowest objective value of -13. However, this success probability value is close to that for a suboptimal solution, potentially hindering the identification of the optimal solution in other cases. Conversely, SPSA with the lowest number of shots (1024) successfully identifies the optimal solution, demonstrating a potential advantage under noise for this problem size. These observations suggest that the optimal number of shots and the impact of noise on optimizer performance depend on the problem size. For smaller problems, noise might have a less pronounced effect. However, for larger problems, a lower number of shots might be beneficial for gradient-based optimizers like SPSA under noise, potentially helping them escape local minima. Additionally, noise appears to have a mixed effect on COBYLA, with a higher number of shots sometimes leading to better performance but also potentially increasing the chance of getting stuck near suboptimal solutions. These findings emphasize the importance of considering noise when optimizing algorithms, particularly for larger and more complex problems.

Our investigation highlights the potential benefits of using SPSA with a lower number of shots for noisy optimizations, especially for larger problems. However, direct hardware optimization through iterative execution on real devices is challenging due to resource constraints and the increased time required by SPSA compared to COBYLA. Therefore, a practical approach for real-world applications might involve optimizing algorithm parameters in a noise-free environment with COBYLA, followed by the implementation of error mitigation or correction techniques during execution on real quantum devices.

3.3.2 Optimal parameters

The performance of QAOA depends on the values of its parameters. Optimal parameters result in a higher quality of performance, which is indicated by a high value of the approximation ratio or success probability. The parameters of QAOA are usually determined by a classical optimizer, but the optimal parameters for QAOA with $p = 1$ can be determined using a grid search. Here, we study the influence of parameters on the performance of QAOA. Specifically, we use grid search to identify the optimal parameters for QAOA at depth $p = 1$. Then, we compare the performance of QAOA employing optimal parameters with those determined by a classical optimizer. In this investigation, we employ COBYLA. Figures 3.5(a) and 3.5(b) show the optimization landscape of 5-qubit QAOA using the Qasm simulator in the absence of noise and on the actual quantum device ibmq_ehningen, respectively. The optimal parameters θ, expectation values E, as well as the approximation ratio and success probability are shown in Table 3.1. We observe that the optimal values of parameters and optimization landscape hardly change, indicating that QAOA at $p = 1$ is stable even in the presence of noise.

We simulate QAOA with the original and optimal initial values using bit-flip, bit-phase flip, and depolarizing error channels indicated by $\mathcal{E}_X$, $\mathcal{E}_Y$, and $\mathcal{E}_D$ [28] with an error rate of p_ϵ. The error channels acting on qubits described by density matrix ρ are defined as in Eqs. 3.1, 3.2, and 3.4. We study the state fidelity of QAOA between the ideal quantum state and the noisy state with an error rate of up to 1%. The state fidelity is defined in Eq. 3.7. In our case, ρ_1 is the final QAOA state in the absence of errors, and ρ_2 is the state for QAOA with an error rate of p_ϵ. We consider discrete values of $p_\epsilon \in \{0, 0.1\%, 0.2\%, ..., 1\%\}$. The maximum fidelity of 1 occurs at $p_\epsilon = 0$.

Table 3.1: Optimal solution founded by grid search with Qasm simulator in absence of errors (error-free) and ibmq_ehningen. θ: values of parameters. E: expectation values.

Parameter	error-free	ibmq_ehningen
θ	$(2.3, 2.1)$	$(2.3, 2.3)$
E	-0.957	-0.726
AR	0.417	0.409
SP	0.235	0.295

Published in Yanjun Ji, Sebastian Brandhofer and Ilia Polian, 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), DOI: 10.1109/QCE53715.2022.00040.

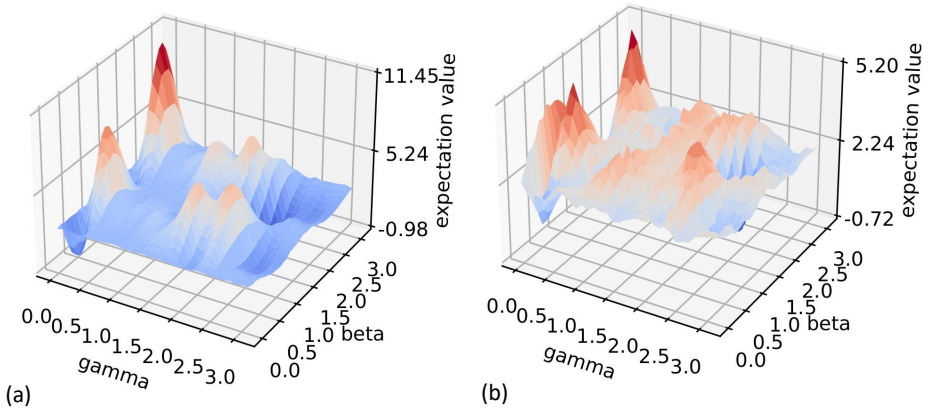

Figure 3.5: Optimization landscape of QAOA for portfolio optimization with (a) Qasm simulator and (b) ibmq_ehningen.

We label original (orig) as original initial values established by the classical optimizer COBYLA, and the optimal (opt) as optimal initial values determined by grid search, i.e., Figure 3.5(a). As shown in Figure 3.6, optimal initial values of QAOA produce a higher fidelity than the original values. Different error channels introduce distinct types of errors in quantum circuits, leading to various effects on the performance of algorithms. The effect of bit-phase flip on fidelity is significant, while the fidelity under bit-flip error varies only slightly. In addition, the type of initial values only produces a small difference under depolarizing error. At a noise level of $p_\epsilon = 1\%$, the fidelity of QAOA with bit-flip errors decreases to approximately 0.95. For depolarizing errors, the fidelity drops to 0.75, and for bit-phase flip errors, it drops to 0.6.

The behavior of approximation ratio and success probability with increased error rates is also investigated. The results of running 10 QAOA using Qasm simulator are shown in Figure 3.7. Without error, we achieve approximation ratios of around 0.42 with optimal and 0.39 with original parameters. The approximation ratio is strongly affected by bit-phase flip error, such as the fidelity in Figure 3.6. The values of approximation ratio at $p_\epsilon = 1\%$ are around 0.33 and 0.30 for optimal and original parameters, respectively. QAOA under noise with optimal parameters outperformed the original. The influence of parameters on approximation ratio is smaller than the success probability. Optimal parameter values produced a significantly better success probability than the original, as shown in Figure 3.7(b). The results demonstrate the importance of optimizing parameters in enhancing the resilience of VQAs.

3.3.3 Compilation quality

Published in Yanjun Ji, Sebastian Brandhofer and Ilia Polian, 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), DOI: 10.1109/QCE53715.2022.00040. ©2022 IEEE.

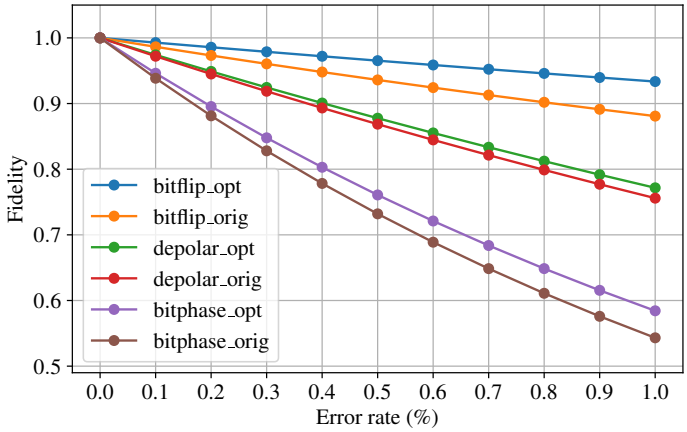

Figure 3.6: Simulation of fidelity of QAOA with original (orig) and optimal (opt) initial values using Qasm simulator under bitflip, depolarizing and bit-phase flip errors as a function of error rate from 0 to 1%.

Qiskit's transpilation procedure includes randomization, and running it multiple times produces different solutions. To improve stability, we implemented an additional check that bounds the maximum increase of CX gates $\texttt{CX}_{\text{inc}}^{\text{max}}$, i.e., discards transpiled circuits with an increase of CX gates exceeding this threshold. Figures 3.8 and 3.9 show the approximation ratio and, respectively, the success probability of 678 repetitions of QAOA on the physical quantum computer ibmq_ehningen with $\texttt{CX}_{\text{inc}}^{\text{max}}$ set to 215%, 185%, 155%, 140%, 75%. For each such restriction, Figures 3.8(a) and 3.9(a) show the 95% confidence interval of approximation ratio and success probability, whereas Figures 3.8(b) and 3.9(b) include the complete histograms. It can be seen that restricting the increase in CX gates tends to improve the performance of QAOA. Therefore, one main goal of our calibration-aware transpilation is to minimize the number of used CX gates (while also improving fidelity). Our results show that QAOA is resilient against noise up to a certain noise level. However, the performance of QAOA deteriorates rapidly with increasing bit-phase flip noise. We also find that the choice of the classical optimization algorithm and the number of iterations of QAOA have a significant impact on its resilience.

3.4 Conclusions

We proposed an evaluation methodology and metrics for assessing the resilience of QOAs under noise simulation and real device execution. Additionally, we presented case studies on the resilience of QAOA for MaxCut and portfolio optimization problems, and discussed the factors that impact the resilience of QOAs. While QOAs currently demonstrate limited resilience, our analysis highlights clear opportunities for improvement by implementing techniques such as efficient compilation. This motivates the development of strategies to enhance algorithm performance on noisy quantum devices.

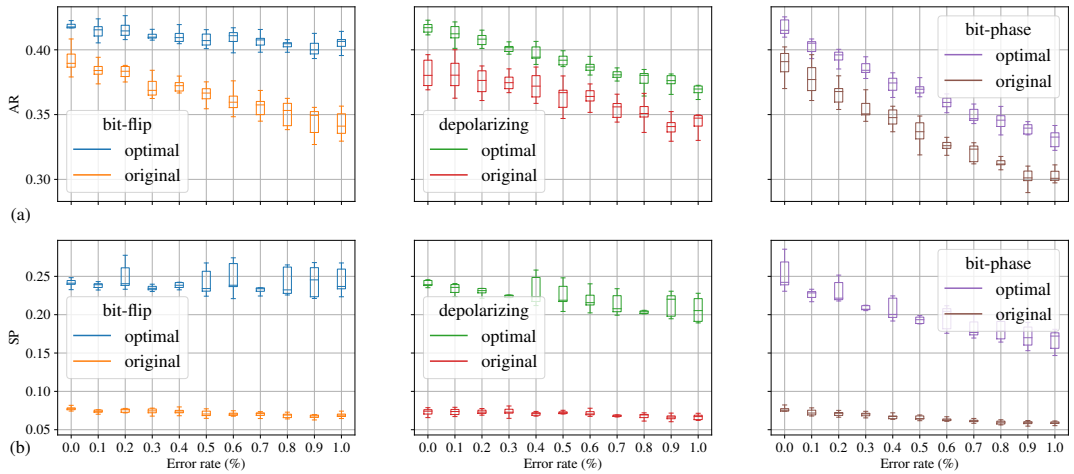

Figure 3.7: Simulation of (a) approximation ratio and (b) success probability of QAOA with 10 repetitions using Qasm simulator under bitflip, depolarizing and bit-phase flip errors as a function of error rate from 0 to 1%.

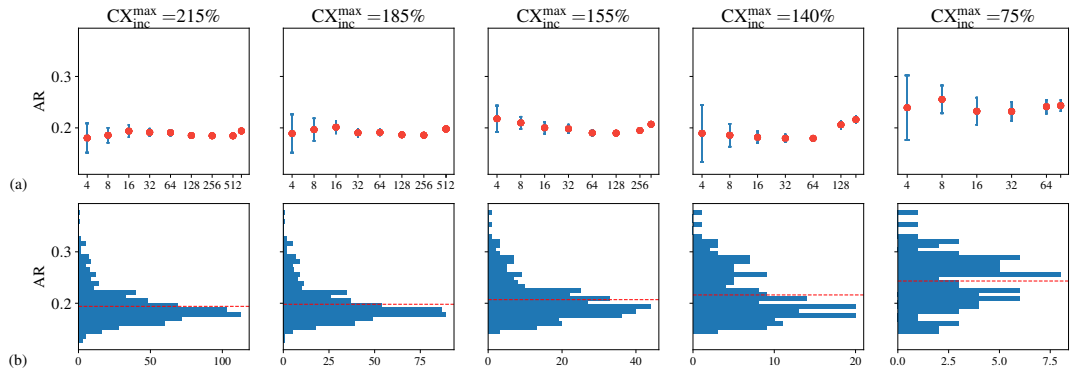

Figure 3.8: Impact of compilation quality on QAOA approximation ratio. (a) 95% confidence interval and (b) histogram with 678 QAOA repetitions using Qiskit transpiler on ibmq_ehningen. With restriction of the amount of maximum increase in CX gates, the approximation ratio is improved.

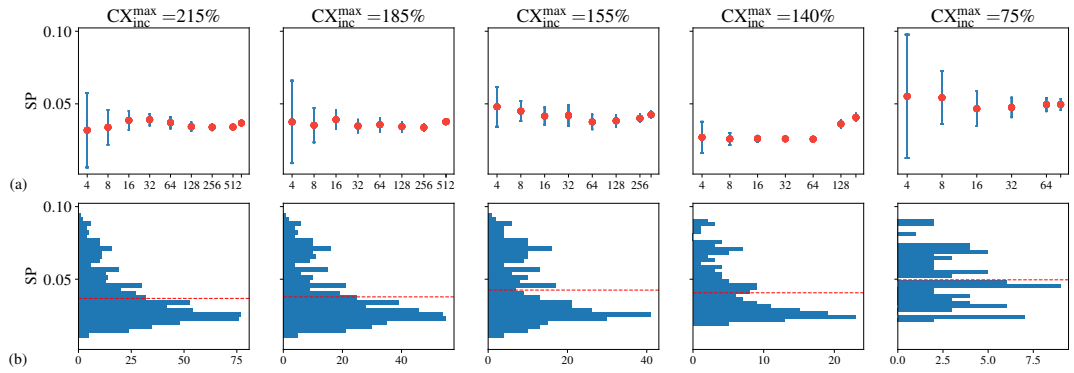

Figure 3.9: Impact of compilation quality on QAOA success probability. (a) 95% confidence interval and (b) histogram with 678 QAOA repetitions using Qiskit transpiler on ibmq_ehningen. Success probability shows a similar behavior to approximation ratio in Figure 3.8.

Part II
Resilience Enhancement Strategies

Preface

Building upon the foundation of resilience assessment established in the first part, the second part delves into various strategies for enhancing the resilience of quantum algorithms, including optimized quantum compilation, selective optimization, error mitigation, and a cooptimization approach. Chapter 4 presents an efficient calibration-aware transpilation method and an algorithm-oriented qubit mapping strategy. Utilizing the proposed compilation method, Chapter 5 further explores selective optimization on bipotent architectures, focusing on optimizing algorithm-native gates at the pulse level. Understanding the factors that affect the efficiency of error mitigation is a crucial aspect of resilience enhancement. This is analyzed in Chapter 6, which investigates the synergistic effect of error mitigation and circuit design, considering both hardware and algorithmic factors. Experiments are conducted on real quantum processing units (QPUs) to validate the findings. Finally, Chapter 7 explores the concept of algorithm-hardware cooptimization, a methodology that incorporates hardware constraints into the design of algorithms. This approach is then applied to various problem instances, including MaxCut on complete and noncomplete graphs, as well as portfolio optimization. Experimental evaluations demonstrate the effectiveness of this cooptimization strategy in reducing noise and improving performance on real quantum hardware. Our comprehensive exploration of resilience enhancement strategies provides a holistic understanding of the techniques that can be used to enhance the resilience of QOAs, ultimately accelerating their integration into practical applications.

Chapter 4

Optimized Quantum Compilation

This chapter presents strategies for optimizing quantum compilation to enhance the performance and robustness of QOAs. First, we introduce a calibration-aware transpilation process to improve compilation efficiency by considering the time-varying error rates of quantum devices and algorithm features. Then, we bridge the gap between heuristic and exact methods to address qubit connectivity constraints by providing optimal and scalable solutions. Finally, we introduce the workflow for optimized compilation and discuss methods for ensuring reliability. Section 4.1 has been published in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)* under the title *Calibration-Aware Transpilation for Variational Quantum Optimization* [6]. Section 4.2 is currently under third review in *Physical Review Applied*.

4.1 Efficient calibration-aware transpilation

The compilation process handles mapping algorithms to executable code for quantum devices, which involves mapping logical qubits in algorithms to physical qubits on quantum devices, inserting SWAP gates to accommodate connectivity constraints, decomposing gates into the device's native gate set, and optimizing the circuit at the gate or pulse level to mitigate the impact of errors. In comparison, transpilation is a critical component of the compilation process. While compilation encompasses broader algorithmic considerations, transpilation specifically focuses on low-level circuit rewrite and optimizations. Moreover, transpilation is frequently employed for IBM quantum devices, where IBM provides a transpiler to automate this process. In some contexts, the terms compilation and transpilation are used interchangeably. In this section, we adopt the term 'transpilation' to emphasize its focus on IBM quantum devices. Specifically, we introduce the calibration-aware transpilation for variational algorithms with parameterized ansatz circuits.

4.1.1 Method advantages

As introduced in Section 2.3.2, the errors of quantum devices vary over time. Figure 4.1 further reports hourly calibration data, which we collected on ibmq_ehningen

Published in Yanjun Ji, Sebastian Brandhofer and Ilia Polian, 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), DOI: 10.1109/QCE53715.2022.00040. ©2022 IEEE.

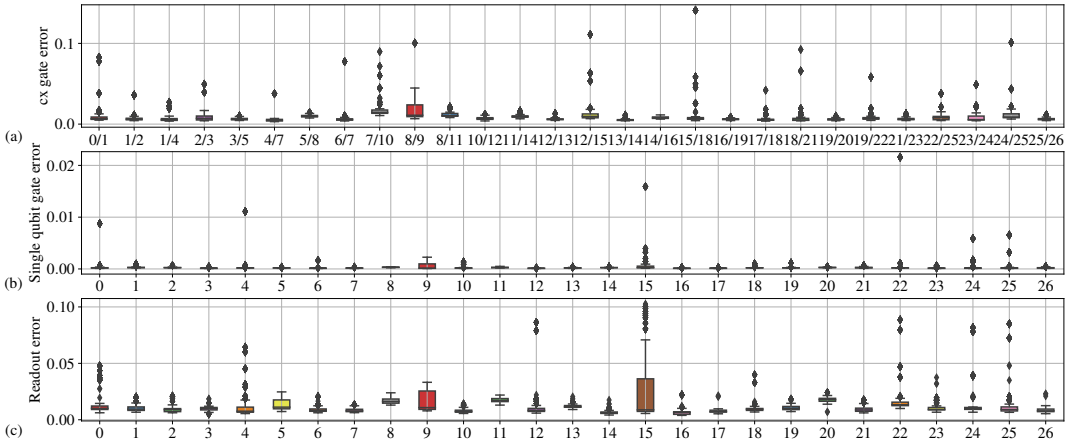

Figure 4.1: Error information of ibmq_ehningen over 39 days.

over a period of 39 days. The highest variations were observed for the CX gate between qubits 8 and 9 (Figure 4.1a) and for the readout errors of qubit 15 (Figure 4.1c). Therefore, periodically updated calibration of quantum devices provides information on time-varying error rates that should be considered during transpilation. Based on these observations, we propose a calibration-aware transpilation process that divides the implementation process into three main steps: (1) noise-unaware and computationally heavy pretranspilation; (2) fast noise-aware matching; and (3) fast decomposition followed by heuristic optimization. The first step performs a noise-unaware and computationally heavy pretranspilation of the quantum circuit to address the connectivity constraints optimally. The second step performs a fast noise-aware matching of the pretranspiled circuit on the topology based on updated noise rates ensuring that the qubits with the highest quality are selected for execution. Finally, the circuit is decomposed into the basis gate set of the target devices, followed by optimization that optimizes the circuit for speed and accuracy.

The calibration-aware transpilation process provides executable circuits with low latency and can be applied to other near-term quantum computers. For a complete run of a variational algorithm under constant error rates, only step (3) needs to be executed for each new ansatz circuit. Step (2) is required only if the error rates reported by calibration have changed significantly since the beginning of the computation. The most expensive step (1) is executed only once for the whole run. This distribution is helpful for incremental, calibration-aware transpilation when the variational algorithm adapts its own execution to changing error rates. While designed to optimize variational quantum algorithms, the calibration-aware transpilation process is also applicable for optimizing general quantum algorithms.

The procedure is outlined in Figure 4.2. The processed ansatz circuit A is parameterized with values θ_i, which, in case of QAOA, are angles determined by the classical optimization step. Transpiled circuits $A(\theta_1), A(\theta_2), \ldots$ differ only minimally and their basic structure must be computed only once for A and can be reused by all circuits. Moreover, transpilation takes error rates ξ^* into account, which are determined by calibration. After each re-calibration, the procedure checks whether new error rates ξ are substantially different from ξ^*. If this is the case, transpilation is not repeated

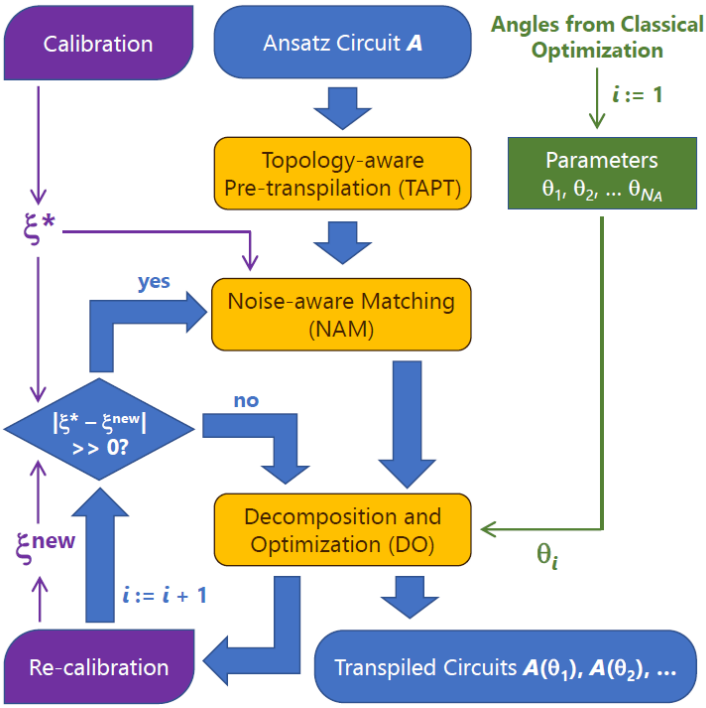

Figure 4.2: Flowchart of calibration-aware transpilation. It consists of three steps: Topology-aware pretranspilation (TAPT), executed one time for the entire run of a variational algorithm and calculating parts of the solution applicable to all ansatz circuits; Noise-aware matching (NAM), invoked only when error rates have changed significantly, and decomposition and optimization (DO), which includes improvements for a specific ansatz circuit $A(\theta_i)$.

from scratch, but the initial, optimized solution is mapped to a different subset of qubits with the same subgraph topology yet better fidelities. Overall, calibration-aware transpilation consists of three steps: topology-aware pre-transpilation (TAPT); noise-aware matching (NAM), and decomposition and optimization (DO).

The main advantage of calibration-aware transpilation is the significantly reduced effort, as all computationally heavy parts are accumulated in the TAPT step that is run only once, and the two remaining steps are lightweight. It is feasible to use NAM and DO in an incremental mode: whenever a new ansatz circuit is ready for execution on the quantum hardware, reserve the quantum computer, acquire calibration data, execute NAM (if needed) and DO, and then immediately execute the transpiled circuit on the reserved computer. This makes sure that the most recent calibration data are used for each ansatz circuit, while the amount of the quantum computer's time "wasted" during reservation is minimal. In the traditional approach with full transpilation of each ansatz circuit, reserving the quantum computer for its complete duration would be unrealistic. The computer would start processing other tasks, and the transpiled ansatz circuit, once it is ready, would be inserted into the regular queue and executed possibly at a time instant when the calibration data is outdated.

In addition, the potential advantage of transpiling one circuit with calibration-aware transpilation is shown in Figure 4.3. The high-level idea is to submit the circuit

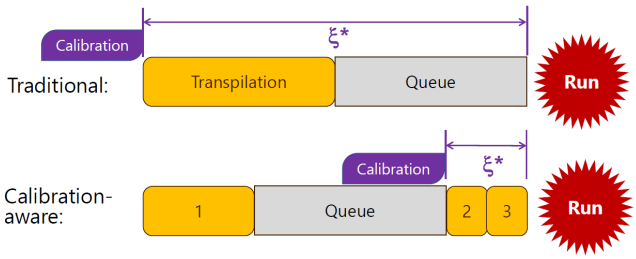

Figure 4.3: Advantage for transpiling one circuit with calibration-aware transpilation compared to traditional: fresher error rate information. 1, 2 and 3 indicate TAPT, NAM and DO processes in CA transpilation, respectively.

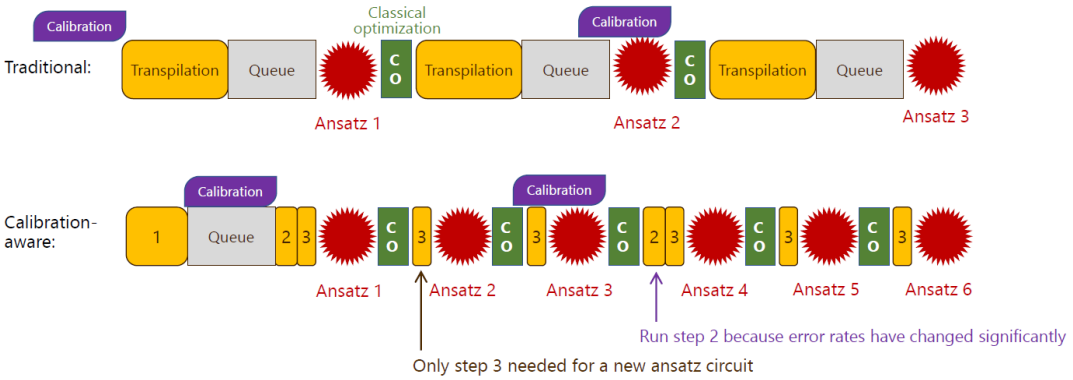

Figure 4.4: Advantage of calibration-aware transpilation compared to traditional for a variational run with multiple ansatz circuits: effort reducing and time saving. Classical optimization is used to calculate new parameters of QAOA.

directly to the queue after step 1, which is the most computationally heavy process. Steps 2 and 3, which take calibration data into account, are performed just before execution, followed by immediate execution of the transpiled circuit. This ensures that CA has a fresher error rate information than the traditional approach where calibration data is acquired at the beginning of transpilation. This is crucial to the performance of algorithms on the NISQ computers as their errors change over time. Furthermore, based on CA's structures, we can significantly reduce effort and save time for a variational run with multiple circuits, see Figure 4.4. While the traditional approach requires each circuit to be transpiled and passed into the queue before execution, CA only needs to pass the first circuit into the queue, and since steps 2 and 3 are fast, the remaining ansatz circuits can continue implementation, resulting in a significant reduction in overheads.

4.1.2 Procedure

As has been discussed above (Figure 4.2), calibration-aware transpilation is organized in three steps:

- TAPT, which can be computationally complex and produces a high-quality (or even optimal) solution that is independent of error rates.

- NAM, which takes the connectivity determined during the first step and maps it to a subgraph of the IBM quantum devices's topology graph with the lowest error rates. This step is simple and needs to be repeated only if the error rates according to the calibration data have changed significantly.
- DO, which is a collection of inexpensive procedures that take the ansatz circuit's parameters (for QAOA: angle θ_i) into account. For example, certain quantum gates can be removed altogether for $\theta_i = 0$.

In the following, we provide details on the three steps.

Topology-aware pre-transpilation (TAPT)

The TAPT process aims at satisfying the connectivity requirements of a quantum algorithm (here: ansatz circuit) at a given architecture. All two-qubit gates must either be mapped to connected qubits of the quantum hardware, or additional SWAP gates must be inserted such as to bring them onto neighboring qubits. On IBM's architecture used in this work, SWAP gates are rather expensive primitives, implemented by 3 CX gates. Therefore, TAPT aims at minimizing the number of required extra SWAP gates, and ultimately the total number of CX gates in the circuit. Note that in general, even an optimized transpiled circuit has more CX gates than before transpilation.

Algorithm 4.1 shows the pseudocode of topology-aware pretranspilation. To obtain an efficient pretranspiled circuit as a starting point, the algorithm starts with finding an initial mapping by *graph placement* [210]. This procedure identifies a subgraph isomorphism between the graph of interacting logical qubits and the connectivity graph of the physical qubits. Before inserting SWAP gates, the circuit is partitioned into sub-circuits bounded by a CX gate on one or both sides. For QAOA ansatz circuits, such efficient sub-circuits have the form CX-R_Z-CX and implement the $Z \otimes Z$ interaction between two qubits.

To insert SWAP gates, we use an optimal method based on the satisfiability modulo theory (SMT) with circuit depth as the optimization objective [216]. The SMT method treats the sub-circuits identified as outlined above as primitive circuit elements. That is, SWAP gates are inserted only between sub-circuits. The rationale behind this procedure is the later application of CX cancellation, where CX gates on the boundaries of sub-circuits can be merged with CX gates that implement SWAP gates. In addition, considering sub-circuits reduces the problem complexity and the runtime of the SMT solver. Thereafter, the sub-circuits and the inserted SWAP gates are decomposed into basis gates of the circuit and undergo optimization, including CX cancellation. The resulting circuit is depth-optimized and executable on (a subgraph of) the target topology graph.

Noise-aware matching (NAM)

The previous process maps the algorithm's qubits to a subgraph of the topology graph, but it does not consider error rates of physical qubits in that subgraph. At the same time, most topology graphs of today's larger-scale quantum computers have a large number of isomorphies and symmetries. For example, the topology graph from Figure 2.3(a) can be understood as consisting of two "tiles" (physical qubits 0, . . . , 14,

Algorithm 4.1: Topology-aware pre-transpilation (TAPT)

Input: Original quantum circuit qc, Coupling map $G(V, E)$
Output: Topology-aware pretranspiled circuit pqc
begin
  $M \leftarrow$ initial mapping;
  $U \leftarrow$ set of sub-circuits between two qubits in qc with different structures;
  for $u \in U$ **do**
    transform all gates in qc with the same structure as u to u with gate parameters;
  end
  Set initial mapping M;
  Route the re-constructed circuit by inserting SWAP gates using SMT based optimal algorithm to minimize depth;
  Decompose parameterized u into basis gates of qc;
  for *each swap gate in the circuit* **do**
    if *swap gate is before the measurement* **then**
      remove SWAP gate and interchange the measurement of two qubits;
    else
      decompose SWAP into three CX gates and optimize with CX cancellation;
    end
  end
  Transform into a logic circuit by removing the idle wires;
  return pqc
end

16 and physical qubits 10, 12, ..., 26). Any algorithm mapped to a subgraph from one "tile" can also run on the other. Moreover, each "tile" is symmetric. For instance, a 5-qubit algorithm mapped to physical qubits (0, 1, 4, 7, 6) is automatically valid for physical qubits (10, 12, 15, 18, 17); (15, 12, 10, 7, 6); (16, 14, 11, 8, 9); (26, 25, 22, 19, 20). Note that larger IBM computers have even more identical "tiles" and offer more valid alternatives for each result of step TAPT.

The NAM process considers up to N alternative subgraphs and selects the one with the highest effective average fidelity. N is a user-defined constant, which trades the likelihood of finding a good matching against the number of necessary computations; the latter can be important when calibration-aware transpilation is used in the incremental mode and the quantum computer waits until the new subgraph is identified. The effective average fidelity of quantum circuit qc with the calibration data $\xi = (f_{\mathtt{u}}, f_{CX}, f_{\mathtt{d}})$ as

$$get\_fidelity(qc, \xi) = \frac{1}{3}\left(\prod_{\mathtt{u} \in qc} f_{\mathtt{u}} + \prod_{CX \in qc} f_{CX} + \prod_{\mathtt{d} \in qc} f_{\mathtt{d}}\right) \tag{4.1}$$

where $f_{\mathtt{u}}$, f_{CX} and $f_{\mathtt{d}}$ are fidelities of single qubit, CX gate and readout gate, respectively.

The NAM process is described in Algorithm 4.2. As input we have pretranspiled circuit, which has satisfied the connectivity of subgraph of topology graph, coupling map, the latest calibration data, the function *get_fidelity* to calculate the fidelity of circuit, and the number of trials N (we used $N = 15$ in our experiments). In order to select high-fidelity qubits, we perform N trial matchings with Qiskit's transpile.

Algorithm 4.2: Noise-aware matching (NAM)

Input: Topology-aware pretranspiled circuit *pqc*, Coupling map $G(V, E)$, Calibration data ξ, Fidelity computation function *get_fidelity*, Number of trials N
Output: Matched circuit *mqc*
begin
  $mqc \leftarrow$ matched *pqc* with Qiskit;
  $c_m \leftarrow$ number of CX gates in *mqc*;
  $f_m \leftarrow get\_fidelity(mqc, \xi)$;
  $j \leftarrow 0$;
  while $j \neq N$ **do**
    $rqc \leftarrow$ re-matched *pqc* with Qiskit;
    $c_r \leftarrow$ number of CX gates in *rqc*;
    $f_r \leftarrow get\_fidelity(rqc, \xi)$;
    if $c_r \leq c_m$ *and* $f_r > f_m$ **then**
      $mqc \leftarrow rqc$
    end
    $j \leftarrow j + 1$;
  end
  return *mqc*
end

With the randomization of Qiskit's transpilation procedure, the pretranspiled circuit is matched to different physical qubits of quantum computer. Effective average fidelities of the matched circuits on physical qubits are calculated and the circuit with the highest fidelity is picked. This circuit contains the information of selected physical qubits in N trials.

Decomposition and optimization (DO)

After the NAM process, the target qubits used to run the algorithm are fixed. Then the quantum algorithm is decomposed into the native gate set supported by IBM quantum devices. In this work, we are using IBM's computers that support single-qubit gates and the CX gate as an entangling gate. After decomposition, we apply optimization techniques *Optimize1qGates*, *CommutationAnalysis*, *CommutativeCancellation*, *CXCancellation*, *RemoveDiagonalGatesBeforeMeasure* and *RemoveResetInZeroState* provided by Qiskit. We repeat this process 15 times aiming to obtain the final transpiled circuit with the least number of CX gates. Note that this step is architecture-specific and would need to be adapted for a different platform; for example, Google's computers use CZ gates as entangled gates.

4.1.3 Benchmarking with QAOA

We benchmark the calibration-aware transpilation with QAOA on four IBM quantum devices, ibmq_ehningen, ibm_cairo, ibm_auckland and ibm_hanoi, all of which have 27 qubits and the same topology graph, as shown in Figure 2.3(a). The initial values of QAOA used here are determined by COBYLA. We use three sets of metrics: the quality of the transpilation process (quantified by percental increase of the circuit's depth $\Delta d\%$, its total number of gates $\Delta g\%$ and number of its CX gates $\Delta g_{CX}\%$ as a

result of transpilation); the runtime of the transpilation procedure; and the quality of QAOA in terms of approximation ratio and success probability achieved on a physical quantum computer. We start with a comparison of CA with a large number of different transpilation methods with respect to approximation ratio and success probability. Then, we pick one of the best methods observed and compare it with CA in-depth. Finally, we outline the potential benefits of CA for runtime of a complete QAOA algorithm.

Approximation ratio, success probability, CX gate count

We compare the achieved approximation ratio in Figure 4.5. In addition to CA, this figure includes Qiskit's built-in transpilation procedure, SMT-based methods [216], two variants of Tket [210], staq [217], a total of 53 composite methods depending on different initial mapping and routing procedures, and swap network (SN) [218]. CA either outperforms other methods or is on a par with the best of them for all four quantum computers.

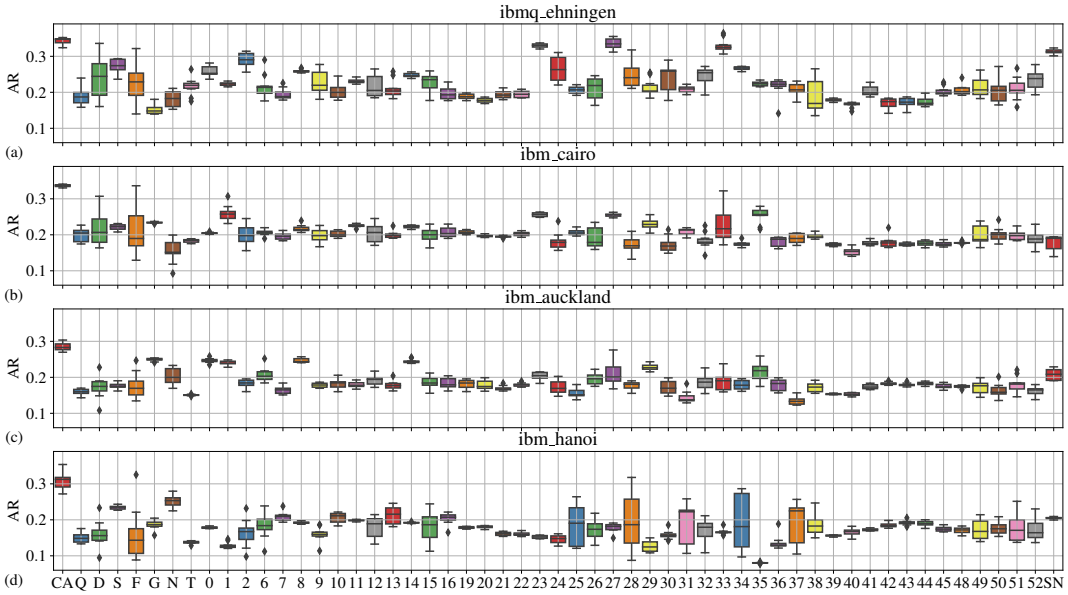

Figure 4.5: Approximation ratio of 10 QAOA runs with different transpilation methods on four IBM quantum devices. CA: Calibration-aware (our approach); Q: Qiskit's built-in transpilation procedure with *optimization_level* 3; D/S/F: SMT-based methods [216] to minimize depth (D), minimize the number of SWAP gates (S), to maximize fidelity (F); G: Tket with initial mapping based on *graph placement* [210]; N: Tket with *noise aware placement* [210]; T: staq [217]; 00...52: methods composed of different initial mapping and routing procedures, some including ZX-calculus optimization [219]; SN: swap network [218]. 10 runs of QAOA per data point.

We believe that this advantage is due to CA's NAM step considering a number of possible sub-graphs, selecting the one with the best fidelity according to a more up-to-data calibration data than other methods. At the same time, the TAPT step produces a robust depth-optimized solution for the connectivity constraints, thus limiting errors that stem from excessive gate-count. We observed that purely fidelity-

oriented transpilation can incur strong variability in gate count for different ansatz circuits $A(\theta_1), A(\theta_2), \ldots$; CA's DO step is applying only minimal modifications to the basic solution from the TAPT step, thus leading to well-aligned transpilation results for different circuits.

Figure 4.6 shows the success probability of QAOA on the same four IBM quantum devices. Similar to the approximation ratio, CA produces a good and robust result. For some reference methods (e.g., method 33 on ibm_cairo), the success probability tends to fluctuate more than approximation ratio, sometimes the opposite occurs (e.g., method 24 on ibmq_ehningen). We observed that for cases where the approximation ratio exceeds 0.3, the success probability on the physical computer roughly reaches its simulated noise-free value with the Qasm simulator.

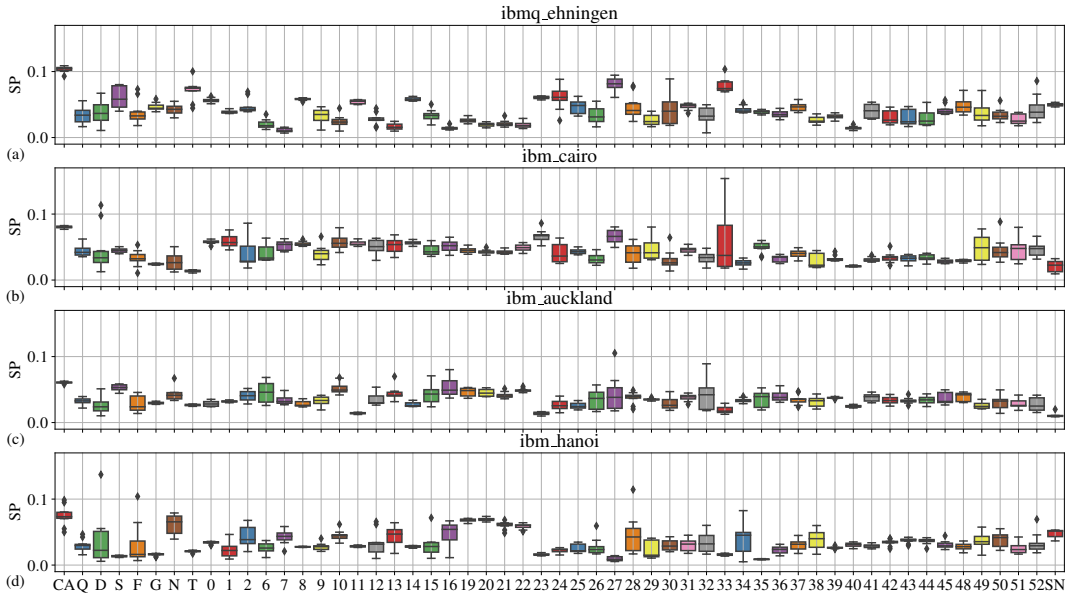

Figure 4.6: Success probability with different transpilation methods on four IBM quantum devices (10 QAOA runs, same symbols as in Figure 4.5).

The percental increase in CX gates after transpilation is reported in Figure 4.7. The numbers differ only minimally among the four quantum computers. Again, CA is consistently best or among the best methods with respect to this metric, while

Table 4.1: Comparison of percentage increase in depth $\Delta d\%$, the total number of gates $\Delta g\%$ and the number of CX gates $\Delta g_{CX}\%$ after transpilation with CA and SF. μ: average value. σ: standard deviation.

Quantum device	$\Delta d\%$				$\Delta g\%$				$\Delta g_{CX}\%$			
	CA		SF		CA		SF		CA		SF	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ibmq_ehningen	126.32	0.0	116.84	31.47	170.00	0.0	104.00	66.78	30.00	0.0	42.00	17.20
ibm_cairo	126.32	0.0	110.00	27.49	170.00	0.0	93.20	63.03	30.00	0.0	48.00	18.87
ibm_auckland	126.32	0.0	116.84	31.47	170.00	0.0	104.00	66.78	30.00	0.0	42.00	17.20
ibm_hanoi	126.32	0.0	116.84	31.47	170.00	0.0	104.00	66.78	30.00	0.0	42.00	17.20

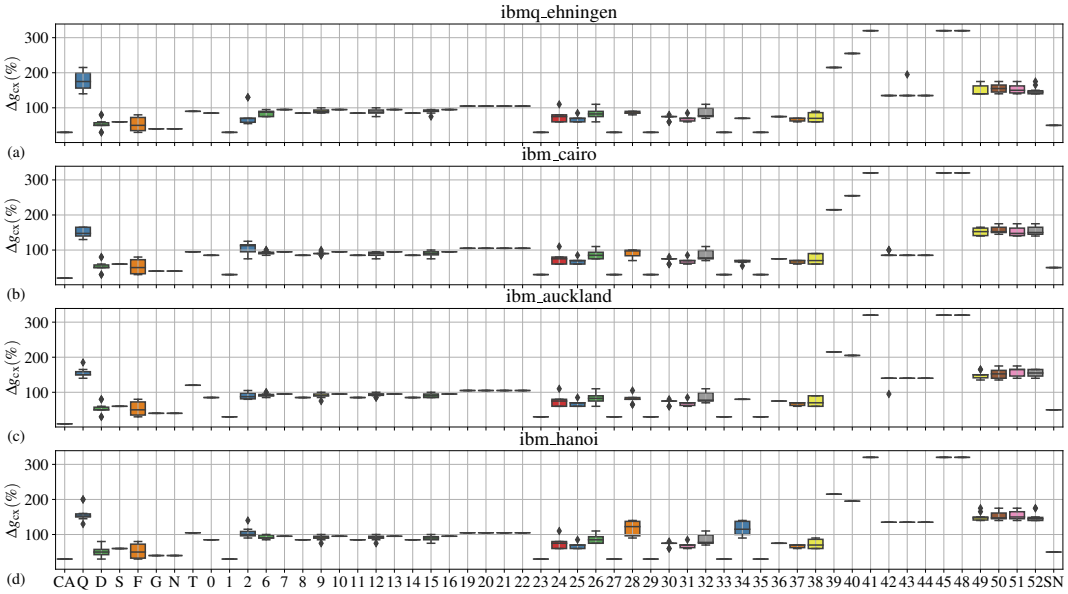

Figure 4.7: Percental increase in the number of CX gates after transpilation (10 QAOA runs, same methods as in Figure 4.5).

other methods produce an increase of up to 330% (more than four times) in CX gates. CA's outcome is also much more stable, since all its ansatz circuits are based on the same high-quality basic solution provided by the TAPT step, whereas, e.g., Qiskit (Q) produces quite different transpilation results for each QAOA run and quantum computer.

Table 7.1 shows a detailed comparison of CA with SF in terms of increase in depth $\Delta d\%$, total number of gates $\Delta g\%$ and number of CX gates $\Delta g_{CX}\%$. The table shows that SF is a good transpilation method with a slightly better $\Delta d\%$, significantly better $\Delta g\%$, but significantly worse $\Delta g_{CX}\%$. SF exposes large-scale variability whereas CA's results are repeatable.

To assess the significance of calibration data and the NAM step, we report in Figure 4.8 the approximation ratio and the success probability of three methods: topology-aware transpilation TA, which is CA that stopped after the TAPT step and did not incorporate any calibration data; SMT-based transpilation SF [216] that maximizes the circuit's fidelity and is used for reference; and the full CA procedure with all its three steps. CA by far outperforms TA and is also consistently better than SF, while both TA and CA are less affected by the variability of the obtained results. We conclude that all three steps of CA are needed to obtain a high-quality solution.

Runtime

One central objective of our calibration-aware transpilation approach is to reduce the runtime of iterative variational algorithms. We now compare CA with SMT-based [216] fidelity maximization (SF). Table 4.2 reports the average runtimes of CA's three steps, their sum (μ) and standard deviation (σ), along with the average runtime and standard deviation for SF, which is not partitioned into steps. It can be seen that the overall runtimes of CA and SF are comparable, suggesting a similar

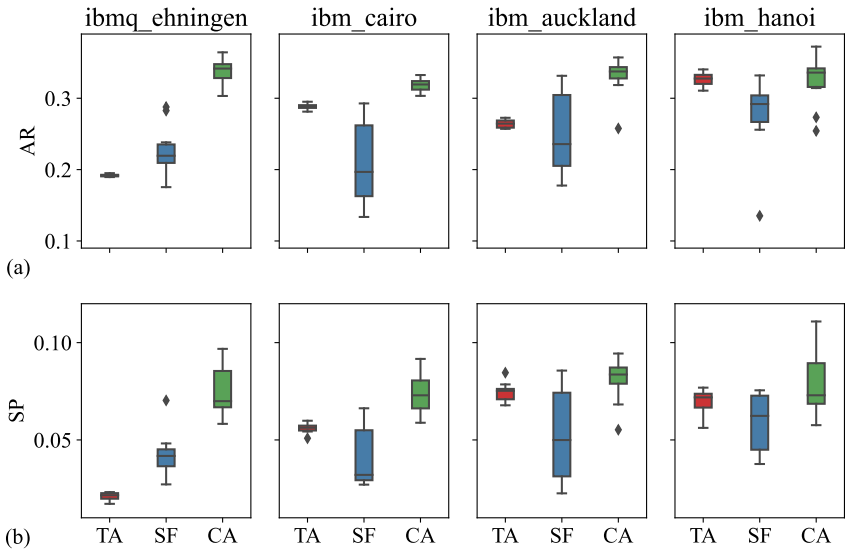

Figure 4.8: Approximation ratio and success probability with IBM quantum devices. TA: topology-aware (noise-unaware) transpilation (TAPT step of CA); SF: SMT-based fidelity maximization; CA: full calibration-aware method (TAPT, NAM, DO).

amount of computational effort being invested. However, CA manages to serialize its computation without a major deterioration of overall runtime.

To illustrate the runtime advantage enabled by CA, assume we have N_A QAOA ansatz circuits to execute. The expectation value of the total runtime for N_A circuits with SF is

$$\mu_{\mathrm{SF}}(N_A) = \mu_{\mathrm{SF}} \times N_A \tag{4.2}$$

with μ_{SF} being the average runtime for one circuit. If the calibration data changes every m iterations, the expected total runtime with CA is

$$\mu_{\mathrm{CA}}(N_A) = \mu_{\mathrm{TAPT}} + \mu_{\mathrm{NAM}} \times \lceil \frac{N_A}{m} \rceil + \mu_{\mathrm{DO}} \times N_A, \tag{4.3}$$

where μ_{TAPT}, μ_{NAM} and μ_{DO} are average runtimes for TAPT, NAM and DO process, respectively. For the special case that calibration data is fixed, we have $m = N_A$ and only the last step DO needs to be executed each time.

Table 4.2: Comparison of runtime of transpilation with CA and SF for 10 QAOA runs (in seconds). μ: average runtime. σ: standard deviation.

	CA					SF	
Quantum device	TAPT	NAM	DO	μ	σ	μ	σ
ibmq_ehningen	18.13	4.41	2.26	24.80	0.22	30.00	3.65
ibm_cairo	23.07	3.59	3.13	29.79	0.26	25.90	1.74
ibm_auckland	22.92	3.44	2.31	28.67	0.14	27.94	5.46
ibm_hanoi	17.44	4.37	2.50	24.31	0.31	29.89	3.20

Table 4.3: Comparison of runtime (in seconds) of CA and SF for $N_A = 5, 100$. CER: Changing error rate after $m = 5$ iterations. FER: Fixed error rate ($m = N_A$). $\Delta\mu$: average time savings compared to SF.

		CA		SF	$\Delta\mu(\%)$	
N_A	Quantum device	CER	FER		CER	FER
5	ibmq_ehningen	33.83	33.83	149.98	-77.44	-77.44
	ibm_auckland	37.91	37.91	139.71	-72.87	-72.87
	ibm_cairo	42.33	42.33	129.48	-67.31	-67.31
	ibm_hanoi	34.32	34.32	149.43	-77.03	-77.03
100	ibmq_ehningen	332.23	248.46	2999.66	-88.92	-91.72
	ibm_auckland	322.67	257.31	2794.29	-88.45	-90.79
	ibm_cairo	408.21	339.99	2589.61	-84.24	-86.87
	ibm_hanoi	355.05	272.03	2988.57	-88.12	-90.90

The projected runtimes of a complete QAOA run with $N_A \in \{5, 100\}$ ansatz circuits are shown in Table 4.3. The data assumes two scenarios: changing error rate (CER), where the calibration data changes after 5 iterations ($m = 5$), and fixed error rate (FER), where the calibration data remains unchanged ($m = N_A$). We see an improvement of up to one order of magnitude due to CA's three-step structure where the most expensive part of the calculation is executed only once.

4.2 Algorithm-oriented qubit mapping

The previous section shows that calibration-aware transpilation strikes a good balance between quality and stability of transpilation results. Experiments on four IBM quantum devices demonstrate the benefits of calibration-aware transpilation for variational algorithms. By accounting for fluctuating noise rates and optimizing the transpilation process accordingly, we can minimize the impact of noise on algorithm performance for specific hardware, resulting in faster execution times and improved fidelity. While the proposed process shows effectiveness in executing quantum algorithms, finding optimal solutions in the TAPT process remains challenging, especially for large-scale problem instances. In this section, we delve into the qubit mapping problem [220, 221] on different subtopologies that determines the assignment of logical qubits in a quantum circuit to the physical qubits with deterministic topology types in the quantum device, a critical step in the transpilation process. The main objective of qubit mapping is to minimize the number of inserted SWAP gates or circuit depth and to maximize circuit fidelity. This problem is identified as NP-hard [222], meaning that there is no known polynomial-time algorithm to find optimal solutions for it, underscoring the necessity for efficient and effective methods to tackle it.

Qubit mapping can be expressed as a mathematical optimization problem and solved using constraint satisfaction techniques. These approaches are called exact methods and have been investigated in various studies [223, 224, 225, 226, 216]. While exact methods provide high quality and stable solutions, their compilation time in-

Under third review in *Physical Review Applied*.

creases exponentially with problem size. In contrast, heuristic approaches [226, 227, 228, 229, 230, 225] prioritize efficiency by providing fast solutions without guaranteeing optimality. Another approach involves constructing swap layers [218, 231, 232, 233, 234] aimed at providing scalable solutions. However, similar to heuristic methods, optimality is not guaranteed.

Here, we propose a strategy called algorithm-oriented qubit mapping (AOQMAP) that aims to bridge the gap between exact and scalable mapping methods by utilizing the inherent structure of algorithms. The proposed strategy follows a two stage approach. First, it maps circuits to subtopologies to meet connectivity constraints. Second, it identifies the optimal qubits for execution using a cost function. Notably, AOQMAP provides both scalable and optimal solutions for variational quantum algorithms with fully connected two qubit interactions on common subtopologies including linear, T-, and H-shaped, minimizing circuit depth. Benchmarking experiments conducted on IBM quantum devices demonstrate significant reductions in gate count and circuit depth compared to Qiskit, Tket, and SWAP network. Specifically, AOQMAP achieves up to an 82% reduction in circuit depth and an average 138% increase in success probability. This scalable and algorithm-specific approach holds the potential to optimize a wider range of quantum algorithms.

The implementation of AOQMAP is publicly available on Github [235].

4.2.1 Methodology overview

This section details our methodology. Figure 4.9 presents algorithm oriented qubit mapping (AOQMAP) flow. The process begins with decomposition of input VQA into a two qubit Hamiltonian. Subsequently, this Hamiltonian is routed onto subtopologies by introducing SWAP gates, ensuring compliance with hardware connectivity constraints. Each of these routed circuits is then decomposed into native basis gates of target QPU and optimized to minimize errors. For verification, a reference circuit is constructed directly from the two-qubit Hamiltonian, matching parameters and gate sequences of routed circuit. The Hellinger distance between output distributions of reference and routed circuits is employed to ensure correctness. Finally, a cost function is utilized to select the optimal set of qubits for execution. After execution on the target QPU, a postselection process can be applied to select the outcomes with different types subtopologies that maximizes algorithm performance, as will be detailed later. This comprehensive flow guarantees effective adaptation, verification, and optimization of quantum circuits within constraints imposed by target hardware topology.

Below, we detail each step. Beginning with identifying three common subtopologies, we present optimal and scalable solutions for routing VQAs on these topologies, focusing on fully connected two-qubit interactions. The circuit verification is then explored to ensure correctness, following a qubit selection strategy that maps subtopology-adapted circuits onto high-quality qubits of QPUs. We conclude by analyzing the optimality and scalability of our proposed methods.

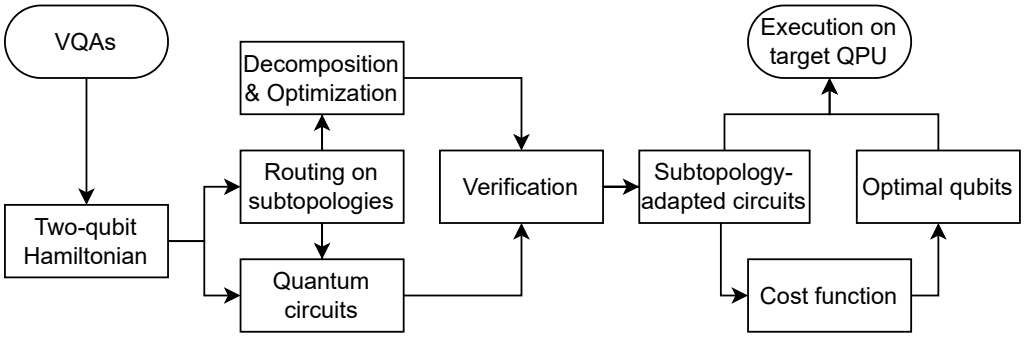

Figure 4.9: Algorithm-oriented qubit mapping (AOQMAP) flow is outlined for mapping VQAs onto a target QPU. The AOQMAP approach differs from traditional qubit mapping methods in that it starts from a two-qubit Hamiltonian, rather than a predefined circuit. The process of AOQMAP involves two main steps: (1) adapting VQA to subtopologies of target QPU; and (2) selecting the optimal mapping scheme to implement subtopology-adapted circuits. The adaptation process ensures algorithm excitability, while the mapping process guarantees the use of high-quality qubits for execution. By first adapting circuits and then choosing an optimal mapping scheme, AOQMAP minimizes errors and optimizes algorithm performance.

4.2.2 Identification of subtopologies

We identify three prevalent subtopologies within IBM QPUs: linear, T-, and H-shaped configurations. Given the NP-hard nature of qubit mapping, we focus on symmetric subtopologies to facilitate the development of optimal and scalable solutions. Linear configurations serve as a fundamental topology, while T- and H-shaped topologies (formally defined in Section 4.2.3) represent the simplest symmetric extensions. Moreover, to achieve optimality and scalability, we will employ exact methods for small-scale problems and then develop scalable approaches (see Section 4.2.3). The symmetric T- and H-shaped subtopologies enable efficient analysis of solutions and offer potential for extending these solutions to other topologies. Future research will explore these extensions. Additionally, our analysis of 27-qubit IBM QPUs reveals that these subtopologies are dominant for the problem sizes considered. Specifically, we exhaustively identify all possible subtopologies of up to seven qubits and calculate their corresponding layouts within the QPU using mapomatic [236]. Figure 4.10 illustrates these subtopologies, and Table 4.4 summarizes the numbers of their corresponding layouts. As shown in Table 4.4, linear topologies exhibit the highest number of layouts, followed by T- and H-shaped configurations.

Table 4.4: Number of layouts within a 27-qubit QPU that enable the execution of circuits satisfying the connectivity constraints of subtopologies shown in Figures 4.10(a-o).

Topology type	3	4	5	6	7
Linear	74 (a)	80 (b)	100 (d)	104 (f)	132 (j)
T-shaped		48 (c)	36 (e)	64 (g)	48 (k)
H-shaped					56 (l)
T-variant				24 (h-i)	44 (m-n), 12 (o)

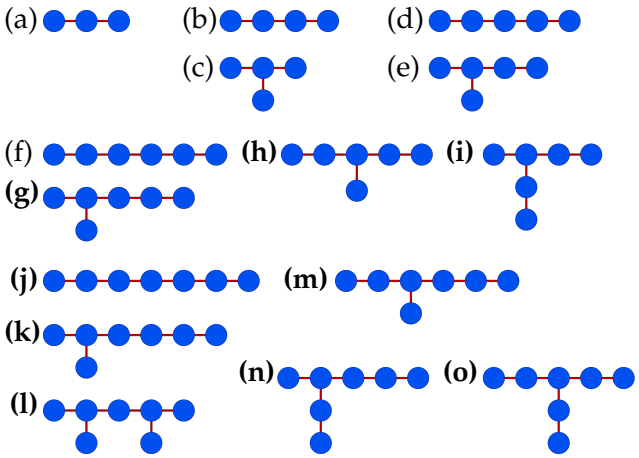

Figure 4.10: All possible subtopologies within the heavy-hex topology for varying numbers of qubits, ranging from 3 to 7. The most common subtopology on a 27-qubit topology (Figure 2.3(a)) is linear, followed by T-shaped and then H-shaped.

To identify the optimal subtopology among the three candidates, a postselection strategy is employed. This approach is essential when the optimal subtopology is not immediately evident. The cost function, which will be detailed in Section 4.2.4, incorporates gate fidelity to select the optimal set of qubits for executing subtopology-adapted circuits. In addition to enabling the selection of qubits for a given topology, this cost function allows for the distinction between various types of subtopologies. However, each subtopology exhibits unique crosstalk effects that are not accounted for in the calibration data of IBM QPUs. Furthermore, as will be demonstrated subsequently, increased qubit connectivity reduces the number of two-qubit gates required but increases circuit depth, creating a trade-off between circuit fidelity and execution time. The cost function based solely on gate fidelity is insufficient for accurately assessing the quality of a circuit. To overcome these limitations, a postselection procedure is performed after executing circuits corresponding to each subtopology on QPUs. The criterion for postselection is based on minimizing the expectation value of the problem Hamiltonian, which allows us to identify the most effective subtopology. This will be investigated in Section 4.2.7. This method will be further demonstrated through the application of AOQMAP to a digitized counterdiabatic quantum optimization algorithm in Chapter 7.

4.2.3 Subtopology-aware circuit adaptation

This section introduces the methodology for subtopology aware circuit adaptation on NISQ devices. We focus on linear, T-, and H-shaped configurations, and develop strategies for adapting VQAs to each subtopology. In this study, we focus on the QAOA applied to dense portfolio optimization problems, as described in Section 3.2.1, where each qubit necessitates interaction with all other qubits.

To achieve optimal qubit mapping solutions, a comprehensive analysis of problem and mixer Hamiltonians described in Eqs. (3.10) and (3.11) is essential. Notable observations guide our approach: (i) The two-qubit gates in Eq. 3.10 exclusively involve ZZ interactions. Their summation form implies that the order of application does not

affect the outcome, allowing flexibility in mapping H_c; (ii) The single rotation gates R_Z and R_X in H_c and H_m can be independently applied to each qubit without affecting others. Hence, qubit mapping is unnecessary for these gates; (iii) For high depths, as depicted in Eq. 3.12, a fixed gate arrangement for two-qubit gates is required at each depth to maintain an equivalent H_c. However, since all ZZ gates commute with each other, they can be assigned in each depth with an arbitrary gate arrangement. This implies that the ZZ gate arrangement needed for implementing QAOA at each depth can be arbitrary, providing additional flexibility in qubit mapping. By leveraging these observations, we can formulate a tailored mapping method for quantum algorithms to optimize their performance, minimizing the number of CX gates, reducing circuit depth, and thereby maximizing the algorithm's overall performance.

In the upcoming sections, we explore qubit mapping solutions on three subtopologies, emphasizing QAOA with all-to-all connected two-qubit gate interactions. Although our analysis focuses on this specific instance of VQAs, the implications extend to algorithms with analogous features. Specifically, we underscore the relevance of our findings to Hamiltonian with partially connected interactions, VQE, and other NISQ devices in Section 4.2.6.

Linear subtopology

In quantum computing, linear subtopology arranges qubits sequentially in a one-dimensional configuration, forming a linear chain or arrangement. To determine optimal mapping on this subtopology, we initially employ an exact method aimed at minimizing the circuit depth [216]. This decision follows the same principle in Section 4.1, which uses circuit depth as the objective function instead of fidelity, thereby not requiring access to real devices at this stage. However, instead of mapping the entire algorithm as in Section 4.1, we focus here solely on mapping the ZZ gates in Eq. 3.10, treating them as single entities and avoiding the decomposition into two CX gates to streamline the mapping process. Additionally, we narrow our attention to the first QAOA depth, significantly reducing the effort required to identify a solution.

The routing solution for ZZ gates in a five-qubit QAOA on linear topology is illustrated in Figure 4.11(a). Assuming an arbitrary initial qubit order $[a, b, c, d, e]$, after each swap layer, qubit orders evolve as follows: $[a, c, b, e, d]$, $[c, a, e, b, d]$, and $[c, e, a, d, b]$. Since each qubit needs to interact with all other qubits, ZZ gate acts on following qubit pairs: for qubit a, (a, b), (a, c), (a, d), and (a, e); for qubit b, (b, c), (b, d), and (b, e); for qubit c, (c, d), and (c, e); and for qubit d, (d, e). Our analysis reveals that all ten ZZ gates required in a five-qubit QAOA can be executed on a linear subtopology, regardless of the initial qubit order. This observation is highly beneficial for achieving scalability, as it facilitates straightforward extension to high QAOA depths by repeating these swap layers. Similarly, as depicted in Figure 4.11(b), the fifteen ZZ gates required for six-qubit QAOA can be implemented using four swap layers. We observe that in an n-qubit QAOA, ZZ gates are organized into n layers to minimize circuit depth. The swap layers, excluding the first and last layers, are positioned after each ZZ layer. As shown in Figure 4.11(c), due to the cancellation of CX gates between ZZ and SWAP gates, each SWAP gate following a ZZ gate introduces only one additional CX gate. The optimal routing solutions of ZZ gates in QAOA on linear subtopology exhibit a structure similar to the SWAP

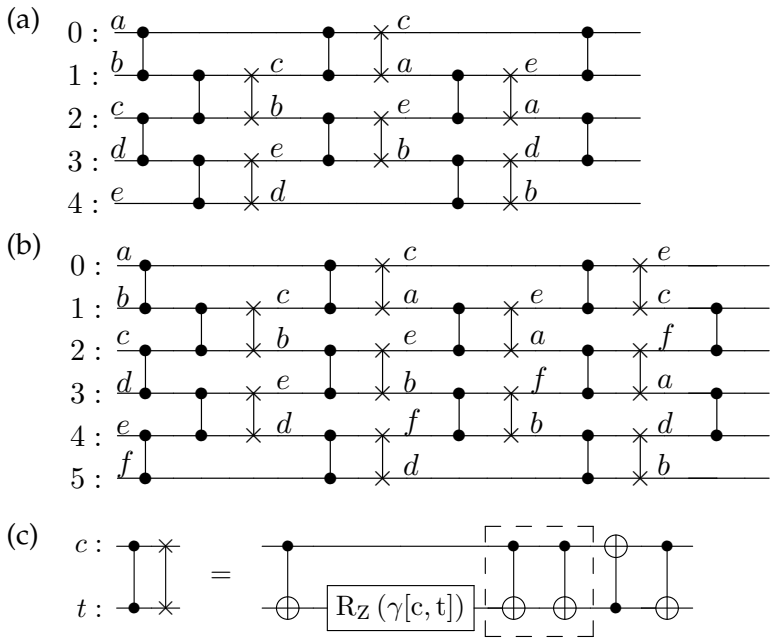

Figure 4.11: Routing solutions of ZZ gates in QAOA at depth $p = 1$ with (a) five and (b) six qubits. (c) ZZ gate on qubit pair (c, t) with rotation angle $\gamma[c, t]$, followed by a SWAP gate, with the box showing CX gate cancellation between ZZ and SWAP gates.

network presented in [218], with the absence of the first and last swap layers. This similarity enables the extension of solutions to an arbitrary number of qubits. The dispensability of the first and last swap layers stems from the flexibility to adjust initial qubit order and measurement order to eliminate SWAP gates in the first and last ZZ layers, leveraging commutativity of ZZ and SWAP gates. Once two-qubit gates are mapped, we can construct the entire QAOA circuit.

Algorithm 4.3 presents pseudocode for mapping QAOA on linear subtopology. The reconstruction process starts with an initialized qubit order $O = \{0, 1, ...i, ..., j, n-1\}$ for n qubits. We first prepare an initial state $|\psi_0\rangle$ by applying a Hadamard gate to each qubit. Then, we apply ZZ or ZZ-SWAP gate layer continuously according to the solutions presented in Figure 4.11. The order of qubits i and j is exchanged only when a SWAP gate is applied to qubit pair (i, j). To improve algorithm efficiency, it is crucial to optimize ZZ and ZZ-SWAP gates using strategies such as gate cancellation or pulse level optimization techniques, which will be explored in Chapter 5. Finally, R_Z gates in problem Hamiltonian and R_X gates in mixer Hamiltonian are implemented on qubits with appropriate parameters. Figure 4.12 shows the resulting circuit of a five-qubit QAOA at depth $p = 1$. All ZZ, R_Z, and R_X gates are executed according to the current qubit order, followed by measurement of qubits.

As demonstrated, an arbitrary qubit order can implement all required ZZ gates in QAOA with n qubits using $n - 2$ swap layers. A solution of depth p can be obtained by repeating p times swap layers in depth $p = 1$ circuit. Inserting these swap layers consecutively, the initial order returns after $2n$ swap layers. Since each QAOA depth introduces $n - 2$ swap layers, circuit repeats every $\text{lcm}(2n, n-2)/(n-2)$ depths,

Algorithm 4.3: AOQMAP for QAOA on linear subtopology

Input: Number of qubits n, QAOA depth p, Parameters $\gamma[c,t]$, $\alpha[i]$, and $\beta[j]$ of gates ZZ on qubit pair (c,t), R_Z on qubit i, and R_X on qubit j, respectively, where $c,t,i,j \in \{0,...,n-1\}$ and $c < t$

Output: Circuit satisfying connectivity constraints

Function `ApplyZZGate`(O, i, j)
  Apply ZZ($\gamma[O[i],O[j]]$) on (i, j)
end
Function `ApplyZZSWAPGate`(O, i, j)
  Apply ZZ($\gamma[O[i],O[j]]$) on (i, j)
  Apply SWAP on (i, j)
  $O[i] \leftrightarrow O[j]$ `// Exchange qubit order`
end
$O \leftarrow \{0,1,...,n-1\}$`// Initialize the qubit order`
Prepare the initial state $|0\rangle^{\otimes n}$
Apply $H^{\otimes n}$
while $p > 0$ **do**
  $s \leftarrow 0$
  while $s < n$ **do**
    for $q := 0$ *to* $n-1$ *step* 2 **do**
      if $s == 0$ *or* $s == n-1$ **then**
        `ApplyZZGate`(O, q, $q+1$)
      else
        `ApplyZZSWAPGate`(O, q, $q+1$)
      end
    end
    $s \leftarrow s+1$
    if $s < n$ **then**
      for $q := 1$ *to* $n-1$ *step* 2 **do**
        if $s == 0$ *or* $s == n-1$ **then**
          `ApplyZZGate`(O, q, $q+1$)
        else
          `ApplyZZSWAPGate`(O, q, $q+1$)
        end
      end
    end
    $s \leftarrow s+1$
  end
  for $k := 0$ **to** n **do**
    Apply $R_Z(\alpha[O[k]])$ on k
    Apply $R_X(\beta[O[k]])$ on k
  end
  $p \leftarrow p-1$
end
Measure the qubits ($[0,...,n-1] \rightarrow [O[0],...,O[n-1]]$)

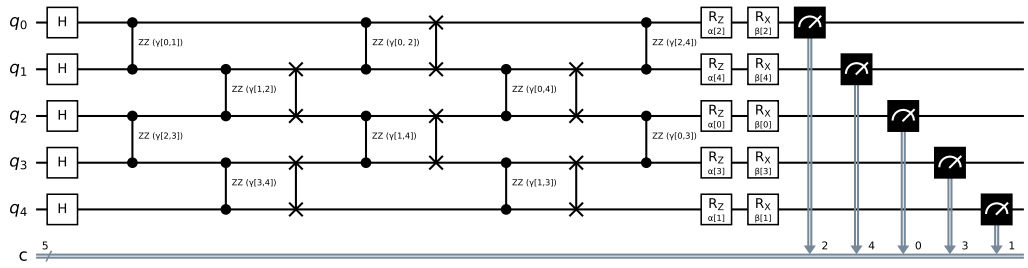

Figure 4.12: Resulting circuit of five-qubit QAOA for portfolio optimization on a linear subtopology using AOQMAP approach. This circuit is further decomposed and optimized for the target QPU. The optimal mapping scheme is then selected for execution based on the QPU's noise information.

where $\text{lcm}(2n, n-2)$ is the least common multiple of $2n$ and $n-2$. For odd numbers of qubits, one repetition of swap layers in depth $p = 1$ circuit demonstrates symmetry, and at depth two, the final qubit order returns directly to the initial qubit order. For even numbers of qubits, one repetition of such swap layers demonstrates alternating odd and even-numbered swap layers, resulting in circuits that repeat a subcircuit with the same gate arrangement every $\text{lcm}(2n, n-2)/(n-2)$ layers. Since all ZZ gates commute with each other, these two constructions of high depth solutions are equivalent. However, we can utilize mirror symmetry of swap layers to construct high depth circuits for even numbers of qubits and obtain circuits repeating every two depths. Additionally, mirror symmetry can also be utilized to construct algorithms with partially connected two qubit gates, which we discuss in Section 4.2.6.

T-shaped subtopology

We now analyze solutions obtained by exact method [216] for two-qubit gates in QAOA with five and six qubits on T-shaped subtopology. A T-shaped subtopology features a central qubit at the apex of "T" shape, serving as the primary qubit. The arms of T-shaped topology consist of one or more qubits that are linearly connected to the central qubit. The qubits in arms typically do not have direct interactions with each other. We note that the minimum number of qubits on a T-shaped subtopology is four. Figure 4.13(a) shows the definition of T-shaped topology for n qubits with qubit 2 as the center qubit. Consider an arbitrary initial qubit order $[a, b, c, d, e]$ for the solution of five-qubit, as shown in Figure 4.13(b). Following each swap layer, the qubit order evolves sequentially: $[a, b, d, c, e]$, $[d, b, a, e, c]$, and $[d, b, e, a, c]$. This sequence allows us to execute all required ten ZZ gates for five-qubit QAOA. Similarly, the four swap layers in six-qubit QAOA are capable of performing the necessary fifteen ZZ gates, as shown in Figure 4.13(c).

Algorithm 4.4 outlines procedure for generating n swap layers for n-qubit QAOA on T-shaped subtopology. As depicted in Figure 4.13(c), the initial swap layer starts at center qubit 2 and consists of SWAP gates on qubit pairs $\{(2,3),(4,5),...,(n_{\text{odd}}-1, n_{\text{odd}})\}$ with $n_{\text{odd}} = (n-1) - 1 + (n-1) \bmod 2$. This layer alternates with another layer that starts from two different qubits in short arms of a T-shaped structure, forming two distinct layers $\{(0,2),(3,4),...,(n_{\text{even}}-1, n_{\text{even}})\}$ and $\{(1,2),(3,4),...,(n_{\text{even}}-$

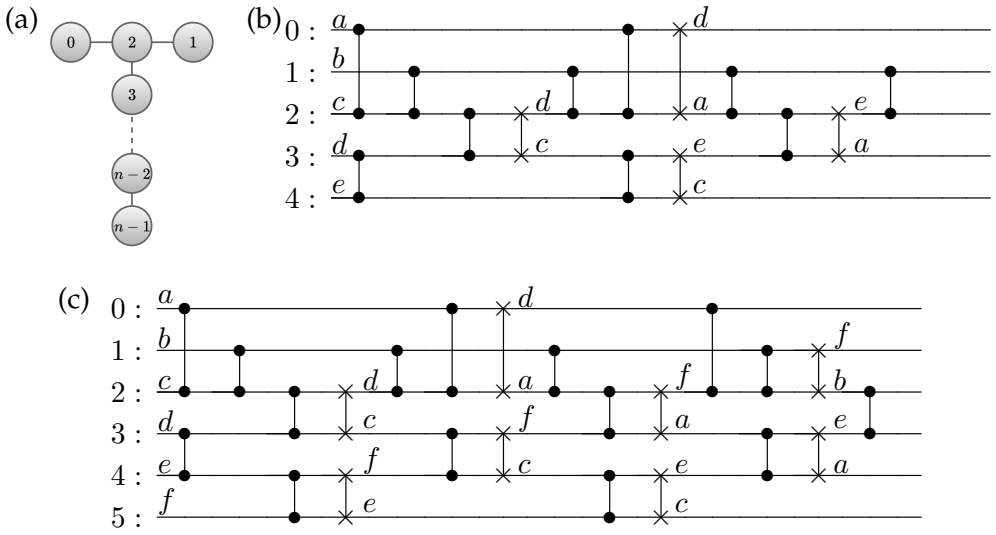

Figure 4.13: Definition of T-shaped topology and corresponding routing solutions of ZZ gates in QAOA at depth $p = 1$. (a) T-shaped topology defined for n qubits with qubit 2 as the center qubit. The connectivity of this subtopology for n qubits is given by $\{(0,2),(1,2),(2,3),(3,4),...,(n-2,n-1)\}$. (b) Five-qubit routing solution. (c) Six-qubit routing solution.

$1, n_{\text{even}})\}$ with $n_{\text{even}} = (n-1) - (n-1) \bmod 2$. After constructing n swap layers, all required ZZ gates can be implemented on connected qubit pairs. Algorithm 4.5 presents pseudocode for AOQMAP applied in n-qubit QAOA at depth $p = 1$ on T-shaped subtopology. We begin the process by initializing a qubit order $O = \{0, 1, ..., i, ..., j, n-1\}$ for n qubits, with qubit 2 being center qubit. This initial qubit order enables executing one layer of ZZ gates. One possible option is gates on qubit pairs of the center qubit and two qubits in short arm, and gates on qubit pairs starting with the first qubit in long arm. For instance, the first ZZ layer in six-qubit QAOA consists of $\{(0,2),(1,2),(3,4)\}$. The second layer starts with center qubits and proceeds to qubits on long arm to form ZZ gates on qubit pairs $\{(2,3),(4,5)\}$, followed by the first swap layer that introduces new qubit order and provides opportunity for additional gate implementation. We first execute remaining ZZ gates that are not on qubit pairs of the second swap layer. Then, we implement remaining ZZ gates on qubit pairs of the second swap layer, followed by SWAP gates. This process ensures that ZZ gate is positioned immediately before SWAP gate on the corresponding qubit pair, thereby enabling CX gate cancellation. We maintain this iterative process until all ZZ gates are implemented.

Similar to linear subtopology, there are two solutions for higher depths: repeating swap layers in depth $p = 1$ circuit and leveraging mirror symmetry of swap layers. Alternating swap layers and their mirrors causes gates in QAOA at depth p to invert gates at the previous depth. For commuting gates such as the ZZ gates used in QAOA, this reversal has no impact on algorithm performance. However, for noncommuting gates, this inversion may suppress Trotter errors [237, 238], making

Algorithm 4.4: Swap layers on T-shaped subtopology

Input: Number of qubits n, Connectivity of T-shaped topology defined in Figure 4.13(a)
Output: List of swap layers S
begin
  $n_{\text{odd}} \leftarrow (n-1) - 1 + (n-1) \bmod 2$;
  $n_{\text{even}} \leftarrow (n-1) - (n-1) \bmod 2$;
  $S \leftarrow$ empty List;
  $j \leftarrow 0$;
  while $j < n$ **do**
    $S[j] \leftarrow S[j] \cup [(2,3),(4,5),...,(n_{\text{odd}}-1, n_{\text{odd}})]$;
    if $++j \geq n$ **then** break;
    $S[j] \leftarrow S[j] \cup [(0,2),(3,4),...,(n_{\text{even}}-1, n_{\text{even}})]$;
    if $++j \geq n$ **then** break;
    $S[j] \leftarrow S[j] \cup [(2,3),(4,5),...,(n_{\text{odd}}-1, n_{\text{odd}})]$;
    if $++j \geq n$ **then** break;
    $S[j] \leftarrow S[j] \cup [(1,2),(3,4),...,(n_{\text{even}}-1, n_{\text{even}})]$;
    if $++j \geq n$ **then** break;
  end
end

AOQMAP promising for optimizing other algorithms. Compared to solutions on linear topology that exhibit a fixed number of swap layers, solutions on T-shaped topology require at least $n-2$ but no more than n swap layers for n-qubit QAOA. If ZZ gates remain after $n-2$ swap layers, SWAP gates in the penultimate and/or last layer become indispensable. It is worth noting that SWAP gates at the end of a circuit can be eliminated, as no remaining two-qubit gates require the introduction of new qubit orders.

The T-shaped subtopology affords enhanced qubit connectivity for the central qubit, reducing required SWAP gates but increasing circuit depth. This implies that T-shaped subtopology is advantageous when the algorithm's fidelity is a primary factor affecting performance. Conversely, linear subtopology is more effective for larger circuit sizes where schedule duration is paramount.

H-shaped subtopology

The H-shaped subtopology shares similarities with T-shaped, but has two central qubits at each end instead of one. The horizontal segments of "H" serve as a bridge connecting central qubits. To implement an H-shaped subtopology, a minimum of six qubits is required. We define the connectivity of an H-shaped subtopology with center qubits 2 and $n-3$ as $\{(0,2),(1,2),(2,3),...,(n-4,n-3),(n-3,n-2),(n-3,n-1)\}$ for n qubits.

Algorithm 4.6 presents pseudocode for generating n swap layers that enable the implementation of Hamiltonian with fully connected two-qubit gates on H-shaped topology. All n swap layers are necessary for this implementation. For odd numbers of qubits, the first and third swap layers differ in initial connections of $(0,2)$ and $(1,2)$, which alternate. Similarly, the second and fourth layers differ in final connections of $(n_{\text{even}}-2, n_{\text{even}}-1)$ and $(n_{\text{even}}-2, n_{\text{even}})$. For even numbers of qubits, the first and

Algorithm 4.5: AOQMAP for QAOA at depth $p = 1$ on T-shaped subtopology

Input: Number of qubits n, Parameters $\gamma[c,t]$, $\alpha[i]$, and $\beta[j]$ of gates ZZ on qubit pair (c,t), R_Z on qubit i, and R_X on qubit j, respectively, where $c,t,i,j \in \{0,...,n-1\}$ and $c < t$
Output: Circuit satisfying connectivity constraints
$O \leftarrow \{0,1,...,n-1\}$// `Initialize the qubit order`
$L \leftarrow$ List of all ZZ gates
$E \leftarrow$ List of connected edges on T-shaped subtopology
$E_O \leftarrow [(O[r],O[s])$ for $(r,s) \in E]$
$S \leftarrow$ List of n swap layers // `Algorithm 4.4`
$S_O \leftarrow [(O[r],O[s])$ for $(r,s) \in S[k]]$
Function `ApplyZZGate`(O, c, t)
  Apply ZZ($\gamma[c,t]$) on $(O.\texttt{index}(c), O.\texttt{index}(t))$
end
Function `ApplySWAPGate`(O, i, j)
  Apply SWAP on (i,j)
  $O[i] \leftrightarrow O[j]$ // `Exchange qubit order`
end
Prepare the initial state $|0\rangle^{\otimes n}$
Apply $H^{\otimes n}$
for $k := 0$ *to* n **do**
  if L is empty **then** continue
  foreach ZZ($\gamma[c,t]$) $\in L$ **do**
    if $(c,t) \in E_O$ and $(c,t) \notin S_O$ **then**
      `ApplyZZGate`(O, c, t)
      L.`remove`(ZZ($\gamma[c,t]$))
    end
  end
  if L is empty **then** continue
  foreach $(i,j) \in S[k]$ **do**
    $c,t = O[i], O[j]$
    if $c > t$ **then**
      $c \leftrightarrow t$ // `Exchange`
    end
    if ZZ($\gamma[c,t]$) $\in L$ **then**
      `ApplyZZGate`(O, c, t)
      L.`remove`(ZZ($\gamma[c,t]$))
      if L is empty **then** continue
      `ApplySWAPGate`(O, i, j)
    end
  end
end
foreach SWAP on (i,j) located at the circuit end **do**
  Remove SWAP
  $O[i] \leftrightarrow O[j]$
end
for $k \leftarrow 0$ **to** n **do**
  Apply $R_Z(\alpha[O[k]])$ on k
  Apply $R_X(\beta[O[k]])$ on k
end
Measure the qubits $([0,...,n-1] \rightarrow [O[0],...,O[n-1]])$

third layers are identical, which include connections between two center qubits. In contrast, the second and fourth layers differ in the first and last connections, where (1,2) and $(n_{\text{odd}} - 2, n_{\text{odd}} - 1)$ are for the second layer and (0,2) and $(n_{\text{odd}} - 2, n_{\text{odd}})$ are for the fourth layer. This alternating pattern of connections between neighboring swap layers efficiently constructs the set of minimized SWAP gates for VQAs with arbitrary numbers of qubits mapped to H-shaped topology.

The procedure to construct a depth-one circuit with arbitrary qubit number n on H-shaped subtopology is similar to Algorithm 4.5 for T-shaped subtopology. The list of connected edges E is updated to reflect the H-shaped connectivity, and the list of n swap layers S is generated according to Algorithm 4.6 for H-shaped subtopology. Similarly, repeating the same swap layers at depth $p = 1$ circuit or leveraging mirror symmetry extends solutions to high depths. Compared to linear and T-shaped topologies, H-shaped topology enables additional connections for two center qubits that reduce required SWAP gates but increase circuit depth.

Verification of circuits

After obtaining routing solutions on various subtopologies, the circuit is decomposed into basis gates of target QPUs, followed by an optimization step aiming to reduce redundant gates in circuit and improve fidelity. This decomposition and optimization process is performed using Qiskit transpiler [208] with default settings (optimization level 1). To maintain effectiveness, it is necessary to enforce the same or reduced number of CX gates during decomposition and optimization. Moreover, we verify the accuracy of adapted circuit by comparing output probability distributions of adapted and original circuits with Hellinger distance [239], which ranges from 0 to 1, with 0 indicating identical distributions. The Hellinger distance of two probability distributions is defined in Eq. 3.9. The verification process begins by simulating each circuit using a simulator on a classical computer to obtain the expected output state in the absence of noise. In this study, we employ the Qasm simulator provided by Qiskit to execute original and mapped circuits. We then calculate the Hellinger distance between probability distributions of simulated output states.

Table 4.5 reports the Hellinger distance between original and mapped circuits generated using AOQMAP on linear subtopology (AOQMAP-L). The benchmark circuits are QAOA for portfolio optimization with 3, 4, 5, 6, and 10 qubits and depths ranging from 1 to 7. Each data point is averaged over 50 circuit repetitions, with a standard error of the means that is more than 15 orders of magnitude smaller than the mean. The Hellinger distance increases with qubit number but remains two orders of magnitude smaller than one. Values greater than 0 are due to measurement noise or shot noise generated when executing circuits using the Qasm simulator. This observation confirms that the mapped circuits are consistent with the originals across the problem instances studied.

While we directly execute adapted and original circuits in the ideal case, without considering noise, and employ Hellinger distance to verify correctness, incorporating noise models is crucial for a more accurate assessment of AOQMAP's performance in real-world scenarios. However, verifying quantum circuits under noise is challenging due to the exponential scaling of traditional strategies such as quantum state tomography [240, 241, 242, 243], quantum process tomography [244, 245], and randomized

Algorithm 4.6: Swap layers on H-shaped subtopology

Input: Number of qubits n
Output: List of swap layers S
begin
  $n_{\text{odd}} \leftarrow (n-1) - 1 + (n-1) \bmod 2$;
  $n_{\text{even}} \leftarrow (n-1) - (n-1) \bmod 2$;
  $S \leftarrow$ empty List;
  $j \leftarrow 0$;
  if $n \bmod 2 \neq 0$ **then**
    while $j < n$ **do**
      $S[j] \leftarrow S[j] \cup [(0,2),(3,4),...,(n_{\text{odd}}-2, n_{\text{odd}}-1)]$;
      if $++j \geq n$ **then** break;
      $S[j] \leftarrow S[j] \cup [(2,3),(4,5),...,(n_{\text{even}}-2, n_{\text{even}}-1)]$;
      if $++j \geq n$ **then** break;
      $S[j] \leftarrow S[j] \cup [(1,2),(3,4),...,(n_{\text{odd}}-2, n_{\text{odd}}-1)]$;
      if $++j \geq n$ **then** break;
      $S[j] \leftarrow S[j] \cup [(2,3),...,(n_{\text{even}}-4, n_{\text{even}}-3),(n_{\text{even}}-2, n_{\text{even}})]$;
      if $++j \geq n$ **then** break;
    end
  else
    while $j < n$ **do**
      $S[j] \leftarrow S[j] \cup [(2,3),(4,5),...,(n_{\text{even}}-2, n_{\text{even}}-1)]$;
      if $++j \geq n$ **then** break;
      $S[j] \leftarrow S[j] \cup [(1,2),(3,4),...,(n_{\text{odd}}-2, n_{\text{odd}}-1)]$;
      if $++j \geq n$ **then** break;
      $S[j] \leftarrow S[j] \cup [(2,3),(4,5),...,(n_{\text{even}}-2, n_{\text{even}}-1)]$;
      if $++j \geq n$ **then** break;
      $S[j] \leftarrow S[j] \cup [(0,2),...,(n_{\text{odd}}-4, n_{\text{odd}}-3),...,(n_{\text{odd}}-2, n_{\text{odd}})]$;
      if $++j \geq n$ **then** break;
    end
  end
end

benchmarking [246, 247, 248] with the size of qubits to be characterized. Moreover, for large numbers of qubits, directly executing reference and original circuits with a Qasm simulator and comparing their probability distributions becomes inaccurate and impractical. Current research on verifying quantum circuits typically involves decomposing the circuit into subcircuits and quantifying the difference between noisy and ideal subcircuit outputs using metrics such as total variation distance [155, 249] and fidelity [250]. The resulting circuits produced by AOQMAP for VQAs on linear, T-, and H-shaped topologies maintain their structure with increasing qubit numbers and VQA depths, enabling efficient identification of subcircuits. These specifically structured circuits can also be employed to test the performance of NISQ devices as they minimize the impact of compilation quality on their performance and contain different types of topologies to capture more noise information such as crosstalk, providing valuable insights into the scalability of VQAs on real devices. Future research can verify subtopology-adapted VQAs on noisy quantum devices and benchmark the scalability of VQAs on real devices.

Table 4.5: Hellinger distance between original and mapped circuits with AOQMAP-L using QAOA at a depth ranging from 1 to 7.

p	1	2	3	4	5	6	7
3-qubit	0.0045	0.0036	0.0033	0.005	0.0036	0.0048	0.0051
4-qubit	0.0044	0.0064	0.0057	0.0051	0.005	0.006	0.0044
5-qubit	0.0087	0.0067	0.0086	0.0103	0.0072	0.0095	0.0071
6-qubit	0.0138	0.0117	0.0138	0.0102	0.0123	0.0131	0.0124
10-qubit	0.0120	0.0127	0.0137	0.0127	0.0106	0.0158	0.0123

4.2.4 Mapping of subtopology-adapted circuits

The next crucial step is to map these subtopology-aware circuits onto target quantum device by selecting an optimal qubit mapping scheme. To identify a high-quality qubit set, we utilize a cost function denoted as C, which takes into account the error rate of each gate and measurement in the circuit and is given by

$$C = 1 - \prod_{i=1}^{N_g}(1 - p_{g_i}) \prod_{j=1}^{N_m}(1 - p_{m_j}), \tag{4.4}$$

where N_g is number of gates in circuit, p_{g_i} is error rate of the ith gate, N_m is number of measurements, and p_{m_j} is error rate of the jth measurement. A lower value of C indicates a higher estimated fidelity. The error rates of gates and measurements can be obtained from device calibration data. To select optimal qubits, we first utilize mapomatic [236] to identify all layouts on QPU matching connectivity of adapted circuit. We evaluate circuit on each layout, choosing the one with the highest fidelity for execution. Since IBM QPUs like Figure 2.3 contain more linear and T-shaped subtopologies than H-shaped, we focus on mapping to linear and T-shaped in our demonstrations. Additionally, H-shaped subtopologies on IBM QPUs are limited to specific qubit numbers such as 7, 9, 11, 13, and 15, whereas linear and T-shaped provide more flexibility.

The methodology presented here closely resembles the NAM process outlined in Section 4.1. However, we depart from the Qiskit-based approach to circuit matching and instead utilize mapomatic. While Qiskit's N-trial matching offers an efficient means of qubit assignment, it does not guarantee optimal circuit fidelity due to the inherent randomness in its transpilation process. By contrast, mapomatic's exhaustive search of all possible layouts and subsequent evaluation of a cost function ensures the identification of the most optimal configuration.

We adopt a postselection process to determine between linear and T-shaped mappings instead of relying solely on cost function evaluation. Specifically, we execute circuits on both subtopologies and use postselection to select the circuit corresponding to the minimum expectation value of the problem Hamiltonian. While fidelity estimates from the cost function are valuable, factors such as gate scheduling can also affect algorithm performance (see Chapter 5). Moreover, different gate arrangements may be equivalent in the absence of noise but behave differently under actual hardware noise. Therefore, we further introduce an additional variant called AOQMAP-LS

which performs AOQMAP on linear subtopology with mirror symmetric swap layers in depth $p = 1$.

4.2.5 Optimality and scalability

The optimal swap strategy on line topology with n qubits has been proven to necessitate $n - 2$ total swap layers to achieve fully connected two qubit gates [233]. This paper utilizes an exact method to obtain optimal solutions for small scale instances. By repeating swap layers or utilizing their symmetry, these solutions can be extended to any depth p. The analysis of these solutions allows for scalability in terms of the number of qubits. Similarly, we are able to obtain solutions for T- and H-shaped topologies for VQAs with arbitrary depths and numbers of qubits. For Hamiltonian with partially connected two-qubit interactions, scalability in terms of depths is maintained through mirror symmetry. However, in terms of qubit numbers, we need to optimize the initial qubit mapping to minimize CX gate count, which we discuss in Section 4.2.6.

The scalability of AOQMAP to large quantum systems, such as those comprising 100 qubits, is essential for its practical applicability. To compare scalability, we measure compilation time to obtain optimal mappings that minimize circuit depth using an exact approach proposed in [216]. The original work on the compiler in [216] focused on an entire topology rather than a specific substructure. Therefore, we map QAOA circuits onto the entire topology of a 27-qubit IBM QPU, as depicted in Figure 2.3(a). For three-qubit QAOA circuits, the compilation time of an exact algorithm increases exponentially with higher depth p, expanding from seconds at depth 1 to over 5 days at depth 7. Similarly, four-qubit compilation requires 13 seconds for depth 1 but grows substantially to over 15 hours for depth 2 and more than a week for depth 3. Furthermore, increasing qubit number from 3 to 9 lengthens compilation from 3 seconds to over 41 hours. In contrast, our approach intrinsically generalizes solutions for arbitrary depth and number of qubits without requiring any computational effort.

AOQMAP achieves both optimality and scalability by analyzing the optimal solutions of small QAOA instances obtained using the exact method [216] in conjunction with the scalable solutions provided by SWAPNK [218]. Depth optimality is achieved by setting the objective to minimize circuit depth in the exact method [216], while scalability is ensured by preserving the structural properties of small-circuit solutions as the system size increases, leveraging symmetric subtopologies (linear, T-, and H-shaped). By integrating insights from both optimal and scalable approaches, AOQMAP provides depth-optimal and scalable solutions. This methodology not only bridges the gap between fully optimal and scalable approaches but also offers significant insights for addressing other routing problems, advancing both theoretical understanding and practical applications.

While the mapping of Hamiltonians to circuits is inherently scalable, other stages, such as verification, qubit selection, and subtopology identification, require careful attention to ensure feasibility at larger scales. The verification stage currently relies on classical simulation to validate the correctness of routing, decomposition, and optimization processes, which is effective for smaller circuits but is not scalable to larger systems due to the exponential growth in computational resources. To overcome this

limitation, the verification step can be validated on small circuits and subsequently omitted for larger ones. Alternatively, techniques such as ZX-calculus-based circuit verification [251] provide a practical solution, having demonstrated the capability to handle systems with hundreds of qubits. Qubit selection in AOQMAP is performed using the mapomatic framework, which has been demonstrated to scale to systems with hundreds of qubits [236], ensuring its suitability for large problem instances. Postselection, the final step in the workflow, evaluates and selects the optimal outcome among the three identified subtopologies by minimizing the expectation value of the problem Hamiltonian. This process imposes minimal computational overhead, supporting the scalability of AOQMAP.

For subtopology selection, AOQMAP currently supports linear, T-, and H-shaped configurations. To meet the demands of larger or more complex systems, future work will extend these subtopologies to include variants with additional qubit connections, providing enhanced flexibility while maintaining computational efficiency. If no predefined subtopologies are sufficient, routing can still be performed on the available subtopologies using compilers such as Qiskit or Tket, which reduce computational complexity compared to routing on the entire topology while maintaining high quality results. By integrating these strategies into the workflow, AOQMAP ensures scalability for large quantum systems. This efficient and adaptive process balances classical and quantum resources, making it well suited for real world problems on near-term quantum devices.

4.2.6 Applications

QAOA for MaxCut on noncomplete graphs

QAOA is a VQA designed specifically to address combinatorial optimization problems, such as the MaxCut problem for graphs [19]. This problem involves partitioning nodes of a graph into two distinct groups to maximize the number of edges that connect these groups [252]. The problem Hamiltonian in QAOA for MaxCut problem is represented as in Eq. 3.14, Compared to QAOA for portfolio optimization (Eq. 3.10), QAOA for the MaxCut problem does not include Pauli Z items, implying the absence of R_Z gates in quantum circuit. QAOA for MaxCut starts with state preparation and then alternately applies the problem and mixer Hamiltonian to evolve the state toward an optimal solution. QAOA for MaxCut problem on complete graphs follows a similar approach to QAOA for portfolio optimization. However, in noncomplete graphs, the lack of connectivity necessitates the exclusion of corresponding ZZ gates, introducing additional challenges for qubit mapping.

For VQAs with partially connected two-qubit gates, swap layers obtained from fully connected interactions can still ensure that the circuit satisfies connectivity constraints. In such cases, ZZ-SWAP gate corresponding to the missing edge is replaced by a SWAP gate, allowing for correct execution of quantum circuits on subtopologies. However, the presence of remaining SWAP gates due to the lack of two-qubit interactions significantly amplifies errors caused by noise. One solution is to optimize initial mapping or initial qubit order such that all SWAP gates are placed at the end of circuit. These end-located SWAP gates can then be removed by adjusting measurement order accordingly. Furthermore, if a SWAP gate is placed behind a ZZ

gate in the first ZZ layer, it can be removed by adjusting the initial qubit order since ZZ and SWAP gates commute.

Algorithm 4.7 presents pseudocode for mapping QAOA for MaxCut on noncomplete graphs by optimizing initial qubit order. For n qubits, there are $n!/2$ distinguished permutations due to symmetry. Different initial qubit orders introduce different gate arrangements, resulting in different numbers of CX gates. By minimizing additional CX gates, we can obtain an optimized qubit mapping. Practically, we can calculate the optimal or minimum number of CX gates in the resulting circuit. This optimal solution arranges all existing ZZ gates in consecutive layers until all gates are implemented. Then, we remove every SWAP gate behind the first ZZ layer and the one located at the end of circuit. This yields solutions with the minimum number of CX gates. To accelerate the search process, we can set the calculated optimal CX gate count as a target and terminate optimization once an initial qubit order achieves this value. It's important to note that the optimal initial order is not unique. Alternatively, we can employ a heuristic approach, where a certain size of initial qubit orders is searched, and the one with the fewest number of CX gates is selected. For higher depth, we can obtain the solution by utilizing mirror symmetry, which involves alternating between swap layers at depth $p = 1$ and their corresponding mirrors, resulting in a circuit repeating every two depths.

Variational quantum eigensolver (VQE)

In the previous section, we presented optimal routing solutions for QAOA on different types of subtopologies, including linear, T-, and H-shaped. One notable advantage of our approach is its applicability to other VQAs, such as VQE, without requiring additional computational resources. This is because both VQAs involve sequences of parameterized single qubit rotations and fixed two qubit operations. Therefore, optimal qubit routing solutions that we derived for QAOA based on subtopology connectivities can be easily adapted.

VQE [92, 253, 254] is designed to determine the ground state energy or eigenvalue of a Hamiltonian. It has broad applications in various fields such as quantum chemistry [14], condensed matter physics [8], and combinatorial optimization [255]. Let H be the Hamiltonian of a quantum system, and $|\psi\rangle$ a trial wavefunction. The Rayleigh-Ritz quotient is bounded below by the ground state energy E_0

$$E_0 \leq \frac{\langle\psi| H |\psi\rangle}{\langle\psi|\psi\rangle}. \tag{4.5}$$

The objective is to determine a quantum state by examining a parameterized ansatz state, denoted as $|\psi(\theta)\rangle = U(\theta)|0\rangle$, to minimize expectation value of Hamiltonian. Here, $|0\rangle$ represents initial state, and $U(\theta)$ is the vector of parameters θ, also known as variational form or ansatz, which represents a parameterized unitary transformation achievable through quantum circuit. The selection of ansatz circuits plays a critical role in determining the efficacy of VQE. Three prominent categories of ansatz circuits include chemically inspired [256], hardware-efficient ansatz (HEA) [257], and Hamiltonian variational [258].

In this study, we examine VQE with full entanglement, as demonstrated in previous research (e.g., [259]). The ansatz circuit begins with a layer of parameterized R_Y gates,

Algorithm 4.7: AOQMAP for QAOA at depth $p = 1$ with partially connected ZZ interactions

Input: Number of qubits n, Parameters $\gamma[c,t]$ and $\beta[j]$ of gates ZZ on qubit pair (c,t) and R_X on qubit j, respectively, where $c,t,j \in \{0,...,n-1\}$ and $c < t$
Output: Circuit satisfying connectivity constraints
$O \leftarrow \{0,1,...,n-1\}$// `Initialize the qubit order`
$L \leftarrow$ List of all existing ZZ gates
Function `MapTwoQubitGates`(O, L, γ)
  $O_0 \leftarrow O$ // `Initial qubit order`
  $qc_{zz} \leftarrow$ Quantum circuit initialized to $|0\rangle^{\otimes n}$
  foreach ZZ *in* L **do**
    Assign ZZ($\gamma[c,t]$) to qc_{zz} according to O_0 and swap layers obtained for fully connected two qubit interactions and update current qubit order O if SWAP is applied
  end
  $O_f \leftarrow O$ // `Final qubit order`
  foreach SWAP on (i,j) located at the end of qc_{zz} **do**
    Remove SWAP
    $O_f[i] \leftrightarrow O_f[j]$
  end
  foreach SWAP on (i,j) located at the end of the first ZZ layer of qc_{zz} **do**
    Remove SWAP
    $O_0[i] \leftrightarrow O_0[j]$
  end
return (qc_{zz}, O_f)
(qc_{zz}^{opt}, O_f) = `MapTwoQubitGates` (O, L)
$n_{cx}^{opt} \leftarrow$ CX gate count of qc_{zz}^{opt}
$\mathcal{O}_q \leftarrow$ List of defined qubit orders
for O_i *in* $\mathcal{O}_q$ **do**
  (qc_{zz}, O_{f_i}) = `MapTwoQubitGates` (O_i, L)
  $n_{cx} \leftarrow$ CX gate count of qc_{zz}
  if $n_{cx} < n_{cx}^{opt}$ **then**
    $n_{cx}^{opt} = n_{cx}$
    $qc_{zz}^{opt} = qc_{zz}$
    $O_f = O_{f_i}$
  end
end
Prepare the initial state $|0\rangle^{\otimes n}$
Apply $H^{\otimes n}$
Apply qc_{zz}^{opt}
for $k \leftarrow 0$ **to** n **do**
  Apply $R_X(\beta[O_f[k]])$ on k
end
Measure the qubits $([0,...,n-1] \rightarrow [O_f[0],...,O_f[n-1]])$

followed by controlled-Z (CZ) gates that serve as entangling gates. Unlike CX gate, which distinguishes between control and target qubits, CZ gate is undirected. After that, another set of parameterized R_Y gates is performed, succeeded by measurement. For higher p, the subcircuit between the first layer and measurement is repeated p times. The circuit with n qubits and depth p contains $(p+1)n$ parameters that require optimization by a classical optimizer. The full entanglement is achieved through $n(n-1)/2$ CZ gates, each comprising one CX and two Hadamard gates.

Algorithm 4.8 describes procedure of AOQMAP for VQE with full entanglement on linear subtopology. It differs from Algorithm 4.3 in the gates used. We initiate the implementation of CZ gate with one CX and two Hadamard gates that act on the physical qubit representing target qubit of CX gate. Subsequently, we perform CZ-SWAP gate by inserting a SWAP gate after CX gate and the second Hadamard gate on the physical qubit representing control qubit of CX instead of target qubit, since the introduced SWAP gate alters qubit order. Analogously, we construct $n-2$ swap layers for n qubits on linear subtopology, meaning that CZ gates are performed on the first and last layers, while the constructed CZ-SWAP gates are implemented on remaining layers. Finally, R_Y gates are assigned accordingly, followed by measurement. The qubit mapping solution for VQE circuits on T- and H-shaped subtopologies can be attained similarly. Additionally, Algorithm 4.7 can be employed to obtain solutions for non-fully entangled VQE. The mapped circuit with depth $p = 1$ on five qubits is depicted in Figure 4.14. Each SWAP gate introduces a new qubit order and adds one additional CX gate, resulting in a total of $p(n-1)^2$ CX gates. The incorporated SWAP gates guarantee that all requisite CZ gates can be executed. R_Y gates and measurement operators are then assigned according to the current qubit order.

The AOQMAP approach offers several advantages. First, solutions for different subtopologies can be easily adapted between VQAs. This means that once solutions are found for a specific algorithm on target subtopology, they can be applied to other VQAs. Second, AOQMAP facilitates individual block optimization. For instance, the optimization of CZ-SWAP gate can be achieved by rearranging SWAP gate before the second Hadamard gate, while altering the qubit that Hadamard gate acts upon. Moreover, this structure enables efficient pulse optimization (Chapter 5) and application of error mitigation strategies (Chapters 6 and 7).

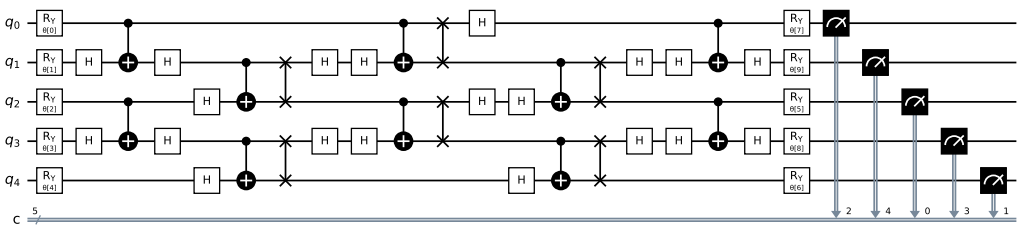

Figure 4.14: Resulting circuit of a five-qubit VQE at depth $p = 1$ with full entanglement on a linear subtopology using AOQMAP. The circuit can be further optimized by canceling two Hadamard gates and performing CX gate cancellation for CX-SWAP gates.

Algorithm 4.8: AOQMAP for VQE on linear subtopology

Input: Number of qubits n, VQE depth p, Vector of parameters θ with dimension $(p+1) \times n$
Output: Circuit satisfying connectivity constraints
Function `ApplyCZGate`(i, j)
  Apply H on j
  Apply CX on (i, j)
  Apply H on j
end
Function `ApplyCZSWAPGate`(O, i, j)
  Apply H on j
  Apply CX on (i, j)
  Apply SWAP on (i, j)
  $O[i] \leftrightarrow O[j]$ `// Exchange qubit order`
  Apply H on i
end
$O \leftarrow \{0, 1, ..., n-1\}$`// Initialize the qubit order`
Prepare the initial state $|0\rangle^{\otimes n}$
for $i := 0$ *to* n **do**
  Apply $R_Y(\theta[i])$ on i
end
$p_{\max} \leftarrow p$
while $p > 0$ **do**
  $s \leftarrow 0$
  while $s < n$ **do**
    for $q := 0$ *to* $n-1$ *step* 2 **do**
      if $s == 0$ *or* $s == n-1$ **then**
        `ApplyCZGate`$(q, q+1)$
      else
        `ApplyCZSWAPGate`$(O, q, q+1)$
      end
    end
    $s \leftarrow s+1$
    if $s < n$ **then**
      for $q := 1$ *to* $n-1$ *step* 2 **do**
        if $s == 0$ *or* $s == n-1$ **then**
          `ApplyCZGate`$(q, q+1)$
        else
          `ApplyCZSWAPGate`$(O, q, q+1)$
        end
      end
    $s \leftarrow s+1$
  end
  for $i := 0$ *to* n **do**
    Apply $R_Y(\theta[i + n(p_{\max} - p + 1)])$ on $O[i]$
  end
  $p \leftarrow p-1$
end
Measure the qubits $([0, ..., n-1] \rightarrow [O[0], ..., O[n-1]])$

NISQ devices and algorithms beyond VQAs

AOQMAP facilitates the transfer of solutions among diverse quantum devices, accounting for their specific architectures and noise characteristics. The routing solutions for VQAs on linear, T-, and H-shaped configurations can be adapted to other superconducting quantum devices. For instance, Google's Sycamore processor [260] employs square lattice qubit connectivity providing more opportunities for the three subtopologies. An H-shaped subtopology can be readily implemented in the processor by identifying two neighboring square faces in the grid and their linear connection. Similarly, Rigetti's processor [261] also exhibits such basic subtopologies. The adaptability is then achieved through a decomposition-optimization-remapping workflow, where the routed circuit on a subtopology is decomposed into basis gates of target device, optimized, and mapped onto hardware, taking into account the latest device calibration data. As novel architectures proliferate, AOQMAP's customized adaptability enables algorithms to be easily adjusted, making it a crucial advantage in the NISQ era.

The proposed method is also adaptable to other gate-based quantum architectures, such as trapped ions [262, 263, 264], where qubit connectivity and gate fidelities may differ significantly. Trapped ion architectures typically support full connectivity on a one-dimensional chain. While this eliminates the need for SWAP layers, first decomposing the circuit into native gates and then selecting high-quality qubits remains essential for improving algorithm performance. Moreover, gate sequences introduced by SWAP layers provide guidance for implementing algorithms. Additionally, research has shown that gate fidelities decrease as the chain size increases [263, 265], limiting scalability to a large number of qubits. An open question is whether focusing on neighboring qubit interactions can maintain high gate fidelities with increasing system size, thereby achieving scalability. Future research can compare the performance of fully connected two-qubit interactions versus neighboring qubit interactions using our routing solutions.

For photonic architectures [266, 267, 268, 269], the universal quantum computing models, such as one-way or measurement-based [270, 271, 272, 273], differ fundamentally from the standard quantum circuit model, requiring tailored compilation strategies for mapping circuits to photonic hardware [274, 275, 276]. While the proposed mapping strategy is not directly applicable to such architectures, this study provides valuable insights into optimizing algorithm performance. For instance, recent work has demonstrated VQA with hardware-efficient ansatzes (HEA) on a quantum photonic platform [277]. Our routing solutions with neighboring interactions can be adapted to build HEA for VQAs, including integrating symmetric structures to construct high VQA layers, as employed in AOQMAP-L.

The AOQMAP flow presented in Figure 4.9 can be generalized to optimize various quantum algorithms. Specifically, the division of the process into subtopology adaptation, verification, and mapping of subtopology-adapted circuits is particularly promising for improving the performance of algorithms on real quantum devices, as it facilitates the efficient utilization of classical and quantum resources while enabling high-quality implementations. One of the most significant advantages of AOQMAP is its ability to find optimal and scalable solutions for VQAs with fully connected two-qubit interactions on linear, T-, and H-shaped subtopologies. These solutions

can be directly adapted to variational-based algorithms, including machine learning models leveraging variational circuits [278, 279, 280, 281]. While these solutions cannot be straightforwardly transformed to algorithms beyond VQAs, such as Grover's algorithm [282], our approach to finding solutions by bridging exact and scalable methods offers valuable guidance: (1) Carefully analyze the specific structure of target algorithm from Hamiltonian to circuit level to identify properties such as symmetry and the most advantageous subtopologies. In the case where the optimal subtopology is not immediately apparent, start with linear, T-, and H-shaped topologies, as they frequently appear in most NISQ devices; (2) Focus on two-qubit interactions in the algorithm and insert SWAP gates to satisfy connectivity constraints. This simplifies the routing problem, as any single qubit gate can be assigned at the algorithmic level, which we have demonstrated with QAOA by first finding routing solutions on subtopologies and then assigning single qubit gates; (3) Prioritize placing SWAP gates behind or before two-qubit gates. This allows for optimization of the two-qubit gate following a SWAP gate by leveraging CX gate cancellation (see Figure 4.14 for CZ-SWAP and Chapter 7 for ZY-SWAP), thereby reducing the impact of noise due to the insertion of SWAP gates; (4) Remove any possible initial and final SWAP gates by adjusting the initial and measurement orders. Integrating these guidelines into the design process of heuristic and exact methods is promising to improve the quality and efficiency of solving qubit mapping problems.

4.2.7 Benchmarking experiments

This section presents benchmarking results of QAOA for portfolio optimization problems on several IBM QPUs. We evaluate the efficiency of our method against three other approaches: Qiskit [208], Tket [210], and SWAP network (SWAPNK) [218].

Table 4.6 summarizes the characteristics of IBM QPUs employed in our benchmarking experiments. It is worth noting that these characteristics fluctuate over time. The processors used in our study range from 7 to 127 qubits. The native gate set on 7- and 27-qubit QPUs is {CX, ID, R_Z, SX, X}, comprising CX, identity gate, single-qubit Z-rotation, $\pi/2$ X-rotation, and Pauli-X. The 127-qubit QPUs implement a basis set of {ECR, ID, R_Z, SX, X}, where ECR is echoed cross-resonance two-qubit gate. The average error rate for single qubit gates varies from 10^{-4} to 10^{-2}, while for two-qubit gates, it is typically an order of magnitude higher, around 10^{-2}. Additionally, average readout error rates are approximately 10^{-2}. The average gate lengths for single-qubit gates range from 32 ns to 60 ns, whereas for two-qubit gates, they range from 306.96 ns to 665.83 ns. Furthermore, the mean energy relaxation time T_1 ranges from 101.06 µs to 227.95 µs across different processors, and the average dephasing time T_2 ranges from 71.04 µs to 166.11 µs. These comprehensive devices allow us to validate our optimization techniques across various qubit numbers, connectivity options, and noise properties.

The portfolio optimization instances utilized in this study are obtained from the supplementary information of [94]. Five different problem sizes are examined, involving the selection of the first 3, 4, 5, 6, and 10 assets from the portfolio optimization dataset. In QAOA, the key parameters (q, B, A, λ) vary depending on problem size. Here, q, B, A, and λ respectively represent risk preference, number of assets to be selected for the portfolio, penalty factor, and global scaling factor. The parameter

Table 4.6: Characterization of IBM QPUs.

Property	ibm_perth	ibm_nairobi	ibmq_ehningen	ibmq_kaolkata	ibm_cusco	ibm_nazca
Qubit number	7	7	27	27	127	127
Single qubit gate error	4.867e-04	3.049e-04	3.348e-04	3.732e-02	2.703e-03	8.61e-04
Two qubit gate error	1.140e-02	8.749e-03	1.025e-02	1.855e-01	8.861e-02	5.49e-02
Readout error	2.443e-02	3.007e-02	1.230e-02	2.600e-02	5.509e-02	5.18e-02
Single qubit gate length	35.56 ns	35.56 ns	32.00 ns	35.56 ns	44.00 ns	60.00 ns
Two qubit gate length	485.93 ns	306.96 ns	346.98 ns	450.29 ns	487.22 ns	658.17 ns
T_1	162.06 µs	101.06 µs	143.82 µs	103.18 µs	131.51 µs	197.66 µs
T_2	123.82 µs	71.04 µs	166.11 µs	82.98 µs	109.17 µs	128.56 µs

values (q, B, A, λ) for each case are as follows: (0.33, 2, 0, 20.97) for three-qubit, (0.33, 2, 0.13, 17.99) for four-qubit, (0.33, 3, 0.07, 17.51) for five-qubit, (0.33, 3, 0.12, 17.73) for six-qubit, and (0.33, 5, 0.05, 6) for ten-qubit scenario. To determine the parameters (β, γ), we employ the constrained optimization by linear approximation (COBYLA) [85] optimizer along with Qiskit's Qasm simulator.

Evaluation metrics

We utilize circuit metrics including CX gate count and circuit depth to evaluate various qubit mapping methods. Additionally, we employ approximation ratio (AR) and success probability (SP) to assess the performance of QAOA, as defined in Section 3.2.1. A higher value of AR or SP implies a more effective algorithm performance, while a lower value of CX gate count or circuit depth indicates a high quality of the qubit mapping solution.

Comparison of qubit mapping strategies

To ensure fair benchmarking, we employ various qubit mapping techniques to map circuits onto target quantum hardware, followed by decomposing and optimizing these circuits using Qiskit [208] with default settings (optimization level 1). For Tket [210], we employ NoiseAwarePlacement to select the lowest-noise qubits based on target QPUs' noise properties. The default Tket routing method *RoutingPass* is employed to insert SWAP gates. For Qiskit [208], we transpile with default settings. Finally, to map SWAPNK [218], we apply the same mapping strategy as AOQMAP, as detailed in Section 4.2.4.

We first compare our approach with SWAPNK [218]. Figure 4.15 presents the reduction in SWAP gates for QAOA at depth $p = 1$ using AOQMAP on linear (AOQMAP-L), T- (AOQMAP-T), and H-shaped (AOQMAP-H) subtopologies compared to SWAPNK. Specifically, AOQMAP-L leads to a decrease of 67% in SWAP gates for three-qubit and 20% for ten-qubit. Moreover, AOQMAP-T reduces SWAP gates by 67% for four-qubit and 29% for ten-qubit, whereas AOQMAP-H reduces SWAP gates by 53% for six-qubit and 36% for ten-qubit. For QAOA with depth p, the reduction in SWAP gate count achieved is p multiplied by the reduction attained at depth $p = 1$.

We then compare our approach with Tket, Qiskit, and SWAPNK on IBM QPUs. The examined QAOA depths range from 1 to 7 for qubit numbers of three, five, six, and ten. Table 4.7 presents the two-qubit gate (CX or ECR) counts for the mapped

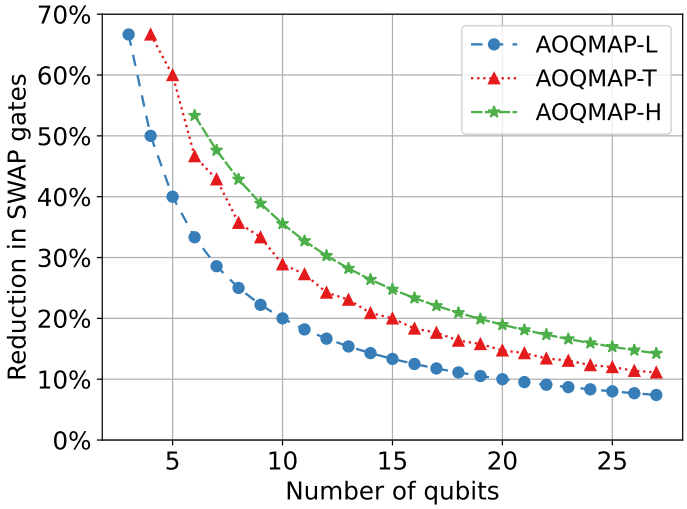

Figure 4.15: Reduction in the number of SWAP gates for QAOA with depth $p = 1$ using AOQMAP-L, AOQMAP-T, and AOQMAP-H compared to SWAPNK.

circuits generated by each strategy across the benchmarked QPUs. We note that the 127-qubit QPUs natively implement ECR gates rather than CX gates. Across all benchmarks, AOQMAP achieves the fewest CX or ECR gates. For three-qubit QAOA, the number of CX gates remains identical for AOQMAP since three-qubit can only form a linear subtopology. However, for five-qubit QAOA with depth p from two to seven on ibm_perth (5Q-Perth), AOQMAP reduces the number of CX gates compared to other QPUs (5Q-Kolkata and 5Q-Cusco) because a circuit adapted to a T-shaped subtopology yields lower expectation value of problem Hamiltonian and is selected. Similar CX reductions occur for ten-qubit QAOA such as on ibm_nazca and ibmq_kolkata at depth three. SWAPNK provides consistent solutions for fixed problem sizes, while Tket and Qiskit produce solutions that vary across QPUs with different qubit counts.

Table 4.8 presents circuit depths of mapped QAOA with different methods. AOQMAP achieves the shortest depth on average. The T-shaped subtopology increases five-qubit depth on ibm_perth but reduces CX gates. For six-qubit QAOA on ibmq_ehningen, AOQMAP uses only the linear subtopology which has the least number of CX gates and the shortest depths than others. For ten-qubit QAOA on 127-qubit QPUs ibm_cusco and ibm_nazca, depth varies slightly due to the randomness of Qiskit's transpile used in the decomposition and optimization process.

The solutions provided by AOQMAP achieve on average the shortest circuit depth and the fewest number of CX or ECR gates compared to other approaches. Specifically, AOQMAP reduces the two-qubit gate count by 28.8% (up to 56.1%) and circuit depth by 30.2% (up to 82.1 %) on average compared to Qiskit, Tket, and SWAPNK.

It is important to note that the solution quality in Qiskit is sensitive to the optimization level settings. As will be shown in Chapter 7, where AOQMAP is applied to improve digitized counterdiabatic QAOA, for QAOA circuits with depth $p = 1$, Qiskit's optimization levels 1 and 2 produce comparable results, while level 3 shows improved performance. However, AOQMAP consistently outperforms all Qiskit optimization levels tested, highlighting its superior efficiency and effectiveness.

Table 4.7: Reduction in CX or ECR gate counts of AOQMAP compared to other qubit mapping methods using QAOA with n qubits (nQ) and depths from 1 to 7 on various IBM QPUs. Higher values indicate better performance of AOQMAP.

Benchmark	Tket	Qiskit	SWAPNK
3Q-Perth	(22%, 22%, 30%, 28%, 31%, 30%, 32%)	(22%, 22%, 30%, 28%, 31%, 30%, 32%)	(22%, 22%, 22%, 22%, 22%, 22%, 22%)
3Q-Kolkata	(22%, 22%, 30%, 28%, 31%, 30%, 32%)	(22%, 22%, 30%, 28%, 31%, 30%, 32%)	(22%, 22%, 22%, 22%, 22%, 22%, 22%)
3Q-Ehningen	(22%, 22%, 30%, 28%, 31%, 30%, 32%)	(30%, 48%, 40%, 45%, 35%, 38%, 42%)	(22%, 22%, 22%, 22%, 22%, 22%, 22%)
3Q-Nairobi	(22%, 22%, 30%, 28%, 31%, 30%, 32%)	(22%, 22%, 30%, 28%, 31%, 30%, 32%)	(22%, 22%, 22%, 22%, 22%, 22%, 22%)
5Q-Perth	(10%, 28%, 28%, 30%, 30%, 31%, 31%)	(19%, 34%, 35%, 37%, 35%, 39%, 36%)	(13%, 20%, 20%, 20%, 20%, 20%, 20%)
5Q-Kolkata	(10%, 22%, 22%, 25%, 26%, 25%, 26%)	(21%, 29%, 30%, 32%, 32%, 34%, 34%)	(13%, 13%, 13%, 13%, 13%, 13%, 13%)
5Q-Cusco	(28%, 25%, 27%, 26%, 27%, 26%, 27%)	(32%, 34%, 38%, 38%, 38%, 39%, 39%)	(13%, 13%, 13%, 13%, 13%, 13%, 13%)
5Q-Ehningen	(7%, 19%, 22%, 24%, 24%, 25%, 25%)	(48%, 50%, 54%, 52%, 53%, 56%, 53%)	(13%, 13%, 13%, 13%, 13%, 13%, 13%)
6Q-Ehningen	(41%, 40%, 41%, 42%, 42%, 42%, 42%)	(39%, 43%, 45%, 43%, 44%, 44%, 45%)	(11%, 11%, 11%, 11%, 11%, 11%, 11%)
10Q-Kolkata	(52%, 50%, 48%, 52%, 53%, 48%, 54%)	(44%, 43%, 45%, 47%, 48%, 47%, 44%)	(10%, 7%, 7%, 7%, 7%, 7%, 10%)
10Q-Nazca	(48%, 53%, 55%, 55%, 56%, 56%, 56%)	(46%, 46%, 48%, 49%, 48%, 49%, 49%)	(7%, 7%, 7%, 7%, 7%, 7%, 7%)
10Q-Cusco	(51%, 53%, 55%, 55%, 56%, 56%, 56%)	(46%, 46%, 48%, 49%, 48%, 49%, 49%)	(7%, 7%, 7%, 7%, 7%, 7%, 7%)

Table 4.8: Reduction in circuit depth. Same benchmarks as in Table 4.7.

Benchmark	Tket	Qiskit	SWAPNK
3Q-Perth	(30%, 21%, 29%, 25%, 29%, 26%, 29%)	(27%, 21%, 22%, 25%, 25%, 26%, 26%)	(10%, 11%, 11%, 11%, 11%, 11%, 11%)
3Q-Kolkata	(30%, 21%, 29%, 25%, 29%, 26%, 29%)	(27%, 21%, 22%, 25%, 25%, 26%, 26%)	(10%, 11%, 11%, 11%, 11%, 11%, 11%)
3Q-Ehningen	(27%, 21%, 28%, 25%, 28%, 26%, 28%)	(14%, 28%, 22%, 26%, 19%, 22%, 25%)	(10%, 11%, 11%, 11%, 11%, 11%, 11%)
3Q-Nairobi	(30%, 21%, 29%, 25%, 29%, 26%, 29%)	(27%, 21%, 22%, 25%, 25%, 26%, 26%)	(10%, 11%, 11%, 11%, 11%, 11%, 11%)
5Q-Perth	(29%, 13%, 18%, 19%, 20%, 20%, 21%)	(43%, 23%, 27%, 34%, 28%, 36%, 28%)	(7%, -26%, -27%, -27%, -27%, -27%, -27%)
5Q-Kolkata	(29%, 36%, 40%, 46%, 47%, 46%, 47%)	(47%, 43%, 51%, 50%, 48%, 54%, 53%)	(7%, 7%, 8%, 8%, 8%, 8%, 8%)
5Q-Cusco	(53%, 50%, 50%, 49%, 50%, 49%, 49%)	(48%, 47%, 48%, 52%, 50%, 52%, 52%)	(13%, 13%, 12%, 12%, 11%, 12%, 11%)
5Q-Ehningen	(21%, 26%, 28%, 29%, 30%, 30%, 31%)	(51%, 55%, 57%, 57%, 57%, 59%, 57%)	(7%, 7%, 8%, 8%, 8%, 8%, 8%)
6Q-Ehningen	(48%, 43%, 44%, 48%, 45%, 46%, 48%)	(44%, 45%, 50%, 50%, 47%, 49%, 49%)	(6%, 6%, 7%, 7%, 7%, 7%, 7%)
10Q-Kolkata	(57%, 69%, 65%, 71%, 71%, 66%, 59%)	(54%, 66%, 67%, 69%, 72%, 71%, 27%)	(-33%, 4%, 4%, 4%, 4%, 4%, -35%)
10Q-Nazca	(74%, 75%, 79%, 80%, 81%, 82%, 82%)	(71%, 71%, 52%, 72%, 71%, 76%, 74%)	(6%, 6%, 6%, 5%, 5%, 5%, 5%)
10Q-Cusco	(75%, 75%, 79%, 80%, 81%, 81%, 81%)	(71%, 71%, 52%, 71%, 71%, 76%, 74%)	(7%, 7%, 6%, 6%, 6%, 7%, 6%)

Computational resources

This section examines the computational resources needed for AOQMAP and compares them with the requirements of Qiskit and Tket. We show that the performance enhancements achieved by AOQMAP do not come at the expense of increased computational complexity, making it a practical approach. Specifically, we first divide the runtime of AOQMAP into three parts: adaptation, verification, and qubit selection. As illustrated in Figure 4.9, VQAs are routed, decomposed, and optimized on subtopologies in adaptation process, ensuring that the circuit can be executed on QPU's subtopologies. Then, the circuit correctness is verified by computing the Hellinger distance of adapted and original circuits using Qasm simulator. Finally, mapomatic [236] is employed to select optimal qubits for execution.

AOQMAP, Qiskit, and Tkit employ the same experimental setup described at the last section. We benchmark the QAOA for MaxCut on complete graphs with varying qubit numbers and depths, mapping these circuits onto a 27-qubit QPU. We take AOQMAP-T as an example, and its analysis can be extended to AOQMAP-L and AOQMAP-H. Tables 4.9 and 4.10 provide a comprehensive overview of runtime for three parts of AOQMAP and an overall comparison, respectively. Our investigation reveals that adaptation in AOQMAP consumes the least time, with its duration increasing minimally as the number of qubits grows. This process is also minimally affected by variations in QAOA depths. In contrast, verification is the most time-intensive, requiring the execution of circuits with Qasm simulator. Moreover, verification runtime increases with the number of qubits but is relatively unaffected by QAOA depths. The runtime for qubit selection increases with both the

Table 4.9: Runtime [s] of three components in AOQMAP on T-shaped subtopology using QAOA with the number of qubits n ranging from 4 to 10 and depth p of 1, 3, 5, and 7. The runtime is divided into adaptation, verification, and qubit selection. Avg denotes the average runtime for different qubit numbers.

p	n	4	5	6	7	8	9	10	Avg
1	Adaptation	0.04	0.05	0.08	0.06	0.12	0.09	0.08	0.07
	Verification	0.13	0.12	0.16	0.16	0.19	0.25	0.21	0.17
	Qubit selection	0.03	0.05	0.11	0.07	0.09	0.09	0.14	0.08
3	Adaptation	0.02	0.02	0.03	0.03	0.05	0.06	0.08	0.04
	Verification	0.12	0.14	0.14	0.24	0.19	0.2	0.23	0.18
	Qubit selection	0.11	0.06	0.09	0.09	0.16	0.14	0.23	0.13
5	Adaptation	0.03	0.03	0.05	0.05	0.06	0.08	0.1	0.06
	Verification	0.12	0.2	0.16	0.31	0.2	0.23	0.28	0.21
	Qubit selection	0.05	0.06	0.11	0.13	0.2	0.2	0.33	0.15
7	Adaptation	0.03	0.03	0.06	0.06	0.08	0.11	0.2	0.08
	Verification	0.13	0.14	0.16	0.19	0.22	0.25	0.28	0.2
	Qubit selection	0.08	0.08	0.12	0.16	0.27	0.25	0.44	0.2

Table 4.10: Runtime comparison of different qubit mapping methods using QAOA, with the number of qubits n ranging from 4 to 10 and depth p of 1, 3, 5, and 7. Avg denotes the average runtime for different qubit numbers.

p	n	4	5	6	7	8	9	10	Avg
1	AOQMAP-T	0.2	0.22	0.34	0.29	0.4	0.44	0.44	0.33
	Qiskit	0.02	0.03	0.03	0.03	0.03	0.04	0.05	0.03
	Tket	0.05	0.36	0.19	0.13	0.64	0.77	2.99	0.73
3	AOQMAP-T	0.25	0.22	0.27	0.36	0.39	0.41	0.55	0.35
	Qiskit	0.02	0.04	0.03	0.13	0.06	0.06	0.08	0.06
	Tket	0.07	0.42	0.27	0.23	1.42	0.92	3.16	0.93
5	AOQMAP-T	0.2	0.23	0.31	0.48	0.47	0.52	0.7	0.42
	Qiskit	0.05	0.05	0.05	0.09	0.08	0.12	0.14	0.08
	Tket	0.13	0.48	0.35	0.34	0.97	1.09	3.19	0.94
7	AOQMAP-T	0.24	0.25	0.34	0.41	0.56	0.62	0.92	0.48
	Qiskit	0.04	0.06	0.06	0.09	0.19	0.13	0.17	0.11
	Tket	0.16	0.53	0.49	0.44	1.13	1.34	3.67	1.11

number of qubits and QAOA depths. We also observe that the runtime for deeper circuits might be shorter than that for shallower circuits. This is potentially due to the stochastic nature of Qiskit transpiler employed for decomposition and optimization. As indicated in Table 4.10, Qiskit demonstrates the shortest runtime. Additionally, AOQMAP outperforms the heuristic strategy Tket. Tket's runtime increases with QAOA depths and qubit numbers, and exhibits significant fluctuations, potentially due to the interface transforming circuits implemented with Qiskit and Tket.

Simulation under noise

To examine the impact of noise on the performance of QAOA with different qubit mapping strategies, we simulate algorithms mapped onto a 27-qubit QPU with the topology shown in Figure 2.3(a). We use a depolarizing noise model [283, 284],

which is a common simple model used to approximate the effects of mixed noise processes in quantum systems. This model captures noise effects by applying random single-qubit bit-flip, phase-flip, and combined bit- and phase-flip errors to each gate, providing a good approximation of the overall noise behavior and impact on algorithm performance.

We compare AOQMAP-L and AOQMAP-T with Tket, Qiskit, and SWAPNK. The mapped circuits are simulated under depolarizing noise with strength 0.005 for two-qubit gates and $\epsilon/10$ for single-qubit gates. Figure 4.16 presents the approximation ratio and success probability of QAOA on 3, 4, 5, and 6 qubits at depths from 1 to 7. Compared to other quantum mapping strategies, AOQMAP-L and AOQMAP-T demonstrate better performance at higher depths and comparable performance at depth $p = 1$. This suggests that increased depth of QAOA leads to a more significant noise effect, emphasizing advantages of AOQMAP. AOQMAP-T requires fewer CX gates during the mapping stage compared to others, which mitigates the detrimental effects of noise accumulation, leading to improved performance. For three-qubit QAOA, AR and SP decrease with increased depth, indicating that noise effects outweigh performance improvement at higher depths. In comparison, for QAOA with 4, 5, and 6 qubits, AR and SP increase and then stabilize or slightly decrease, suggesting a balance between performance improvements and noise effects.

Demonstration on IBM quantum devices

This section evaluates the performance of QAOA for portfolio optimization across six IBM QPUs, comparing different mapping approaches on 7-qubit ibm_perth and ibm_nairobi, 27-qubit ibmq_kolkata and ibmq_ehningen, and 127-qubit ibm_nazca and ibm_cusco. The problem sizes QAOA include 3, 4, 5, 6, and 10 qubits, and depths ranging from 1 to 7.

As discussed in Section 4.2.3, two methods exist for constructing higher QAOA depth, including repeating swap layers in depth $p = 1$ circuit and leveraging their symmetry. To examine the influence of symmetry, we conduct a comparative analysis of AOQMAP on linear subtopology with repetitive routing at depth $p = 1$ (AOQMAP-L) and with mirror-symmetric swap layers (AOQMAP-LS), as well as AOQMAP on T-shaped subtopology (AOQMAP-T). We consider QAOA instances with four qubits and depths from 1 to 7. Figure 4.17 compares these variants to Tket, Qiskit, and SWAPNK on the 7-qubit ibm_nairobi. AOQMAP on linear subtopology utilizing symmetry significantly enhances the performance, increasing success probability by an average of 82% (up to 1.83×) compared to AOQMAP-L, 76% (up to 2.76×) compared to AOQMAP-T, 1.72× (up to 3.80×) compared to Tket, 1.77× (up to 3.21×) compared to Qiskit, and 73% (up to 1.41×) compared to SWAPNK. These results demonstrate the effectiveness of symmetry for improving performance on NISQ devices, validating the advantage of AOQMAP's symmetry-incorporated mapping approach.

For the demonstration on ibmq_ehningen, we directly compare AOQMAP-L with other methods, while for evaluations on other QPUs, a postselection process is utilized to generate the AOQMAP solution. Specifically, we construct the algorithms with AOQMAP-L, AOQMAP-LS, and AOQMAP-T. Each adapted circuit is mapped to the target device using the method described in Section 4.2.4. The demonstration

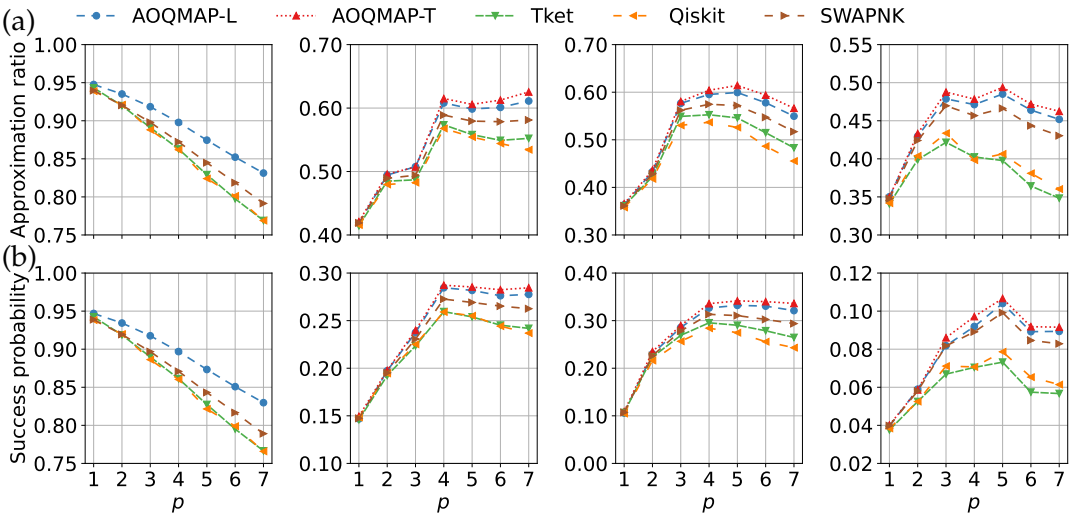

Figure 4.16: Performance comparison of QAOA for portfolio optimization mapped with AOQMAP-L, AOQMAP-T, Tket, Qiskit, and SWAPNK under depolarizing noise. (a) Approximation ratio and (b) success probability for QAOA with three, four, five, and six qubits. Three-qubit QAOA has an absence of AOQMAP-T, as the minimum number of qubits required for the T-shaped subtopology is four.

solution corresponding to the minimum expectation value of the problem Hamiltonian is then chosen. As shown in Figure 4.18(a), AOQMAP achieves the highest performance on all QPUs. Tket and Qiskit perform comparably on all QPUs, while SWAPNK performs better on ibm_perth and ibmq_ehningen than ibm_kolkata and ibm_nairobi. The approximation ratio decreases with increased depth, indicating that the negative impact of noise dominates the improved performance at higher depths. For five-qubit QAOA (Figure 4.18(b)), AOQMAP maintains the highest AR across all QPUs. Tket outperforms Qiskit on ibmq_kolkata and ibmq_ehningen, whereas Qiskit performs better on ibm_cusco. SWAPNK has a better performance on ibmq_perth and ibmq_kolkata, while it performs worse on ibmq_ehningen. For six-qubit QAOA on ibmq_ehningen, AOQMAP has a significantly higher AR value than Tket, Qiskit, and SWAPNK (Figure 4.18(c)). For the largest ten-qubit QAOA, AOQMAP consistently achieves the highest AR across all tested QPUs. SWAPNK has a higher AR only on ibm_nazca, while Qiskit and Tket perform worse on all QPUs. On ibmq_kolkata, AOQMAP achieves the highest AR value at depth two, outperforming others. With increased depth, the noise in large-scale circuit increases significantly, resulting in a reduced AR value. For ten-qubit QAOA on ibm_cusco, AOQMAP has the highest AR value at depth $p = 1$. The presence of the second-highest AR at depth six demonstrates a trade-off between improved performance and introduced noise of the increased QAOA depth.

These findings demonstrate that AOQMAP provides a significant improvement in approximation ratio and success probability, surpassing other popular qubit mapping approaches. In particular, AOQMAP improves AR by an average of 54% and SP by an average of 138% compared to Qiskit, Tket, and SWAPNK, demonstrating robust high performance on NISQ devices. Furthermore, our experiments on various QPUs highlight the limitations of solely relying on noise model simulations for assessing

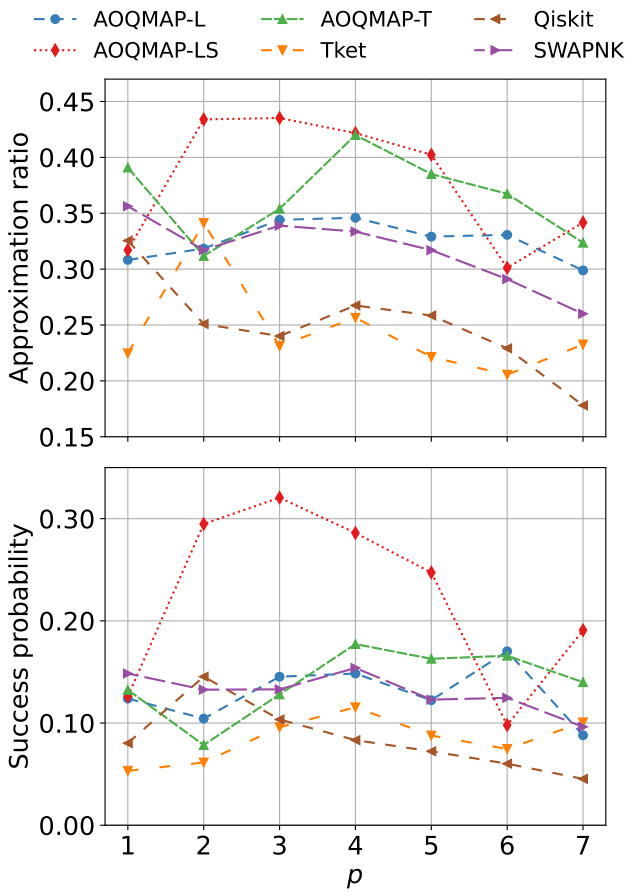

Figure 4.17: Demonstration of 4-qubit QAOA on a 7-qubit QPU ibm_nairobi with various qubit mapping methods. (a) Approximation ratio and (b) success probability. QAOA depths range from 1 to 7.

circuit performance. While such simulations provide valuable insights, they often fail to accurately capture the complex and multifaceted noise characteristics inherent to real quantum hardware. Although different qubit mapping strategies exhibit comparable behavior under simulated noise conditions, their performance diverges significantly when executed on physical devices. This disparity emphasizes the critical role of efficient qubit mapping in achieving high algorithm performance on NISQ devices.

4.3 Optimized compilation workflow

Based on previously extensive experiments involving the practical execution of variational quantum algorithms on real quantum devices, this section presents a framework for the resilient compilation of general quantum algorithms, emphasizing enhanced reliability and robustness against noise. The objective is to optimize the compilation flow by taking into account the constraints within hardware. Additionally, we propose metrics for the evaluation of compilation quality.

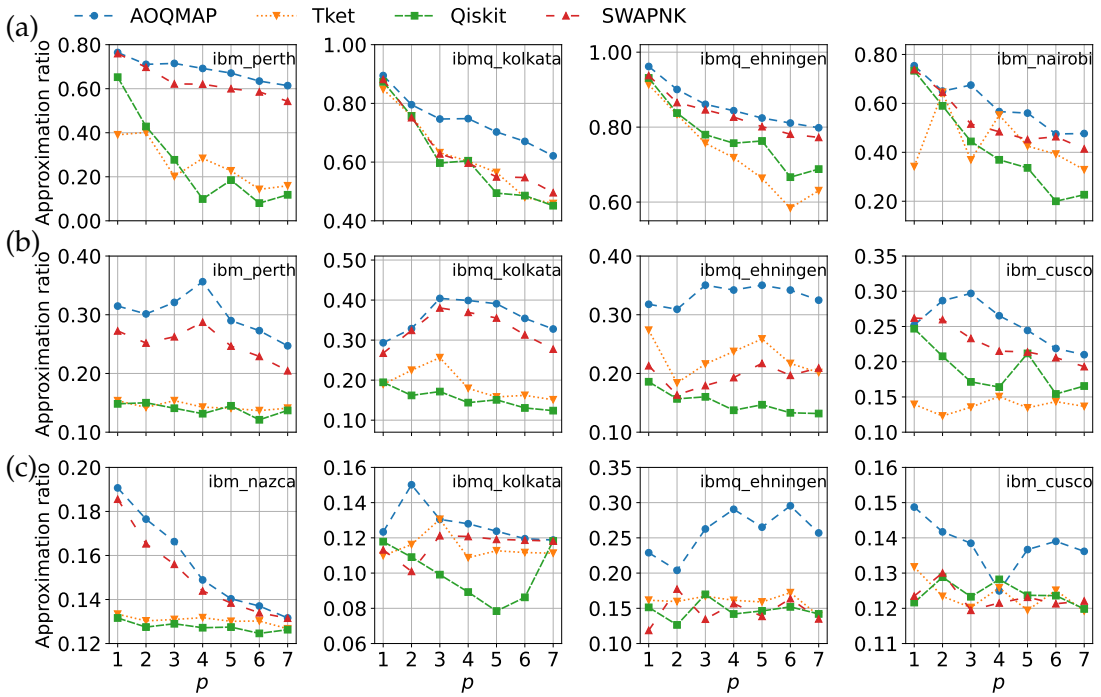

Figure 4.18: Approximation ratio of QAOA obtained on various IBM QPUs using AOQMAP, Tket, Qiskit, and SWAPNK. The numbers of qubits are (a) three, (b) five, and (c) ten (six for ibmq_ehningen). AOQMAP demonstrates the highest performance on all QPUs in comparison to other approaches.

4.3.1 Proposed workflow

To optimize the execution of quantum algorithms, we integrate various strategies developed in previous sections. As illustrated in Figure 4.20, the workflow comprises several key steps: (1) adapting circuits to compatible subtopologies of QPUs, (2) modeling noise to capture realistic errors in real quantum devices, (3) formulating cost functions to guide qubit selection, and (4) selecting optimal qubits for circuit execution. In the subtopology-aware circuit adaptation process, quantum algorithms are transformed into circuits compatible with specific subtopologies of QPUs, leveraging the topology and native gate set of the quantum devices. Subsequently, an error model describing noise in the system is employed, along with a designed cost function accounting for factors such as fidelity and schedule duration of gates. Realistic noise models and effective cost functions are crucial for selecting qubits that are less susceptible to errors during circuit execution. To refine the error model for accuracy, we recommend analyzing historical calibration data of the devices and executing benchmarking circuits on the target QPU for validation. Additionally, error mitigation strategies can be integrated into the cost function design and noise modeling process, as different qubits may exhibit varying mitigation effectiveness. Finally, the predefined cost function and error model are used to guide the selection of optimal qubits that satisfy the subtopology connectivity constraints by leveraging updated calibration data provided by the QPUs. Employing different methods to select qubits for execution can significantly influence algorithm performance, which

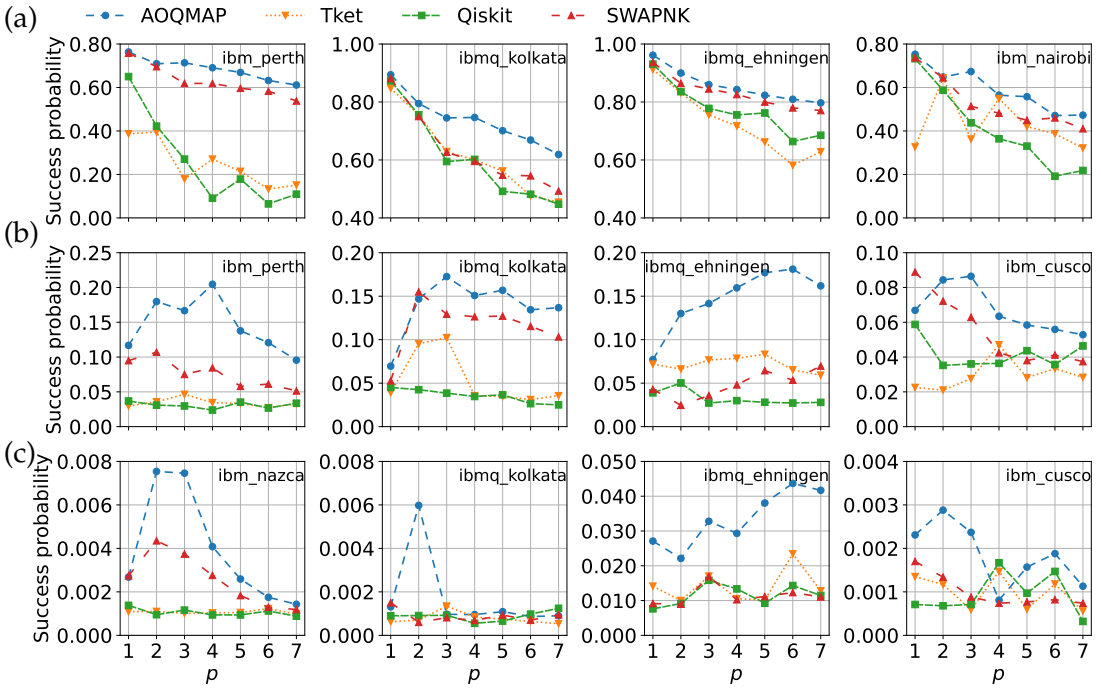

Figure 4.19: Success probability of QAOA obtained on various QPUs with AOQMAP, Tket, Qiskit, and SWAPNK. The labels are the same as Figure 4.18.

we will study in Chapter 5. Our framework offers a comprehensive strategy to fully exploit the capabilities of existing quantum hardware resources.

4.3.2 Reliability and quality

Reliable and high-quality quantum compilation are essential for maximizing the performance of algorithms. For statistical validation of circuits tailored to specific subtopologies, we can employ the Hellinger distance (Eq. 3.9) to compare the compiled circuit's outputs with the results from original circuit. A near-zero value indicates reliable compilation. Additionally, randomized benchmarking and quantum tomography offer verification of pulse-level optimizations. These techniques provide estimates independent of calibration data for optimized circuits. While challenging to apply to large-scale circuits, they verify specific gate operations on actual devices and the accuracy of gate optimizations throughout compilation. To quantify the quality of compilation, we propose a comprehensive set of metrics.

Compilation quality metrics

The following metrics can be used to assess the efficiency of the compilation process and the quality of compiled circuits:

- **Local preparation time**: The time required to prepare the compilation process locally on classical computers.

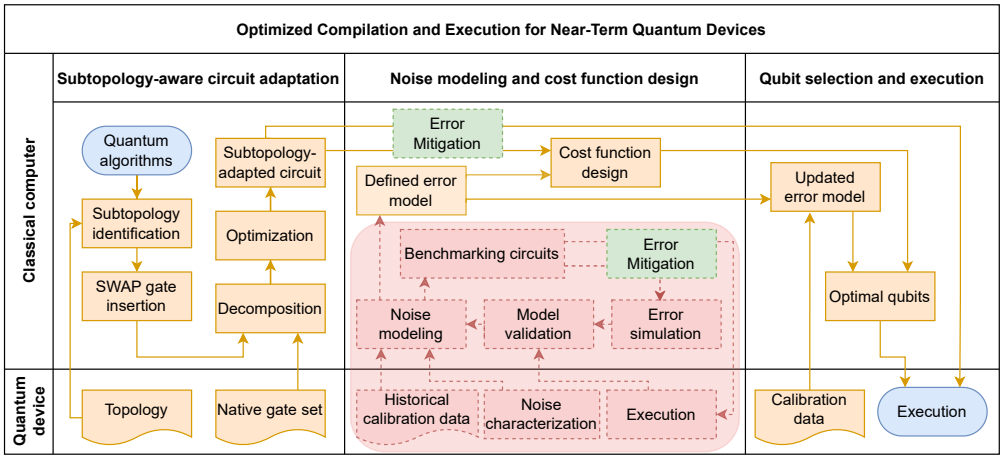

Figure 4.20: Optimized compilation and execution for near-term quantum devices. This flowchart outlines a process to optimize the execution of quantum algorithms on QPUs. Key steps include subtopology-aware circuit adaptation, noise modeling, cost function design, and qubit selection based on updated calibration data.

- **Quantum device access time**: The time required to generate the final compiled circuit after accessing real quantum devices.
- **Quantum state and process fidelity**: The fidelity between the compiled and original circuits' quantum states, simulated under varying noise conditions.
- **Increase in total gates**: The relative increase in the total gate count of the compiled circuit compared to the original.
- **Increase in two-qubit gates**: The relative increase in the two-qubit gate count of the compiled circuit compared to the original.
- **Increase in circuit depth**: The relative increase in the number of layers or depth of the compiled circuit compared to the original.

We introduce compilation time in detail. Compilation time consists of two primary components: local preparation time and quantum device access time. As illustrated in Figure 4.21, local preparation time addresses device connectivity constraints and refines the error model and cost function using classical computers, considering the target device's topology and native gate set. Historical characterization data can be leveraged to further enhance the precision of these models. Subsequently, quantum device access involves utilizing real-time calibration data to update the models for qubit selection and potentially performing individual qubit optimizations. For instance, selective optimization at the pulse level is advantageous by optimizing only quantum gates with a high error rate. Finally, the compiled circuit is executed on the quantum device. Local preparation time doesn't directly impact execution quality, but it affects the quality of the adapted circuits and the accuracy of the models guiding qubit selection. In contrast, quantum device access time directly influences execution quality. Extended access times may lead to outdated calibration data, and

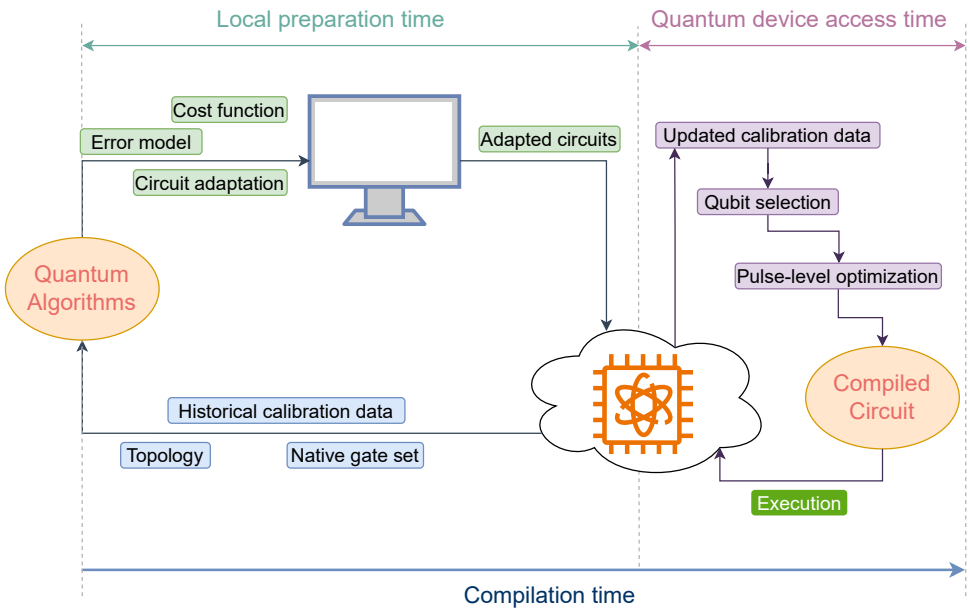

Figure 4.21: Breakdown of quantum circuit compilation time. Quantum circuit compilation time comprises two main components: local preparation time and quantum device access time. Local preparation time encompasses tasks such as: (i) adapting circuits to the specific topologies based on native gate sets and historical calibration data of the QPU; (ii) developing an error model that captures realistic noise; and (iii) formulating a custom cost function for qubit selection. Quantum device access time includes utilizing up-to-date calibration data and performing optimizations at the pulse level.

the preselected qubits may now have high error rates. We exclude queue time from job submission to execution; reserving quantum computers for execution can mitigate outdated calibration data issues. Minimizing quantum device access time ensures optimal utilization of the resource.

Based on our examination of diverse strategies aimed at improving the compilation of variational quantum circuits in Sections 4.1 and 4.2, a framework has been devised for the optimization of quantum algorithm compilation. This approach possesses the capacity to substantially enhance both the efficiency of compilation and the robustness of quantum circuits, thereby establishing a solid basis for the advancement of forthcoming compilers. In the subsequent chapter, our focus turns to the optimization of circuits at the pulse level and the augmentation of performance at the algorithmic level.

Chapter 5

Selective Optimization on Bipotent Architectures

Selective optimization is a strategy proposed for optimizing quantum circuits on bipotent architectures, where the same logical gate is natively implemented using two different pulse sequences with distinct quality on various qubit connections. Strategically optimizing specific gates within the circuit reduces possible computation efforts while enhancing performance, as only gates with lower quality can benefit significantly from optimization, while those with higher quality may not require it. While the focus here is on bipotent architectures, the underlying principles and insights can be applied to a broader range of quantum algorithms and hardware configurations. This chapter has been published in *Physical Review A* under the title *Optimizing quantum algorithms on bipotent architectures* [285].

5.1 Preliminaries

We first introduce the fundamental background of our methodology. We begin by providing motivation for our research. Then, we analyze different gate implementations and optimizations. Finally, we present features of bipotent architecture.

5.1.1 Motivation

Computational capabilities of today's quantum architectures are severely limited by comparatively high error rates of their hardware components. This shortcoming is commonly addressed on two levels: by providing improved implementations of qubits and quantum gates, and by adapting a given quantum circuit to error mechanisms and error rates of a specific quantum computer. A recent example from the first category is IBM's direct-CX gate that exhibits a shorter duration and a better fidelity than the previous entangling gate design based on the Echo Cross-Resonance (ECR) principle [31]. Approaches from the second category include noise-aware transpilation methods [225] and pulse-level optimization of algorithm primitives [286, 202].

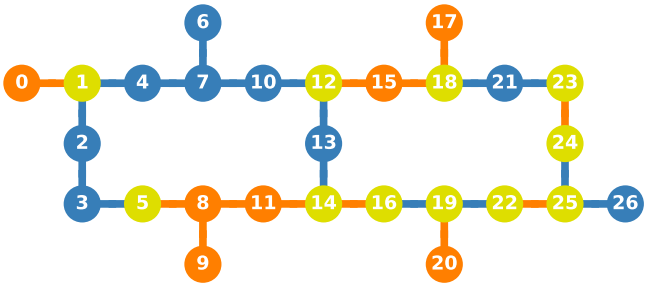

Figure 5.1: Bipotent architecture of ibmq_ehningen. It contains two types of CX gates: direct-CX, represented by blue or dark gray edges, and ECR-CX, represented by orange or medium gray edges. Qubits are depicted as circles, the lines between the qubits are connections where a CX gate can be applied. When only one type of CX gate can be applied, the qubit has the same color as the CX gate type, otherwise it is green (light gray).

Vigorous optimizations of quantum gates can lead to architectures where the optimized gate is not available for all existing qubits or qubit connections. For example, Figure 5.1 shows the topology of a QPU ibmq_ehningen. In this diagram, qubit pairs on which a direct-CX gate implementation is available share a blue (dark gray) edge, whereas qubit pairs on which only the ECR-CX gate implementation is available share an orange (medium gray) edge. While direct-CX gates tend to have better raw error rates, ECR-CX gates offer a significant advantage: they support bespoke pulse-level optimization, which is currently not available for direct-CX gates unless additional calibration is performed. Pulse-level optimizations have proven to be highly effective for regular quantum circuits, such as the QAOA [19], where frequently occurring sub-circuits are mapped to efficient, bespoke sequences of pulses [287, 286].

Here, we investigate the qubit mapping of QAOA onto quantum computers that provide two different implementations of the same gate on different pairs of qubits. We call such quantum architectures *bipotent*. Figure 5.1 is an example of a bipotent architecture, where two different CX gate types are available on different qubit pairs. Similarly, if there are more than two implementations of the same gate, we call such architectures *multipotent*. The reason for adopting a bipotent architecture may be the need to balance the quality of all qubits in engineering or manufacturing and the difficulty of having equally high-quality qubits due to various noises. The qubit mapping on bipotent architectures should balance between hardware-level and algorithm-level improvements, that is, trade the lower error rate of direct-CX gates for pulse-level optimization supported by ECR-CX gates. We focus on a specific quantum algorithm: QAOA for portfolio optimization (PortOpt) and maximum-cut (MaxCut) instances with denser connectivity.

We start by algorithm-agnostically benchmarking direct-CX versus ECR-CX gates, generating data specific to the platforms being used. We then introduce pulse-level optimizations for QAOA circuits applied to PortOpt. In particular, mapping a dense PortOpt problem to a rather sparse topology map such as in Figure 5.1 will require many SWAP gates, leading to a substantial contribution to the entire circuit's error rate. Good pulse-level optimizations are known for such SWAP gates. Note that

finding optimal pulse sequences is computationally expensive; for example, the compilation time of *gradient ascent pulse engineering* (GRAPE) scales exponentially in the size of the quantum algorithm [288]. For this reason, we only optimize the pulses of frequently occurring primitives of QAOA. The size of the primitives does not scale with the size of the algorithm. Moreover, we focus on improvements that rely on available calibration data and use pulse-level optimizations where possible, but do not consider solutions that require additional calibrations.

We report a comprehensive set of demonstrations on two of IBM's bipotent architectures. For the first time, we investigate not only the fidelities of QAOA circuits, but also their gate types and actual schedule durations on a bipotent architecture. Our results indicate that, for today's error rates, pulse-level optimization leads to a stronger error-rate decrease than using improved but monolithic native gates in most cases. This comes with a few unexpected observations. For example, the relative impact of pulse-level optimization and monolithic gates varies for circuits of different sizes; pulse-level optimizations tend to outperform direct-CX gates for medium-scale circuits. Moreover, the Control-Target polarity of CX gates, which is usually ignored by implementing algorithms, has a significant influence on the quality of the results.

The key contributions and novelties are as follows.

- For the first time, the features of a bipotent architecture are studied and used to optimize the implementation of algorithms on the QPUs.
- We improve the performance of QAOA for the first time by selectively optimizing partial gates implemented on some specific instead of all qubit pairs. This optimization technique can also be applied to other quantum algorithms.
- We describe how vigorously optimized quantum gates lead to bipotent quantum architectures and identify the conflict between using highly calibrated monolithic gates on the one hand, and pulse-level optimization on the other.
- We extensively study the performance of QAOA on noisy quantum computers for different problem sizes and depths using different strategies for selecting direct-CX or/and ECR-CX gates.
- We provide practical advice on how to map QAOA instances onto a given bipotent quantum architecture, thus improving the performance of QAOA on said architectures.
- All methods presented in this chapter do not require any additional calibration.

We use an exact and scalable approach on a linear topology aiming to minimize the circuit depth to pretranspile the circuit to satisfy the connectivity constraints. Then, we map the pretranspiled circuit to different qubits according to the circuit fidelity, gate type, and schedule duration. We provide first the features of a bipotent architecture. Then, we discuss pulse-level optimizations in QAOA and demonstrate the performance of QAOA with different optimizations on two QPUs. The case beyond QAOA and error mitigation techniques are then discussed. Additional details about the QPUs used in this study are presented in the Appendix A.1.

5.1.2 Gate design and optimization

Quantum gates are a crucial component of quantum algorithms, and their quality is paramount to algorithm performance. Researchers have explored various methods for enhancing gate quality, including investigating different measurement durations to improve readout fidelity [289] and implementing pulse-level optimizations to enhance gate fidelity. For example, GRAPE can be used to translate quantum algorithms directly into optimal hardware control pulses with shorter pulse lengths than a gate-based compilation, thereby improving fidelity, but with exponentially increasing compilation times. A partial compilation of variational algorithms [290] was developed to achieve the pulse speedups of GRAPE by precomputing optimal pulses for parametrization-independent blocks of gates. In addition, aggregating small gates into larger operations can reduce the compilation latency [201].

Compared to directly optimizing algorithms that require extensive computation, optimizing specific gates can improve the performance of the algorithm more effectively. In [286], the authors show a pulse-efficient transpilation approach that requires no additional calibration efforts by scaling Cross-Resonance (CR) pulses and removing redundant single-qubit operations to reduce the overall schedule duration of the gate and thus improves its fidelities. Faster and more reliable SWAP gates were developed at pulse-level with CR native gates on IBM's QPUs [291]. Moreover, using hardware primitive gates can reduce errors and run times resulting in improved performance [202]. In addition, a hybrid gate-pulse model [292] was proposed to improve the performance of QAOA on IBM's QPUs. Recently, the authors in [293] demonstrated an improvement in performance by achieving shorter pulse schedules through the optimization of pulse amplitude and duration.

We now introduce the pulse-level structure of CX gates. Figure 5.2(a) is a general symbol of the CX gate on IBM's QPUs. The direct-CX and ECR-CX gates are indistinguishable at the gate-level. However, there are differences in the pulse-level. Figures 5.2(b) and 5.2(c) show the implementations (up to a global phase) of a direct-CX gate on qubits [1, 4] and an ECR-CX gate on qubits [1, 0], respectively. While the direct-CX consists of one CR gate and one R_Z gate, the ECR-CX consists of one ECR gate and three single-qubit gates. A closer look into the hardware implementation, the ECR gate is implemented by two CR gates and one R_X gate on the control qubit, as shown in Figure 5.2(d). Their corresponding schedules consisting of several microwave pulses are shown in Figure 5.3. The direct-CX with a schedule duration of 245.3 ns is implemented by one Gaussian square pulse on the connecting channel U3, a parallel echoed Gaussian square pulse on the drive channel D4 of target qubit 4, and a virtual (digital) R_Z rotation (denoted by a ↻). In comparison, the ECR-CX gate with a duration of 320 ns consists of two CR pulses and four single-qubit gates. The schedule duration of the virtual R_Z gate is 0 ns, while that of R_Y and R_X gates is 32 ns.

Although direct-CX has a shorter schedule duration than ECR-CX, gates implemented by ECR-CX gates may perform better than those implemented by direct-CX, since they can be further optimized directly at the pulse-level by employing existing techniques in Qiskit [208]. Specifically, the ZZ gate implemented by ECR-CX gates allows pulse-level optimization without any additional calibration [286]. Compared to this, optimizing gates implemented by direct-CX gates currently requires additional calibration (e.g., [294]). Figures 5.2(e)-5.2(h) show the different implementations of ZZ

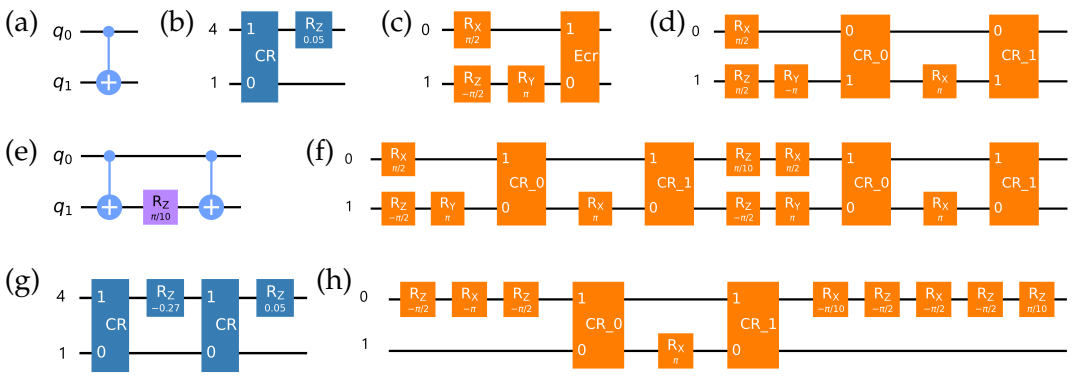

Figure 5.2: Different implementations of CX and ZZ gates on IBM's QPUs. (a) Symbol of CX gate. (b) direct-CX gate on qubits [1, 4]. (c) ECR-CX gate on qubits [1, 0]. (d) Hardware implementation of the ECR-CX gate. (e) Decomposition of ZZ gate. ZZ gate implemented by (g) direct-CX gates on qubits [1, 4] and (f) ECR-CX gates on qubits [1, 0]. (h) ZZ_{OPT} gate, the pulse-level optimization of (f).

gate on IBM's QPUs. It achieves a schedule duration of 490 ns on qubits [1, 4] using direct-CX gates, compared to a value of 640 ns on qubits [1, 0] using ECR-CX gates. However, the ZZ gate implemented by ECR-CX gates can be optimized at pulse-level resulting in a reduced duration of 241.8 ns. We refer to the ZZ gate optimized at pulse-level as ZZ_{OPT}. In addition, single-pulse-gates R_X and R_Y in ECR-CX enable gate cancellations, while a virtual R_Z gate with 0 ns schedule duration in direct-CX can not be further optimized.

As shown in [31], the direct-CX gate seems to have better quality. At the same time, the ECR-CX gate gives the user flexibility for pulse-level optimization [286], which will be discussed in Section 5.1.3.

5.1.3 Features of bipotent architecture

We now investigate the features of a bipotent architecture ibmq_ehningen, as shown in Figure 5.1, on which two types of CX gates are present: direct-CX and ECR-CX. Additionally, we denote the qubits as Q-ECR (orange, medium gray), Q-Direct (blue, dark gray), and Q-Bipotent (green, light gray), depending on whether the qubit is connected to only ECR-CX, only direct-CX, or both.

We collected hourly calibration data of ibmq_ehningen throughout September 2022. Figure 5.4 shows the CX gate error rate and gate execution time of ECR-CX and direct-CX. While the error rate varies over time, the gate time is constant. Their average values are summarized in Table 5.1. Compared to ECR-CX, the average error rate of direct-CX is slightly reduced by 4.82%, while the average gate time is significantly reduced by 32.79%. The data show that direct-CX is generally of better quality than ECR-CX.

Figures 5.5(a) and 5.5(b) show the decoherence times T_1 and T_2, respectively, of the 27 qubits on ibmq_ehningen. Qubits connected to two types of CX gates (Q-Bipotent) have a higher average value of T_1 and T_2 compared to qubits connected only to ECR-CX (Q-ECR) or only to direct-CX gates (Q-Direct). While the precise reason

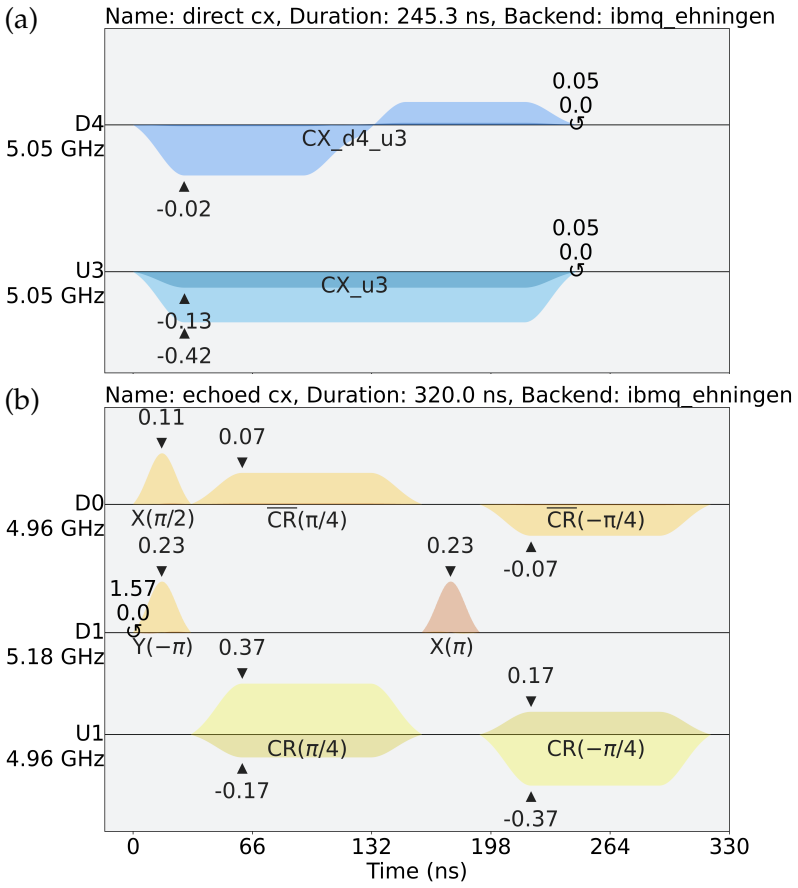

Figure 5.3: Schedules of (a) direct-CX on qubits [1, 4] and (b) ECR-CX on qubits [1, 0].

for this trend is unknown, it appears logical that more robust qubits are considered suitable for implementing both types of gates and are designated Q-Bipotent. This implies the advantage of using a combination of both types of CX gates. Single-qubit gate error and readout error are shown in Figures 5.5(c) and 5.5(d), respectively. The mean gate errors are comparable for three qubit types. However, the Q-ECR has a higher readout error overall than the Q-Direct and the Q-Bipotent. Throughout this chapter, we use the default readout error mitigation provided by Qiskit [208] to reduce the impact of measurements on results. The averages of decoherence times and error rates of qubits are summarized in Table 5.2.

The study on bipotent architecture shows that direct-CX gates have significantly reduced schedule duration. In addition, Q-Bipotent has the highest decoherence

Table 5.1: Properties of ECR-CX and direct-CX gates.

Parameter	ECR-CX	direct-CX	Reduction (%)
Gate error (%)	0.83	0.79	4.82
Gate time (ns)	382.22	256.89	32.79

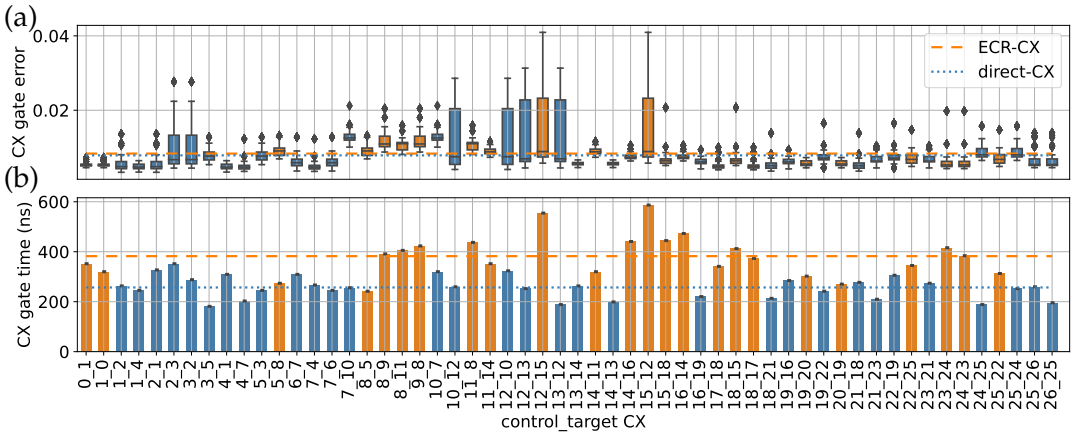

Figure 5.4: Properties of ECR-CX (orange, medium gray) and direct-CX (blue, dark gray) gates on ibmq_ehningen. (a) Gate error rate. (b) Execution time in nanoseconds.

Table 5.2: Properties of qubits.

Parameter	Q-ECR	Q-Direct	Q-Bipotent
T_1 (µs)	122.59	154.29	161.16
T_2 (µs)	132.94	118.02	175.16
Gate error (%)	0.021	0.023	0.026
Readout error (%)	1.386	1.009	1.077

times T_1 and T_2 overall, while on average Q-Direct has a better T_1 than Q-ECR that has a better T_2.

5.2 Optimizing algorithm native gates at pulse level

In this section, we explore potential optimization strategies at the pulse level, leveraging the features of the bipotent architecture. Our investigation begins by analyzing gates within the algorithm. Subsequently, we focus on optimizing these identified gates and demonstrate the results experimentally.

5.2.1 Algorithm-native gate set

Studying the properties of an algorithm is crucial to improve the performance of its implementation. One of the most important properties is the Algorithm-Native Gate Set (ANGS), which is the set of gates used to construct the algorithm. Efficient implementations of these gates will facilitate transpilation, enable exploitation of pulse-level optimization, and ultimately result in better performance.

Figure 5.6 shows the pre-transpiled circuit of QAOA for PortOpt with 5 qubits and $p = 1$ on a linear topology using the strategy AOQMAP presented in Section 4.2 that provides optimal and scalable solutions. The resulting circuit has a structure similar to a swap network (see e.g., [218]), but two layers are skipped per p, namely SWAP gates after the first and last layers of ZZ gates in every p. Thus, in the gate set {H, R_X, R_Z, ZZ, ZZ-SWAP}, the circuit with n qubits has a depth of $2 + (n + 2) \times p$ including initial state and measurement operators.

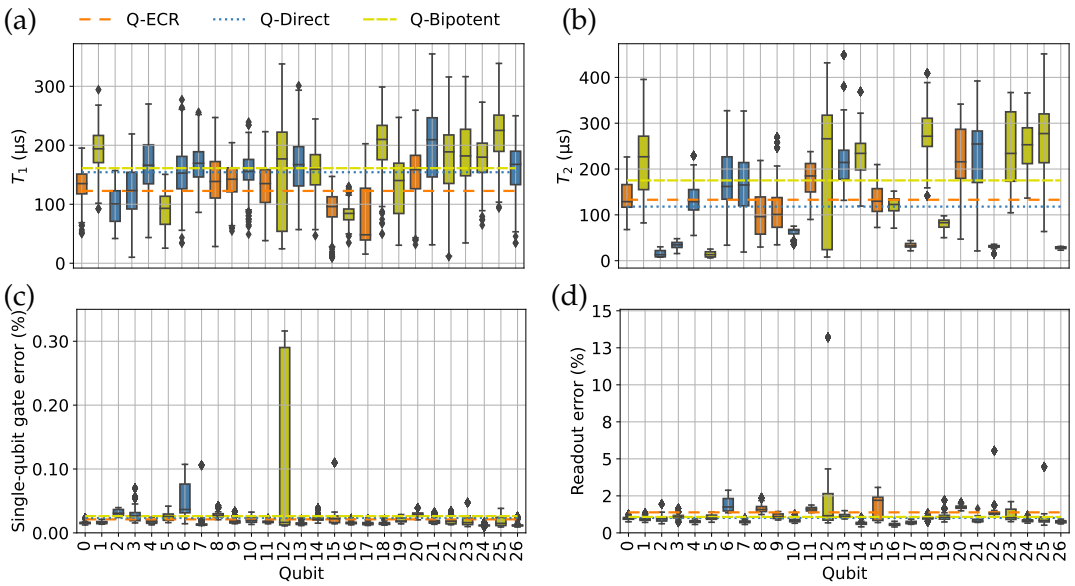

Figure 5.5: Properties of 27 qubits on ibmq_ehningen. (a) T_1 and (b) T_2 decoherence times. A higher value is better. (c) Single-qubit gate error and (d) readout error. A lower value is better. We label qubits connected to two types of CX gates as Q-Bipotent (green, light gray) and qubits connected only to ECR-CX or direct-CX gates as Q-ECR (orange, medium gray) or Q-Direct (blue, dark gray), respectively.

With the overall structure of QAOA circuit in Figure 5.6, the ANGS of QAOA for PortOpt is {H, ZZ, SWAP-ZZ, R_Z, R_X}, while that of QAOA for MaxCut is {H, ZZ, SWAP-ZZ, R_X}, as no R_Z is required for Eq. 3.14. Therefore, we focus on improvements of these gate types based on different CX primitives.

5.2.2 Analyzing two-qubit gates

We now analyze two-qubit gates. The CZ, ZZ, and ZZ-SWAP gates are both undirected, which means that a more efficient implementation is possible with the native

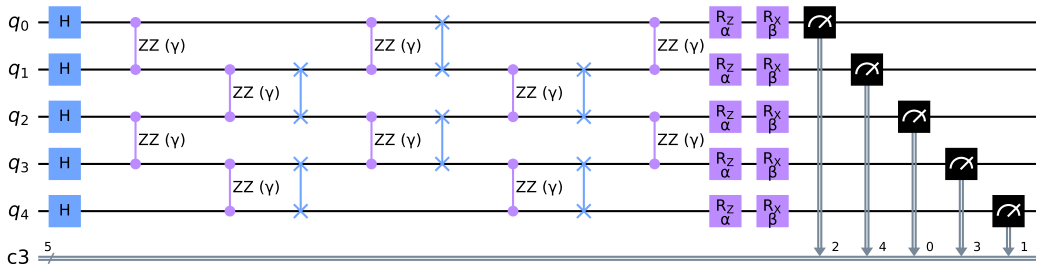

Figure 5.6: Pre-transpiled circuit of QAOA for PortOpt with 5 qubits and $p = 1$ on a linear topology with AOQMAP proposed in Section 4.2. For higher p, the circuit between the initial state and measurement is repeated p times, each time with new parameters.

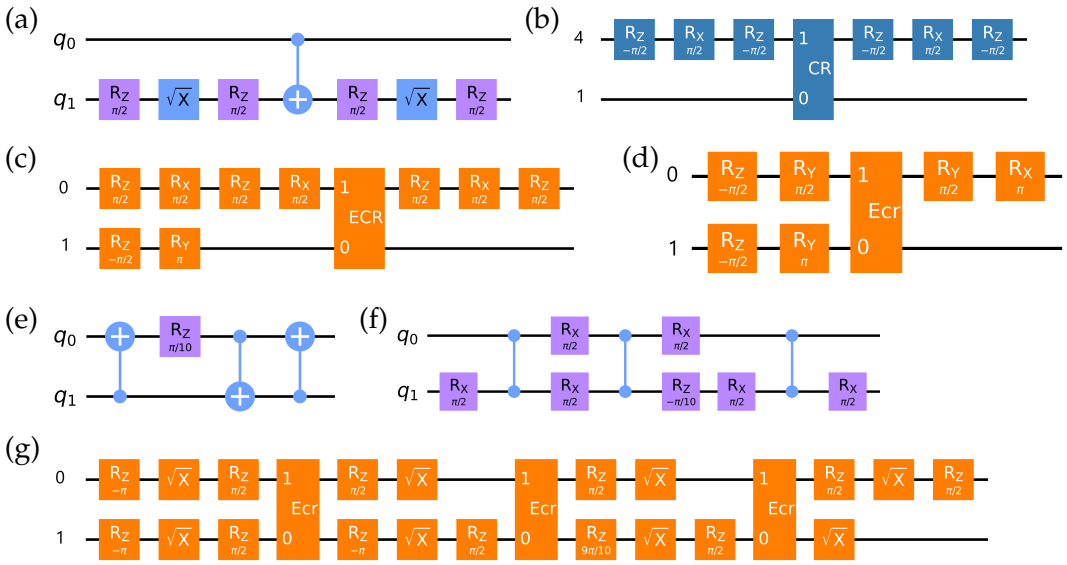

Figure 5.7: Different implementations (up to a global phase) of CZ and ZZ-SWAP gates on IBM's QPUs. (a) Decomposition of CZ gate. CZ gate implemented by (b) direct-CX gate on qubits [1, 4] and (c) ECR-CX gate on qubits [1, 0]. (d) CZ_{OPT} gate, the pulse-level optimization of (c). (e) Decomposition of ZZ-SWAP gate. (f) Alternative decomposition of the ZZ-SWAP gate into CZ gates. (g) ZZ-$SWAP_{OPT}$ gate, the implementation of (f) on qubits [1, 0] using ECR-CX gates with the pulse-level optimization.

CX gate on the QPU. Additionally, gate cancellation at pulse-level provides opportunities for optimizations of gates implemented by ECR-CX gates.

Figure 5.7(a) presents the decomposition of CZ gate on the IBM quantum device. Figure 5.7(b) shows the default implementation of a CZ gate using direct-CX on qubits [1, 4] with a schedule duration of 309.3 ns. Although the implementation using ECR-CX on qubits [1, 0] shown in Figure 5.7(c) has a longer duration of 384 ns, it can be optimized at the pulse-level (CZ_{OPT}) by reducing three single-qubit gates, resulting in a duration of 352 ns, as shown in Figure 5.7(d). The decompositions of ZZ-SWAP gate into CX gates and CZ gates [295] are shown in Figures 5.7(e) and 5.7(f), respectively. In contrast to the CZ gate, the CX gate is directed. The ZZ-SWAP gate has a schedule duration of 992 ns on qubits [1, 0] and 800 ns on qubits [1, 4]. To implement the ZZ-SWAP gate with CZ gates, we use CZ_{OPT} implemented by ECR-CX since the CZ gate is not a basis gate of IBM's QPUs, and then take advantage of single-qubit cancellation at the pulse-level, resulting in the same duration as the standard ZZ-SWAP gate. The optimized ZZ-SWAP gate (ZZ-$SWAP_{OPT}$) is shown in Figure 5.7(g).

The data show that implementing gates using vigorously optimized monolithic gates results in a shorter duration, while using ECR-CX gates allows for user-friendly pulse-level optimization without requiring additional calibration.

5.2.3 Experimental results

We have discussed the pulse-level optimization of gates implemented by ECR-CX gates. We now use the *process fidelity* to benchmark the performance of default and

optimized gates on the qubits that support ECR-CX gates. The process fidelity can be determined by the Quantum Process Tomography (QPT) using Qiskit's standard implementation. The QPT is a method to characterize the actual behavior of quantum gates on QPUs. For QPT experiments, several initial states need to be prepared and then evolved, followed by numerous measurements at different measurement bases. The quantum channel is then reconstructed from the measurement data. The process fidelity between two quantum channels η and ξ is given by:

$$F(\eta, \xi) = \mathrm{Tr}\left[\sqrt{\sqrt{\rho_\eta}\rho_\xi\sqrt{\rho_\eta}}\right]^2, \tag{5.1}$$

where ρ_η and ρ_ξ are the normalized *Choi* matrices for the channel η and ξ, respectively. In our case, ρ_η is the *Choi* matrix of the ideal operator, while ρ_ξ is the *Choi* matrix determined by the QPT process on the QPU.

The ZZ and ZZ-SWAP denote gates implemented by the default basis gates of IBM's QPUs, while the $\mathrm{ZZ}_{\mathrm{OPT}}$ and ZZ-$\mathrm{SWAP}_{\mathrm{OPT}}$ refer to gates with pulse-level optimizations. In addition, we employ the notations CT and TC to distinguish between the gate implemented by CX gates with Control-Target (CT, in hardware native CX direction) and Target-Control (TC, opposite to hardware native CX direction).

Figure 5.8(a) shows the infidelity $(1 - F)$ of ZZ gate on qubits [5, 8] (left) with a gate repetition of 1 and on qubits [1, 0] (right) with a gate repetition of 10, respectively, as a function of angle. The schedule duration of $\mathrm{ZZ}_{\mathrm{OPT}}$-CT and $\mathrm{ZZ}_{\mathrm{OPT}}$-TC increases with angle, while that of ZZ-CT and ZZ-TC is constant since CX gate has a fixed duration and $\mathrm{R_Z}$ gate is virtual. $\mathrm{ZZ}_{\mathrm{OPT}}$-CT has the best overall performance as its duration is the shortest. Figure 5.8(b) shows the infidelity of the ZZ-SWAP gate on qubits [0, 1] with gate repetitions of 10 (left) and 20 (right), respectively. We benchmark the performance of ZZ-SWAP gate implemented by default CX gates considering CT and TC and by $\mathrm{CZ}_{\mathrm{OPT}}$ gates considering CT. The schedule durations of ZZ-SWAP-CT and ZZ-$\mathrm{SWAP}_{\mathrm{OPT}}$-CT are the same, while that of ZZ-SWAP-*TC* is longer. The ZZ-$\mathrm{SWAP}_{\mathrm{OPT}}$ gate yields the best performance by decomposing the ZZ-SWAP into CZ gates, performing pulse-level optimization, and taking into account the Control-Target polarity of CX gates.

The results confirm that using hardware native polarity Control-Target to implement undirected gates, such as ZZ and SWAP-ZZ, is more efficient than using Target-Control polarity which results in a longer duration. In addition, gates optimized at pulse-level perform better than their default implementation. In the benchmarking below, we implement the undirected gates with Control-Target.

5.3 Methodology and benchmark experiments

In this section, we outline our methodology for demonstrating selective optimization and evaluate it on several IBM quantum devices.

5.3.1 Methodology

To implement QAOA, we first prepare pretranspiled circuits. This process ensures that the circuits satisfy the connectivity constraints on a linear topology. The next

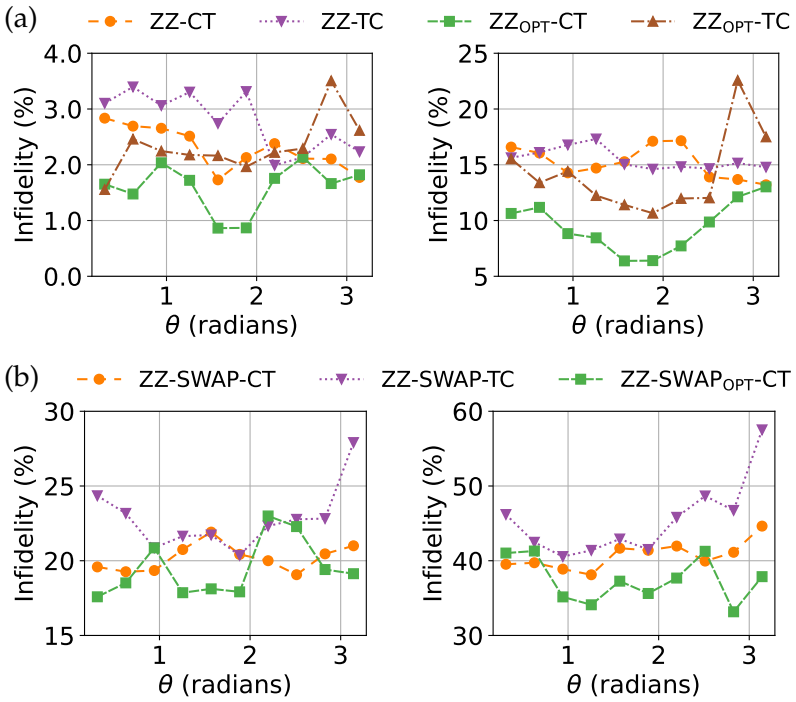

Figure 5.8: Infidelity determined by QPT on the ibmq_ehningen. For (a), the ZZ gate acts on qubits [5, 8] with a gate repetition of 1 (left) and [1, 0] with a gate repetition of 10 (right). For (b), the ZZ-SWAP gate acts on qubits [1, 0] with a repeated number of gates of 10 (left) and 20 (right).

crucial step is to identify high-quality qubits to execute them. To evaluate the qubit quality, we first consider only the circuit fidelity. As illustrated in Figure 5.9, we consider four families of QAOA circuits for evaluation. ECR-, Direct- and Global-circuits connected with violet (dark gray) lines use the qubits selected by mapomatic [236] aiming to maximize the fidelity of the circuit. In comparison, the Bipotent-circuits connected with green (light gray) lines take into account the schedule duration, fidelity, and gate type. Moreover, ECR-circuits utilize only the ECR-CX gates (with or without pulse-level optimizations). Direct-circuits are composed of only direct-CX gates, while Global-circuits and Bipotent-circuits contain mixtures of both gate types.

To construct an ECR-circuit with k qubits, mapomatic [236] selects a maximum-fidelity linear arrangement of k qubits such that ECR-CX gates are available on neighboring qubits. We use three versions of such circuits: ECR-Default that employs default implementations of ZZ and ZZ-SWAP gates; ECR-ZZ$_{\mathrm{OPT}}$ that utilizes pulse-level-optimized ZZ gate performed with the reduced CR gate; and ECR-ZZ-SWAP$_{\mathrm{OPT}}$ that leverages pulse-level optimizations for ZZ gates and reconstructs ZZ-SWAP gates with CZ$_{\mathrm{OPT}}$ gates. We maintain the same set of qubits for each data point if optimizations are implemented.

Direct-circuits exclusively utilize direct-CX gates, and best-fidelity linear arrangements of qubits that support such gates are selected, again using mapomatic. It is worth noting that there are no pulse-level optimizations in Direct-circuits. Global-circuits are best-fidelity linear arrangements of arbitrary qubits, regardless of whether

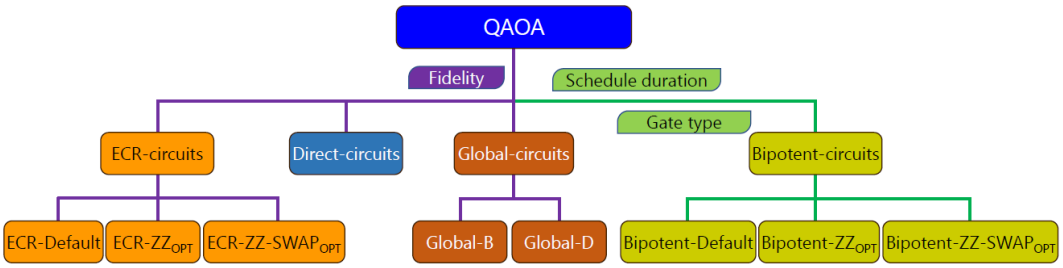

Figure 5.9: QAOA circuits. ECR-circuits, Direct-circuits, and Global-circuits consider fidelity and restrict the CX gate type in the circuit to only ECR-CX, only direct-CX, and regardless of its type, respectively, whereas Bipotent-circuits have a trade-off between fidelity, schedule duration, and gate type.

their connections support ECR-CX or direct-CX. It is possible, but not necessary, that all selected qubits of a Global-circuit have only direct-CX links. We call such configurations Global-D (for "direct"), and configurations with mixtures of direct-CX and ECR-CX Global-B (for "bipotent"). We observe that due to the higher error rate of ECR-CX gates, as shown in Figure 5.4, only ECR-CX-linked qubits are not selected in Global-circuits.

In addition to those circuits where qubits are selected based on the circuit fidelity, we consider a further bipotent arrangement, called Bipotent-circuits. As the schedule duration of a circuit on the selected qubits has a fixed value and can be determined directly by Qiskit, Bipotent-circuit optimizes duration while controlling its fidelity. We focus only on qubits with below-average single-qubit and two-qubit error rates. Among these qubits, we select a linear arrangement on which the pulse-level optimized circuit has the shortest schedule duration. Moreover, we enforce bipotency by requiring at least one ECR-CX and at least one direct-CX gate. Pulse-level optimizations can be applied to ECR-CX gates within such circuits; the resulting circuits are labeled Bipotent-Default, Bipotent-ZZ_{OPT} and Bipotent-ZZ-$SWAP_{OPT}$, analogously to ECR-circuits above.

5.3.2 Benchmarks and performance metrics

We consider as benchmarks the circuits with different numbers of qubits and depths p. The maximum number of qubits that can be connected linearly using only ECR-CX gates is five, e.g., [9, 8, 11, 14, 16], as illustrated in Figure 5.1. Therefore, we limited the number of qubits to five to allow a full comparison of qubits with different CX gate types.

The five-asset PortOpt instance that runs on five qubits (5Q) is taken from [94], and its first three and four assets are used for the three- (3Q) and four-qubit (4Q) instances, respectively. The values of the variables q, B, A, and λ vary across the three cases considered. For 3Q, they are 0.33, 2, 0, and 20.97, respectively. For 4Q, they are 0.33, 2, 0.13, and 17.99, respectively. Finally, for 5Q, they are 0.33, 3, 0.07, and 17.51, respectively. For MaxCut we consider the *complete graph*, which results in the same connectivity requirements as the dense PortOpt problem instances.

To evaluate the performance of QAOA for PortOpt, we define the approximation ratio (AR) and success probability (SP). A higher value of AR or SP implies better

performance. Compared to the previous results, here we employ the post-selected results that satisfy the budget constraint (all feasible solutions) to calculate AR of n assets $\{z_1, ..., z_n\}$, defined as follows:

$$r(z_1, \ldots, z_n) = C(z_1, \ldots, z_n)/C_{\text{opt}}, \tag{5.2}$$

where $C(z_1, ..., z_n)$ and C_{opt} are the mean value found by QAOA and optimal value, respectively. The SP is defined as the probability of optimal solution. Similarly, the AR of QAOA for MaxCut is defined by:

$$r(z_1, ..., z_n) = C(z_1, ..., z_n)/C_{\text{max}}, \tag{5.3}$$

with $C(z_1, ..., z_n)$ the mean value found by QAOA and $C_{\text{max}} = \max C(z_1, ..., z_n)$ the optimal value.

We simulate the circuits in the noiseless case with the Qasm simulator to get the values of AR and SP of QAOA for PortOpt and MaxCut that can be used as a baseline. For the demonstrations on QPUs, we set the number of shots to 50000 for each circuit. Table 5.3 presents the simulated values of the AR and SP of QAOA for PortOpt with $p \in \{1, 2, ..., 5\}$, while Table 5.4 displays the results of QAOA for MaxCut. The data show that QAOA with 5 qubits for PortOpt exhibits smaller AR and SP values compared to MaxCut for the specific instances considered.

In addition to AR and SP, we use the schedule duration and the number of CX gates to study the properties of the circuit. A lower schedule duration means that the circuit executes faster, while a reduced number of CX gates implies pulse-level optimizations.

5.3.3 Results analysis

QAOA for portfolio optimization

We first benchmark the performance of QAOA for PortOpt with ECR-circuits, Global-circuits, and Direct-circuits. Then, we demonstrate the performance of QAOA with Bipotent-circuits and explore the pulse-level optimizations.

The results of QAOA-PortOpt with 3Q, 4Q, and 5Q are shown in Figures 5.10(a–c), respectively. The ECR-Default has the largest schedule duration. In the ECR-ZZ_{OPT}, the number of CX gates is reduced as the ZZ gates are optimized. Furthermore, the ZZ-SWAP_{OPT} gate is implemented by CZ_{OPT} gates, therefore there are no CX gates in the ECR-ZZ-SWAP_{OPT}. In Figure 5.10(a), the qubits used in the ECR-circuits are

Table 5.3: AR and SP of QAOA for PortOpt without noise.

p	1	2	3	4	5
3Q-AR	0.993	0.996	0.998	1.000	1.000
3Q-SP (%)	99.34	99.57	99.83	99.99	99.98
4Q-AR	0.525	0.674	0.892	0.930	0.935
4Q-SP (%)	21.18	29.04	44.30	45.61	46.97
5Q-AR	0.513	0.819	0.828	0.898	0.909
5Q-SP (%)	17.70	48.61	45.28	55.33	58.35

Table 5.4: AR and SP of QAOA for MaxCut without noise.

p	1	2	3	4	5
5Q-AR	0.914	0.972	0.996	0.997	0.997
5Q-SP (%)	87.79	96.99	99.45	99.62	99.72

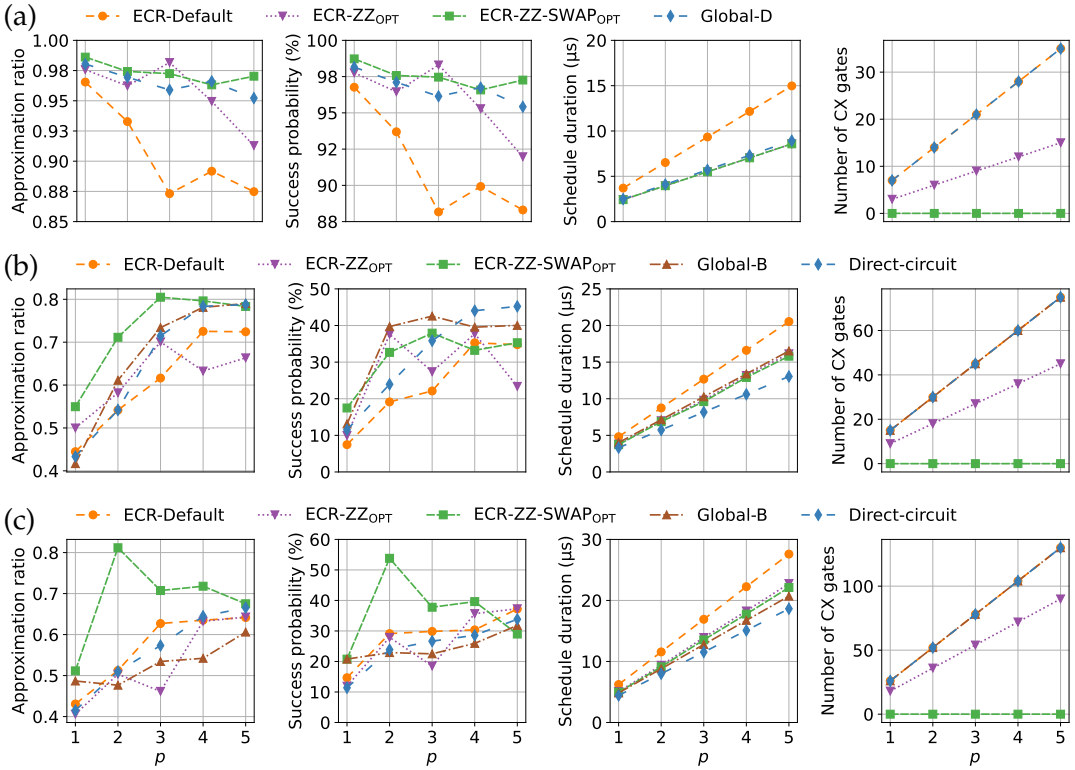

Figure 5.10: Approximation ratio, success probability, schedule duration, and CX gate count of QAOA for PortOpt with p from 1 to 5 using ECR-, Global-, and Direct-circuits on ibmq_ehningen for (a) 3Q, (b) 4Q, and (c) 5Q. The average CX gate error rates are 0.48% in all Global-circuits, they are 0.78%, 0.87%, and 0.89% in ECR-circuits for 3Q, 4Q, and 5Q, respectively, while 0.48% and 0.60% in Direct-circuits for 4Q and 5Q, respectively.

[16, 14, 11], while in Global-D they are [18, 21, 23]. The ECR-ZZ-SWAP$_{OPT}$ has the best performance, although the average CX gate error rate in ECR-circuits is 1.6 times higher than that in the Global-D. Optimizing both ZZ and ZZ-SWAP gates yields better and more stable AR and SP than optimizing only ZZ gates. Figure 5.10(b) shows the results of QAOA using ECR-circuits with qubits [16, 14, 11, 8], Global-B with qubits [18, 21, 23, 24], and Direct-circuit with qubits [1, 4, 7, 6]. Despite having the highest fidelity, Global-B does not perform the best. In comparison, ECR-ZZ-SWAP$_{OPT}$ achieves the highest AR values, although its average CX gate error rate is 1.8 times higher and the duration is much longer. We believe this advantage comes from the combination of ZZ$_{OPT}$ and ZZ-SWAP$_{OPT}$ gates. The Direct-circuit has a shorter schedule duration than Global-B resulting in comparable AR values and better SP values for higher p. Figure 5.10(c) shows a performance comparison of 5Q-QAOA for PortOpt using ECR-circuits with qubits [16, 14, 11, 8, 9], Global-B with

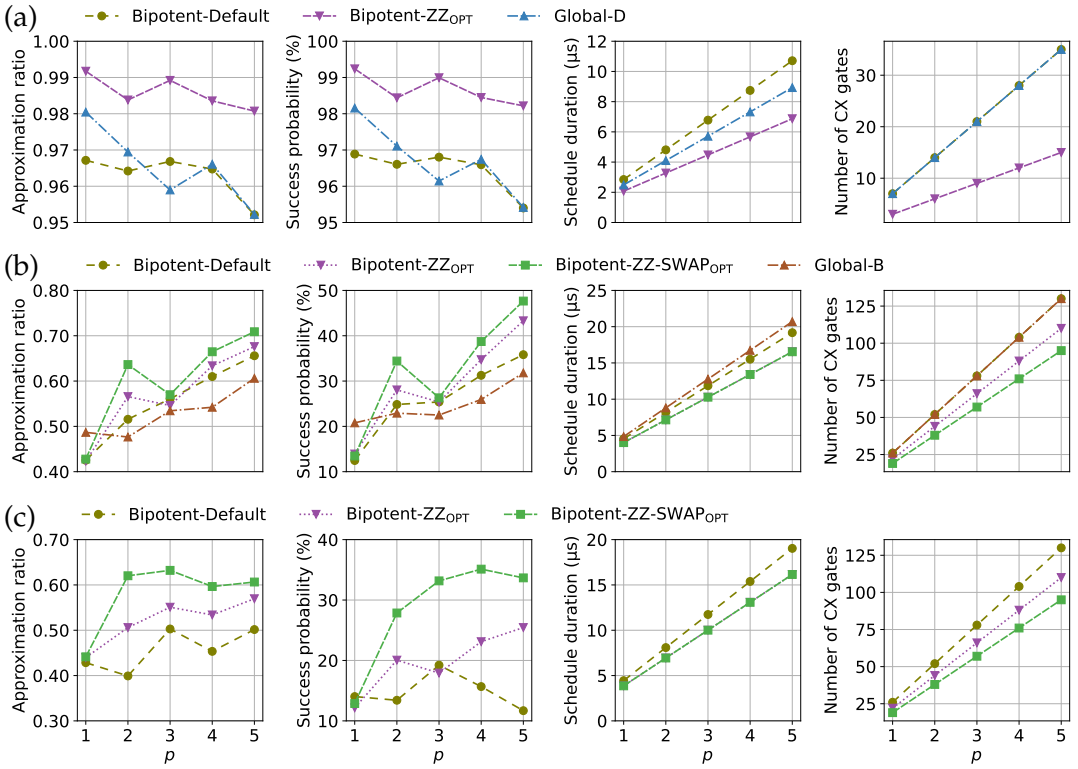

Figure 5.11: Benchmarking results of QAOA for PortOpt using Bipotent- and Global-circuits on ibmq_ehningen for (a) 3Q, (b) 5Q, and (c) on ibm_auckland for 5Q. The average CX gate error rates for each data point in (a), (b), and (c) are 0.60%, 0.65%, and 0.67%, respectively.

qubits [17, 18, 21, 23, 24] including two ECR-CX gates, and Direct-circuit with qubits [6, 7, 4, 1, 2]. While the Global-B and Direct-circuit exhibit better fidelity and shorter schedule durations, ECR-ZZ-SWAP$_{OPT}$ outperforms them with the highest AR values. Therefore, despite its longer duration, ECR-ZZ-SWAP$_{OPT}$ can be considered as a promising alternative for achieving higher accuracy in quantum computing. A possible explanation is that combining ZZ$_{OPT}$ and ZZ-SWAP$_{OPT}$ produces a pulse that is more resilient to noise than the others.

We now benchmark the performance of QAOA with Bipotent-circuits (optimized for schedule duration rather than for fidelity). Figure 5.11(a) shows the results of 3Q-QAOA on ibmq_ehningen. The qubits used in Bipotent-circuits are [22, 25, 26] including one ECR-CX on qubits [22, 25] and one direct-CX on qubits [25, 26]. Compared to Bipotent-Default and Global-D, Bipotent-ZZ$_{OPT}$ demonstrates better performance, as indicated by its higher AR and SP values. The decrease in the number of CX gates achieved through the use of optimized ZZ gates suggests that partial ZZ gates originally implemented with default CX gates have been optimized at the pulse-level. This optimization leads to a reduction in schedule duration. The Bipotent-circuits in Figure 5.11(b) employ a set of five qubits, namely [11, 14, 13, 12, 10], among which only a single pair of qubits is capable of performing an ECR-CX gate. The optimization of ZZ gates implemented using ECR-CX gates in Bipotent-ZZ$_{OPT}$ leads to a reduction in the number of CX gates and a shorter schedule duration. Bipotent-ZZ-SWAP$_{OPT}$ further optimizes ZZ-SWAP gates by replacing

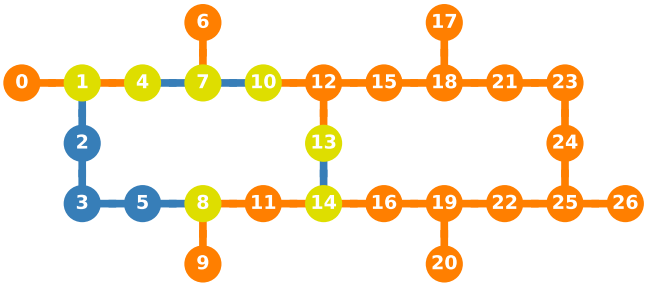

Figure 5.12: Bipotent architecture of ibm_auckland. It has the same label as ibmq_ehningen.

certain ECR-CX-based ZZ-SWAP gates with CZ_{OPT}-based ZZ-$SWAP_{OPT}$ gates, thus further decreasing the number of CX gates. Although Global-B has better fidelity, the demonstrations show that Bipotent-circuits outperform it for $p \geq 2$. Overall, among all the methods tested, Bipotent-ZZ-$SWAP_{OPT}$ exhibits the best performance in terms of the improvement in both the AR and SP metrics, making it a promising approach for quantum circuit optimization. Figure 5.11(c) presents the benchmark results of 5Q-QAOA using qubits [11, 8, 5, 3, 2] on ibm_auckland. It is worth noting that ibm_auckland has a different bipotent architecture compared to ibmq_ehningen, as illustrated in Figure 5.12. The results on ibm_auckland are consistent with the trend of 5Q-QAOA for PortOpt observed on ibmq_ehningen. Specifically, we observe that QAOA with Bipotent-ZZ-$SWAP_{OPT}$ outperforms the others. These results highlight the potential of the proposed optimization approach in improving the performance of quantum algorithms on other QPUs.

The data show that leveraging a combination of ZZ_{OPT} and ZZ-$SWAP_{OPT}$ gates can significantly enhance both AR and SP, surpassing the performance achieved by solely optimizing ZZ gates. This approach also outperforms Global-circuits with the highest fidelity. Specifically, when compared to Global-circuits, we observe an improvement in AR of up to 70%, on average 29%, for ECR-ZZ-$SWAP_{OPT}$ at $p = 2$. Similarly, on ibm_auckland, Bipotent-ZZ-$SWAP_{OPT}$ shows an improvement of up to 55%, on average 27%, compared to Bipotent-Default. On ibmq_ehningen, Bipotent-ZZ-$SWAP_{OPT}$ achieves an improvement of up to 34%, on average 14%, compared to Global-B.

QAOA for MaxCut

We now investigate the performance of QAOA for MaxCut with 5 qubits. The qubits used in ECR-circuits, Global-B, and Direct-circuit are [16, 14, 11, 8, 9], [6, 7, 4, 1, 0], and [6, 7, 4, 1, 2], respectively. Figure 5.13(a) shows that Global-B, which has the highest fidelity, performs better for smaller p, whereas Direct-circuit, which has the lowest schedule duration, performs better for larger p. These results suggest that, when using default IBM gates, fidelity is more critical for medium-scale circuits, while a shorter schedule duration is more important for larger circuits. One possible explanation is that, as the size of the circuit increases, the effects of decoherence become dominant and prevail over the variability in qubit error rates. ECR-circuits with higher error rates and longer schedule durations lead to lower AR compared to other methods.

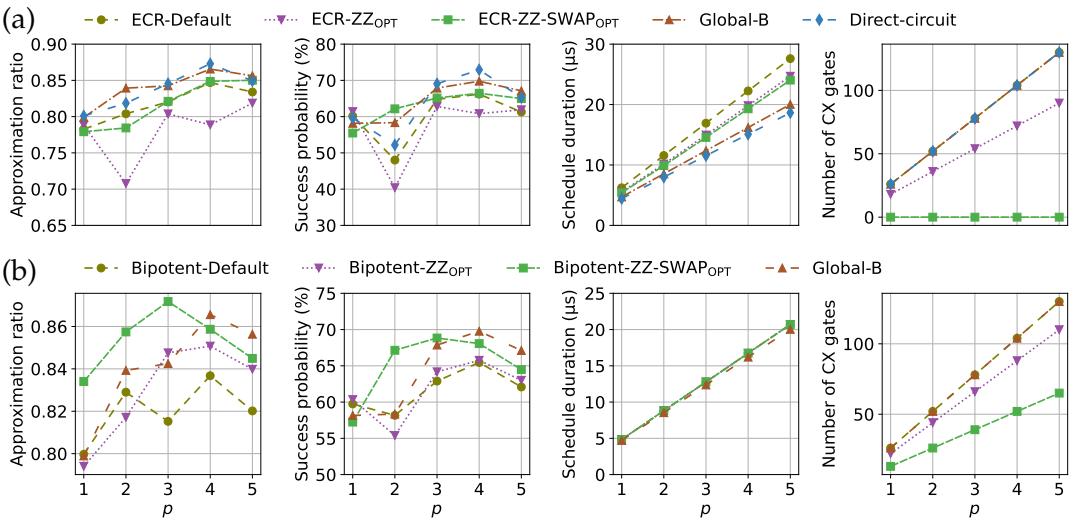

Figure 5.13: Benchmarking results of 5Q-QAOA for MaxCut on ibmq_ehningen using two sets of circuits: (a) ECR-, Global-, and Direct-circuits, and (b) Bipotent- and Global-circuits. The average CX gate error rates in the ECR-circuits, Global-B, and Direct-circuit are 0.89%, 0.48%, and 0.60%, respectively, while that of each data point in (b) is 0.48%.

However, by combining ZZ$_{OPT}$ and ZZ-SWAP$_{OPT}$, ECR-ZZ-SWAP$_{OPT}$ achieves stable performance in both AR and SP. Figure 5.13(b) shows the results of QAOA using Bipotent-circuits with qubits [17, 18, 21, 23, 24] including two ECR-CX and two direct-CX gates. While all three types of Bipotent-circuits have the same schedule duration, Bipotent-ZZ-SWAP$_{OPT}$ exhibits a clear advantage, as demonstrated by significant improvements in both AR and SP. However, Global-B exhibits better performance for larger values of p due to its shorter duration.

Our study highlights the importance of efficient gate implementation, achieved through increasing fidelity and reducing schedule duration, in enhancing the algorithm's performance. However, we found that the use of different gate types in a circuit can result in significantly divergent performances, even when the duration is identical. Therefore, careful consideration must be given to the selection of gate types to optimize the performance.

5.3.4 Discussion

Our results demonstrate that selective pulse optimization tailored to the algorithm structure outperforms the direct use of high-quality native gates. Furthermore, the analysis reveals that primitive gate fidelities alone do not determine algorithm performance. The gate type and circuit schedule duration should also be considered for optimal compilation. While the primary focus of this research is on optimizing QAOA on bipotent architectures without requiring additional calibration, the insights presented here have broader implications for other quantum algorithms and architectures. Specifically, some two-qubit gates in an algorithm may only achieve high fidelity after a time-consuming gate calibration, resulting in a notable increase in overhead costs [296]. However, our study suggests a *selective calibration* approach that can reduce the potential overhead by calibrating only the two-qubit gates implemented

on specific qubit pairs with a higher CX gate error rate, instead of calibrating all two-qubit gates. This approach enhances algorithm performance by enabling more efficient resource utilization, leading to faster and more accurate results. Moreover, pulse-level optimization and various gate decomposition strategies can be combined to significantly improve algorithm performance. By investigating the resilience of different pulse shapes to noise, we can identify more robust pulse shapes that are less affected by noise and errors in the computation. Consequently, by leveraging these resilient pulse shapes, we can achieve better performance of quantum algorithms.

We optimize the algorithm on cross-resonance based hardware. The optimization is also applicable to other architectures, such as tunable couplers [297] that support interactions like iSWAP and CZ gates [298, 299]. The implementation of ZZ or ZZ-SWAP gates can be realized using three sets of gates: CZ gates, iSWAP gates, and a combination of CZ and iSWAP gates. By tailoring distinct gate decompositions to the properties of the two-qubit gates on the target hardware, such as fidelity and schedule duration, we can achieve improved performance. Furthermore, integrating selective calibration at the pulse-level makes this approach particularly promising for near-term quantum computing. Additionally, our study shows that decomposing ZZ-SWAP gates into CZ gates helps achieve improved results. Therefore, we expect that architectures with a native CZ gate may benefit even more from this optimization. Importantly, these techniques are not restricted to a particular algorithm and can be employed in various other quantum algorithms.

We applied readout error mitigation in Qiskit to mitigate measurement errors. However, there are additional techniques that can be utilized to suppress other types of errors in bipotent architectures. Here, we discuss two such techniques: dynamical decoupling [300] and zero-noise extrapolation [301]. Dynamical decoupling is a widely used strategy to suppress decoherence errors [302] by applying pulse sequences to idle qubits. This strategy will be detailed investigated in Chapter 6. In bipotent architectures, the presence of two distinct CX gate types with unique pulse schedules can lead to significant variations in pulse lengths across different qubit pairs within the algorithm. Applying dynamical decoupling pulses to the idle regions of shorter pulses can be particularly advantageous for such architectures. Moreover, diverse gate decompositions on different qubit pairs and selective calibration can also result in noticeable differences in pulse lengths and larger idle regions. This underscores the importance of incorporating dynamical decoupling into this optimization strategy. The ZNE technique estimates the noiseless result by utilizing expectation values measured at different noise levels. The impact of ZNE and the combined impact of DD and ZNE will be studied in Chapter 7. In bipotent architectures, two types of CX gates introduce different errors, which can be suppressed using ZNE. Additionally, ZNE can help identify the optimal gate decomposition for each qubit pair among various decomposition strategies used on different qubit pairs. Thorough analysis, experimentation, and adaptation of these techniques to the unique characteristics of bipotent architectures are crucial for achieving improved performance on such architectures.

Chapter 6

Synergistic Error Mitigation and Circuit Design

This chapter investigates the synergistic effects of dynamical decoupling (DD) and optimized circuit design for maximizing the performance and robustness of quantum algorithms on near-term devices. We leverage eight IBM quantum processors to explore the impact of device characteristics and algorithm design on the effectiveness of DD for error mitigation. This analysis takes into account hardware factors including circuit fidelity, scheduling duration, and the native gate set of quantum devices. Additionally, we examine the influence of algorithmic implementation details, including specific gates used, the chosen DD sequence, and the level of circuit optimization applied. This chapter has been published in the special issue on Quantum Computing in the NISQ Era of the journal *Entropy* under the title *Synergistic Dynamical Decoupling and Circuit Design for Enhanced Algorithm Performance on Near-Term Quantum Devices* [303].

6.1 Methodology

We experimentally investigate the impact of hardware and algorithmic factors on the effectiveness of dynamical decoupling in superconducting quantum processes. Hardware factors, such as circuit fidelity, schedule duration, and native gate sets, define the fundamental capabilities of a quantum device. Conversely, algorithmic factors encompass the design choices made during algorithm implementation (specific sequence of quantum operations), error suppression strategy (selection of a DD sequence), and circuit optimization techniques. These algorithmic choices ultimately determine how efficiently the inherent capabilities of the hardware are utilized for optimal performance. Understanding these factors can potentially improve the practical implementation of algorithms on near-term quantum devices.

6.1.1 Benchmark circuits and metrics

We first present benchmark circuits used in our demonstration and metrics employed to assess algorithm performance and DD effectiveness.

Published in Yanjun Ji and Ilia Polian, Synergistic dynamical decoupling and circuit design for enhanced algorithm performance on near-term quantum devices, *Entropy*, 26(7), 2024.

QAOA for portfolio optimization

We employ QAOAs for portfolio optimization as benchmarks, which can be found in Section 3.2.1. Our study uses a QAOA with qubit numbers ranging from 3 to 12 and a depth of 1. To establish a baseline for experimental results, we present the simulation results conducted in a noise-free environment with Qiskit's Qasm simulator in Table 6.1. All data use 30,000 circuit repetitions (shots). We observe that the approximation ratio and success probability decrease as the number of qubits increases.

Metric definition

To quantify the impact of noise, we introduce two normalized metrics: the normalized approximation ratio (NAR) and the normalized success probability (NSP). These metrics are defined as

$$\mathrm{NAR} = r_\epsilon / r_0, \tag{6.1}$$
$$\mathrm{NSP} = p_\epsilon / p_0, \tag{6.2}$$

where r_ϵ and p_ϵ represent the approximation ratio and success probability obtained under noise conditions. r_0 and p_0 denote corresponding values obtained from the simulated noise-free case (Table 6.1). For the same problem instances, we use identical values of r_0 and p_0 for evaluation. Due to noise, the NAR and NSP typically exhibit values less than one. However, in rare instances, these metrics may exceed unity, indicating that specific noise patterns or randomness enhance performance compared to the simulated noise-free case.

Without any error mitigation being applied, the NAR and NSP for algorithms executing on real noisy quantum devices are given by

$$\mathrm{NAR_B} = r_b / r_0, \tag{6.3}$$
$$\mathrm{NSP_B} = p_b / p_0, \tag{6.4}$$

where B represents results obtained from real hardware without error mitigation. r_b and p_b represent the corresponding approximation ratio and success probability, respectively. When a specific error mitigation strategy, such as the DD sequence, is applied, the NAR and NSP are given by

$$\mathrm{NAR_{DD}} = r_d / r_0, \tag{6.5}$$
$$\mathrm{NSP_{DD}} = p_d / p_0, \tag{6.6}$$

Table 6.1: Noise-free simulation results of the QAOA for portfolio optimization using Qiskit's Qasm simulator with 30,000 shots.

Number of Qubits	3	4	5	6	7	8	9	10	11	12
Approximation ratio	0.9751	0.4342	0.3776	0.3734	0.3589	0.3144	0.2806	0.3241	0.2933	0.3161
Success probability	0.9747	0.1536	0.1131	0.0422	0.0227	0.0124	0.0065	0.0057	0.0018	0.0006

where DD denotes the application of DD sequences. r_d and p_d represent the approximation ratio and success probability, respectively, with error mitigation. To assess DD effectiveness in error mitigation, we introduce two additional metrics: Δ_{NAR} and Δ_{NSP}. These metrics are defined as the difference between the corresponding values obtained with and without DD sequences:

$$\Delta_{\text{NAR}} = \text{NAR}_{\text{DD}} - \text{NAR}_{\text{B}}, \tag{6.7}$$

$$\Delta_{\text{NSP}} = \text{NSP}_{\text{DD}} - \text{NSP}_{\text{B}}. \tag{6.8}$$

A positive value for Δ_{NAR} and Δ_{NSP} indicates a successful improvement in performance resulting from the utilization of DD sequences.

We further introduce the concept of error mitigation success rate (EMSR) to quantify the robustness of an error mitigation strategy. EMSR is defined as the percentage of experimental trials where error mitigation improved the outcome compared to no mitigation. A high EMSR indicates consistent performance improvement while a low EMSR suggests limited effectiveness or potential performance degradation. We employ two EMSR metrics in this study, EMSR_{AR} and EMSR_{SP}, depending on whether the approximation ratio or success probability is used. Positive values of Δ_{NAR} and Δ_{NSP} contribute to increased EMSR_{AR} and EMSR_{SP}, respectively.

6.1.2 Hardware considerations

This section provides information about the IBM quantum devices used in our experiments. The QPUs with 27 qubits, namely ibmq_mumbai, ibmq_kolkata, ibm_cairo, and ibmq_ehningen, operate using basis gates {CX, ID, R_Z, SX, X}, where ID represents identity gate, R_Z performs a single qubit rotation around the z-axis, X is the NOT gate, and SX is the square root of X. On the other hand, the QPUs with 127 qubits, specifically ibm_kyoto, ibm_cusco, ibm_brisbane, and ibm_sherbrooke, utilize basis gates {ECR, ID, R_Z, SX, X}, where ECR is the echoed cross-resonance gate.

Figure 6.1(a) illustrates the schedule of a native CX gate on ibm_cairo. This implementation employs a single-pulse gate duration of 112 dt and a cross-resonance (CR) gate duration of 544 dt, leading to a total duration of 1312 dt, where dt represents the system cycle time. In contrast, Figure 6.1(b) depicts the schedule of a native ECR gate on ibm_cusco. Here, the single-pulse and CR gate durations are 88 dt and 416 dt, respectively, resulting in a shorter total duration of 920 dt. The CX gate is typically implemented using one ECR gate and multiple single-qubit gates. A further decomposition of the CX gate into ECR and single-pulse gates at pulse level enables the elimination of single-pulse gates during circuit optimization, as demonstrated in Chapter 5. Additionally, CX-based IBM QPUs, where CX is the native gate, support the operation of CX in two directions. On the other hand, ECR-based QPUs, where ECR is the native gate, typically only support the ECR gate in one direction. Consequently, quantum circuits designed for these latter devices need to be decomposed into sequences of gates that include only the supported direction of the ECR.

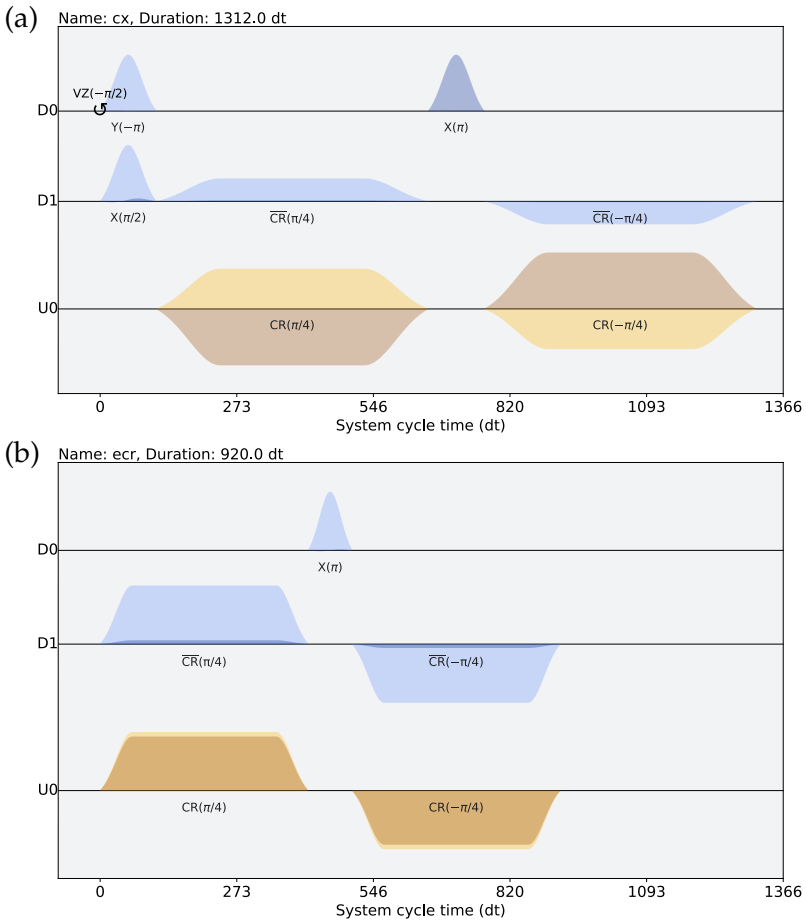

Figure 6.1: Schedule of hardware-native two-qubit gates: (a) CX gate on qubit pair (0,1) of ibm_cairo, and (b) echoed cross-resonance (ECR) gate on qubit pair (0,1) of ibm_cusco. The system cycle time (1dt) is $2/9$ ns ≈ 0.22 ns in ibm_cairo, while it is 0.50 ns in ibm_cusco. D_i represents the drive channel acting on qubit i, and U_j is the control channel for a corresponding qubit pair (c, t) driving the control qubit c at the frequency of the target qubit t.

6.1.3 Synergistic design approach

This section describes a synergistic design approach for maximizing the performance and robustness of algorithms on near-term quantum devices. This approach acknowledges the critical interplay between the hardware's capabilities and the design choices made in the software implementation. The quality of algorithm implementation directly affects the performance. Key aspects include the efficiency of transpilation processes, specific gate types used, and the overall symmetry of the algorithm structure. For instance, the algorithm-oriented qubit mapping (AOQMAP) method introduced in Section 4.2 offers advantages in transpilation for variational quantum algorithms (VQAs) [88] over popular compilers such as Qiskit [208] and Tket [210] by introducing fewer two-qubit gates, maintaining a shallower circuit depth, and promoting higher symmetry.

Here, we utilize the AOQMAP method proposed in Section 4.2 to efficiently map circuits onto hardware, aiming to minimize SWAP gates and circuit depth on linear

topologies. Subsequently, we examine two implementations of QAOAs for portfolio optimization on CX-based IBM QPUs. These implementations differ in their choice of gate decompositions within the algorithms. The first implementation, referred to as CX implementation, directly decomposes the gates in the QAOA into basis gates of the QPUs using Qiskit's transpiler with optimization level 3. In comparison, the second implementation, referred to as CZ implementation, initially decomposes gates in algorithms into CZ and single-qubit gates. Then, Qiskit's transpiler with optimization level 3 is used to perform optimization and decomposition into basis gates of QPUs. In Chapter 5, we have demonstrated that this CZ decomposition approach outperforms CX decomposition for ZZ and ZZ-SWAP gates in the QAOA on IBM QPUs. During implementation, we also explore different optimization level settings in Qiskit and investigate their impact. To ensure consistency in evaluations, we employ identical benchmark circuits and parameters. Additionally, we consistently use 30,000 shots for each demonstration. Within the Qiskit framework, we default to using optimization level 3.

We also investigate the effectiveness of two well-established DD sequences: CPMG and XY4. As illustrated in Figure 6.2, the CPMG sequence applies two X pulses separated by a delay of $\frac{t}{2}$, with additional delays of $\frac{t}{4}$ at the beginning and end. The parameter t represents the time interval during which the qubit remains idle, excluding the duration of single-qubit pulses and, in the case of CPMG, two X pulses. In comparison, the XY4 sequence utilizes two X and two Y pulses, each separated by a delay of $\frac{t}{4}$, with additional delays of $\frac{t}{8}$ at the beginning and end. Additionally, the "alap" (as late as possible) scheduling method, which schedules the stop time of instructions as late as possible, is used for scheduling gates and inserting DD sequences throughout our study. Figure 6.3 showcases the resulting implementations of a three-qubit QAOA with a CPMG sequence on a 27-qubit QPU ibm_kolkata using both CX and CZ implementations. Compared to CX implementation, CZ implementation employs all the same directed CX gates. Additionally, we observe an X gate inserted between control qubits of CX gates in the CZ implementation, potentially suppressing idle errors and improving performance. Moreover, CZ implementation shows improved symmetry compared to CX implementation. By simultaneously optimizing both hardware and software aspects through careful algorithm design, efficient transpilation techniques, and DD sequences, it is possible to fully exploit the capabilities of near-term quantum algorithms.

In the following we delve into the examination of multiple factors that influence algorithm performance and DD effectiveness. These factors are classified into two main categories: hardware and algorithmic. More specifically, we thoroughly analyze the

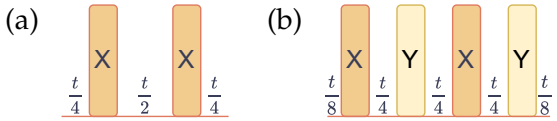

Figure 6.2: Two types of dynamical decoupling (DD) sequences: (a) Carr–Purcell–Meiboom–Gill (CPMG) and (b) XY4. The delay time t represents the idle time of the qubit minus the duration of the corresponding X or Y pulses.

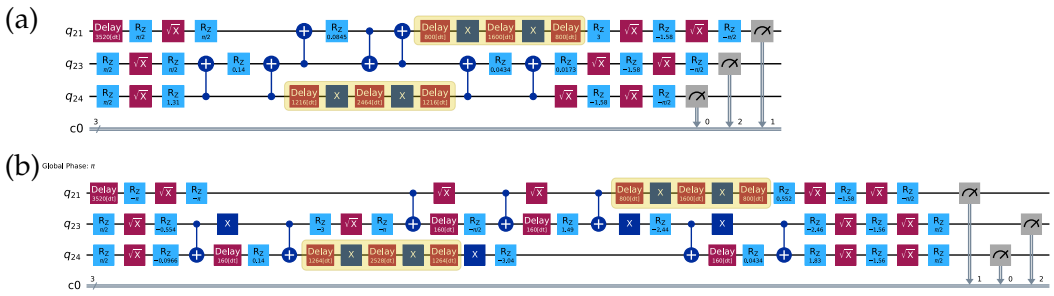

Figure 6.3: Two implementations of a three-qubit QAOA: (a) CX implementation and (b) CZ implementation, where both are decomposed and optimized using Qiskit's transpiler with optimization level 3. Highlighted yellow boxes represent CPMG sequences.

influence of circuit fidelity, schedule duration, and native gate set on the performance of DD sequences. Furthermore, we explore the impact of algorithm implementation, choice of DD sequence, and level of optimization on algorithm performance and DD effectiveness. The objective of these analyses is to provide invaluable insights into the design and optimization of algorithm implementation for achieving efficient execution on quantum devices.

6.2 Impact of hardware factors

Our investigation begins by examining how hardware characteristics affect algorithm performance and DD effectiveness. The CPMG sequence is chosen for our study due to its widespread adoption and relative simplicity, allowing us to gain fundamental insights. We analyze these factors using extensive datasets. The experiments involve varying the qubit counts from 3 to 12 and investigating different combinations of algorithm implementations, including the CX and CZ versions of the QAOA, as well as optimization levels 1 and 3 within the Qiskit framework. We conduct these experiments using eight QPUs, which consist of four 27-qubit devices—ibmq_mumbai, ibmq_kolkata, ibm_cairo, and ibmq_ehningen—and four 127-qubit devices: ibm_kyoto, ibm_cusco, ibm_brisbane, and ibm_sherbrooke. These extensive datasets provide a solid foundation for analyzing the impact of hardware factors on algorithm performance.

6.2.1 Circuit fidelity

We first explore the impact of circuit fidelity on the performance. The fidelity of a circuit qc, denoted as $\mathcal{F}_{qc}$, measures the agreement between the actual operation of a quantum circuit and its ideal operation. The circuit fidelity can be mathematically represented as

$$\mathcal{F}_{qc} = \prod_{G_s \in qc} f_{G_s} \prod_{G_t \in qc} f_{G_t} \prod_{G_m \in qc} f_{G_m}, \tag{6.9}$$

where f_{G_s}, f_{G_t}, and f_{G_m} denote fidelities of a single-qubit gate G_s, a two-qubit gate G_t, and the measurement G_m, respectively, in the circuit. We focus on circuits with fidelities between 0.01 and 1. This broader range of fidelities establishes a solid basis

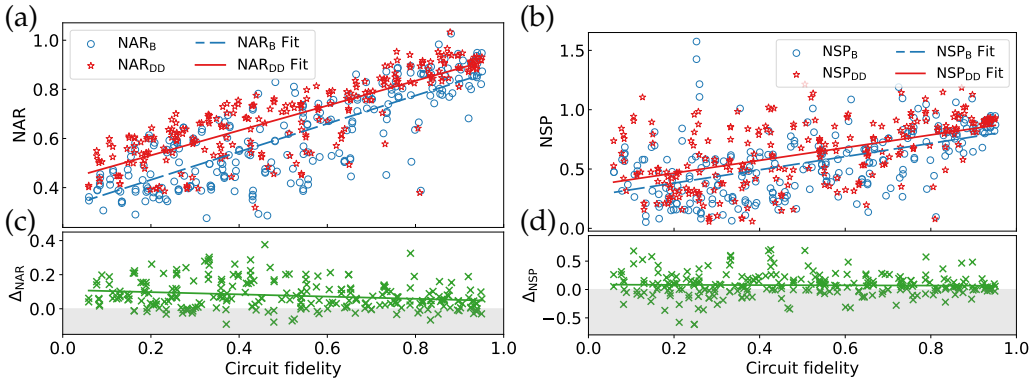

Figure 6.4: Impact of circuit fidelity on algorithm performance and DD effectiveness: (a) normalized approximation ratio (NAR), (b) normalized success probability (NSP), (c) improvement in NAR after applying DD (Δ_{NAR}), and (d) improvement in NSP after applying DD (Δ_{NSP}). Higher values of NAR_B, NAR_{DD}, NSP_B, and NSP_{DD} indicate better performance on actual quantum devices. Values exceeding unity indicate that the performance achieved on the IBM quantum hardware surpasses the results obtained from the noise-free simulation. Positive values of Δ_{NAR} and Δ_{NSP} demonstrate improvements due to DD. The CPMG sequence is used for all data points. Each line in the graph represents a linear fit of the data. The reported $EMSR_{AR}$ and $EMSR_{SP}$ are 85.55% and 66.8%, respectively.

for investigating the effectiveness of DD sequences under realistic noise conditions. We analyze various metrics defined in Section 6.1.1, including NAR_B, NAR_{DD}, NSP_B, NSP_{DD}, Δ_{NAR}, and Δ_{NSP}. The measured data and the corresponding circuit fidelity are fitted using a linear function. The correlation coefficient C_r and p-value are computed to assess the quality of linear approximation. C_r measures the strength and direction of the linear relationship, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 indicating no association. The absolute value of C_r reflects the correlation strength: very strong (0.9–1.0), strong (0.7–0.9), moderate (0.4–0.7), weak (0.2–0.4), and very weak (0–0.2). It is important to note that correlation does not imply causation. The p-value is a complementary statistical measure that evaluates the strength of evidence against the null hypothesis of no correlation. A low p-value (typically below 0.05) suggests a statistically significant correlation, possibly not due to random chance. Linear fitting builds on correlation by determining the best-fit line equation, enabling predictions based on the observed relationship.

Figure 6.4 depicts a general trend of improved algorithm performance with increasing circuit fidelity. DD sequences enhance both the NAR and NSP on average. However, the NSP exhibits a wider range of variation for a given circuit fidelity com-

Table 6.2: Parameters derived from the analysis of Figure 6.4.

Metric	Mean	Fit Function	Correlation Coefficient	p-Value
NAR_B	0.610	$y = 0.572x + 0.317$	0.809	0
NAR_{DD}	0.687	$y = 0.506x + 0.430$	0.827	0
NSP_B	0.556	$y = 0.558x + 0.270$	0.530	0
NSP_{DD}	0.631	$y = 0.537x + 0.358$	0.521	0
Δ_{NAR}	0.077	$y = -0.065x + 0.111$	-0.209	0.00075
Δ_{NSP}	0.075	$y = -0.021x + 0.086$	-0.027	0.66997

pared to the approximation ratio, particularly at lower fidelities. Table 6.2 presents the average value, fitted function, correlation coefficient, and p-value for each metric. A stronger correlation is observed between NAR and circuit fidelity compared to NSP, suggesting a more pronounced dependence of NAR on fidelity. In contrast, the correlation between Δ_{NSP} and circuit fidelity is very weak, as evidenced by the low value of C_r and high p-value. This suggests that the observed decrease in DD effectiveness might be due to random fluctuations or other factors not captured by circuit fidelity alone. The average improvement in NAR and NSP due to DD sequences is approximately 0.08. Moreover, the negative coefficients of the fitted lines for Δ_{NAR} and Δ_{NSP} suggest a potential decrease in DD effectiveness as circuit fidelity increases.

6.2.2 Schedule duration

We now investigate the influence of schedule duration τ and, in particular, the logarithmic transformation of schedule duration $\ln(\tau/\mathrm{dt})$ on algorithm performance and DD effectiveness. Schedule duration reflects the total time required to execute a quantum circuit and depends on the number and execution time of individual gates. Shorter durations potentially improve circuit fidelity by reducing the system's exposure to decoherence errors, but achieving them necessitates faster gates, which can be hardware-limited.

Figure 6.5 depicts the impact of $\ln(\tau/\mathrm{dt})$ on the defined metrics using the same datasets as in Figure 6.4. We observe that the algorithm performance degrades with increasing schedule duration, while DD effectiveness improves. However, Δ_{NSP} exhibits larger fluctuations for longer durations, suggesting that while DD sequences mitigate decoherence errors, potentially improving performance at longer durations, they could also introduce other error mechanisms, such as operation errors, that counteract this improvement. Table 6.3 summarizes corresponding parameters. The coefficients of the linear function for $\mathrm{NAR_B}$, $\mathrm{NAR_{DD}}$, $\mathrm{NSP_B}$, and $\mathrm{NSP_{DD}}$ indicate a suppressed decay in performance with increasing schedule duration by applying DD sequences. The correlation coefficients between these metrics and $\ln(\tau/\mathrm{dt})$ further support the effectiveness of DD sequences in reducing the dependence of performance (both NAR and NSP) on schedule duration. Δ_{NAR} exhibits a statistically

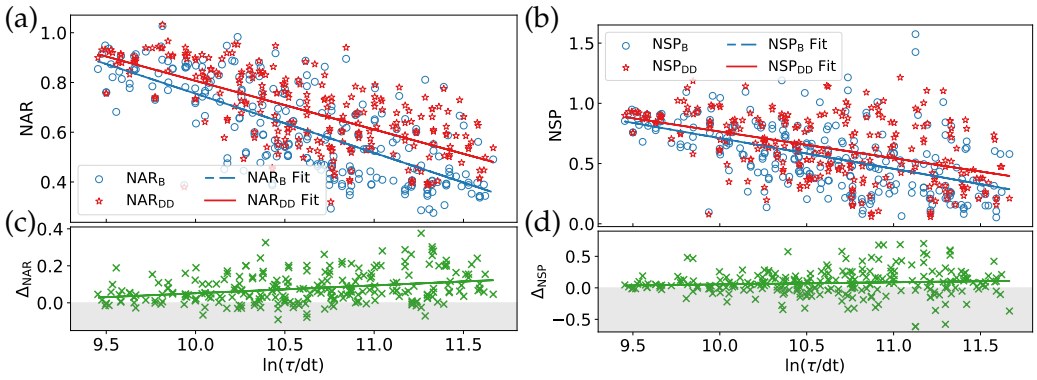

Figure 6.5: Influence of logarithmic transformation of circuit schedule duration ($\ln(\tau/\mathrm{dt})$) on algorithm performance and DD effectiveness with the same datasets as in Figure 6.4: (a) NAR, (b) NSP, (c) Δ_{NAR}, and (d) Δ_{NSP}.

Table 6.3: Parameters derived from the analysis of Figure 6.5.

Metric	Fit Function	Correlation Coefficient	p-Value
NAR_B	$y = -0.238x + 3.136$	-0.731	0
NAR_{DD}	$y = -0.196x + 2.764$	-0.691	0
NSP_B	$y = -0.254x + 3.253$	-0.524	0
NSP_{DD}	$y = -0.221x + 2.975$	-0.464	0
Δ_{NAR}	$y = 0.042x - 0.372$	0.295	0
Δ_{NSP}	$y = 0.033x - 0.278$	0.092	0.14281

weak correlation with $\ln(\tau/\mathrm{dt})$, while Δ_{NSP} shows a very weak correlation. This observation aligns with the findings for circuit fidelity. However, Δ_{NSP} appears more sensitive to schedule duration compared to circuit fidelity, as indicated by a larger absolute correlation coefficient ($|C_r|$) and lower p-value.

Figures 6.6(a–d) illustrate the impact of circuit fidelity and schedule duration on NAR_{DD}, NSP_{DD}, Δ_{NAR}, and Δ_{NSP}, respectively. As observed in Figure 6.6(a), NAR_{DD} exhibits degradation with increasing logarithmic schedule duration ($\ln(\tau/\mathrm{dt})$) and decreasing circuit fidelity. High performance is concentrated in the region where the schedule duration τ is below $e^{10.5}$dt and circuit fidelity surpasses 0.5. Conversely, low performance is primarily observed for τ exceeding $e^{10.5}$dt and fidelities below 0.5. A similar trend is evident for NSP_{DD} in Figure 6.6(b). However, unlike NAR_{DD}, achieving a high NSP_{DD} value remains feasible even for longer schedule durations and lower fidelities. This suggests that NSP_{DD} is less sensitive to these factors compared to NAR_{DD}. Figures 6.6(c–d) further demonstrate the effectiveness of DD sequences, particularly at longer durations. This is potentially due to the ability of DD sequences to mitigate decoherence errors that become more prominent at these timescales.

6.2.3 Native gate sets

This section explores the performance of quantum algorithms implemented on two distinct sets of IBM QPUs. The first set comprises four 27-qubit devices utilizing the CX gate as their native two-qubit gate, while the second set consists of four 127-qubit devices employing the ECR gate as their native two-qubit gate.

Benchmark results obtained from the two QPU sets are presented in Figure 6.7. Table 6.4 summarizes average values of metrics and parameters derived from the linear fits of these metrics against the number of qubits. As shown in Figures 6.7(a) and 6.7(b), applying DD sequences generally improves the NAR and NSP, respectively, for both native gate sets. The highest performance is achieved with NAR_{DD}^{ECR} and NSP_{DD}^{ECR}, which leverages ECR-based QPUs and incorporates DD sequences. Moreover, QPUs utilizing the ECR gate exhibit consistently higher baseline performance, NAR_B^{ECR} and NSP_B^{ECR}, compared to those with the CX gate, NAR_B^{CX} and NSP_B^{CX}, even surpassing those with DD sequences, NAR_{DD}^{CX}, and NSP_{DD}^{CX}. This observation suggests that the selection of QPUs may be more critical for achieving optimal performance than relying solely on DD techniques. As shown in Figures 6.7(c) and 6.7(d), DD effectiveness improves as the qubit count increases for both gate sets. Moreover, the CX gate set exhibits higher DD effectiveness than the ECR gate set. As illustrated in Figures 6.7(e) and 6.7(f), the ECR gate produces an overall higher circuit fidelity

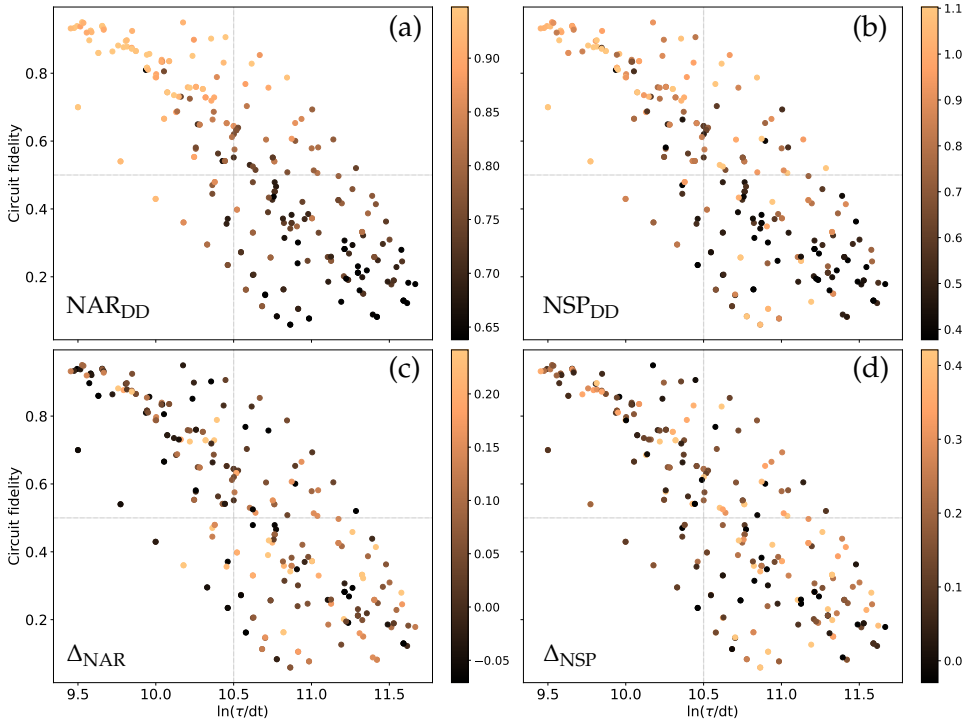

Figure 6.6: Influence of circuit fidelity and $\ln(\tau/\mathrm{dt})$ on algorithm performance and DD effectiveness: (a) $\mathrm{NAR}_{\mathrm{DD}}$, (b) $\mathrm{NSP}_{\mathrm{DD}}$, (c) Δ_{NAR}, and (d) Δ_{NSP}.

and lower schedule duration, potentially contributing to higher performance. Analyzing the proportion of positive outcomes in the experimental data presented in Figures 6.7(c) and 6.7(d), we observe that the reported values of $\mathrm{EMSR}_{\mathrm{AR}}$ and $\mathrm{EMSR}_{\mathrm{SP}}$ are 92.5% and 72.5% for the CX gate, respectively, whereas the corresponding values are 75% and 62.5%, respectively, for the ECR gate, suggesting that DD sequences are more robust in mitigating errors for the CX gate set.

The correlation coefficient presented in Table 6.4 reveals a very strong negative correlation between $\mathrm{NAR}_{\mathrm{DD}}^{\mathrm{ECR}}$ and the number of qubits compared to $\mathrm{NAR}_{\mathrm{B}}^{\mathrm{ECR}}$. Similarly, a very strong negative correlation is observed between circuit fidelity and qubit count for both CX and ECR gate sets, indicating a significant decrease in circuit fidelity as the number of qubits increases. Moreover, we observe a weaker correlation between NSP and qubit number compared to NAR, implying that the impact of qubit count on success probability is less pronounced than its effect on approximation ratio. Additionally, the high p-values for $\Delta_{\mathrm{NAR}}^{\mathrm{ECR}}$ and $\Delta_{\mathrm{NSP}}^{\mathrm{ECR}}$ suggest that the effectiveness of DD sequences for ECR gates is more susceptible to random fluctuations. This can be attributed to the interplay between the intended decoherence suppression capabilities of DD sequences and the additional gate errors that they introduce. As shown in Figures 6.7(e) and 6.7(f), circuits utilizing ECR gates demonstrate higher fidelities while requiring shorter execution times. The inherent advantage of shorter circuits may limit the potential for further enhancement through the use of DD sequences. In such cases, incorporating extra DD pulses could even lead to a decrease in overall algorithm performance.

Our demonstrations on eight IBM QPUs highlight the importance of circuit fidelity, schedule duration, and DD sequences in optimizing algorithm performance.

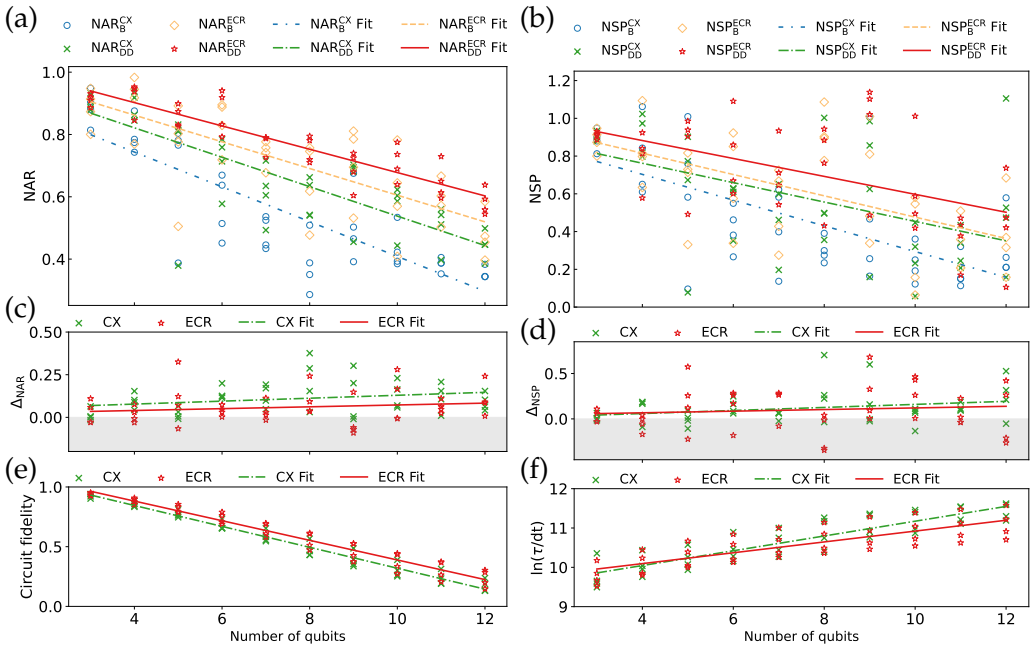

Figure 6.7: Comparison of two hardware-native gate sets: {CX, ID, R_Z, SX, X} and {ECR, ID, R_Z, SX, X}, denoted as CX and ECR gate sets, respectively. Results obtained using four 27-qubit quantum processing units (QPUs) ibmq_mumbai, ibmq_kolkata, ibm_cairo, and ibmq_ehningen for the CX gate set, and four 127-qubit QPUs ibm_kyoto, ibm_cusco, ibm_brisbane, and ibm_sherbrooke for the ECR gate set: (a) NAR, (b) NSP, (c) Δ_{NAR}, (d) Δ_{NSP}, (e) circuit fidelity, and (f) $\ln(\tau/\mathrm{dt})$. The CPMG sequence is used for all data points. Each line represents a linear fit to the corresponding data.

As circuit fidelity decreases and schedule duration increases, DD sequences become increasingly important for mitigating errors and identifying optimal solutions. Furthermore, the results suggest that ECR-based QPUs offer advantages over CX-based QPUs. This is primarily due to the inherently shorter schedule durations and higher circuit fidelities associated with ECR gates. However, CX-based QPUs benefit more significantly from DD sequences in terms of error mitigation.

6.3 Impact of algorithmic factors

6.3.1 Algorithm implementations

We compare the performance of CX and CZ implementations of the QAOA on four 27-qubit IBM QPUs, as detailed in Section 6.1.3. The CPMG sequence is consistently utilized throughout this analysis. As shown in Figures 6.8(a) and 6.8(b), CZ implementation with DD sequences achieves the highest average values for both the NAR ($\mathrm{NAR}^{\mathrm{CZ}}_{\mathrm{DD}}$) and NSP ($\mathrm{NSP}^{\mathrm{CZ}}_{\mathrm{DD}}$). However, for the NSP at qubit counts exceeding 9, CZ implementation without DD sequences, $\mathrm{NSP}^{\mathrm{CZ}}_{\mathrm{B}}$, outperforms that with DD sequences, $\mathrm{NSP}^{\mathrm{CZ}}_{\mathrm{DD}}$, potentially due to the introduction of significant errors by DD sequences themselves. Furthermore, CX implementation exhibits increasing DD effectiveness as the qubit number grows (Figures 6.8(c) and 6.8(d)), whereas Δ_{NSP} for CZ implementation exhibits a slight decrease. Additionally, $\mathrm{EMSR}_{\mathrm{AR}}$ and $\mathrm{EMSR}_{\mathrm{SP}}$ are

Table 6.4: Parameters derived from the analysis of Figure 6.7.

Metric	Mean	Fit Function	Correlation Coefficient	p-Value
$\mathrm{NAR}_{\mathrm{B}}^{\mathrm{CX}}$	0.548	$y = -0.056x + 0.967$	-0.832	0
$\mathrm{NAR}_{\mathrm{DD}}^{\mathrm{CX}}$	0.656	$y = -0.047x + 1.010$	-0.812	0
$\mathrm{NAR}_{\mathrm{B}}^{\mathrm{ECR}}$	0.712	$y = -0.043x + 1.033$	-0.766	0
$\mathrm{NAR}_{\mathrm{DD}}^{\mathrm{ECR}}$	0.771	$y = -0.037x + 1.052$	-0.911	0
$\mathrm{NSP}_{\mathrm{B}}^{\mathrm{CX}}$	0.464	$y = -0.068x + 0.976$	-0.678	0
$\mathrm{NSP}_{\mathrm{DD}}^{\mathrm{CX}}$	0.582	$y = -0.051x + 0.967$	-0.504	0.00092
$\mathrm{NSP}_{\mathrm{B}}^{\mathrm{ECR}}$	0.618	$y = -0.056x + 1.041$	-0.579	0.00009
$\mathrm{NSP}_{\mathrm{DD}}^{\mathrm{ECR}}$	0.716	$y = -0.047x + 1.072$	-0.529	0.00044
$\Delta_{\mathrm{NAR}}^{\mathrm{CX}}$	0.108	$y = 0.009x + 0.043$	0.27	0.09252
$\Delta_{\mathrm{NAR}}^{\mathrm{ECR}}$	0.059	$y = 0.005x + 0.019$	0.167	0.30429
$\Delta_{\mathrm{NSP}}^{\mathrm{CX}}$	0.118	$y = 0.017x - 0.008$	0.277	0.08407
$\Delta_{\mathrm{NSP}}^{\mathrm{ECR}}$	0.098	$y = 0.009x + 0.031$	0.108	0.50688
Circuit fidelity (CX)	0.539	$y = -0.088x + 1.195$	-0.984	0
Circuit fidelity (ECR)	0.594	$y = -0.082x + 1.211$	-0.975	0
$\ln(\tau/\mathrm{dt})$ (CX)	10.703	$y = 0.188x + 9.294$	0.891	0
$\ln(\tau/\mathrm{dt})$ (ECR)	10.577	$y = 0.138x + 9.543$	0.791	0

consistently higher for CX implementation (92.5% and 80%, respectively) compared to CZ implementation (75% and 57.5%, respectively), suggesting a higher robustness of DD sequences for CX implementation. As depicted in Figures 6.8(e) and 6.8(b), CX and CZ implementations exhibit comparable circuit fidelities. For a larger number of qubits, CX implementation even achieves slightly higher fidelities. Moreover, CX implementation demonstrates a consistently shorter schedule duration compared to CZ implementation (Figure 6.8(f)). This difference in schedule duration is attributed to the increased number of single-qubit gates required by CZ implementation (details in Figure 6.3).

Table 6.5 summarizes the average value of each metric along with the linear fit parameters extracted from Figure 6.8. The negative coefficient associated with $\Delta_{\mathrm{NSP}}^{\mathrm{CZ}}$ suggests a decrease in DD effectiveness, as measured by success probability, with an increasing qubit count for CZ implementation. This behavior can be attributed to the intricate interplay between the optimization landscape and gate errors. As the number of qubits involved increases, the cumulative error introduced by DD sequences becomes more pronounced. This significantly alters the optimization landscape, rendering the pre-optimized parameters of the QAOA in the noiseless case no longer suitable. Furthermore, correlation coefficients between the NAR ($\mathrm{NAR}_{\mathrm{B}}$ and $\mathrm{NAR}_{\mathrm{DD}}$) and qubit count reveal that applying DD sequences weakens the correlation for CX implementation while strengthening it for CZ implementation. This trend is also observed for $\mathrm{NSP}_{\mathrm{B}}$ and $\mathrm{NSP}_{\mathrm{DD}}$. An additional observation is the high p-values associated with $\Delta_{\mathrm{NAR}}^{\mathrm{CZ}}$ and $\Delta_{\mathrm{NSP}}^{\mathrm{CZ}}$, indicating the potential dominance of random fluctuations in these metrics for CZ implementation.

The results suggest that although CX implementation offers more advantages in terms of DD effectiveness, the higher performance of CZ implementation highlights the significance of circuit structure in executing the QAOA. In certain instances, the inherent advantage of a more symmetrical circuit structure, as exhibited by CZ implementation, can outweigh the benefits of strong DD mitigation achieved with CX implementation. However, the optimal selection of gate decomposition ultimately relies on the specific algorithm being implemented, the hardware capabilities

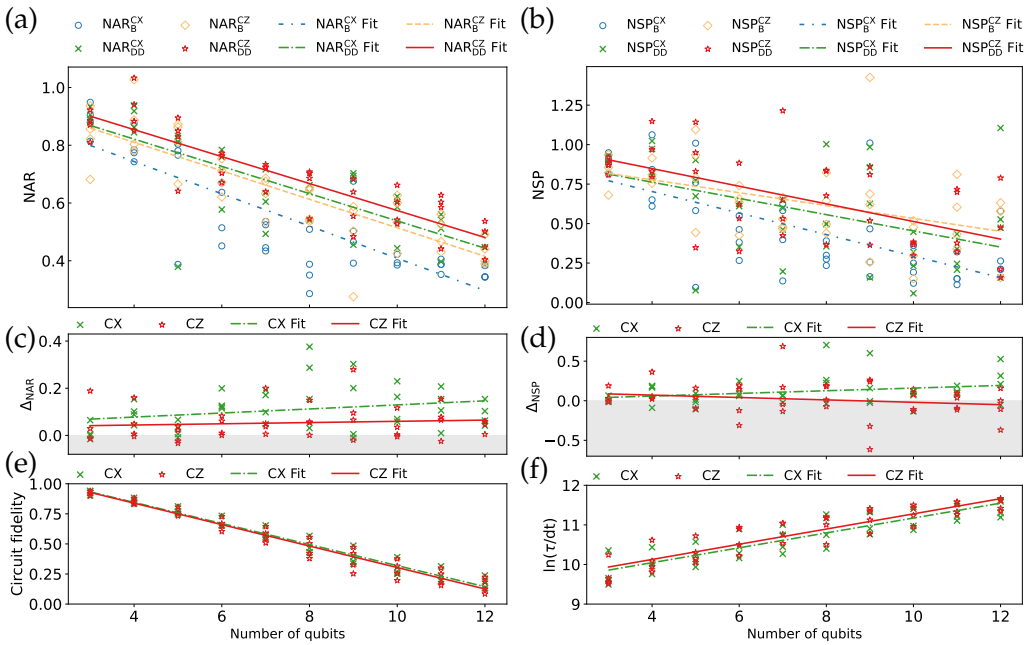

Figure 6.8: Comparison of CX and CZ implementations of QAOA across four 27-qubit IBM QPUs ibmq_mumbai, ibmq_kolkata, ibm_cairo, and ibmq_ehningen: (a) NAR, (b) NSP, (c) Δ_{NAR}, (d) Δ_{NSP}, (e) circuit fidelity, and (f) $\ln(\tau/\mathrm{dt})$. The CPMG sequence is used for all data points. Each line represents a linear fit of the data.

available, and the desired balance between overhead caused by DD and potential performance gains.

6.3.2 DD sequences

This section evaluates the performance of two DD sequences, CPMG and XY4, for mitigating decoherence errors during QAOA execution with CX implementation. The evaluation leverages data from seven IBM QPUs. XY4, which employs four single-qubit pulses, might be more effective for qubits with extended idle times than CPMG, which utilizes two single-qubit pulses. While DD sequences may introduce crosstalk errors, the use of the same transpiled circuits minimizes the impact of this potential crosstalk on our evaluation.

Figures 6.9(a) and 6.9(b) show that both CPMG and XY4 contribute to improved algorithm performance. While CPMG and XY4 achieve comparable NAR values, XY4 exhibits a better NSP for a larger number of qubits, but with noticeable fluctuations. CPMG and XY4 demonstrate comparable DD effectiveness for a small number of qubits (Figures 6.9(c) and 6.9(d)), with XY4 showing a slight advantage for larger qubit counts. However, CPMG exhibits greater robustness, as evidenced by its higher $\mathrm{EMSR}_{\mathrm{AR}}$ (84.29%) and $\mathrm{EMSR}_{\mathrm{SP}}$ (75.71%) compared to XY4's values (67.14% and 64.29%, respectively). Figures 6.9(e) and 6.9(f) illustrate comparable circuit fidelity and schedule duration for both CPMG and XY4. As before, the data are fitted with a linear function. The resulting parameters are summarized in Table 6.6. The coefficients of the linear function suggest that both CPMG and XY4 effectively suppress the decrease in NAR and NSP as the qubit count increases. The correlation

Table 6.5: Parameters derived from the analysis of Figure 6.8.

Metric	Mean	Fit Function	Correlation Coefficient	p-Value
$\mathrm{NAR}_{\mathrm{B}}^{\mathrm{CX}}$	0.548	$y = -0.056x + 0.967$	-0.832	0
$\mathrm{NAR}_{\mathrm{DD}}^{\mathrm{CX}}$	0.656	$y = -0.047x + 1.010$	-0.812	0
$\mathrm{NAR}_{\mathrm{B}}^{\mathrm{CZ}}$	0.638	$y = -0.049x + 1.007$	-0.850	0
$\mathrm{NAR}_{\mathrm{DD}}^{\mathrm{CZ}}$	0.691	$y = -0.047x + 1.041$	-0.885	0
$\mathrm{NSP}_{\mathrm{B}}^{\mathrm{CX}}$	0.464	$y = -0.068x + 0.976$	-0.678	0
$\mathrm{NSP}_{\mathrm{DD}}^{\mathrm{CX}}$	0.582	$y = -0.051x + 0.967$	-0.504	0.00092
$\mathrm{NSP}_{\mathrm{B}}^{\mathrm{CZ}}$	0.635	$y = -0.041x + 0.940$	-0.473	0.00208
$\mathrm{NSP}_{\mathrm{DD}}^{\mathrm{CZ}}$	0.653	$y = -0.056x + 1.071$	-0.589	0.00006
$\Delta_{\mathrm{NAR}}^{\mathrm{CX}}$	0.108	$y = 0.009x + 0.043$	0.27	0.09252
$\Delta_{\mathrm{NAR}}^{\mathrm{CZ}}$	0.053	$y = 0.003x + 0.034$	0.109	0.50206
$\Delta_{\mathrm{NSP}}^{\mathrm{CX}}$	0.118	$y = 0.017x - 0.008$	0.277	0.08407
$\Delta_{\mathrm{NSP}}^{\mathrm{CZ}}$	0.018	$y = -0.015x + 0.132$	-0.202	0.21130
Circuit fidelity (CX)	0.539	$y = -0.088x + 1.195$	-0.984	0
Circuit fidelity (CZ)	0.527	$y = -0.089x + 1.195$	-0.977	0
$\ln(\tau/\mathrm{dt})$ (CX)	10.703	$y = 0.188x + 9.294$	0.891	0
$\ln(\tau/\mathrm{dt})$ (CZ)	10.799	$y = 0.192x + 9.360$	0.901	0

coefficients and p-values suggest a stronger correlation between $\Delta_{\mathrm{NAR}}^{\mathrm{XY4}}$ and qubit count compared to $\Delta_{\mathrm{NAR}}^{\mathrm{CPMG}}$, $\Delta_{\mathrm{NSP}}^{\mathrm{XY4}}$, and $\Delta_{\mathrm{NSP}}^{\mathrm{CPMG}}$.

The results indicate that DD sequences are generally recommended for improving circuit performance. They allow for achieving acceptable results in a wider range of circuits. For instance, the QAOA with DD can reach higher NAR values for more qubits. However, the effectiveness and robustness of DD sequences can vary. While XY4 offers slightly better performance improvements in terms of NAR and NSP, CPMG demonstrates higher robustness as measured by EMSR. This highlights the importance of considering both performance gains and mitigation robustness when choosing a DD sequence.

6.3.3 Optimization levels

We investigate the influence of optimization levels on the performance and DD effectiveness using five IBM QPUs. Two optimization levels within Qiskit's transpiler are considered: level 1 (Opt1) and level 3 (Opt3), representing the default and highest settings, respectively. It is worth noting that different optimization levels do not affect the number of two-qubit gates in our demonstration. This is because we are considering benchmark circuits that have already undergone the AOQMAP approach, which effectively ensures adherence to connectivity constraints and eliminates the need for additional SWAP gates. In our case, different optimization levels influence the selected qubits for circuit execution and the count of single-qubit gates, which affects the circuit fidelity and schedule duration. The CPMG sequence and CX implementation of the QAOA are employed throughout this analysis.

Figures 6.10(a) and 6.10(b) demonstrate that Opt3 with DD sequences achieves the highest overall performance in terms of NAR and NSP, followed by Opt1 with DD sequences. Without error mitigation, Opt3 outperforms Opt1 for all tested qubit counts in terms of NAR, whereas, for NSP, Opt3 exhibits an advantage only for a small number of qubits, with comparable performance achieved at larger qubit

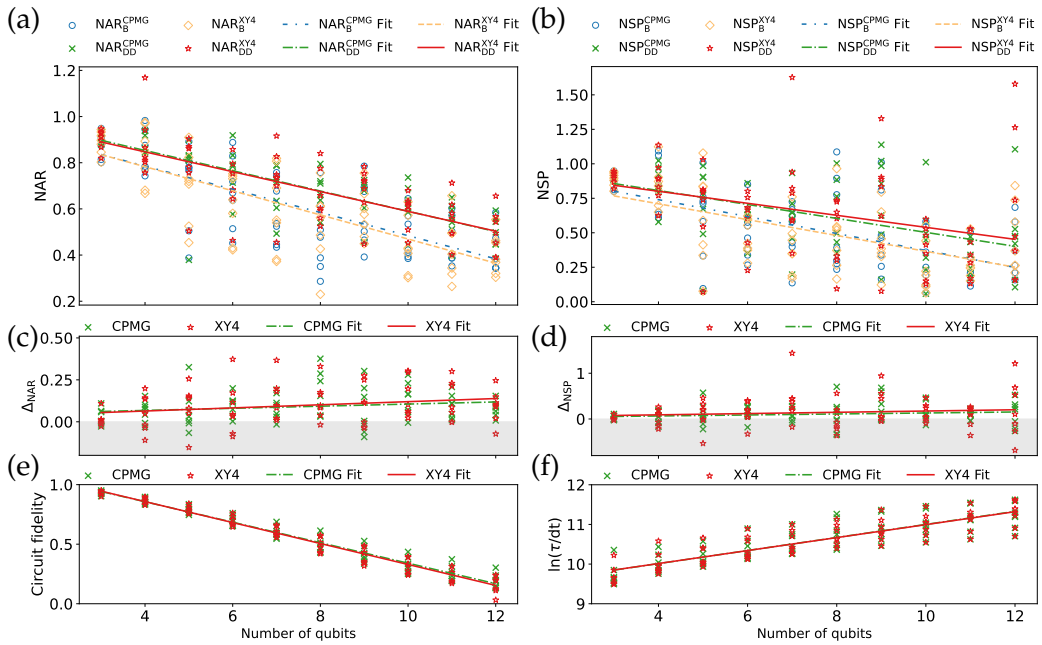

Figure 6.9: Comparison of CPMG and XY4 sequences across seven IBM QPUs ibmq_mumbai, ibmq_kolkata, ibm_cairo, ibmq_ehningen, ibm_kyoto, ibm_cusco, and ibm_brisbane: (a) NAR, (b) NSP, (c) Δ_{NAR}, (d) Δ_{NSP}, (e) circuit fidelity, and (f) $\ln(\tau/\mathrm{dt})$. Each line represents a linear fit of the data.

counts. The average improvement in NAR due to DD (Δ_{NAR}) is generally higher for Opt1 compared to Opt3 (Figure 6.10(c)). However, for NSP, DD initially benefits Opt1 more, but this advantage shifts toward Opt3 for larger qubit counts (Figure 6.10(d)). Furthermore, the reported average $\mathrm{EMSR}_{\mathrm{AR}}$ and $\mathrm{EMSR}_{\mathrm{SP}}$ are 92% and 74% for Opt1, respectively, compared to 80% and 60% for Opt3, suggesting a higher robustness of DD for Opt1. It is important to note that Opt3 exhibits higher circuit fidelity (Figure 6.10(e)) and a shorter schedule duration (Figure 6.10(f)), which are crucial for high algorithm performance.

The detailed linear fit parameters for the optimization levels are provided in Table 6.7. While DD effectively mitigates the decrease in NAR for both Opt1 and Opt3 with increasing qubits, its impact on NSP differs. For Opt3, DD suppresses the decrease in NSP, whereas, for Opt1, it appears to exacerbate the decay. The correlation between NSP and qubit count for Opt1 is very weak (low C_r and high p-value), suggesting minimal influence from qubit count on NSP for this optimization level. In contrast, Opt1 prioritizes schedule duration, exhibiting a stronger correlation with qubit count, while Opt3 prioritizes circuit fidelity, showing a stronger correlation between fidelity and qubit count.

This analysis demonstrates a trade-off between the optimization level and DD effectiveness. While Opt3 offers superior overall performance with DD sequences, Opt1 exhibits higher DD effectiveness. Furthermore, DD sequences become increasingly beneficial for Opt3 at larger qubit counts for finding optimal solutions.

Table 6.6: Parameters derived from the analysis of Figure 6.9.

Metric	Mean	Fit Function	Correlation Coefficient	p-Value
$\mathrm{NAR}_{\mathrm{B}}^{\mathrm{CPMG}}$	0.610	$y = -0.050x + 0.986$	-0.749	0
$\mathrm{NAR}_{\mathrm{DD}}^{\mathrm{CPMG}}$	0.699	$y = -0.044x + 1.029$	-0.802	0
$\mathrm{NAR}_{\mathrm{B}}^{\mathrm{XY4}}$	0.6	$y = -0.052x + 0.992$	-0.752	0
$\mathrm{NAR}_{\mathrm{DD}}^{\mathrm{XY4}}$	0.697	$y = -0.043x + 1.019$	-0.750	0
$\mathrm{NSP}_{\mathrm{B}}^{\mathrm{CPMG}}$	0.525	$y = -0.061x + 0.986$	-0.613	0
$\mathrm{NSP}_{\mathrm{DD}}^{\mathrm{CPMG}}$	0.629	$y = -0.051x + 1.009$	-0.513	0.00001
$\mathrm{NSP}_{\mathrm{B}}^{\mathrm{XY4}}$	0.509	$y = -0.057x + 0.940$	-0.591	0
$\mathrm{NSP}_{\mathrm{DD}}^{\mathrm{XY4}}$	0.648	$y = -0.044x + 0.975$	-0.373	0.00148
$\Delta_{\mathrm{NAR}}^{\mathrm{CPMG}}$	0.09	$y = 0.006x + 0.042$	0.189	0.11748
$\Delta_{\mathrm{NAR}}^{\mathrm{XY4}}$	0.097	$y = 0.009x + 0.027$	0.23	0.05502
$\Delta_{\mathrm{NSP}}^{\mathrm{CPMG}}$	0.104	$y = 0.011x + 0.023$	0.144	0.23425
$\Delta_{\mathrm{NSP}}^{\mathrm{XY4}}$	0.139	$y = 0.014x + 0.035$	0.116	0.33818
Circuit fidelity (CPMG)	0.555	$y = -0.086x + 1.201$	-0.979	0
Circuit fidelity (XY4)	0.55	$y = -0.088x + 1.209$	-0.981	0
$\ln(\tau/\mathrm{dt})$ (CPMG)	10.589	$y = 0.165x + 9.354$	0.851	0
$\ln(\tau/\mathrm{dt})$ (XY4)	10.586	$y = 0.163x + 9.360$	0.858	0

6.4 Discussion

Our comprehensive study, conducted on eight IBM quantum devices, reveals that the application of DD sequences can significantly enhance the performance and robustness of algorithms on near-term quantum devices. However, the effectiveness of DD sequences varies depending on hardware and algorithmic factors. A key finding is the observed inverse relationship between DD effectiveness and the original performance of algorithms without error mitigation. This implies that algorithms with higher inherent performance (measured without DD sequences) exhibit lower DD effectiveness. For instance, ECR-based QPUs offer superior native performance but reduced DD effectiveness compared to CX-based QPUs. Similarly, the CZ implementation of a QAOA exhibits higher native algorithm performance but lower average DD effectiveness compared to CX implementation. Moreover, optimization level 3 produces higher algorithm performance, but level 1 exhibits higher DD effectiveness. This inverse behavior can be attributed to the fact that algorithms with high performance typically have a lower intrinsic error rate, including the decoherence errors targeted by DD sequences. Furthermore, the introduction of DD pulse sequences can lead to new gate operation errors that decrease algorithm performance and potentially limit their effectiveness for certain algorithms.

DD sequences are typically more effective for algorithms with lower fidelity and a longer schedule duration, but their impact on approximation ratio and success probability differs. The results indicate that while algorithms with lower circuit fidelity struggle to achieve high approximation ratio values, the application of DD sequences allows for achieving the simulated success probability. This finding suggests a potential methodology for obtaining the desired probabilities by studying different algorithm implementations and DD sequences and selecting the measure that minimizes system energy.

Table 6.8 summarizes the observed effects of investigated hardware and algorithm factors on the effectiveness and robustness of DD. Without error mitigation, the ECR

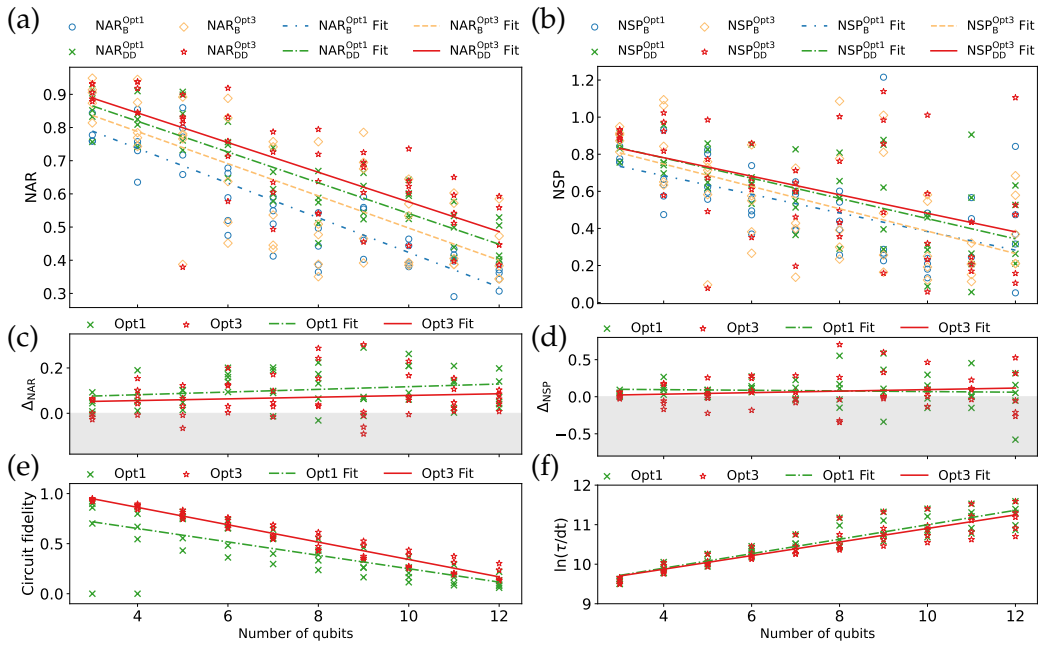

Figure 6.10: Comparison of optimization level 1 (Opt1) and optimization level 3 (Opt3) across five IBM QPUs ibmq_kolkata, ibm_cairo, ibmq_ehningen, ibm_cusco, and ibm_kyoto: (a) NAR, (b) NSP, (c) Δ_{NAR}, (d) Δ_{NSP}, (e) circuit fidelity, and (f) $\ln(\tau/\mathrm{dt})$. The CPMG sequence is used for all data points. Each line represents a linear fit of the data.

native gate set achieves the highest average value of normalized approximation ratio ($\mathrm{NAR_B}$), while the CZ implementation of the QAOA exhibits the highest normalized success probability ($\mathrm{NSP_B}$). However, when applying DD error mitigation, the CX native gate set and CX implementation of the QAOA demonstrate the greatest increase in NAR, while the XY4 sequence leads to the most significant NSP improvement. Additionally, the linear fit slope coefficients suggest that circuit fidelity has a stronger influence on approximation ratio, whereas schedule duration more significantly impacts success probability. Furthermore, as the qubit count increases, the CX gate set, CX implementation, and XY4 sequence benefit more from DD mitigation compared to the ECR gate set, CZ implementation, and CPMG sequence. Notably, the CX implementation exhibits the highest overall robustness for the DD strategy, followed by the CX gate set and circuit optimization level 1 (Opt1).

One significant reason for the higher performance of algorithms on ECR-based QPUs is the inherently higher circuit fidelity and shorter schedule duration. However, another potential factor could be the inherent advantages of ECR gates. In these devices, ECR gates are only allowed for one direction, meaning that any two-qubit gate is directly decomposed into ECR gates with the same direction on the qubit pair. While the hardware-native CX gate also has a native direction, the reverse direction is supported, requiring additional single-qubit gates and the hardware-native CX gate for implementation. Directly decomposing algorithms into one-directional ECR gates could be more advantageous than using CX gates with bidirectional capability at the gate level and then transforming them into native CX gates at the hardware pulse level. A similar trend is observed for CZ implementation, which produces a higher

Table 6.7: Parameters derived from the analysis of Figure 6.10.

Metric	Mean	Fit Function	Correlation Coefficient	p-Value
NAR_B^{Opt1}	0.554	$y = -0.052x + 0.946$	-0.891	0
NAR_{DD}^{Opt1}	0.657	$y = -0.046x + 1.004$	-0.886	0
NAR_B^{Opt3}	0.618	$y = -0.048 + 0.982$	-0.720	0
NAR_{DD}^{Opt3}	0.688	$y = -0.045x + 1.023$	-0.770	0
NSP_B^{Opt1}	0.509	$y = -0.050x + 0.885$	-0.596	0
NSP_{DD}^{Opt1}	0.589	$y = -0.055x + 0.998$	-0.667	0
NSP_B^{Opt3}	0.536	$y = -0.060x + 0.988$	-0.586	0.00001
NSP_{DD}^{Opt3}	0.606	$y = -0.050x + 0.981$	-0.471	0.00056
Δ_{NAR}^{Opt1}	0.103	$y = 0.006x + 0.058$	0.218	0.12785
Δ_{NAR}^{Opt3}	0.069	$y = 0.004x + 0.041$	0.126	0.38513
Δ_{NSP}^{Opt1}	0.08	$y = -0.004x + 0.113$	-0.065	0.65396
Δ_{NSP}^{Opt3}	0.07	$y = 0.010x - 0.007$	0.136	0.34466
Circuit fidelity (Opt1)	0.417	$y = -0.067x + 0.919$	-0.729	0
Circuit fidelity (Opt3)	0.559	$y = -0.087x + 1.209$	-0.973	0
$\ln(\tau/dt)$ (Opt1)	10.540	$y = 0.183x + 9.168$	0.919	0
$\ln(\tau/dt)$ (Opt3)	10.475	$y = 0.172x + 9.186$	0.901	0

Table 6.8: Impact of hardware and algorithm factors on DD effectiveness and robustness.

Factor	Mean				Slope Coefficient		EMSR	
	NAR_B	Δ_{NAR}	NSP_B	Δ_{NSP}	Δ_{NAR}	Δ_{NSP}	$EMSR_{AR}$	$EMSR_{SP}$
Circuit fidelity Schedule duration	0.610	0.077	0.556	0.075	-0.065 0.042	-0.021 0.033	85.55%	66.80%
CX gate set	0.548	0.108	0.464	0.118	0.009	0.017	92.50%	72.50%
ECR gate set	0.712	0.059	0.618	0.098	0.005	0.009	75.00%	62.50%
CX implementation	0.548	0.108	0.464	0.118	0.009	0.017	92.50%	80.00%
CZ implementation	0.638	0.053	0.635	0.018	0.003	-0.015	75.00%	57.50%
CPMG sequence	0.610	0.090	0.525	0.104	0.006	0.011	84.29%	75.71%
XY4 sequence	0.600	0.097	0.509	0.139	0.009	0.014	67.14%	64.29%
Opt1	0.554	0.103	0.509	0.080	0.006	-0.004	92.00%	74.00%
Opt3	0.618	0.069	0.536	0.070	0.004	0.010	80.00%	60.00%

algorithm performance than CX implementation. The CX gate is directed, whereas the CZ gate is undirected, allowing for the decomposition of all two-qubit gates with one directed CZ gate on one or more qubit pairs. Utilizing gates in the same direction could lead to a higher symmetry in the circuit, both in terms of single- and two-qubit gates, potentially contributing to error suppression and improved algorithmic performance. This highlights the importance of considering native gate properties at the pulse level and maintaining circuit structure symmetry during algorithm design. Our findings hold broad applicability across various quantum algorithms, including the VQE [92, 253]. Prioritizing native gates and maintaining circuit symmetry during VQE and other quantum algorithm executions can enhance performance and mitigate errors across diverse quantum computing platforms.

Chapter 7

Algorithm and Hardware Cooptimization: Case Study of DC-QAOA

This chapter explores the cooptimization strategy in the QOA implementation process in order to develop a noise-resistant execution direct from the algorithm design stage. We demonstrate both algorithmic and hardware level improvements. The digitized counterdiabatic QAOA (DC-QAOA) is taken as an example for our demonstration, but the methodology proposed here can be extended to other variational quantum algorithms. We also investigate the performance of various error mitigation strategies. This chapter has been published in *Physical Review A* under the title *Improving the performance of digitized counterdiabatic quantum optimization via algorithm-oriented qubit mapping* [304].

7.1 Introduction to DC-QAOA

The NISQ devices encounter significant hurdles in realizing practical QOAs. A primary constraint lies in the limited qubit connectivity and inherent noise susceptibility of these devices, hindering their ability to fully exploit the potential of optimization algorithms [17]. Additionally, the transformation from analog to digital quantum computing introduces approximations and errors. This process relies on Trotterization, or the Suzuki-Trotter decomposition [305, 306, 307], a cornerstone technique for approximating complex unitary time evolution operators. Trotterization bridges the gap between intricate quantum system dynamics and their digital simulation by decomposing them into sequences of elementary quantum logic gates. This iterative approach allows the implementation of non-native unitary operations using available hardware gate sets, making it invaluable for studying systems with many-body effects, strong correlations, and rich collective phenomena. The potential of efficient digital quantum simulation extends across quantum chemistry, condensed matter physics, and materials science, promising profound new insights.

To efficiently execute algorithms on NISQ devices, it is crucial to incorporate hardware constraints into the algorithm design process. This necessitates optimizing the algorithm's performance while adhering to the specific architecture and limitations

Published in Yanjun Ji, Kathrin F. Koenig, and Ilia Polian, PhysRevA.110.032421, 2024.

of the underlying quantum computer. Here, we propose a novel methodology to enhance algorithm performance by cooptimizing gate sequences and algorithmic parameters while explicitly considering the limited connectivity constraints of current NISQ devices. By considering target device connectivities, our approach improves performance at the algorithmic level through optimized gate sequences and algorithm parameters while minimizing errors and noise introduced during the compilation process. This leads to improved performance, accuracy, and reliability for solving real-world problems. We demonstrate their effectiveness through three scenarios: MaxCut on complete graphs, MaxCut on non-complete graphs, and portfolio optimization.

To illustrate these strategies, we employ the MaxCut problem on complete graphs as our problem instance. The MaxCut problem is a fundamental optimization problem in computer science that involves partitioning the nodes of a graph into two disjoint sets in a way that maximizes the number of edges crossing between the two sets. Due to its computational complexity, MaxCut is classified as NP-hard. The Hamiltonian for the MaxCut problem H_c is given in Eq. 3.14. We use the standard mixer Hamiltonian H_m (Eq. 3.11) [19]. The counterdiabatic (CD) driving Hamiltonian H_{cd} is derived from adiabatic gauge potentials [308, 309, 310]. In this work, we investigate two forms of the CD driving term: $\sum_{i,j} Z_i Y_j$ and $\frac{1}{2}\sum_{i,j}(Z_i Y_j + Y_i Z_j)$. The evolution of DC-QAOA with depth p is given by

$$\left|\psi_f\right\rangle = U_{\mathrm{DC}}(\gamma_p, \beta_p, \alpha_p) \ldots U_{\mathrm{DC}}(\gamma_1, \beta_1, \alpha_1) \left|\psi_0\right\rangle, \tag{7.1}$$

where $|\psi_0\rangle$ denotes the ground state of H_m and $U_{\mathrm{DC}}(\gamma, \beta, \alpha) = U_{cd}(\gamma) U_m(\beta) U_c(\alpha)$ with $U_c(\alpha) = e^{-i\alpha H_c}$, $U_m(\beta) = e^{-i\beta H_m}$, and $U_{cd}(\gamma) = e^{-i\gamma H_{cd}}$.

The objective of DC-QAOA is to identify a set of angles that minimizes the expectation value of the problem Hamiltonian $\left\langle\psi_f\right| H_c \left|\psi_f\right\rangle$ using classical optimization algorithms, such as gradient-based optimizers [76] or gradient-free optimizers like the constrained optimization by linear approximation (COBYLA) [85]. These optimizers iteratively adjust the angles to approximate the ground state of H_c, thereby providing an approximate solution to the MaxCut problem. The approximation ratio of QAOA or DC-QAOA for MaxCut is defined as

$$r = E / E_0, \tag{7.2}$$

where E denotes the expectation value of the problem Hamiltonian obtained through QAOA or DC-QAOA, and E_0 represents the ground state energy of the system.

7.2 Methodology

7.2.1 Incorporating hardware constraints

In the context of DC-QAOA, the Suzuki-Trotter decomposition implies that the gate sequence for a particular Trotter step remains constant. To satisfy the decomposition requirements, we employ a fixed gate sequence for a specified depth in all instances of DC-QAOA leveraging AOQMAP as proposed in Section 4.2. We explore four variations of DC-QAOA: DC-QAOA with a ZY CD driving term both with and

without the problem Hamiltonian; and DC-QAOA with a ZY-YZ CD driving term both with and without the problem Hamiltonian.

The routing solutions for QAOA and DC-QAOA with the AOQMAP approach on a linear topology are illustrated in Figure 7.1. Figure 7.1(a) depicts the routing solution for fully connected ZZ gates in an n-qubit QAOA circuit. Each layer of ZZ gates is followed by a SWAP layer, except for the first and last layers, where the SWAP gate can be removed by adjusting the initial qubit order and the measurement order. Note that ZZ and SWAP gates are both undirected and commute, making the ZZ-SWAP gate equivalent to the SWAP-ZZ gate. For QAOA with partially connected ZZ gates, the solution is obtained by optimizing the initial qubit order such that all existing ZZ gates are arranged as forwards as possible, and the end-located SWAP gates can be removed by adjusting the measurement order. An example of QAOA for MaxCut on noncomplete graphs is presented in Figure 7.1(b). In comparison to Figure 7.1(a), where an arbitrary initial qubit order enables the implementation of all ZZ gates, Figure 7.1(b) requires a specific initial qubit order to minimize the additional CX gate count. However, this deterministic initial qubit order is not unique, allowing for exploring different ZZ gate sequences that may result in distinct efficiencies of optimization performance, which we discuss later.

To construct the solution for DC-QAOA, we leverage the mirror symmetry of swap layers, which enables the generation of two distinct gate sequences, denoted as AOQ-FS and AOQ-SF, as illustrated in Figures 7.1(c) and 7.1(d), respectively. These two gate sequences are equivalent for all-to-all commuting gates such as ZZ gates but exhibit differences for non-all-to-all commuting gates such as ZY gates. Figure 7.1(e) presents general solutions for applying AOQMAP to DC-QAOA. The two-qubit gates in problem and/or CD driving Hamiltonians are represented by U_1 and U_2. For instance, in the case of DC-QAOA with a ZY CD driving term including problem Hamiltonian, U_1 and U_2 correspond to ZZ and ZY gates with respective parameters. Figures 7.1(f–i) depict the decomposition of ZZ, SWAP, ZZ-SWAP, and ZY-SWAP gates, respectively. In particular, Figure 7.1(h) highlights that a SWAP gate directly following a ZZ gate introduces one additional CX gate due to CX gate cancellation, as depicted in the enclosed box. To optimize the SWAP gate following a ZY gate, this SWAP gate is repositioned before the final single qubit gate within the ZY gate, with the position of the corresponding single qubit gate adjusted accordingly, as illustrated in Figure 7.1(i). This optimization reduces CX gates involved.

The use of mirror symmetry in AOQMAP presents a distinctive characteristic, where the qubit order reverts to its initial state every two two-qubit Hamiltonian blocks. Specifically, when a depth-one circuit comprises an odd number of two-qubit Hamiltonian blocks, the final qubit order returns to its initial state until depth two. Although the Suzuki-Trotter decomposition mandates a fixed gate sequence for each Trotter step, variations in gate sequences within a single Trotter step can result in distinct Trotter errors. However, for a Hamiltonian composed solely of ZZ gates, gate sequences have no impact on Trotter error since ZZ gates commute with each other.

In the case of DC-QAOA with a ZY CD driving term but lacking problem Hamiltonian, maintaining a fixed gate sequence for each depth can be challenging. One potential solution is to employ the mirror symmetry of swap layers after each depth, reversing the final qubit mapping back to the initial one. However, this approach introduces a significant increase in noise. An alternative strategy is to utilize these

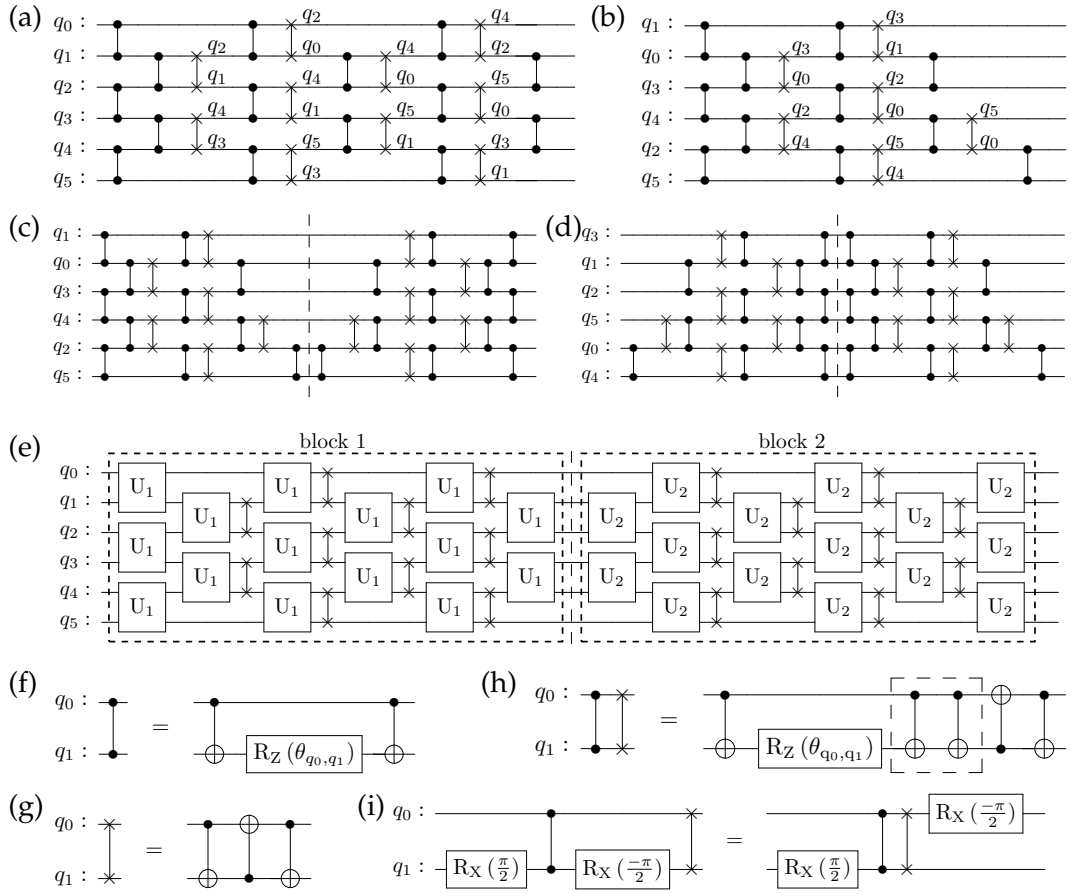

Figure 7.1: AOQMAP approach for two-qubit Hamiltonian in QAOA and DC-QAOA on a linear topology. (a) Routing solution for all-to-all connected ZZ gates in six-qubit QAOA at $p = 1$. While an arbitrary initial qubit order allows for the execution of all two-qubit gates, it results in different gate sequences. (b) Routing solution for partially connected ZZ gates in six-qubit QAOA at $p = 1$. The initial qubit order is optimized to minimize the total increase in the number of CX gates due to the insertion of SWAP gates. The mirror symmetry of swap layers is used to construct a solution for the next QAOA depth. Two constructions of mirror symmetry with varying initial mappings are displayed in (c) and (d); their corresponding gate sequences are referred to as AOQ-FS and AOQ-SF, respectively. (e) Solution for DC-QAOA leveraging mirror symmetry. The two-qubit gates in DC-QAOA are represented by U_1 and U_2 in blocks 1 and 2, respectively. Decomposition of (f) ZZ, (g) SWAP, and (h) ZZ-SWAP. Each SWAP gate adjacent to a ZZ gate introduces only one additional CX gate due to CX gate cancellation. (i) Optimization of ZY-SWAP gate. A ZY gate followed by a SWAP gate can be optimized by operating the SWAP gate directly behind the ZZ gate and altering the position of $R_x(\frac{-ß}{2})$ gate accordingly.

swap layers to construct a solution that incorporates per layer reversed ZY gates at depth two. The utilization of reversed gate sequences has the potential to mitigate Trotter error [237, 238, 311], making AOQMAP a promising approach. In the case where problem Hamiltonian is included, ZZ and ZY terms at depth one reverse the final qubit mapping directly to the initial one, resulting in a fixed gate sequence for each depth.

For DC-QAOA with a ZY-YZ CD driving term but excluding problem Hamiltonian, ZY and YZ terms maintain a fixed gate sequence at each depth. However, when problem Hamiltonian is included, DC-QAOA involves three terms: ZZ, ZY, and YZ. Alternating between swap layers and their mirror symmetry ensures a consistent gate sequence every two depths.

In summary, by exploiting swap layers and their symmetry, the AOQMAP strategy ensures an efficient construction of DC-QAOA circuits adhering to the noncommutative nature of CD driving terms. This approach results in a fixed gate sequence that depends on the initial qubit order. While the choice of gate sequence has no impact on the performance when gates within each block commute, the effectiveness of the parameter optimization process may vary when not all gates commute, which we discuss in the following section.

7.2.2 Performance improvement strategies

This section explores optimization strategies to enhance the performance of DC-QAOA with AOQMAP. The integration of SWAP gates during the qubit mapping process generates a deterministic gate sequence. In contrast, conventional DC-QAOA implementation does not impose a specific gate sequence and follows a sequential order where qubit 0 is successively connected to remaining qubits, followed by the sequential connection of qubits 1, 2, and so on, until reaching the final qubit n. We refer to this gate sequence as the ORIG sequence. Although the performance of DC-QAOA generally remains robust to variations in Hamiltonian and gate sequence within a certain range of Trotter error, these factors can influence the algorithm's optimization efficiency, presenting opportunities for further performance enhancements.

Iterations and initial guess values

The gradient descent optimization method is used to minimize the expectation value of the problem Hamiltonian, corresponding to the objective function. This study investigates the influence of two critical factors during the optimization process: the maximum number of iterations and the number of random initial guess values.

Figure 7.2 compares different settings for the maximum iteration and the number of random initial guess values in the optimization process. The first setting, denoted as 100(*10), uses 100 iterations and 10 random initial guess values. Each data point is repeated 10 times, and the error bar represents one standard error of the mean (SEM). The second setting, denoted as 100(*10)_1000, maintains 100 iterations and 10 random guess values and increases the number of iterations for the optimized initial guess value to 1000, improving both ORIG and AOQ-FS gate sequences. In addition, AOQ-FS shows significantly high approximation ratios at lower depths. However, optimizing DC-QAOA becomes more difficult at larger depths, resulting in lower

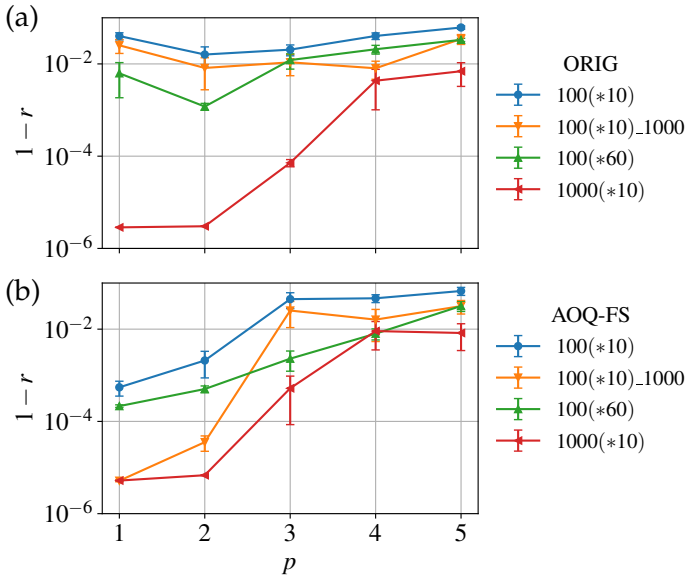

Figure 7.2: Comparison of different settings of the maximum number of iterations and the number of random initial guess values in the optimization process for the six-qubit DC-QAOA with a ZY-ZZ-X Hamiltonian sequence. Deviation $1-r$ of approximation ratio for two different gate sequences: (a) ORIG and (b) AOQ-FS. A lower value of $1-r$ indicates a higher approximation ratio, signifying improved performance. All error bars represent one standard error of the mean (SEM) across 10 independent repetitions.

approximation ratios. Increasing the number of random initial guess values to 60, represented as 100(*60), leads to improved performance compared to 100(*10). The best performance is observed when setting the maximum number of iterations to 1000 and the number of initial guess values to 10, yielding the highest approximation ratio for both ORIG and AOQ-FS gate sequences.

The results demonstrate that despite sharing the same order of Trotter errors, different gate sequences significantly influence optimization efficiency. The AOQ-FS gate sequence generally outperforms the ORIG gate sequence for these settings. To minimize the impact of the optimization on algorithm performance, we use the setting 1000(*10) during the optimization process.

Hamiltonian and gate sequences

We investigate the impact of varying Hamiltonian and gate sequences on the performance of DC-QAOA with the ZY CD driving term. Firstly, we examine the effects of applying Hamiltonians in different orders when the problem Hamiltonian is absent. This leads to two possible sequences: ZY-X and X-ZY. Subsequently, we incorporate the problem Hamiltonian, resulting in six distinct Hamiltonian sequences: ZY-ZZ-X, ZY-X-ZZ, X-ZY-ZZ, ZZ-ZY-X, ZZ-X-ZY, and X-ZZ-ZY.

In addition to examining Hamiltonian sequences, we investigate two gate sequences: ORIG and AOQ. For both the first and second two-qubit Hamiltonians, the ORIG sequence employs the original gate sequence, while AOQ utilizes the sequence depicted in block 1 of Figure 7.1(e). Within AOQ gate sequence, two specific configurations exist: AOQ-FS and AOQ-SF, illustrated in Figures7.1(c) and 7.1(d), respectively.

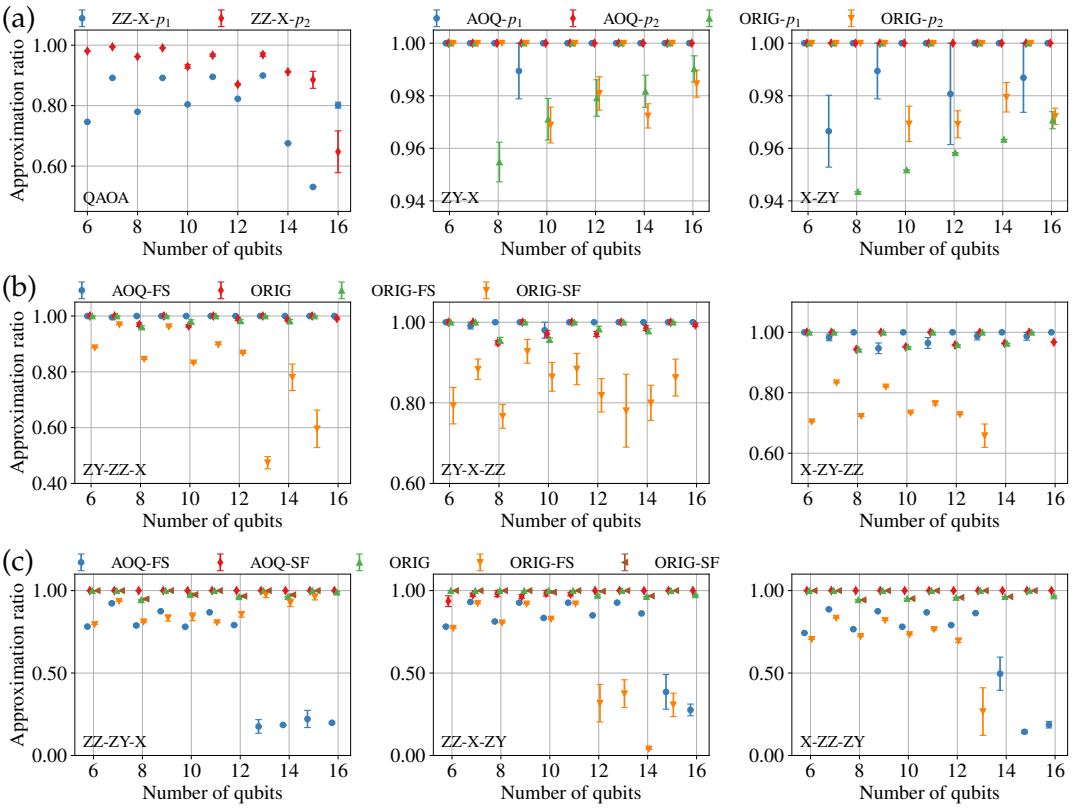

Figure 7.3: Approximation ratio of QAOA and DC-QAOA with a ZY CD driving term for MaxCut on complete graphs. The Hamiltonian sequences differ depending on the order in which they operate on the initial state. For instance, ZY-X indicates that DC-QAOA, excluding problem Hamiltonian, applies the CD driving term first, followed by the mixer Hamiltonian. (a) QAOA with ZZ-X and DC-QAOA with ZY-X and X-ZY at $p = 1$ (p_1) and $p = 2$ (p_2). (b) DC-QAOA with ZY before ZZ term at $p = 1$. (c) DC-QAOA with the ZZ before ZY term at $p = 1$. AOQ refers to the gate sequence introduced by the AOQMAP approach, while ORIG denotes the original gate sequence. For ZY-X and X-ZY, AOQ-p_2 employs the same gate sequence in each depth. All error bars represent one SEM across 10 independent repetitions.

Similarly, within ORIG gate sequence, two configurations are considered: ORIG-FS and ORIG-SF. In ORIG-FS, two-qubit gates in the first two-qubit Hamiltonian term are assigned using ORIG sequence, while the second term employs the reverse order of ORIG sequence. Conversely, in ORIG-SF, the assignment is reversed: the first term utilizes reverse order, while the second term employs ORIG gate sequence. Despite both AOQ-SF and ORIG-SF involving gate sequence reversal, they differ in their approach, with AOQ-SF employing per-layer reversal and ORIG-SF utilizing per-gate reversal.

Figure 7.3 presents the approximation ratio of standard QAOA and DC-QAOA with ZY CD driving term for MaxCut on complete graphs. The number of qubits considered ranges from 6 to 16. For QAOA and the DC-QAOA without problem Hamiltonian, depths are limited to two, while for DC-QAOA including problem Hamiltonian, the depth is set to one. As shown in Figure 7.3(a), approximation ratio of QAOA fluctuates with the number of qubits, exhibiting better performance

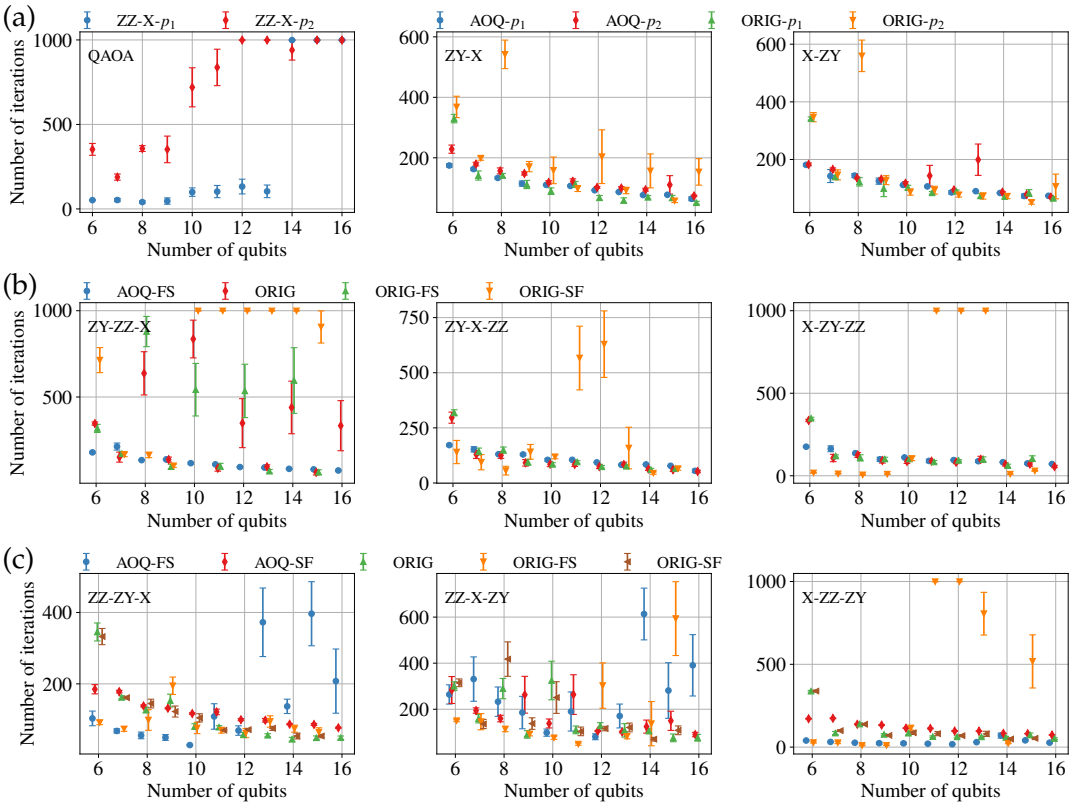

Figure 7.4: Number of iterations in the optimization process corresponding to Figure 7.3.

at depth two. The ORIG and AOQ gate sequences in QAOA do not influence the performance since each ZZ gate commutes with the remaining ZZ gates.

For DC-QAOA without problem Hamiltonian, we compare ZY-X and X-ZY with different gate sequences. With ZY applied first (ZY-X), AOQ achieves an overall more stable and improved approximation ratio compared to ORIG. Similarly, AOQ-p_2 delivers the highest and most stable approximation ratio for X-ZY. In AOQ-p_2, ZY gates are constructed by repeating the gates from AOQ-p_1 for both ZY-X and X-ZY sequences. Implementing this gate sequence requires maintaining swap layers in block 2 of Figure 7.1(e) to reverse to the initial qubit order, which is impractical due to noise. However, we can utilize these swap layers in block 2 to construct solutions at depth two, enabling a symmetric implementation of ZY gates, which we discuss later.

Figures 7.3(b) and 7.3(c) depict the approximation ratio of DC-QAOA when ZY CD driving term is positioned before or after problem Hamiltonian, respectively. It is observed that sequences ZY-ZZ-X with AOQ-FS, ZZ-ZY-X with AOQ-SF, and X-ZZ-ZY with AOQ-SF exhibit better performance. In terms of gate sequence within ZY term, following the first block of Figure 7.1(e) yields better results than following the second one. Specifically, when ZY precedes ZZ term (Figure 7.3(b)), ORIG-FS gate sequence outperforms ORIG-SF. Similarly, when ZZ term precedes ZY term (Figure 7.3(c)), AOQ-SF, ORIG, and ORIG-SF show better performance than AOQ-FS and ORIG-FS. Moreover, AOQ-SF exhibits the most stable and highest performance, while ORIG-SF performs comparably to ORIG.

Figure 7.4 presents the number of iterations required for the optimization process corresponding to results in Figure 7.3. As depicted in Figure 7.4(a), the number of iterations for QAOA increases rapidly with growing number of qubits, eventually reaching the maximum iteration setting for larger sizes. This trend, coupled with a decrease in approximation ratio, indicates inefficient scaling for QAOA. In contrast, for ZY-X and X-ZY sequences, the number of iterations decreases as qubit number increases, demonstrating better scalability. Furthermore, the majority of iterations remain below 200, providing an advantage over QAOA in terms of computational efficiency.

In DC-QAOA with ZY term applied first (Figure 7.4(b)), ZY-ZZ-X with AOQ-FS demonstrates a decreasing number of iterations as qubit count increases, while maintaining a consistently high approximation ratio. Although sequences ZY-X-ZZ and X-ZY-ZZ with AOQ-FS, ORIG, and ORIG-FS exhibit low and stable numbers of iterations, their approximation ratio fluctuates across different qubit counts. This highlights the benefit of utilizing an end located mixer in achieving stable performance across all problem instances. On the other hand, ORIG-SF exhibits a fluctuating iteration, aligning with variations in approximation ratio. In sequences where ZZ term is applied before ZY term (Figure 7.4(c)), ZZ-ZY-X generally yields better results compared to ZZ-X-ZY and X-ZZ-ZY, again demonstrating the advantage of employing an end located mixer Hamiltonian. Moreover, ZZ-ZY-X and X-ZZ-ZY, both with AOQ-SF, demonstrate decreasing numbers of iterations as qubit number increases, while maintaining high and stable approximation ratios. Although X-ZZ-ZY with AOQ-FS exhibits low and stable numbers of iterations, the corresponding approximation ratio varies and remains low.

The results highlight the substantial impact of Hamiltonian and gate sequences on algorithm performance. In particular, gate sequence of ZY CD driving term plays a crucial role in achieving high and stable performance in quantum algorithms. Applying the problem Hamiltonian before CD driving term generally leads to higher approximation ratios. In addition, gate sequences introduced by AOQMAP demonstrate an average improved performance compared to original gate sequences.

Combined and separated implementations

In the implementation of DC-QAOA with ZY CD driving term including problem Hamiltonian, we can employ a combined gate implementation (see, e.g., [310]), where a ZZ gate is followed by a ZY gate, or vice versa. This approach referred to as combined implementation offers the advantage of eliminating the need for the second block of Hamiltonian during the mapping process. Consequently, only block 1 in Figure 7.1(e) is required along with U_1 gate, which corresponds to the combination of ZZ and ZY gates. In the following, we evaluate the performance of the combined implementation of ZZ and ZY gates against the separated implementation of ZZ and ZY terms.

Figures 7.5(a) and 7.5(b) illustrate approximation ratios of ZZ-ZY-X and ZY-ZZ-X, respectively. For original gate sequence, combined (ORIG-T) and separated (ORIG) implementations of ZZ-ZY-X exhibit comparable performance, while ORIG outperforms ORIG-T for ZY-ZZ-X. Moreover, for ZZ-ZY-X, separated implementation AOQ-SF surpasses combined implementations AOQ-T-F and AOQ-T-S, which em-

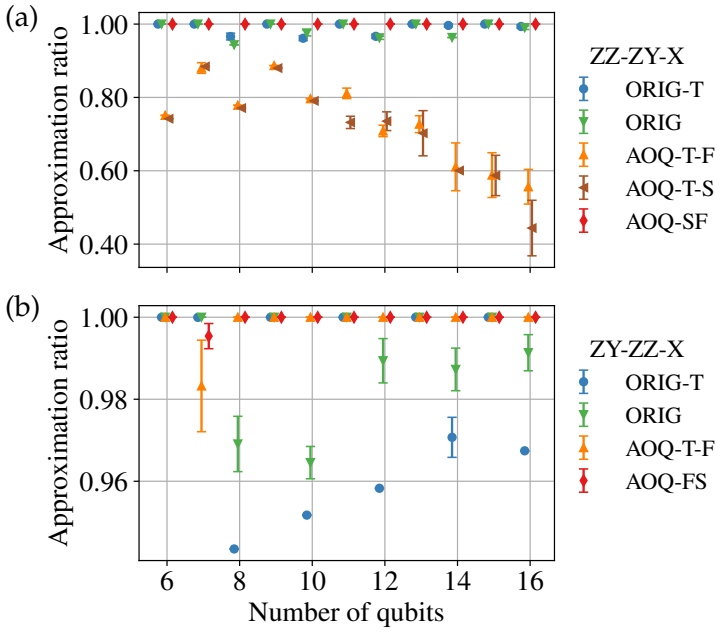

Figure 7.5: Comparison of implementing the ZZ and ZY gates as a combined unit versus separating the implementation with all ZZ gates applied first followed by all ZY gates (or vice versa). ORIG-T denotes the implementation of (a) ZZ-ZY or (b) ZY-ZZ gate as a unit using the original gate sequence, while ORIG represents the separated implementation. AOQ-T-F and AOQ-T-S correspond to the combined implementations using the first and the second block gate sequences from AOQMAP, respectively. AOQ-FS and AOQ-SF refer to the separated implementations. All error bars represent one SEM across 10 independent repetitions.

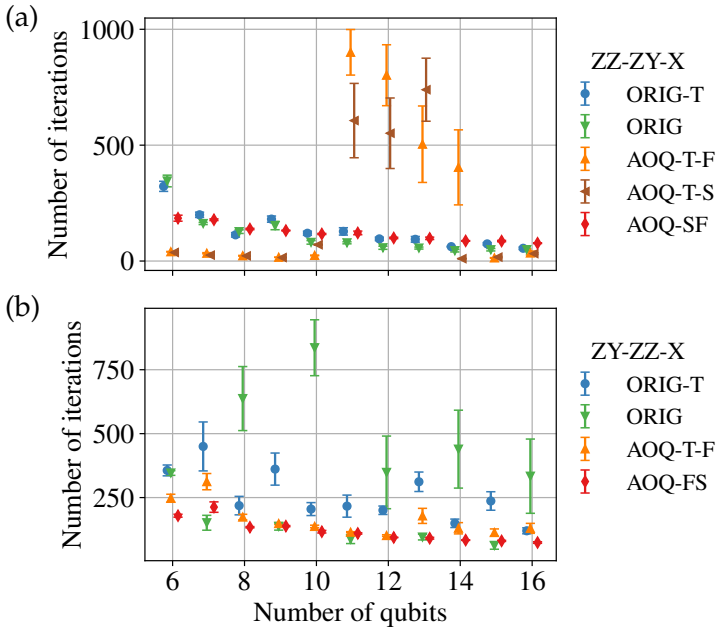

Figure 7.6: Number of iterations in the optimization process corresponding to Figure 7.5.

ploy gate sequences from block 1 and block 2 of Figure 7.1(e), respectively. Similarly, for ZY-ZZ-X, the separated implementation AOQ-FS also outperforms the combined implementation AOQ-T-F.

Figure 7.6 provides the number of iterations required in the optimization process, corresponding to results presented in Figure 7.5. In Figure 7.6(a), AOQ-T-F and AOQ-T-S, which exhibit lower and fluctuating approximation ratios, demonstrate fluctuations, whereas ORIG-T, ORIG, and AOQ-SF show a more stable and decreasing trend as qubit number increases. For ZY-ZZ-X (Figure 7.6(b)), AOQ-T-F and AOQ-FS, where ZY term follows the sequence in block 1, consistently require a low number of iterations. In contrast, ORIG-T and ORIG produce fluctuating outcomes, aligning with the trends observed in approximation ratios shown in Fig 7.5(b).

The results demonstrate that ZZ-ZY-X with AOQ-SF and ZY-ZZ-X with AOQ-FS consistently achieve the highest approximation ratios and require fewer iterations as the number of qubits increases. These findings underscore the advantage of employing separate implementations to enhance the performance of DC-QAOA.

Symmetry in implementation

As previously discussed, ZY-X and X-ZY with AOQ gate sequence at depth two (AOQ-p_2 in Figure 7.3(a)) utilize an identical gate sequence for each depth, necessitating the use of swap layers in block 2 of Figure 7.1(e) to restore the initial qubit order. In this study, we investigate alternative implementations of ZY-X and X-ZY at depth two. Specifically, we examine the effects of symmetric implementations, which hold the potential to improve performance at both the algorithmic level by suppressing Trotter error [312, 237, 238] and the hardware level by enhancing noise resilience in quantum device, as demonstrated in Chapter 4. Moreover, symmetry has demonstrated advantages in variational quantum machine learning [281] and variational quantum optimization [313, 314, 315].

Figures 7.7(a) and 7.7(b) present the approximation ratio achieved using ORIG-FS and AOQ-FS gate sequences, respectively. AOQ-FS generally outperforms ORIG-FS. For ORIG-FS (Figure 7.7(a)), ZY-X-ZY performs better than other sequences but fluctuates with the number of qubits. For AOQ-FS (Figure 7.7(b)), X-ZY-ZY-X achieves the highest performance, highlighting the effectiveness of the symmetric sequence of ZY gates combined with the mixer Hamiltonian at the beginning and end of the sequence. Additionally, ZY-X-ZY-X demonstrates better performance compared to X-ZY-X-ZY, suggesting that mixer terms located at the end tend to outperform those located at the beginning.

Figure 7.8 shows the number of iterations corresponding to results in Figure 7.7. With ORIG-FS (Figure 7.8(a)), ZY-X-ZY exhibits a decreasing number of iterations as qubit number increases, consistent with its high performance in Figure 7.7(a). In contrast, sequences containing ZZ term demonstrate fluctuating iterations, aligning with their lower approximation ratios. For AOQ-FS (Figure 7.8(b)), DC-QAOA with ZY terms requires fewer than 300 iterations. The sequence X-ZY-ZY-X achieves the best performance, requiring fewer than 200 iterations.

These findings underscore the crucial role of the gate sequence in determining the performance of DC-QAOA with symmetric implementation. Strategic placement

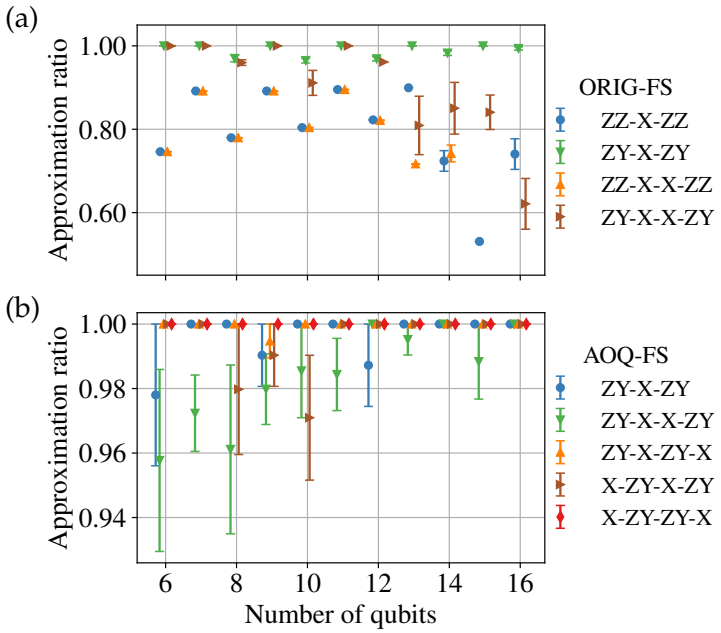

Figure 7.7: Comparison of symmetric implementation using (a) ORIG-FS and (b) AOQ-FS. All error bars represent one SEM across 10 independent repetitions.

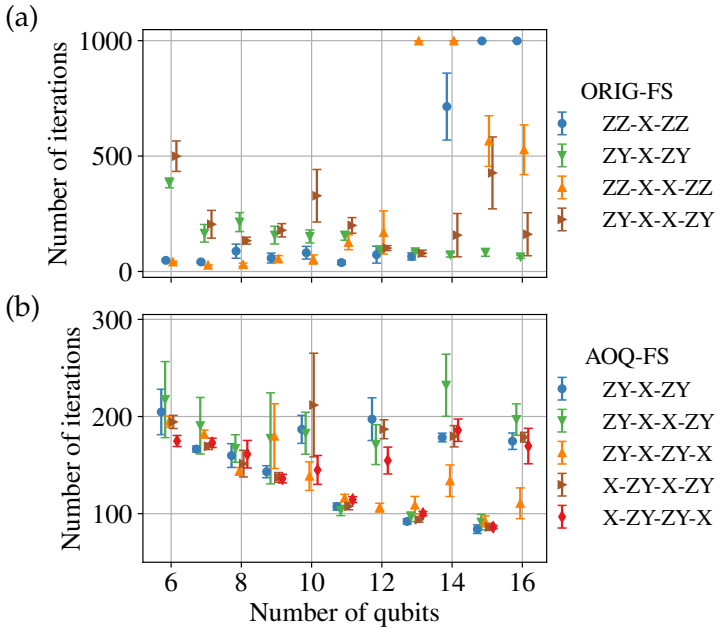

Figure 7.8: Number of iterations in the optimization process corresponding to Figure 7.7.

of mixer terms within the sequence can lead to improved approximation ratios and reduced numbers of iterations.

7.2.3 Approach to cooptimization

The previous results demonstrate that as qubit count increases, a reduction in the number of required iterations can result in stable and high performance with enhanced scalability. Gate sequences generated by AOQMAP provide advantages over original gate sequences in optimizing digitized counterdiabatic quantum algorithms. Moreover, AOQMAP employs a per-layer gate reversal strategy, which outperforms the per-gate reversal strategy used in original gate sequence, as evidenced by enhanced performance of AOQ-FS compared to ORIG-FS for ZY-ZZ-X, and AOQ-SF compared to ORIG-SF for ZZ-ZY-X. In all implementations, gate sequence of non-all-to-all commuting gates, such as ZY gates, significantly influences the algorithm performance. Although ZZ-ZY-X with AOQ-SF and ZY-ZZ-X with AOQ-FS achieve comparable high performance, ZY-ZZ-X with AOQ-FS can be obtained with an initialized qubit order $\{0, 1, 2, ..., n-1\}$, whereas ZZ-ZY-X with AOQ-SF requires an optimized initial qubit order (see Figure 7.1(e)), which may necessitate more time and effort to determine.

Based on these findings, we propose the method to enhance the performance of digitized counterdiabatic quantum algorithms, which involves cooptimizing algorithm parameters and gate sequences while satisfying hardware connectivity. Figure 7.9 illustrates the process for applying AOQMAP to the digitized counterdiabatic quantum algorithms with a deterministic Hamiltonian sequence. We first evaluate routing solutions on subtopologies using different swap layers such as for linear, T-shaped, and H-shaped configurations. Each initial qubit order introduces a unique gate sequence, leading to a fixed number of CX gates. For VQAs with fully connected two-qubit interactions, any arbitrary initial qubit order permits the implementation of all required two-qubit gates. However, when dealing with partially connected two-qubit interactions, finding the optimal solution becomes more difficult. Optimizing the initial qubit order becomes crucial for reducing the number of CX gates required for the implementation. Moreover, there may be a better solution that takes advantage of the incomplete connection requirement.

After minimizing CX gate count, the next step involves optimizing algorithm parameters using a classical optimizer to minimize the expectation value of the problem Hamiltonian or the energy. A specific number of initial qubit orders, which minimize CX gate count, are evaluated to identify the gate sequence that minimizes the energy. The optimized gate sequences generated from different types of subtopologies may exhibit varying algorithm performances. Therefore, a preselection can be conducted to identify gate sequences from different subtopologies that minimize energy and demonstrate better performance. These selected sequences are then executed on the target QPU with a cost function to assess qubit quality. After the execution, a post-selection step can be performed to identify the best mapping scheme from various types of subtopologies by selecting the one that corresponds to the minimum energy.

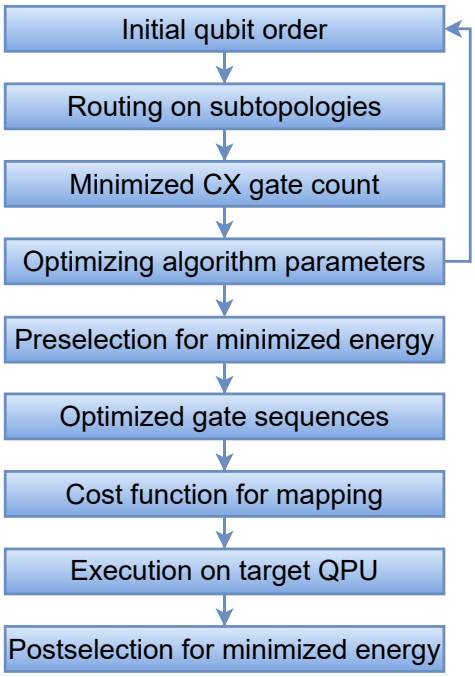

Figure 7.9: Optimization approach for applying AOQMAP to digitized counterdiabatic quantum optimization algorithms with a deterministic Hamiltonian sequence.

7.3 Applications and evaluation

We validate the effectiveness of our optimization approach through experimental evaluations conducted on three problem instances: unweighted MaxCut on complete graphs, unweighted MaxCut on noncomplete graphs, and portfolio optimization. The problem Hamiltonian for MaxCut on complete graphs shares similarities with the Hamiltonian representing portfolio optimization. In particular, portfolio optimization corresponds to the weighted MaxCut problem on complete graphs, where edges connecting two nodes have varying weights. This introduces additional complexities compared to the unweighted MaxCut problem.

7.3.1 MaxCut on complete graphs

We examine two variations of DC-QAOA for MaxCut on complete graphs. The first variation, ZY-ZZ-X, employs a ZY CD driving term and includes problem Hamiltonian. The second variation, ZY-YZ-X, utilizes a ZY-YZ CD driving term but excludes problem Hamiltonian. Both sequences follow a consistent Hamiltonian sequence, starting with the CD driving term, followed by problem Hamiltonian, and concluding with the mixer. This specific order provides advantages in dealing with non-all-to-all commuting gates in CD driving terms, which necessitates the optimization of gate sequences. We set the maximum number of iterations to 200 and employ 10 random initial guesses for the optimization process. Furthermore, we explore 10 initial qubit orders, starting with the default order $\{0, 1, ..., n-1\}$ and generating random permutations for the remaining orders. The order that minimizes the expectation value of the problem Hamiltonian is selected.

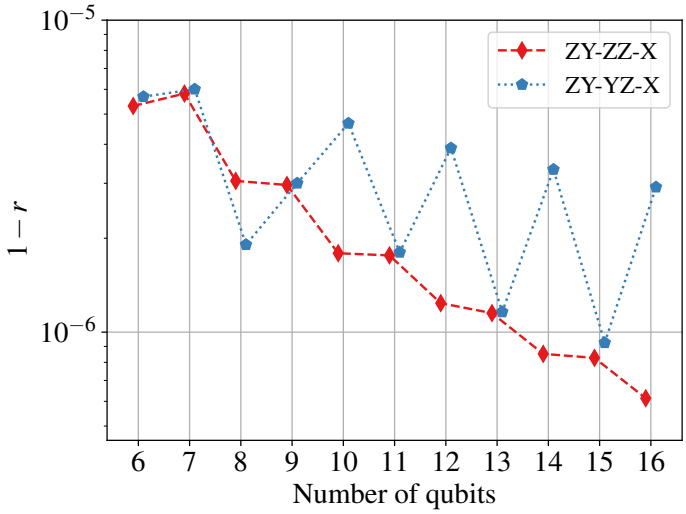

Figure 7.10: Deviation $1 - r$ of approximation ratio of DC-QAOA for MaxCut on complete graphs at $p = 1$ with the AOQ-FS gate sequence on a linear topology. A lower deviation indicates a higher approximation ratio, implying improved performance.

Figure 7.10 depicts the deviation of the approximation ratio of DC-QAOA for the MaxCut problem on complete graphs, with qubit number ranging from 6 to 16. The gate sequence is according to AOQ-FS on a linear topology. We observe that the deviation for each data point is less than 10^{-5}, demonstrating a high approximation ratio. Additionally, ZY-ZZ-X sequence exhibits a decreasing deviation as qubit count increases, implying improved performance. In comparison, ZY-YZ-X sequence demonstrates fluctuating results. One possible explanation is that non-all-to-all commuting YZ gates introduce different Trotter errors for odd and even numbered qubits. Among ten initial qubit orders examined, the default qubit order consistently produces the lowest expectation value for all data points in Figure 7.10. This finding underscores the robustness and efficacy of the default qubit order in achieving optimal results for this specific problem. Employing a predetermined initial qubit order facilitates the acceleration of the optimization process, as the search for an optimal initial qubit order is eliminated. Compared to routing solutions obtained using AOQMAP on linear topologies, solutions on T- and H-shaped topologies exhibit a reduced CX gate count but an increased circuit depth due to their enhanced connectivity, as demonstrated in Section 4.2. However, identifying an optimal initial qubit order that minimizes energy on these subtopologies may pose a challenge due to their unique characteristics.

7.3.2 MaxCut on noncomplete graphs

The MaxCut problem on noncomplete graphs is more general and challenging compared to the MaxCut problem on complete graphs. In noncomplete graphs, the absence of edges leads to the lack of corresponding ZZ gates, resulting in reduced symmetry and presenting significant challenges for both classical and quantum optimization algorithms. Furthermore, in the MaxCut problem on complete graphs, any initial qubit order yields the same additional CX gate count. However, for non-

complete graphs, different initial qubit orders result in varying additional CX gate counts. Consequently, there exists a trade-off between minimizing CX gate count and optimizing gate sequence. As shown in Figure 7.9, our approach starts by searching for an efficient mapping solution that minimizes CX gates on a subtopology. Here, we focus on T-shaped subtopology, but the methodology is also applicable to other subtopologies. We generate a set of N_o random initial qubit orders and select the order that yields the lowest CX gate count. Increasing N_o improves the probability of finding an optimal solution but also poses challenges in handling large problem instances. We set a maximum value of N_o to 3000 for problem instances investigated. This restriction allows us to determine a fixed gate sequence that minimizes CX gates among N_o tests conducted.

QAOA and DC-QAOA use the ORIG gate sequence, while DC-QAOA-OPT uses the AOQ-FS gate sequence. Furthermore, DC-QAOA employs the default Hamiltonian sequence ZZ-X-ZY, whereas DC-QAOA-OPT utilizes the ZY-ZZ-X. All algorithms utilize a gradient descent optimizer with 10 random initial guesses and a maximum number of iterations of 200. Figure 7.11 demonstrates a high performance of DC-QAOA-OPT compared to DC-QAOA and QAOA for the MaxCut problem on three regular graphs. The consistent improvement in the approximation ratio achieved by DC-QAOA-OPT underscores the effectiveness of the proposed optimization approach. We further highlight the advantages of AOQMAP in reducing CX gate count compared to Qiskit and Tket. In Qiskit, we use the default setting to compile the circuit directly on a 27-qubit QPU, with the default Hamiltonian and gate sequences. By default, Qiskit employs an optimization level of 1 to compile quantum circuits directly. This default level is chosen to strike a balance between optimization and computational effort. Higher levels may produce higher quality circuits but require more time and computational resources. In Tket, we use the NoiseAwarePlacement initial mapping and its default routing method *RoutingPass*. Then, we apply Tket's own optimization strategies including *DecomposeBRIDGE*, *DecomposeSWAPtoCX*, and *RemoveRedundancies*. Finally, the default setting of Qiskit's transpiler is used to decompose gates into the backend's basis gate set. Figure 7.12 shows comparison results. We observe that AOQMAP consistently produces the fewest number of CX gates compared to Qiskit and Tket. Specifically, AOQMAP achieves an average reduction of 27.2% in CX gate count. These results highlight the improved performance of AOQMAP at the algorithmic level, while also providing efficient qubit mapping solutions.

7.3.3 Portfolio optimization

We now investigate the application of QAOA and DC-QAOA in solving portfolio optimization problem, which aims to identify a portfolio that maximizes expected return while minimizing risk, subject to certain constraints [316, 94] (see Section 3.2.1). Specifically, we evaluate the performance of QAOA, DC-QAOA, and DC-QAOA-OPT for portfolio optimization instances with a qubit count ranging from 3 to 5 and a depth ranging from 1 to 3. Our primary focus is on assessing the effectiveness of the optimized DC-QAOA utilizing AOQMAP in comparison to conventional DC-QAOA and standard QAOA approach. The default DC-QAOA utilizes ZZ-Z-X-ZY Hamiltonian sequence with ORIG gate sequence, repeatedly applied p times to reach

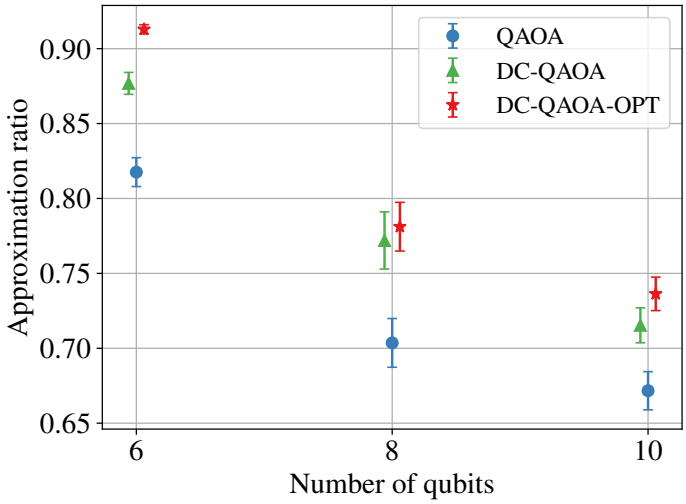

Figure 7.11: Approximation ratio of QAOA, DC-QAOA, and DC-QAOA-OPT at $p = 2$ for MaxCut on 10 randomly generated 3-regular graphs. All error bars represent one SEM.

a depth of p. In contrast, DC-QAOA-OPT explores three Hamiltonian sequences with AOQ-FS gate sequence: ZY-ZZ-Z-X, ZZ-Z-X-ZY, and ZY-YZ-Z-X. The sequence minimizing energy is selected. For ZY-ZZ-Z-X, ZY CD driving term is applied first, followed by problem Hamiltonian, and finally the mixer Hamiltonian. In ZZ-Z-X-ZY, ZY CD driving term is placed at the end. In comparison, ZY-YZ-Z-X employs a ZY-YZ CD driving term while lacking the two-qubit Hamiltonian in problem Hamiltonian. In both DC-QAOA and DC-QAOA-OPT, CD driving term follows the same coefficient as the ZZ term in problem Hamiltonian (Eq. 3.10).

The COBYLA algorithm is employed in the optimization process, with a maximum number of function evaluations (maxiter) set to 1000. The optimization strategy for QAOA follows the approach outlined in [94]. For DC-QAOA and DC-QAOA-OPT, the strategy for the QAOA part (ZZ-Z-X) is maintained, while randomness is introduced into the CD driving term. To further enhance the exploration of the solution space, N_I randomly generated initial guess values are additionally incorporated into the optimization process for QAOA, DC-QAOA, and DC-QAOA-OPT. In our case, N_I is set to 30. Similar to the MaxCut on complete graphs, any initial qubit order allows for the implementation of all two-qubit gates in DC-QAOA for portfolio optimization but results in different gate sequences, leading to variations in optimization efficiency. DC-QAOA-OPT employs the optimization process shown in Figure 7.9 to generate an optimized gate sequence. This involves evaluating N_o initial qubit orders to identify the one that minimizes the expectation value of problem Hamiltonian. In this study, ten different initial qubit orders are utilized to optimize the gate sequence on a linear topology. The optimization process is repeated ten times, and results are presented with error bars that represent one SEM.

Figure 7.13 depicts the approximation ratio and success probability of QAOA, DC-QAOA, and DC-QAOA-OPT for portfolio optimization with varying numbers of qubits. The left, middle, and right subfigures correspond to instances with three, four, and five qubits, respectively. The results clearly demonstrate that DC-QAOA-OPT outperforms DC-QAOA and QAOA in terms of both approximation ratio and success

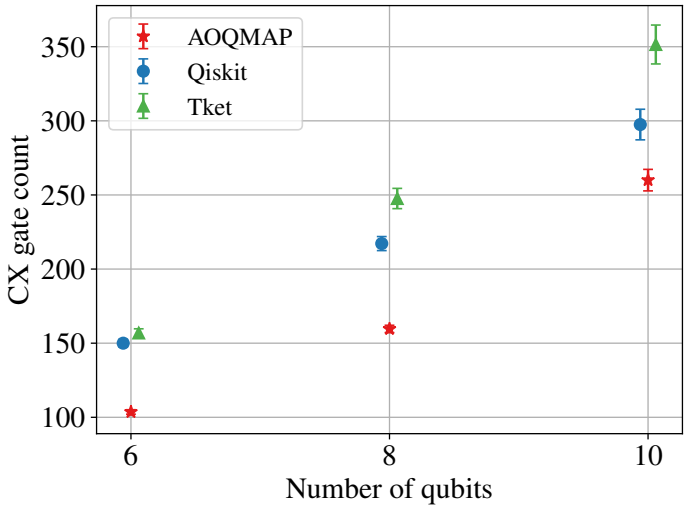

Figure 7.12: Number of CX gates in the DC-QAOA circuit at $p = 2$ for MaxCut on 10 randomly generated 3-regular graphs mapped using AOQMAP, Qiskit, and Tket. All error bars represent one SEM.

probability, highlighting the advantages of cooptimizing algorithm parameters, gate sequences, and qubit mapping.

To summarize, we propose a methodology for efficiently applying the AOQMAP to DC-QAOA, leveraging the Suzuki-Trotter decomposition to optimize qubit mapping and gate sequences. The AOQMAP approach introduces a customized gate sequence tailored to the initial qubit order and Hamiltonian sequence. Our approach generates optimized gate sequences that minimize SWAP gates, improving the resilience of algorithms on noisy quantum devices. Furthermore, these optimized gate sequences facilitate efficient optimization of algorithm parameters, leading to improved performance at the algorithmic level compared to the default sequence. Extensive experiments demonstrate that optimizing sequences of Hamiltonian and non-all-to-all commuting gates leads to substantial performance gains, as evidenced by increased approximation ratio and reduced iterations as the qubit count increases. The effectiveness of our approach is further confirmed by its successful application to other problem instances, showcasing its ability to enhance performance and pave the way for further advancements in DC-QAOA.

Table 7.1: Average (avg) and maximum (max) reduction in CX gates and circuit depth using AOQMAP-L compared to other qubit mapping strategies in Figure 7.14.

	Qiskit-Opt1		Qiskit-Opt2		Qiskit-Opt3		Tket		2QAN	
	avg	max	avg	max	avg	max	avg	max	avg	max
CX count	65.1%	77.1%	65.2%	77.2%	43.4%	47.7%	27.3%	31.7%	8.8%	16.3%
Depth	79.0%	90.5%	79.2%	90.8%	83.3%	87.9%	34.4%	35.9%	72.6%	84.9%

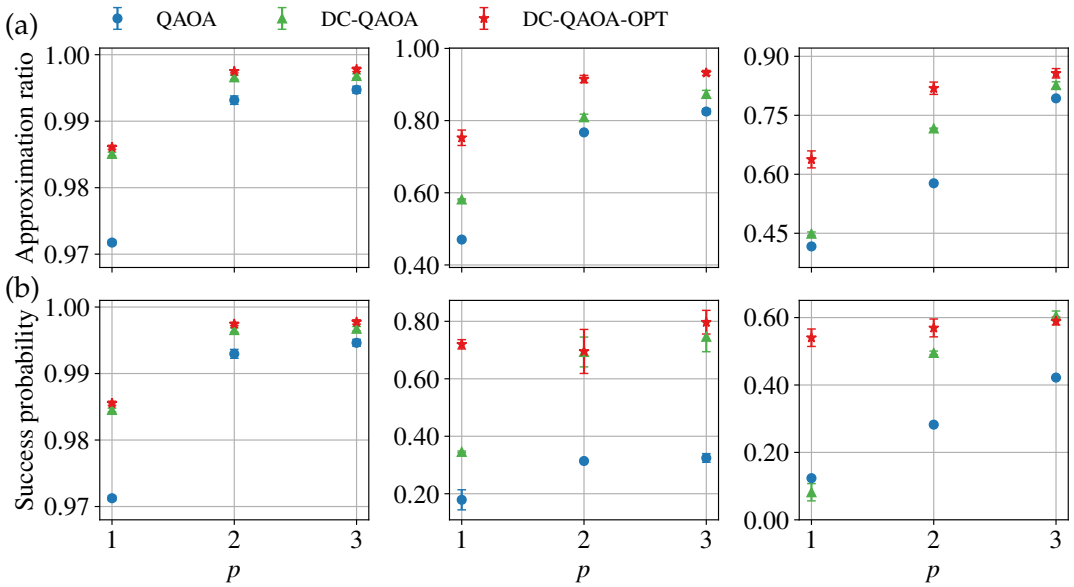

Figure 7.13: (a) Approximation ratio and (b) success probability of QAOA, DC-QAOA, and DC-QAOA-OPT for portfolio optimization with three qubits (left), four qubits (middle), and five qubits (right). DC-QAOA utilizes the ZZ-Z-X-ZY Hamiltonian sequence with the ORIG gate sequence, while DC-QAOA-OPT is optimized over the ZY-ZZ-Z-X, ZZ-Z-X-ZY, and ZY-YZ-Z-X Hamiltonian sequences with AQO-FS gate sequence. All error bars represent one SEM across 10 independent repetitions.

7.3.4 Qubit mapping strategy evaluation

In addition to demonstrating performance improvement of our optimized gate sequence over the default one, we compare our optimization approach to commonly used qubit mapping strategies, including industrial compilers Qiskit [208] and Tket [210], as well as the application-specific compiler 2QAN [317]. Effectiveness is evaluated by measuring circuit depth and CX gate count, with lower values indicating higher efficiency. The 2QAN method leverages the flexibility of permuting gates within Hamiltonian to find solutions that minimize SWAP gates [317]. It compiles only the first Trotter step and utilizes the same circuit for odd steps while reversing gate order for even steps, exhibiting a similarity to AOQMAP's utilization of mirror effects. This suggests that for comparison of 2QAN and AOQMAP with respect to circuit properties (excluding optimized gate sequences), evaluating depth-one QAOA circuits suffices. However, unlike our approach that also optimizes gate sequences, 2QAN solely minimizes SWAP gates. For Qiskit and Tket, we apply our optimized sequences to original QAOA circuits for fairness. Again, depth-one QAOA circuits remain sufficient for evaluation. This is because, for depth-two QAOA, Qiskit and Tket typically require more than double the number of SWAP compared to depth one, while 2QAN and AOQMAP guarantee double SWAP gates for depth two. Finally, focusing solely on QAOA is sufficient since any solution found for QAOA can be extended to DC-QAOA using the strategy proposed in Figure 7.1.

The QAOA for MaxCut on complete graphs are first employed as benchmarks. In this case, AOQMAP provides direct solutions for these all-to-all connected two-qubit

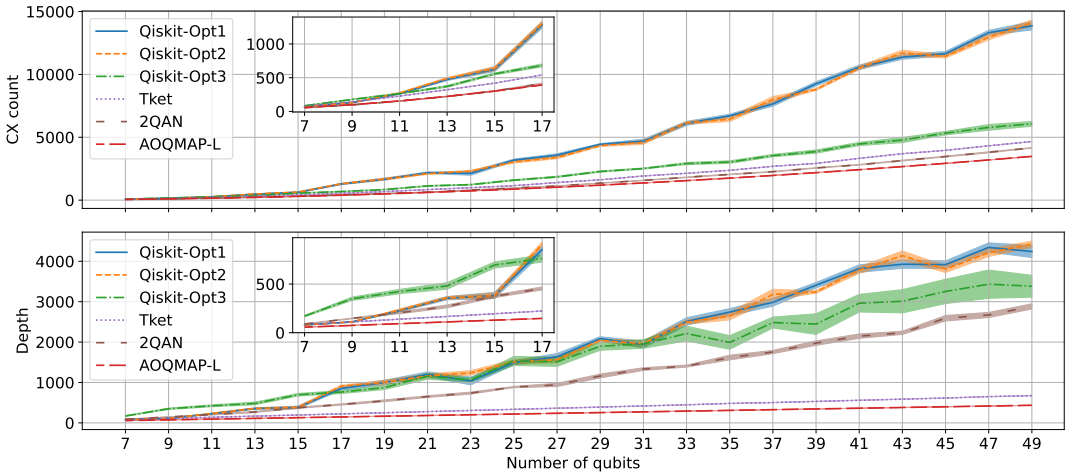

Figure 7.14: Number of CX gates and circuit depth of mapped QAOA for MaxCut on complete graphs with $p = 1$ and qubit counts ranging from 6 to 50 using Qiskit with optimization levels 1 (Opt1), 2 (Opt2), and 3 (Opt3), Tket, 2QAN, and AOQMAP-L. Lower values mean better results.

gates. We map QAOA circuits with a depth of one onto IBM's 127-qubit QPU without considering device's noise characteristics. Its native gate set includes {ECR, ID, R_Z, SX, X}, where ID is identity gate, R_Z is single-qubit rotation gate around z-axis, X is Pauli X gate, and SX is square root of X gate. However, here we decompose QAOA into gate set {CX, R_Z, R_X, R_Y}, where R_X and R_Y are single rotation gates around x-axis and y-axis, respectively. This choice prioritizes the mapping process by minimizing influence of decomposition and optimization introduced by Qiskit's transpiler. For comparison, we consider Qiskit with optimization levels from 1 to 3. For Tket, we utilize GraphPlacement for initial qubit mapping, followed by its default RoutingPass for routing. Its subsequent optimization strategies are described in Section 7.3.2. For 2QAN, the initial mapping strategy employs Qiskit's SABRE mapper due to the computational inefficiency of QAP within 2QAN framework for circuits with larger numbers of qubits and QPUs with many qubits, followed by 2QAN's own routing strategy. In AOQMAP, we employ the solutions on linear (L) subtopologies. T- and H-shaped sub-topologies offer increased connectivity, reducing the number of SWAP gates required while increasing the depth of mapped circuits. Qiskit's transpiler with an optimization level of 1 performs the final decomposition and optimization for Tket, 2QAN, and AOQMAP.

Figure 7.14 presents the number of CX gates and circuit depth of mapped QAOA for qubit numbers ranging from 7 to 49 with various mapping strategies. Within Qiskit, optimization levels 1 (Qiskit-Opt1) and 2 (Qiskit-Opt2) show minimal differences and produce the most CX gates and deepest circuit depths, indicating the lowest solution quality. While Qiskit-Opt3 offers improvement, Tket demonstrates lower CX counts and shallower circuit depths than Qiskit-Opt3. In comparison, 2QAN achieves fewer CX gates but deeper circuit depths than Tket. Finally, AOQMAP-L demonstrates the highest performance with the fewest CX gates and shallowest circuit depths. Table 7.1 summarizes the reduction in CX count and circuit depth achieved by AOQMAP-L compared to other methods. On average, AOQMAP-L reduces CX gates by 65%

(up to 77%) compared to Qiskit-Opt1 and Qiskit-Opt2. Similarly, the circuit depth reduction is substantial, averaging 79% with a maximum of 91%. Compared to Qiskit-Opt3, AOQMAP-L achieves an average reduction of 43% in CX count and 83% in circuit depth. Furthermore, AOQMAP-L outperforms Tket with an average 27% reduction in the number of CXs and 34% reduction in circuit depth. Finally, compared to 2QAN, AOQMAP-L shows an average reduction of 9% in CX count and 73% in circuit depth, with maximum reductions of 16% and 85%, respectively. Overall, AOQMAP-L demonstrates a significant average reduction of 42% in CX count and 70% in circuit depth compared to others.

We now evaluate the performance of QAOA for MaxCut problem on 3-regular graphs. In this study, we employ AOQMAP on T-shaped topologies (AOQMAP-T) to generate mapped circuits. The strategy involves optimizing the initial qubit order to minimize the number of CX gates. It's important to note that for an n-qubit QAOA, there are $n!/2$ distinct initial qubit orders due to the inherent symmetry, which is challenging, particularly for larger qubit systems. Therefore, it is important to develop efficient heuristic algorithms for optimizing this task, which is a promising area for future research.

To ensure a fair comparison across different strategies, all methods (Qiskit, Tket, and 2QAN) utilize identical, optimized gate sequences generated by AOQMAP-T. As depicted in Figure 7.14, AOQMAP-T generates circuits with the fewest CX gates and the shallowest depth, followed by 2QAN. While Qiskit with optimization level 3 (Qiskit-Opt3) achieves a lower CX count than Tket, it produces circuits with the deepest depth. Optimization levels 1 and 2 in Qiskit exhibit similar performance but result in the highest CX count. Notably, AOQMAP-T achieves an average maximum reduction of 52.2% in CX count compared to Qiskit-Opt1 and an average maximum decrease of 54.3% in circuit depth compared to Qiskit-Opt3, as detailed in Table 7.2.

7.4 Benchmarking experiments

In this section, we assess the efficiency of various qubit mapping techniques for DC-QAOA applied to MaxCut on complete graphs and portfolio optimization. We conduct a comprehensive benchmarking of their performance across a range of metrics, analyze the influence of noise, evaluate the effectiveness of error mitigation strategies, and demonstrate their implementation on IBM QPUs, providing valuable insights into the practical applicability of these mapping strategies.

Table 7.2: Average (avg) and maximum (max) reduction in CX gates and circuit depth using AOQMAP-T compared to other qubit mapping strategies in Figure 7.15.

	Qiskit-Opt1		Qiskit-Opt2		Qiskit-Opt3		Tket		2QAN	
	avg	max	avg	max	avg	max	avg	max	avg	max
CX count	52.2%	57.5%	51.7%	58.5%	20.8%	27.8%	36.9%	44.4%	7.9%	11.0%
Depth	50.5%	54.0%	51.0%	55.8%	54.3%	54.8%	44.7%	53.5%	24.0%	32.4%

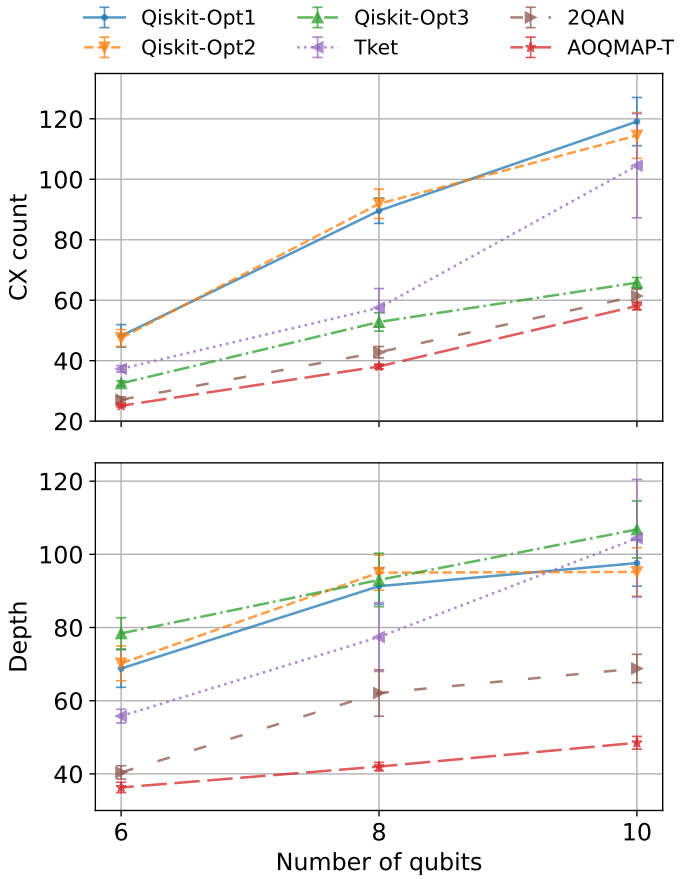

Figure 7.15: Number of CX gates and circuit depth of mapped QAOA for MaxCut on 10 randomly generated 3-regular graphs with $p = 1$ and qubit counts ranging from 6 to 10 using Qiskit with optimization levels 1 (Opt1), 2 (Opt2), and 3 (Opt3), Tket, 2QAN, and AOQMAP-L. Lower values mean better results. The results are obtained with the maximum numbers of initial qubit orders set to 3000, 20000, and 30000 for 6-qubit, 8-qubit, and 10-qubit QAOA.

7.4.1 Experimental setup and evaluation metrics

For AOQMAP, we employ the optimization strategy developed in Section 7.2 to determine the optimal parameters and gate sequence on a linear subtopology. For Qiskit, we utilize the default setting (optimization level 1). For Tket, we use the same settings as in Section 4.2.6. While Qiskit and Tket directly map circuit onto target QPU, AOQMAP requires selecting high-quality qubits for execution, which is performed in two stages (see Section 4.2). First, mapomatic [236] is employed to identify all subtopologies that satisfy connectivity constraints of the resulting circuit. Then, a cost function is used to select the subtopology that minimizes error by calculating circuit's error on each subtopology. For a fair evaluation, all mapped circuits are decomposed onto hardware native basis gate set and optimized using Qiskit transpiler with the default setting. The properties of QPUs can be found in Appendix A.2. It is worth noting that as late as possible (ALAP) scheduling method, customizable in Qiskit's transpiler, is consistently applied across all experiments.

Several metrics are employed to assess the performance of different qubit mapping strategies. The number of CX gates and circuit depth in the resulting circuit are used to compare the efficiency of qubit mapping approaches. Additionally, the approximation ratio is used to evaluate the algorithm performance. A lower CX gate count and shorter circuit depth indicate a more efficient mapping approach, while a higher approximation ratio suggests improved performance. In the absence of noise, Qiskit and Tket should produce similar results as they map the same original circuit. Conversely, AOQMAP generates optimized gate sequences that differ from the original one. In the presence of noise, the performance with different qubit mapping strategies varies due to the distinct number of CX gates in the resulting circuits, reflecting the quality of the mapping process.

7.4.2 Circuit properties

We now investigate circuit properties of DC-QAOA with AOQMAP and compare them to properties obtained using Qiskit and Tket with the experimental setup described previously. Table 7.3 presents the numbers of CX gates and circuit depths in the final circuit of DC-QAOA with qubit count ranging from 1 to 3 for MaxCut on complete graphs using AOQMAP, Qiskit, and Tket on a 7-qubit QPU ibm_perth. We observe that AOQMAP consistently produces the fewest number of CX gates and the shortest circuit depth compared to Qiskit and Tket. Specifically, AOQMAP reduces CX gate count by an average of 28.6% and 18.7%, and circuit depth by 34% and 32% compared to Qiskit and Tket, respectively. These results highlight the effectiveness of AOQMAP in optimizing the circuit properties, resulting in more efficient implementations of algorithms on the QPU.

Table 7.4 presents results for the same problem instances as Table 7.3 on a 27-qubit QPU ibmq_ehningen. While CX gate count remains unchanged with AOQMAP, circuit depth varies slightly between different QPUs. In contrast, Qiskit and Tket result in different CX gate counts and circuit depths on two devices. Furthermore, AOQMAP achieves an average reduction of 20% and 19.7% in CX gate count, and an average reduction of 36% and 32.6% in circuit depth compared to Qiskit and Tket, respectively. These findings demonstrate the consistent ability of AOQMAP to enhance circuit properties across different QPUs.

Tables 7.5 and 7.6 present circuit properties of DC-QAOA for portfolio optimization on two 27-qubit QPUs ibm_cairo and ibmq_ehningen, respectively. Similarly, AOQMAP consistently achieves the lowest CX gate count and the shortest circuit depth compared to Qiskit and Tket. Specifically, on ibm_cairo, AOQMAP reduces CX gate count by 29.3% and 18.2%, and circuit depth by 36% and 30% compared to Qiskit

Table 7.3: Circuit properties of n-qubit DC-QAOA at $p = 1$ for MaxCut on complete graphs with AOQMAP, Qiskit, and Tket approaches on a 7-qubit QPU ibm_perth.

	AOQMAP		Qiskit		Tket	
n	CX count	depth	CX count	depth	CX count	depth
3	14	35	18	44	18	44
4	30	46	46	68	33	73
5	52	59	73	116	69	96

Table 7.4: Circuit properties on a 27-qubit QPU ibmq_ehningen. Same as in Table 7.3.

	AOQMAP		Qiskit		Tket	
n	CX count	depth	CX count	depth	CX count	depth
3	14	35	18	44	18	44
4	30	47	33	73	33	73
5	52	56	73	116	72	96

and Tket, respectively. On ibmq_ehningen, AOQMAP achieves an average reduction of 29.3% and 28.2% in CX gate count compared to Qiskit and Tket, respectively, and 33.9% and 33.2% in circuit depth.

In summary, the AOQMAP approach yields an average reduction of 28.8% in the CX gate count and 33.4% in circuit depth across all problem instances when compared to Qiskit and Tket on the tested QPUs, demonstrating its potential to enhance the performance of DC-QAOA in various applications.

7.4.3 Noise simulation and error mitigation

To assess the impact of noise on the performance of mapped circuits presented in Section 7.4.2, we simulate their behavior under noise models. We also explore the effectiveness of various error mitigation strategies including readout error mitigation (REM), dynamical decoupling (DD), and zero-noise extrapolation (ZNE) in suppressing these noise effects. The detailed information on strategies can be found in Section 2.3.3. In this study, we leverage the CPMG pulse sequence to mitigate errors within the quantum circuit. The CPMG pulse sequence (Figure 6.2(a)), denoted as $\tau/4 - \mathrm{X} - \tau/2 - \mathrm{X} - \tau/4$, involves applying two X pulses, separated by a delay $\tau/2$, with additional delays of $\tau/4$ at the beginning and end of the pulse sequence. The delay time τ is the idle time of the qubit minus the duration of two X pulses. For ZNE, we focus on folding two-qubit gates, particularly CX gates. We utilize the linear fitting method due to its simplicity and effectiveness. Moreover, we investigate the influence of REM on the algorithm performance by employing the matrix-free measurement mitigation (M3) technique [150]. M3 is a scalable method for mitigating measurement errors that eliminates the need to form or invert the entire assignment matrix. Instead, it operates within a subspace defined by the noisy input bit strings.

We initially employ a widely used depolarizing noise model, as described in Section 3.1.2, that introduces random single qubit bit flip, phase flip, and combined bit and phase flip errors into each gate, effectively capturing the effects of mixed noise

Table 7.5: Circuit properties of n-qubit DC-QAOA with a depth ranging from 1 to 3 for portfolio optimization using AOQMAP, Qiskit, and Tket approaches on a 27-qubit QPU ibm_cairo.

	AOQMAP		Qiskit		Tket	
n	CX count	depth	CX count	depth	CX count	depth
3	(14, 28, 42)	(32, 60, 88)	(18, 39, 60)	(42, 83, 124)	(18, 39, 60)	(42, 83, 124)
4	(30, 60, 90)	(47, 85, 144)	(42, 84, 126)	(75, 150, 221)	(42, 84, 126)	(75, 142, 208)
5	(52, 104, 156)	(67, 108, 180)	(76, 153, 237)	(107, 199, 322)	(71, 149, 224)	(82, 175, 265)

Table 7.6: Circuit properties on a 27-qubit QPU ibmq_ehningen. Same as in Table 7.5.

	AOQMAP		Qiskit		Tket	
n	CX count	depth	CX count	depth	CX count	depth
3	(14, 28, 42)	(32, 60, 88)	(18, 39, 60)	(42, 83, 124)	(18, 39, 60)	(42, 83, 124)
4	(30, 60, 90)	(47, 89, 138)	(42, 91, 126)	(75, 131, 221)	(42, 84, 126)	(77, 142, 212)
5	(52, 104, 156)	(66, 108, 180)	(73, 152, 228)	(94, 205, 301)	(73, 148, 223)	(94, 184, 274)

processes in quantum systems. In addition to depolarizing noise, we also consider thermal relaxation noise [283], which arises from the interaction between physical qubits and environment. The device's calibration data is incorporated to evaluate algorithm performance under realistic noise levels. This dataset encompasses essential qubit and gate properties, such as error rates, energy relaxation time T_1, and dephasing time T_2.

Figure 7.16 presents simulation results of DC-QAOA for portfolio optimization with five qubits and depths ranging from 1 to 3 on ibmq_ehningen. We begin by simulating the performance under depolarizing noise and subsequently applying error mitigation techniques including DD and ZNE. In Figure 7.16(a), we present the results without error mitigation, with DD, with ZNE, and with a combination of DD and ZNE. We observe that ZNE proves to be more effective than DD in mitigating errors from depolarizing noise. When both DD and ZNE are applied, the insertion of DD sequences introduces noise, resulting in a slight reduction in approximation ratio. Furthermore, Figure 7.16(b) depicts results for the combination of depolarizing and thermal relaxation noise. While the application of DD in Qiskit enhances approximation ratio of DC-QAOA at high depth ($p = 3$), it conversely diminishes the approximation ratio at low depth ($p = 1$). This observation highlights a trade-off between mitigating decoherence errors caused by thermal relaxation noise and the introduction gate errors due to insertion of pulse sequences from DD strategy. Additionally, the results demonstrate that a combination of DD and ZNE strategies yields the highest performance.

These findings provide valuable insights into the effectiveness of error mitigation techniques in enhancing the performance of DC-QAOA on QPUs. ZNE proves to be more efficient than DD in mitigating gate errors resulting from depolarizing noise. Moreover, the study highlights the importance of minimizing CX gates and circuit depth to achieve improved performance, as evidenced by higher approximation ratios of AOQMAP, emphasizing the critical role of optimizing circuit properties during compilation process.

7.4.4 Demonstration on IBM quantum devices

Following simulation results presented in the previous section, we now demonstrate the performance of DC-QAOA and error mitigation strategies on IBM QPUs, aiming to validate the effectiveness of our optimization strategies in achieving high performance on near-term quantum devices.

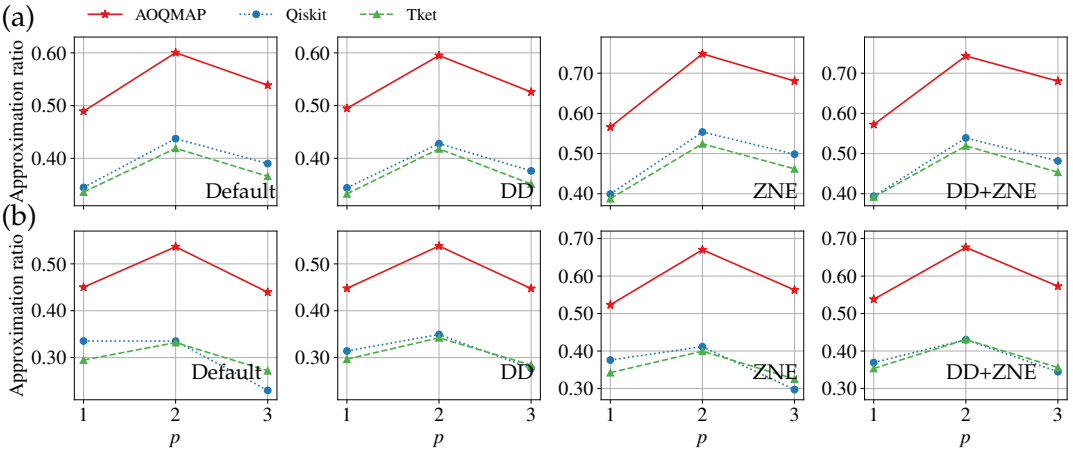

Figure 7.16: Noise simulation and mitigation of DC-QAOA for portfolio optimization with five qubits using AOQMAP, Qiskit, and Tket. Approximation ratio under (a) depolarizing noise and (b) depolarizing and thermal relaxation noise. We compare four situations from left to right: no error mitigation, applying DD, applying ZNE, and applying both DD and ZNE.

MaxCut on complete graphs

We begin by investigating the MaxCut problem on complete graphs, focusing on evaluating DC-QAOA on QPUs ibm_perth and ibmq_ehningen. For noiseless simulations, both DC-QAOA and DC-QAOA-OPT achieve an approximation ratio of 1.00 for tested problem instances, serving as a baseline for their subsequent implementation on QPUs. During execution, we set the number of shots to 50000 for ibm_perth and 20000 for ibmq_ehningen, ensuring sufficient coverage. In the ZNE error mitigation process, we use scale factors of {1, 2, 3} for ibm_perth and {1, 1.5, 2, 2.5, 3} for ibmq_ehningen.

Figure 7.17 presents the results of DC-QAOA and DC-QAOA-OPT on ibm_perth. Specifically, Figures 7.17(a) and 7.17(b) illustrate the approximation ratio obtained without and with REM, respectively. The data show that the application of REM leads to a generally enhanced performance, while the relative behavior of different qubit mapping strategies remains consistent. AOQMAP achieves overall the highest approximation ratio across all situations, demonstrating its effectiveness with and without applying error mitigation. In comparison, for Qiskit at four and five qubits (Figure 7.17(a)), applying DD results in a substantial improvement in approximation ratio, increasing it from zero to around 0.5, while ZNE does not exhibit any noticeable impact. A similar trend is observed for Tket at five qubits, highlighting the effectiveness of DD in mitigating decoherence errors. In contrast, ZNE proves beneficial in improving the performance of AOQMAP with the shortest circuit depth, indicating its value in situations where circuit optimization is prioritized. The combination of DD and ZNE yields the highest overall performance. Incorporating REM further enhances performance. However, for the three qubits case, we observe that the approximation ratio exceeds one when both ZNE and REM are employed. This discrepancy may arise from potential over-mitigation during the postprocessing stage of ZNE and REM techniques.

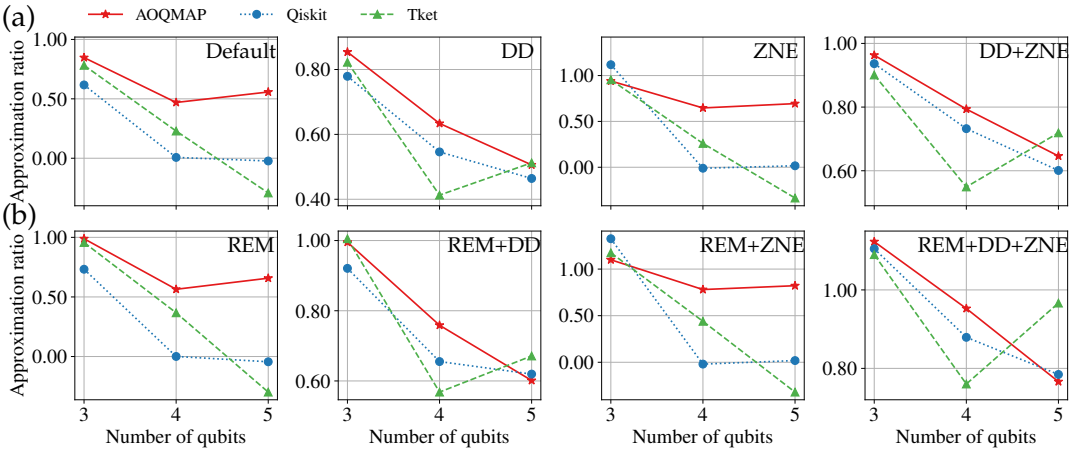

Figure 7.17: Approximation ratio of DC-QAOA for MaxCut on complete graphs using AOQMAP, Qiskit, and Tket on ibm_perth. (a) Results from left to right: no error mitigation, applying DD, applying ZNE, and applying both DD and ZNE. (b) Results corresponding to (a) and additionally applying REM.

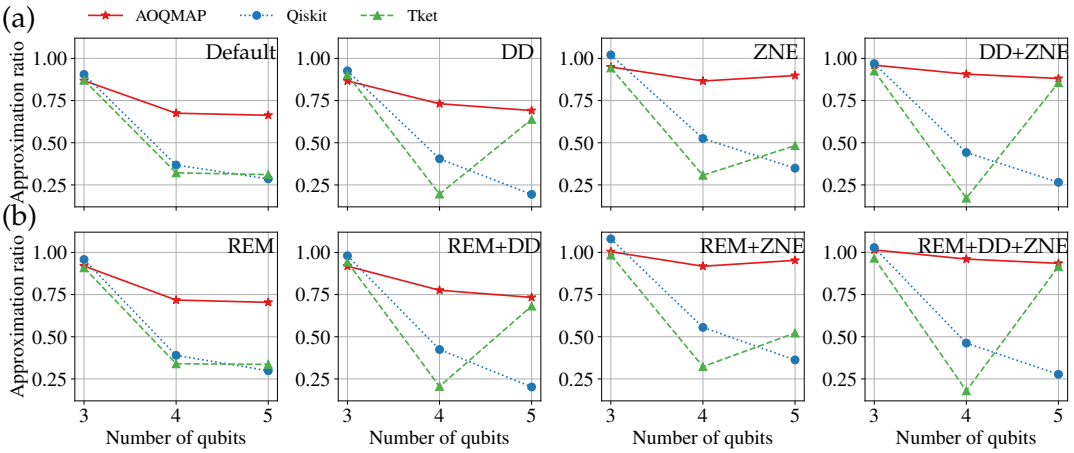

Figure 7.18: Demonstration on ibmq_ehningen. Same as in Figure 7.17.

Figure 7.18 illustrates results obtained on ibmq_ehningen using same problem instances as in Figure 7.17. AOQMAP consistently outperforms Qiskit and Tket, achieving the highest approximation ratio when combining DD, ZNE, and REM. Similar to observations presented in Figure 7.17, applying REM enhances approximation ratios, while the relative performance across various qubit mapping approaches remains unchanged. Furthermore, DD demonstrates significant effectiveness for Tket on five-qubit instances. A potential explanation is that qubits utilized in the execution are particularly sensitive to decoherence noise, rendering DD a beneficial strategy in this scenario.

Table 7.7: Approximation ratio of n-qubit DC-QAOA and DC-QAOA-OPT with a depth ranging from 1 to 3 for portfolio optimization in the absence of noise using the Qasm simulator.

n	3	4	5
DC-QAOA	[0.985, 0.997, 0.997]	[0.575, 0.821, 0.845]	[0.428, 0.718, 0.829]
DC-QAOA-OPT	[0.987, 0.998, 0.998]	[0.739, 0.924, 0.919]	[0.615, 0.882, 0.946]

Portfolio optimization

We now evaluate the performance of DC-QAOA for portfolio optimization on two 27-qubit QPUs: ibm_cairo and ibmq_ehningen. The specific instances and parameters for portfolio optimization problems can be found in Section 4.2. Table 7.7 presents noiseless simulation results for the default DC-QAOA with ORIG gate sequence and the optimized DC-QAOA-OPT with AOQMAP. DC-QAOA-OPT consistently exhibits an average higher approximation ratio compared to the default DC-QAOA, demonstrating improved performance at algorithmic level. For the implementation on both QPUs, we configure the number of shots to 20000, which ensures an adequate number of repetitions for statistical sampling on the given problem instances. Additionally, we employ a set of scale factors {1, 1.5, 2, 2.5} in ZNE on both QPUs.

Figures 7.19 and 7.20 display approximation ratio on ibm_cairo and ibmq_ehningen, respectively. REM is not employed in this demonstration. As depicted in Figure 7.19, AOQMAP consistently outperforms Qiskit and Tket across all problem instances. For three-qubit (Figure 7.19(a)), ZNE proves to be more efficient than DD for AOQMAP, while DD performs better for larger circuit depths produced by Qiskit and Tket. The combination of DD and ZNE yields better results compared to other cases. For four-qubit (Figure 7.19(b)), DD demonstrates better performance than ZNE across all qubit mapping strategies. Combining DD and ZNE leads to a significant enhancement in AOQMAP across all depths, with a particularly notable improvement at shallower depths for Tket and Qiskit. A similar trend is observed for five-qubit case (Figure 7.19(c)). The results highlight the effectiveness of DD in mitigating decoherence errors for deeper circuits and ZNE in mitigating gate errors for shallower circuits.

On ibmq_ehningen, for three-qubit (Figure 7.20(a)), ZNE outperforms DD at low depths, while DD performs better at high depths. Combining DD and ZNE achieves the highest performance. For four-qubit (Figure 7.20(b)), all mitigation strategies improve the performance of AOQMAP at each depth. DD demonstrates effectiveness at $p = 2$ for Qiskit, while the impact of error mitigation on Tket is not significant. For five-qubit (Figure 7.20(c)), all strategies also significantly improve the performance of AOQMAP. DD proves to be effective for Qiskit and Tket exhibits no significant changes under error mitigation.

Our findings demonstrate that while error mitigation techniques can enhance algorithm performance on near-term quantum devices, optimizing circuits for reduced depth and two-qubit gate count is essential to effectively leverage these techniques. Moreover, combining different error mitigation techniques presents an efficient approach to further enhance performance. DD proves to be particularly effective for longer circuit depths, whereas ZNE is more beneficial for shorter circuit depths. The combination of DD and ZNE yields consistent improvements in performance across

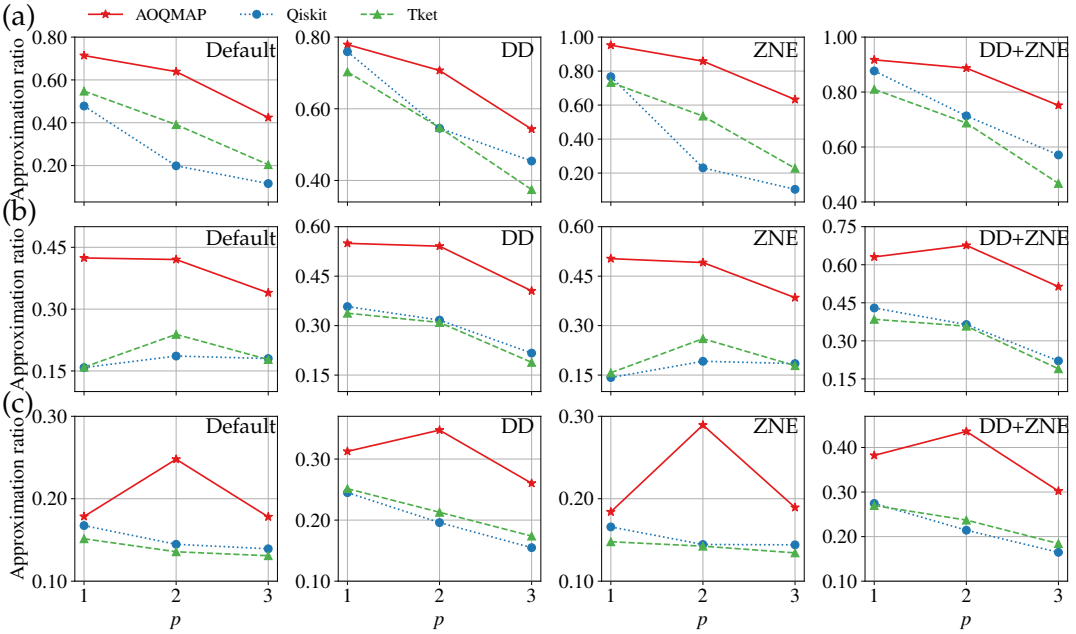

Figure 7.19: Approximation ratio of DC-QAOA for portfolio optimization on ibm_cairo with (a) three qubits, (b) four qubits, and (c) five qubits. From left to right: no error mitigation, applying DD, applying ZNE, and applying both DD and ZNE.

the board, highlighting the synergistic benefits of employing multiple error mitigation techniques.

Results analysis

We now present a detailed analysis of the specific achievements of AOQMAP, highlighting its advantages compared to Qiskit and Tket. To ensure a fair comparison, any approximation ratio values below zero were uniformly adjusted to a minimum value of 0.01. This adjustment allows for a consistent and meaningful comparison of the results across different scenarios.

Table 7.8 summarizes performance improvements obtained by applying error mitigation to AOQMAP, Qiskit, and Tket. The results show that REM yields an average improvement of 12.3% and is more effective for Tket and AOQMAP compared to Qiskit. On the other hand, DD significantly enhances performance, resulting in a 22.6% increase for AOQMAP, a 2.42× increase for Tket, and an impressive 5.65×

Table 7.8: Average increase in approximation ratio with error mitigation applied compared to the case without error mitigation.

	REM	DD	ZNE	DD+ZNE
AOQMAP	12.2%	22.6%	23.9%	51.9%
Qiskit	6.8%	5.65×	18.7%	7.62×
Tket	17.8%	2.42×	13.4%	3.48×
Mean	12.3%	2.77×	18.7%	3.87×

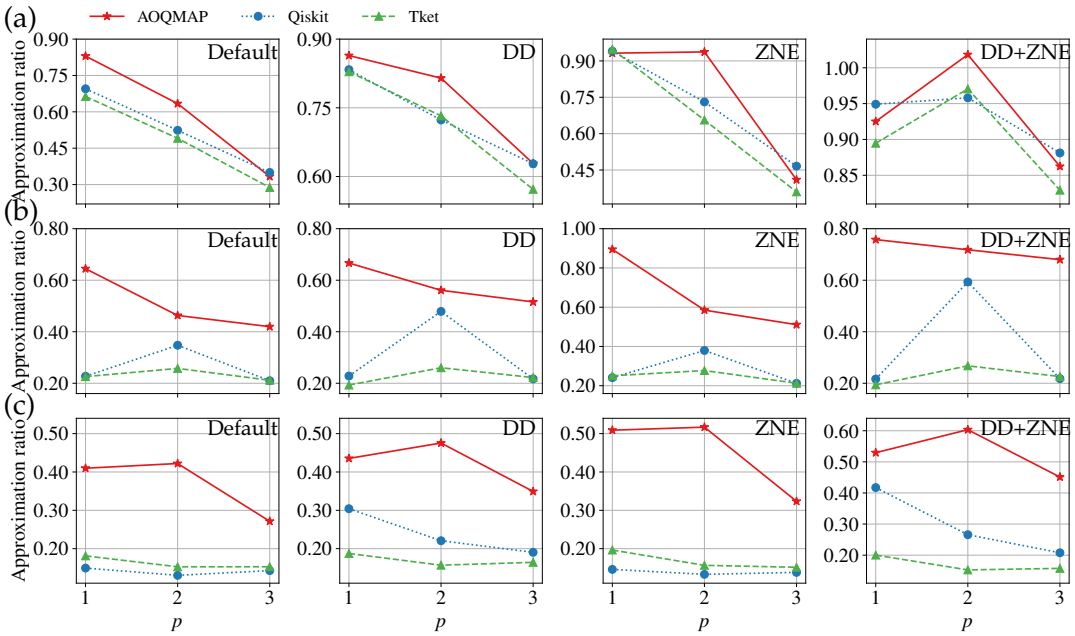

Figure 7.20: Demonstration on ibmq_ehningen. Same as in Figure 7.19.

Table 7.9: Average increase in the approximation ratio achieved with AOQMAP compared to Qiskit and Tket.

	Default	DD	ZNE	DD+ZNE
Qiskit	5.95×	58.6%	5.59×	69.3%
Tket	3.04×	73.9%	3.83×	100.4%
Mean	4.49×	66.3%	4.71×	84.8%

increase for Qiskit. This discrepancy can be attributed to the deeper circuits generated by Qiskit and Tket. In contrast, ZNE exhibits higher efficacy for AOQMAP, resulting in an average increase of 23.9% in approximation ratio. The combination of DD and ZNE demonstrates the highest efficiency, which yields a 3.87× increase.

Table 7.9 presents an average improvement in approximation ratio achieved by AOQMAP compared to Qiskit and Tket across tested QPUs. Without error mitigation (default), AOQMAP demonstrates an average increase of 4.49× in approximation ratio. When employing ZNE alone, AOQMAP exhibits an average improvement of 4.71×. When only DD is applied, AOQMAP still maintains an increase of 66.3%. Combining DD with ZNE further enhances the average improvement to 84.8%.

These results underscore the effectiveness of AOQMAP compared to Qiskit and Tket, both with and without error mitigation. The higher quality of circuits generated by AOQMAP offers significant benefits when implementing error mitigation strategies, making the combination of DD and ZNE, as well as their separate applications, particularly efficient. In contrast, Qiskit and Tket rely on a single predominant strategy, which in this study is DD.

7.4.5 Discussion

Our methodology integrates algorithm-oriented qubit mapping with optimization strategies to enhance the performance of digitized counterdiabatic quantum optimization algorithms on near-term quantum devices. It optimizes both algorithm design and hardware implementation to achieve significant performance improvements. The standard QAOA algorithm employs all-to-all commuting ZZ gates, resulting in gate sequences that do not impact its performance. However, DC-QAOA introduces non-commuting gates in counterdiabatic driving term, making it crucial to optimize Hamiltonian and gate sequence for improved performance. Our findings indicate that different gate sequences, despite having the same Trotter error order, can exhibit varying efficacy during parameter optimization. Based on this observation, we propose an efficient approach that optimizes Hamiltonian, gate sequence, and algorithm parameters while adhering to hardware connectivity constraints. This comprehensive approach leads to performance gains at both algorithmic and hardware levels.

By benchmarking the optimized and mapped gate sequence obtained through AOQMAP with the original sequence mapped using Qiskit and Tket, we show that the optimized gate sequence leads to improved performance at the algorithmic level compared to the original one. When considering the impact of noise, the quality of qubit mapping becomes crucial. High-quality solutions produced by AOQMAP result in fewer CX gates and shallower depth, leading to lower noise accumulation and improved resilience against errors. Experimental demonstrations on IBM QPUs further validate the effectiveness of our approach. The optimized implementation DC-QAOA-OPT achieves the highest performance compared to the default implementation. Moreover, circuits mapped with AOQMAP benefit from each error mitigation technique, demonstrating efficiency of AOQMAP in leveraging different mitigation approaches, unlike circuits mapped by Qiskit and Tket, which rely on specific strategies.

Our study demonstrates significant performance improvements achieved through the combination of algorithm-oriented qubit mapping and algorithmic level optimization for digitized counterdiabatic quantum algorithms on near-term quantum devices. The cooptimization of qubit mapping, gate sequences, and algorithm parameters effectively reduces inherent errors, thus enhancing overall performance. These strategies present promising opportunities for addressing real-world optimization problems using NISQ devices. Future research can explore the extension of this cooptimization approach to other quantum algorithms. Moreover, the development of algorithms that strike a balance between optimizing gate sequences and minimizing CX gates for partially connected two-qubit Hamiltonians holds value.

7.5 Conclusions

We have investigated strategies to enhance the performance and noise resilience of QOAs on near-term quantum devices. These strategies encompass optimized quantum compilation, selective circuit optimization, synergistic error mitigation and circuit design, and algorithm-hardware cooptimization. We initially explored the compilation process of variational algorithms, proposing calibration-aware transpilation to efficiently utilize classical and quantum resources, considering algorithm

features and time-varying error rates. Addressing the NP-hard nature of qubit mapping, we introduced algorithm-oriented qubit mapping, providing optimal and scalable solutions for VQAs with fully connected two-qubit interactions on identified subtopologies. Based on observations from executing VQAs on near-term devices, we developed a resilience-optimized compilation workflow for general quantum algorithms, along with quality metrics for assessing compilation outcomes. These strategies contribute to resilient algorithm execution on NISQ devices. We further adapted our compilation process to optimize VQAs on a specific architecture called the bipotent architecture. This involves employing selective optimization after the qubit selection process. The selective optimization of quantum circuits shows promise in achieving useful large-scale quantum computing. This is because, for circuits with a high number of qubits, exclusively optimizing gates implemented by high-error hardware-native gates reduces calibration efforts and improves performance.

We then explored the significant impact of various factors on algorithm performance and the effectiveness of error mitigation across eight IBM QPUs. These factors include circuit fidelity, schedule duration, choice of hardware-native gates, algorithm implementations, types of dynamic decoupling (DD) sequences, and optimization levels. While the results highlight the general enhancement of algorithm performance and robustness through DD sequences, achieving significantly improved performance relies more critically on factors such as high-quality native gates of QPUs, symmetric algorithm implementation, and effective circuit optimization techniques. Therefore, a holistic approach considering both hardware characteristics and software optimization strategies is essential when designing quantum algorithms to maximize reliability and efficacy. Finally, we proposed an integrated algorithmic and hardware codesign approach to enhancing QOA resilience and performance by tuning algorithm parameters, leveraging efficient Trotterization, and tailoring solutions to quantum hardware constraints. Rigorous analysis and case studies demonstrate substantial improvements in QOA performance and solution quality on current quantum devices. By holistically addressing the multifaceted challenges across the quantum stack layers through a systematic workflow, we can unlock the potential of QOAs in the NISQ era.

Part III
Practical Applications

Preface

In the last part, we explore the practical application of QOAs. Chapter 8 delves into real-world use cases, outlining the challenges and techniques for implementing QOAs. Using portfolio optimization as a representative problem instance, we analyze the impact of parameter settings within algorithms on performance and resilience. By bridging QOAs with practical applications, Part III enables a comprehensive end-to-end resilience analysis of algorithms, which is essential for determining the feasibility and effectiveness of QOAs in solving real-world problems.

Chapter 8

Real-World Applications

Although QOAs have shown great promise in solving complex optimization problems in various fields, their application in real-world scenarios remains a significant concern. In this chapter, we discuss the challenges in quantum optimization, investigate techniques for encoding real world problems, and explore the effect of parameter settings in problem instances on the algorithm's performance.

8.1 Challenges and techniques

8.1.1 Challenges in advancing quantum optimization

Quantum optimization algorithms hold great promise for tackling complex real-world optimization problems that classical computational methods struggle with. However, significant challenges remain in advancing and implementing quantum optimization due to the limitations of current quantum technologies and methodologies.

The present state of quantum hardware poses considerable challenges for practical quantum optimization applications. Quantum computers are still in the early stages of development, characterized by a restricted number of qubits and high error rates [17]. Quantum systems are highly susceptible to environmental noise, leading to decoherence and the loss of quantum information, which undermines the reliability and accuracy of quantum operations necessary for complex optimization tasks. While error mitigation strategies show promise, as discussed in Chapter 6, more powerful QPUs and efficient circuit implementation are more crucial for achieving high performance. The limited connectivity between qubits within current quantum hardware architectures also presents a challenge. Many optimization problems require interactions between qubits that are not directly connected within the quantum hardware, requiring additional SWAP gates or circuit compilation techniques to map the problem onto the target hardware topology. These additional operations introduce overhead and impact the efficiency of the quantum optimization process. The optimized compilation proposed in Chapter 4 presents scalable and optimal solutions for VQAs with fully connected two-qubit interactions. However, generalizing these solutions to other algorithms requires further analysis. As the problem size increases, the scalability of current quantum hardware, along with corresponding strategies

for error mitigation, correction, and compilation, presents a significant hurdle in resolving large-scale optimization problems.

Another fundamental challenge is the development of efficient quantum algorithms for optimization. Although the QAOA and the VQE exhibit promising results, their effectiveness largely relies on the careful design of the problem Hamiltonian and the selection of appropriate variational ansatzes [19, 92]. Designing effective ansatzes that capture the problem structure and are implementable on near-term quantum devices remains an open research question [88]. Furthermore, VQE approaches often necessitate meticulous parameter tuning to achieve the desired performance. Determining suitable parameters can be computationally demanding and nontrivial. The Barren Plateau phenomenon [318, 319], in which the landscape for parameter optimization becomes exponentially flat and lacking in distinguishing features as the problem size increases, poses a significant obstacle. The selection of ansatz, initial state, and hardware noise can all contribute to the occurrence of this phenomenon. The development of algorithms capable of adaptive parameter tuning during execution is still in its early stages.

Efficient problem encoding and mapping from classical formulations to quantum architectures are vital for the success of quantum optimization. Many classical optimization problems need to be reformulated to fit the constraints of quantum hardware. Inefficient encoding techniques can result in suboptimal utilization of quantum resources and may negate the advantages offered by quantum computing. It is crucial to improve problem mapping and encoding strategies to maximize the effectiveness of quantum algorithms. We will discuss the problem landscape and corresponding encoding strategies in the next section.

Finally, verifying and validating quantum optimization results pose significant challenges. Due to the exponentially large Hilbert spaces in which quantum systems operate, classical simulation of quantum optimization becomes intractable for large problem sizes. This impedes the benchmarking and verification of the results obtained from quantum optimization algorithms [320, 321]. Developing efficient verification techniques and leveraging classical-quantum hybrid approaches are active areas of research aimed at addressing this challenge.

8.1.2 Problem landscape and encoding techniques

Real-world optimization problems encompass a diverse landscape, broadly categorized into three main classes: combinatorial, constrained, and machine learning optimization. Combinatorial problems, exemplified by the traveling salesman problem [322], involve finding the optimal configuration from a finite set of possibilities. Constrained optimization (e.g., portfolio optimization) seeks solutions that minimize an objective function while adhering to predefined constraints. Machine learning optimization aims to identify optimal parameters for models by minimizing a loss function, encompassing tasks like training neural networks [323] and support vector machines [324]. These problems often pose significant computational challenges for classical algorithms due to their inherent characteristics. Combinatorial problems exhibit NP-hardness, implying their solution time scales exponentially with problem size. Non-convexity and high dimensionality further complicate constrained and machine learning optimization problems.

To harness the power of QOAs for real-world problems, a critical initial step involves encoding these problems into a format compatible with available quantum hardware. This entails representing problem variables as qubits and transforming objectives and constraints into quantum cost functions. Two common formulations for encoding real-world optimization problems are the quadratic unconstrained binary optimization (QUBO) and the Ising model. In QUBO, problem variables are represented as binary vectors (0 or 1), and objectives and constraints are expressed as a quadratic cost function:

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i \leq j} Q_{i,j} x_i x_j, \tag{8.1}$$

where $x_i \in \{0, 1\}$, and $Q_{i,i}$ and $Q_{i,j}$ are the linear and quadratic coefficients, respectively. The Ising model represents the problem as a spin system where spins can take values of $+1$ (spin up) or -1 (spin down). Many NP problems can be formulated as Ising models [3]:

$$E(s) = \sum_i h_i s_i + \sum_{i \leq j} J_{i,j} s_i s_j, \tag{8.2}$$

where $s_i \in \{-1, +1\}$, h_i are the linear coefficients corresponding to qubit biases, and $J_{i,j}$ are the quadratic coefficients corresponding to coupling strengths between spins. The objective is to minimize $E(s)$. Both QUBO and Ising formulations serve as a bridge between the classical and quantum domains. They can be directly solved using quantum annealing [325] or transformed into Hamiltonian forms [326] for gate-based quantum algorithms, where binary variables are mapped to qubits and the cost function is encoded into a Hamiltonian representing the system's energy.

Optimizing encoding for quantum computing platforms is crucial. One commonly used approach is penalty-based encoding, where constraint violations are penalized in the objective function [327]. An example of this can be found in Section 2.2.4. However, this method can be sensitive to the chosen penalty strength and might yield suboptimal performance. In addition to constraints, the choice of variable encoding also substantially impacts the efficacy of quantum optimization algorithms [328]. Sparse encodings, like one-hot encoding, can impose additional constraints, whereas other encodings, such as binary encoding, may lead to smoother energy landscapes. It is important to carefully design the problem encoding and employ techniques to enforce constraints in order to successfully apply quantum optimization algorithms in real-world scenarios.

8.2 Parameters in problem instances

We now investigate the influence of different parameters in problem instances. Specifically, based on the problem formulation of portfolio optimization in Section 2.2.4, we explore the specific parameter settings.

8.2.1 Energy dependence on asset selection

We first explore the dependence of the system's exact energy on the number of selected assets B. With n available assets, $C(n, B) = \binom{n}{B}$ combinatorial choices exist

for selecting B assets. We analyze the minimum and maximum energy difference for two cases: solutions satisfying the budget constraint (Eq. 2.9) and the global case considering all possibilities. Additionally, the number of combinatorial choices is presented.

Figure 8.1 presents the results for n assets, where n ranges from 3 to 12, and B varies from 1 to n. As depicted in Figure 8.1(a), the constrained energy difference, considering only solutions satisfying the budget constraint, is zero at $B = n$ for all n values due to the selection of all n assets, resulting in a single choice. Initially increasing with B, the constrained energy difference peaks around $B \approx \lceil n/2 \rceil$ before decreasing. This peak indicates that selecting approximately half of the assets maximizes the range of energy differences, suggesting significant variation among possible portfolio options and potential for effective diversification. In contrast, Figure 8.1(b) shows the global energy difference, considering all possible solutions irrespective of the budget constraint, first decreases and then increases. This suggests that around $B \approx \lceil n/2 \rceil$, the difference between maximum or minimum global and constrained energies is minimized, indicating the budget constraint has minimal impact on achieving the best or worst overall energy outcome. Table 8.1 summarizes the positions where the maximum or minimum constrained and global energy differences occur in Figs 8.1(a) and 8.1(b). For n ranging from 3 to 11, these positions coincide. However, for $n = 12$, they differ by one. This observation consistently suggests that selecting half of the assets optimizes diversification among feasible solutions while minimizing the influence of unfeasible solutions due to the budget constraint. Furthermore, Figure 8.1(c) illustrates the number of possibilities for choosing B assets from a set of n assets. This value initially increases with B, reaches a peak at $B \approx \lceil n/2 \rceil$, and then decreases. The rise in the number of choices around this point complicates the identification of optimal solutions due to the greater number of potential portfolio configurations.

By examining the impact of the number of assets and budget size on portfolio optimization, we found that selecting roughly half the assets maximizes diversification (constrained energy difference) while minimally affecting the achievable energy range (global energy difference). However, this choice also coincides with the highest number of possible portfolio configurations, potentially complicating solution identification. These findings emphasize the need to balance diversification and solution complexity when constructing portfolios under budget constraints.

8.2.2 Impact of parameter settings

This section investigates the impact of parameters on the performance of the QAOA with the standard mixer in portfolio optimization problems. The parameters of interest are the risk preference parameter q, penalty factor A, and global factor λ. The goal is to optimize these parameters to enhance the algorithm's performance.

Table 8.1: Positions of the maximum constrained energy difference and the minimum global energy difference for n assets, as depicted in Figures 8.1(a) and 8.1(b), respectively.

n	3	4	5	6	7	8	9	10	11	12
Constrained	2	2	2	3	4	4	5	5	6	7
Global	2	2	2	3	4	4	5	5	6	6

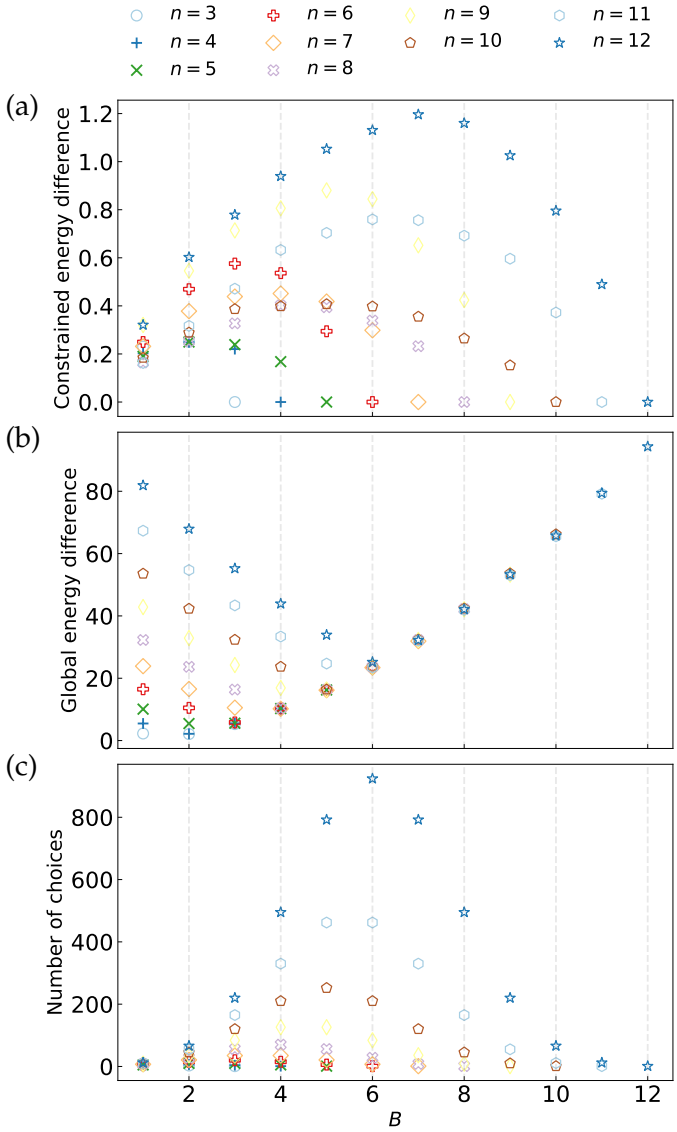

Figure 8.1: System energy difference and number of choices for choosing B assets from n assets: (a) Difference between the maximum and minimum energy of solutions satisfying the budget constraint, (b) difference between the global maximum and minimum energy of solutions regardless of budget constraints, and (c) number of choices that satisfies $C(n, B)$.

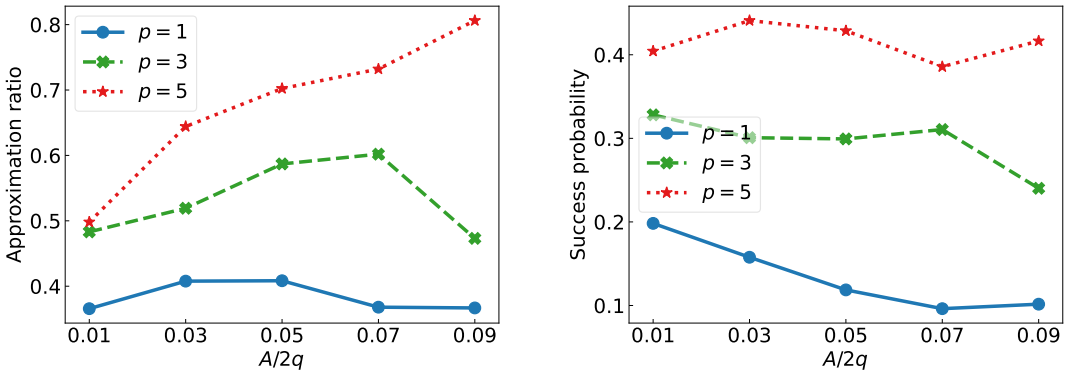

Figure 8.2: Simulation results of 5-qubit QAOA for portfolio optimization with $A/2q$ from 0.01 to 0.09, $q = 1/3$, $B = 2$, and $\lambda = 10$ using Qiskit's Qasm simulator with 2048 shots.

In the experiments, we consider a scenario with $B = 2$ assets selected from a set of 5 randomly generated assets. The risk preference parameter is set to $q = 1/3$, with a higher value of $1 - q$ indicating a greater risk preference. Three different QAOA depths, namely 1, 3, and 5, are explored. The COBYLA method is employed to optimize the angles $\vec{\gamma}$ and $\vec{\beta}$ for each QAOA instance, following the strategy outlined in [94]. The Qiskit's Qasm simulator with shots 2048 is used for all simulations.

We begin by examining the relationship between q and A. Figure 8.2 depicts the results for varying $A/2q$ from 0.01 to 0.09, while fixing λ at 10. For depths 1 and 3, the approximation ratio exhibits an initial increase followed by a decrease. Conversely, the success probability demonstrates a consistent decline with increasing A. Specifically, the probability at $p = 1$ is most susceptible to changes in A, showing a substantial decrease, whereas at $p = 3$, the decrease is slightly. In contrast, for depth 5, the approximation ratio steadily increases with a rising A, reaching its peak at $A = 0.09$. The success probability for depth 5 remains relatively stable around 0.4. These findings suggest that a higher penalty factor A is beneficial for achieving better performance with higher QAOA depths p, while a lower penalty factor is more suitable for shallower circuits. This implies that tailoring the penalty factor to the specific QAOA depth can be crucial for optimizing performance.

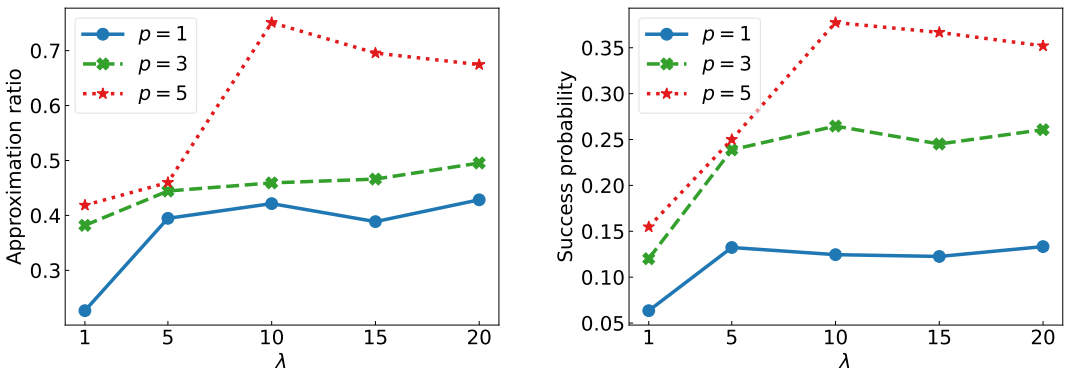

Figure 8.3: Simulation results of 5-qubit QAOA for portfolio optimization with λ from 1 to 20, $B = 2$, $q = 1/3$ and $A/2q = 0.05$ using Qiskit's Qasm simulator with 2048 shots.

We further investigate the influence of the global factor λ by fixing $A/2q = 0.05$ and keeping all other parameters unchanged. Figure 8.3 presents the simulation results for λ ranging from 1 to 20. The results indicate that a larger value of λ generally leads to improved performance. At $\lambda = 1$, both the approximation ratio and success probability are low, suggesting an inefficient classical optimization process. For depths 1 and 3, the approximation ratio exhibits a slight increase and remains nearly saturated when $\lambda > 5$. Similarly, the success probability for depths 1 and 3 shows minimal variation with increasing λ. However, for depth 5, a higher λ leads to a significant improvement in both approximation ratio and success probability, reaching a maximum at $\lambda = 10$ before a slight decline. These observations suggest that an optimal value of the global factor exists for different QAOA depths. This implies that individual parameter optimization for each QAOA depth, rather than using a uniform value for λ, is essential for achieving the highest performance.

This study highlights the critical role of parameter selection in maximizing the performance of the QAOA-based portfolio optimization approach. Specifically, we found that the optimal parameters are not consist for different QAOA depth, indicating importance of individually optimizing parameters in problem instances for each deterministic depth rather than using same problem instance parameters for all QAOA depth. By carefully tailoring the risk preference parameter, penalty factor, and global factor to the specific QAOA circuit depth, significant improvements in both approximation ratio and success probability can be attained.

8.3 Resilience analysis

This section analyzes the resilience of QAOA with different mixers and depths. We first simulate the optimization landscape under different noise conditions, followed by performance evaluation. We employ a composed bit-flip (Eq. 3.1) and phase-flip (Eq. 3.3) error channel with independent noise strengths for each type of error, ϵ_1 and ϵ_2, respectively. This channel acting on a single-qubit gate introduces bit-flip and phase-flip errors with a combined probability of $\epsilon_1\epsilon_2$. Additionally, it introduces independent bit-flip errors with a probability of $\epsilon_1(1-\epsilon_2)$ and phase-flip errors with a probability of $(1-\epsilon_1)\epsilon_2$. Finally, there is a residual probability of $(1-\epsilon_1)(1-\epsilon_2)$ for no error to occur. This model allows independent control over error types and extends to multi-qubits as a tensor product. Here, we focus on a scenario with equal error strengths $\epsilon_1 = \epsilon_2$.

8.3.1 Optimization landscapes

One crucial aspect of characterizing the efficacy of QAOA is the optimization landscape. This landscape visualizes the objective function's behavior across its input parameters, in this case, the angle parameters γ and β. The topography of this landscape, ranging from smooth with a single global minimum (ideal) to complex with numerous local minima (challenging), significantly impacts the efficacy of optimization algorithms. This section investigates the optimization landscape of depth-one QAOA, where the objective function represents the Hamiltonian's expectation value to be minimized.

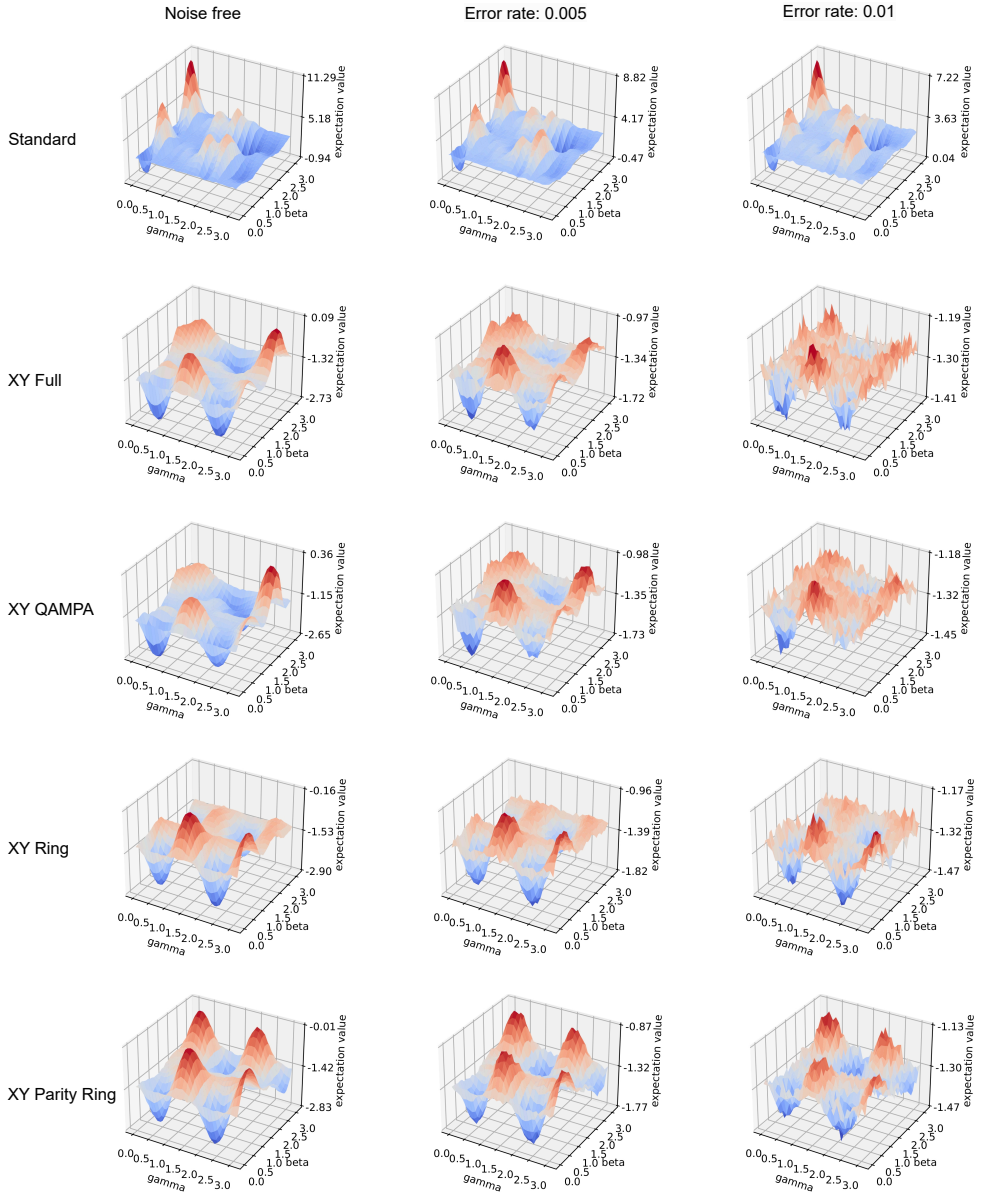

Figure 8.4: Optimization landscape of 5-qubit QAOA with different mixers under varying noise conditions using Qiskit's Qasm simulator with 8192 shots. From left to right: noise free, bit phase flip error with a strength of 0.005, and bit phase flip error with a strength of 0.01.

Figure 8.4 presents the optimization landscapes for various noise levels and mixer types used in the QAOA circuit. In the absence of noise, the QAOA circuit with the XY ring mixer achieves the minimum expectation value of the Hamiltonian (-2.90) at the optimal angles (γ, β) of $(2.2, 0.5)$. Conversely, the standard mixer achieves the highest expectation value (-0.94) at $(2.3, 2.1)$, indicating lower performance. This suggests that limited depth-one QAOA with a standard mixer might not achieve optimal performance, and increasing depth may be beneficial. In contrast, XY mixers offer a higher probability of finding optimal solutions. We note that the optimal expectation value for the used problem instance found by brute force search, corresponding to the optimal solution satisfying budget constraints, is -3.39.

As the noise strength increases, an expected increase in the expectation value is observed for all mixer types, confirming the detrimental effect of noise. However, the optimal angles for both standard and XY mixers remain relatively unchanged despite noise. For instance, the optimal angles for the XY ring mixer are $(2.2, 0.5)$ without noise, $(2.2, 0.5)$ for $\epsilon = 0.005$, and $(2.1, 0.5)$ for $\epsilon = 0.01$. The XY QAMPA mixer exhibits similar behavior and remains the same at $(0.3, 0.5)$. However, the XY full mixer shows a larger shift in optimal angles, from $(2.4, 0.4)$ in the noise-free case to $(0.5, 0.3)$ for $\epsilon = 0.005$, and $(0.6, 0.3)$ for $\epsilon = 0.1$. This suggests that XY full mixer is more sensitive to noise. Compared to the standard mixer, XY mixers consistently achieve lower expectation values across all noise levels. Moreover, the XY ring and parity ring mixers perform similarly across all noise levels, resulting in lower expectation values (higher performance) than XY full and QAMPA mixers. Additionally, the XY full mixer is more susceptible to noise, with lower expectation values than QAMPA in the noise-free case but higher values for $\epsilon = 0.005$ and $\epsilon = 0.01$.

The analysis of the optimization landscape highlights the critical interplay between mixer selection and noise mitigation for achieving optimal performance in noisy QAOA circuits. While XY ring and XY parity ring mixers offer inherent advantages in terms of lower expectation values, all mixer types suffer from performance degradation in the presence of noise. This emphasizes the need to explore robust noise mitigation techniques alongside careful mixer selection for practical QAOA implementations.

8.3.2 Mixers in QAOA

This section explores the influence of mixer selection and QAOA depth on quantum circuit properties and noise performance.

Table 8.2: Circuit depth for 5-qubit QAOA with different mixers and depths.

p	1	2	3	4	5	6	7
Standard	19	36	53	70	87	104	121
XY Full	86	119	152	185	218	251	284
XY QAMPA	88	124	160	196	232	268	304
XY Ring	90	132	173	215	255	297	339
XY Parity Ring	82	111	140	169	198	227	256

Table 8.3: Total number of gates for 5-qubit QAOA with different mixers and depths.

p	1	2	3	4	5	6	7
Standard	50	90	130	170	210	250	290
XY Full	156	244	332	420	508	596	684
XY QAMPA	171	276	381	486	591	696	801
XY Ring	138	215	291	368	443	520	597
XY Parity Ring	138	208	278	348	418	488	558

Table 8.4: Number of CX gates for 5-qubit QAOA with different mixers and depths.

p	1	2	3	4	5	6	7
Standard	20	40	60	80	100	120	140
XY Full	68	108	148	188	228	268	308
XY QAMPA	58	88	118	148	178	208	238
XY Ring	58	87	116	145	174	203	232
XY Parity Ring	58	88	118	148	178	208	238

The initial state preparation differs between mixer types. The standard mixer utilizes the simple $|+\cdots+\rangle$ state, whereas XY mixers necessitate the Dicke state [102]. This requirement introduces additional two-qubit gates, inherently increasing circuit complexity. Tables 8.2, 8.3, and 8.4 compare circuit depth, total gate count, and CX gate count for various mixers and depths using 5-qubit QAOA. Our analysis reveals a significant rise in circuit depth for XY mixers compared to the standard mixer. On average, XY Full mixers lead to a depth increase of 194.41%, followed by XY QAMPA (209.3%) and XY Ring (233.25%). XY Parity Ring, which exploits gate commutativity to optimize gate ordering, demonstrates the least depth increase (171.85%). A similar trend is observed for the total gate count, with XY QAMPA exhibiting the highest increase (194.81%) and XY Parity Ring showing the lowest (116.05%), whereas XY Full and XY Ring have an increase of 157.39% and 125.72%, respectively. The impact on CX gates, the most noise-sensitive operation, is also noteworthy. XY Full mixers incur the most significant increase in CX count (151.86%). XY Ring has the least increase (98.71%), while XY QAMPA and XY Parity Ring show a 101.86% increase. It is important to note that all data includes the gates for preparing the Dicke state. Qiskit is employed for state initialization and transpilation (level 3) into a consistent gate set (CX, R_Z, R_X, R_Y, H) for a fair comparison. While Qiskit's Dicke state preparation might not be optimal, developing strategies for lower two-qubit gate count in XY mixers is valuable.

Figure 8.5 presents simulation results for approximation ratio and success probability under varying noise conditions. In the absence of noise, we observe a performance improvement across all mixers with increasing QAOA depth. XY mixers demonstrably achieve superior approximation ratios and success probabilities compared to the standard mixer. Among XY mixers, XY Full exhibits the highest overall approximation ratio, while XY Parity Ring achieves a lower ratio but a higher success probability. This discrepancy might be attributed to the approximation ratio definition, where the objective value of solutions not satisfying budget constraints are set to 0, leading to not fully reflection of system. We also observe that higher QAOA depth is crutial for

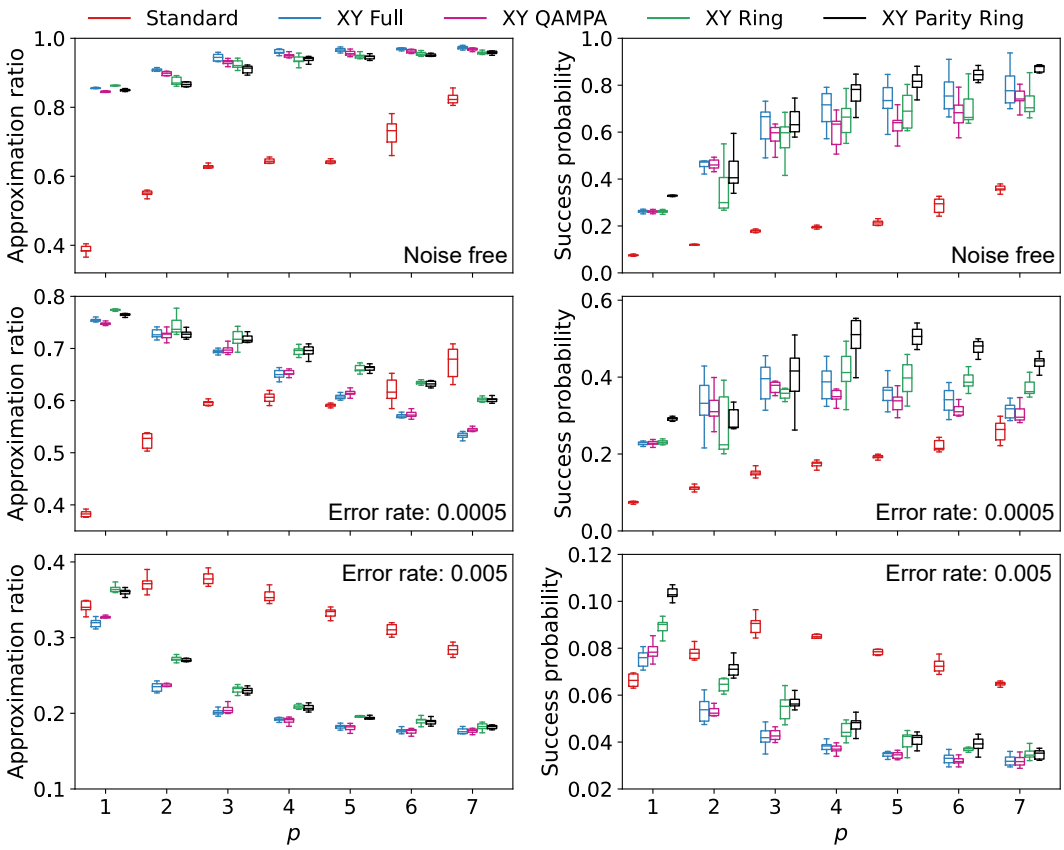

Figure 8.5: Simulation results of approximation ratio and success probability for 5-qubit QAOA with different mixers and depths using Qiskit's Qasm simulator with 8192 shots. The combined bit flip and phase flip error channel is employed.

achieving higher success probability, which increasing from around 0.2 at $p = 1$ to approx 0.9 at $p = 7$ for XY mixers.

When low noise is introduced (error rate 0.0005), XY mixers initially outperform the standard mixer. However, as depth (p) increases, their approximation ratio deteriorates. At $p = 7$, the standard mixer surpasses XY mixers. Nevertheless, XY mixers maintain a higher success probability across all depths. Notably, XY Ring and XY Parity Ring demonstrate higher performance than XY Full and XY QAMPA, especially at higher depths, with XY Parity Ring achieving the highest average success probability. For high noise (error rate 0.005), all mixers exhibit comparable approximation ratios at $p = 1$. For $p \geq 2$, the standard mixer significantly outperforms XY mixers in terms of approximation ratio. The success probability of the standard mixer follows a similar trend, peaking at $p = 3$ before decreasing. Conversely, all XY mixers show a decreasing success probability with increasing depth. However, XY Parity Ring still exhibits the highest success probability at $p = 1$. For $p \geq 2$, the standard mixer outperforms all XY mixers in terms of success probability.

These findings suggest a trade-off between circuit complexity (depth and gate count) and noise resilience for XY mixers. While XY mixers achieve superior per-

formance in noise-free and low-noise scenarios, their advantage diminishes with increasing noise strength. To improve the practicality of XY mixers in noisy environments, future research should focus on developing efficient Dicke state preparation techniques and exploring optimized implementation strategies such as AOQMAP (Chapter 4) and cooptimization with hardware constraints (Chapter 7).

8.3.3 Large-size problem instance

This section investigates the influence of noise on the QAOA with different numbers of qubits. For portfolio optimization problems, QAOA implementations become dense, requiring interactions between each qubit and all others. This translates to a quadratic increase in two-qubit gates with the number of qubits n, following the formula $n(n-1)/2$. To illustrate this effect, we compare 5-qubit and 10-qubit QAOA instances.

Figure 8.6 explores the performance of 5-qubit and 10-qubit QAOA under varying noise strengths using the previously described composed bit-flip and phase-flip error channel. For shallow depths ($p \leq 3$), the 5-qubit QAOA exhibits superior performance compared to the 10-qubit configuration across all noise levels. This is evident in both the approximation ratio and success probability. Notably, at depth one ($p = 1$) the success probability of the 10-qubit QAOA approaches zero. This can be attributed to two factors: (1) limitations of the optimization process at low depths, and (2) inherent limitations of the standard mixer at $p = 1$ in finding optimal solutions, similar to the observations in Figure 8.4 for the 5-qubit case. This highlights the importance of increasing QAOA depth for improved performance, especially with larger qubit numbers.

In the absence of noise, the approximation ratios of 5-qubit and 10-qubit QAOA become comparable for deeper circuits ($p \geq 4$). However, the 10-qubit configuration exhibits a higher success probability, suggesting a potential benefit for using higher depths with large-scale problems. The introduction of noise significantly alters the performance landscape. Even at a low noise strength of 0.01%, the advantage of the 10-qubit configuration diminishes. At a higher noise strength of 0.05%, the 5-qubit QAOA demonstrably outperforms the 10-qubit version across all depths. This underlines the increased susceptibility of more complex 10-qubit circuits to noise. These findings emphasize the need for a careful trade-off between achieving better performance with higher depths and the increased impact of noise in realistic scenarios.

8.4 Conclusions

We analyzed the resilience and practical utility of QOAs for tackling real-world problems. Our findings highlight the crucial role of optimizing parameters in problem instances to achieve improved algorithm performance. Notably, optimal parameters may vary depending on the depth of the QAOA, indicating the potential for depth-specific parameter optimization. Additionally, we can expand upon our previously developed approach to cooptimize hardware and algorithms at the application level, optimizing gate sequences and parameters of problem instances, while also considering hardware connectivity constraints. By exploring problem-specific adaptations and

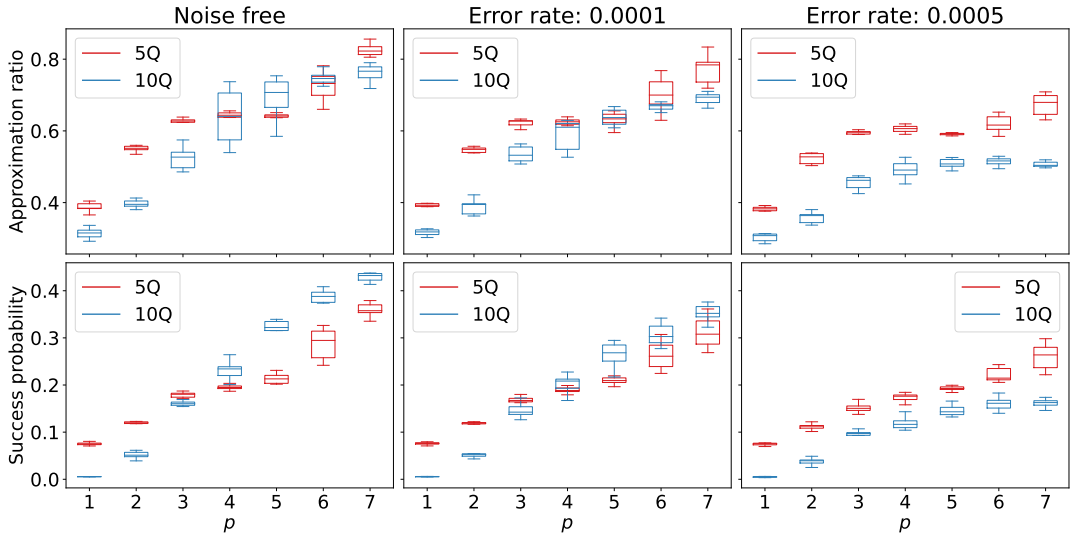

Figure 8.6: Simulation results of 5-qubit (5Q) and 10-qubit (10Q) QAOA with the standerd mixer under varying noise conditions using Qiskit's Qasm simulator with 8192 shots. From left to right: noise free, bit phase flip error with a strength of 0.0001, and bit phase flip error with a strength of 0.0005.

presenting case studies that showcase the efficacy of QOAs in the finance domain, we bridge the gap between theoretical advancements and practical applications, offering valuable insights into the practical implications of quantum optimization in diverse fields.

Chapter 9

Conclusion

This dissertation investigates the resilience of quantum optimization algorithms on near-term noisy quantum devices. We first developed a methodology to quantify resilience by comparing noise-free and noisy executions using metrics such as approximation ratio and success probability. Our findings demonstrate that compilation quality, measurement shots, and classical optimization strategies significantly influence the performance of QOAs. A notable discovery, demonstrated on the real IBM quantum device, is the resilience of QAOA with shallow depths to noise due to the fewer number of gates. This observation suggests that for deeper QAOA circuits, robust compilation techniques, and error mitigation strategies are crucial to fully harness their potential on NISQ devices.

To enhance the performance and resilience of quantum algorithms, we developed various strategies ranging from algorithmic level to hardware pulse level. First, we proposed a calibration-aware transpilation process for VQAs. This process leverages the time-varying error rates of quantum devices and the iterative nature of VQAs to minimize the impact of errors while maximizing the utilization of classical and quantum resources. Demonstrations on various real IBM quantum hardware validate the effectiveness and efficiency of this strategy. Second, we addressed the NP-hard qubit routing problem for VQAs with fully connected two-qubit interactions. Our proposed algorithm-oriented qubit mapping (AOQMAP) approach leverages specific hardware topologies and optimal methods for small-size VQAs to derive scalable and optimal solutions for VQAs with arbitrary numbers of qubits and VQA depths. These solutions minimize the increase in two-qubit gates and can be readily adapted to any quantum device with similar subtopologies, such as Google's quantum hardware. Third, we introduced a framework for optimizing the compilation process of general quantum algorithms. This framework focuses on preparing circuits to address connectivity constraints locally and then selecting high-quality qubits for execution. The noise modeling and cost function design can be further refined to guide the qubit selection process, providing avenues for future research. Guided by the optimized compilation, we investigated selective optimization on bipotent architectures. Our demonstrations on IBM quantum devices show that optimizing a subset of two-qubit gates with higher error rates can significantly improve the performance of VQAs. This finding has important implications for large-scale quantum computing, as it allows for selective optimization of gates to reduce calibration efforts while achieving improved performance.

Furthermore, we explored the factors influencing error mitigation efficiency. While error mitigation is a promising strategy, our exploration of the interplay between error mitigation and circuit design reveals that algorithms with higher original performance benefit less from error mitigation. Achieving significantly improved performance relies more critically on factors such as high-quality native gates of QPUs, symmetric algorithm implementation, and effective circuit optimization techniques. Therefore, a holistic approach that considers both hardware characteristics and software optimization strategies is essential for designing quantum algorithms to maximize reliability and efficacy on NISQ devices. Finally, we proposed a novel cooptimization framework that integrates algorithmic and hardware considerations for enhanced QOA performance. By optimizing algorithm parameters and gate sequences while adhering to hardware connectivity constraints, the performance is improved at the algorithmic level and two-qubit gates are reduced during compilation. This approach leverages algorithmic advantages at the Hamiltonian level and the inherent topology of the hardware, providing insights for cooptimization in other quantum algorithms.

To demonstrate the applicability of QOAs, we outlined the challenges in advancing quantum optimization and introduced problem encoding techniques. Our investigation of problem instance parameters, such as the number of assets, penalty factor, and global factor, reveals that optimal values vary with QAOA depth p, suggesting the need for specific optimization for a fixed p. We also studied the resilience of QOAs with different mixers. XY mixers, while requiring more two-qubit gates, outperform standard mixers but are more noise-sensitive. Moreover, a high value of p is crucial for achieving high success probabilities. Our findings suggest promising strategies for better utilizing near-term devices including optimizing problem instance parameters for a target p and integrating resilience enhancement strategies into the application level, including cooptimizing algorithm design and hardware implementations, synergizing error mitigation and circuit design, and selectively optimizing quantum circuits. By providing theoretical foundations, performance evaluations, optimization techniques, and practical demonstrations of QOAs, this research contributes to the development of robust QOAs for near-term quantum devices, highlighting their potential for practical quantum computing, and providing insights for optimizing other algorithms.

The potential of quantum optimization holds considerable promise for the future; however, several significant challenges should be overcome to fully realize its potential. Developing resilient and scalable quantum devices is paramount for practical applications. Resilience requires improvements in hardware, algorithms, robust error correction and mitigation strategies, and efficient resource management. This involves enhancing qubit coherence time and gate fidelity, designing noise-resilient quantum algorithms specifically tailored for optimization problems, and exploring hybrid quantum-classical approaches. Scalability demands advancements in qubit design, fabrication, innovative architectures, topologies, and control systems. Hybrid quantum devices [329, 330, 331, 332, 333, 334, 335, 336], combining the strengths of different quantum platforms, offer a promising solution to overcome individual limitations. Estimating the quantum advantage for optimization problems remains a complex task. Defining definitive benchmarks and quantifying the advantage over classical algorithms are challenging endeavors. Establishing standardized benchmarks and metrics will facilitate precise assessments of quantum advantage.

Bibliography

[1] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE Computer Society, 1994.

[2] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

[3] Andrew Lucas. Ising formulations of many np problems. *Frontiers in physics*, 2:5, 2014.

[4] Charles Moussa, Henri Calandra, and Vedran Dunjko. To quantum or not to quantum: towards algorithm selection in near-term quantum optimization. *Quantum Science and Technology*, 5(4):044009, 2020.

[5] Román Orús, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4:100028, 2019.

[6] Daniel J. Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yndurain. Quantum computing for finance: State-of-the-art and future prospects. *IEEE Transactions on Quantum Engineering*, 1:1–24, 2020.

[7] Dylan Herman, Cody Googin, Xiaoyuan Liu, Yue Sun, Alexey Galda, Ilya Safro, Marco Pistoia, and Yuri Alexeev. Quantum computing for finance. *Nature Reviews Physics*, 5(8):450–465, 2023.

[8] Bela Bauer, Sergey Bravyi, Mario Motta, and Garnet Kin-Lic Chan. Quantum algorithms for quantum chemistry and quantum materials science. *Chemical Reviews*, 120(22):12685–12717, 2020.

[9] Nathalie P De Leon, Kohei M Itoh, Dohun Kim, Karan K Mehta, Tracy E Northup, Hanhee Paik, BS Palmer, Nitin Samarth, Sorawis Sangtawesin, and David W Steuerman. Materials challenges and opportunities for quantum computing hardware. *Science*, 372(6539):eabb2823, 2021.

[10] Lindsay Bassman Oftelie, Miroslav Urbanek, Mekena Metcalf, Jonathan Carter, Alexander F Kemper, and Wibe A de Jong. Simulating quantum materials with digital quantum computers. *Quantum Science and Technology*, 6(4):043002, 2021.

[11] He Ma, Marco Govoni, and Giulia Galli. Quantum simulations of materials on near-term quantum computers. *npj Computational Materials*, 6(1):85, 2020.

[12] Benjamin P Lanyon, James D Whitfield, Geoff G Gillett, Michael E Goggin, Marcelo P Almeida, Ivan Kassal, Jacob D Biamonte, Masoud Mohseni, Ben J Powell, Marco Barbieri, et al. Towards quantum chemistry on a quantum computer. *Nature chemistry*, 2(2):106–111, 2010.

[13] Ivan Kassal, James D Whitfield, Alejandro Perdomo-Ortiz, Man-Hong Yung, and Alán Aspuru-Guzik. Simulating chemistry using quantum computers. *Annual review of physical chemistry*, 62:185–207, 2011.

[14] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019.

[15] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, 2020.

[16] Alexey Pyrkov, Alex Aliper, Dmitry Bezrukov, Yen-Chu Lin, Daniil Polykovskiy, Petrina Kamya, Feng Ren, and Alex Zhavoronkov. Quantum computing for near-term applications in generative chemistry and drug discovery. *Drug Discovery Today*, page 103675, 2023.

[17] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[18] Daniel Stilck França and Raul Garcia-Patron. Limitations of optimization algorithms on noisy quantum devices. *Nature Physics*, 17(11):1221–1227, 2021.

[19] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[20] Cheng Xue, Zhao-Yun Chen, Yu-Chun Wu, and Guo-Ping Guo. Effects of quantum noise on quantum approximate optimization algorithm. *Chinese Physics Letters*, 38(3):030302, 2021.

[21] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. SIAM, 2019.

[22] Enrico Fontana, Nathan Fitzpatrick, David Muñoz Ramo, Ross Duncan, and Ivan Rungger. Evaluating the noise resilience of variational quantum algorithms. *Phys. Rev. A*, 104:022403, Aug 2021.

[23] Suguru Endo, Simon C. Benjamin, and Ying Li. Practical quantum error mitigation for near-future applications. *Phys. Rev. X*, 8:031027, Jul 2018.

[24] Abhinav Kandala, Kristan Temme, Antonio D Córcoles, Antonio Mezzacapo, Jerry M Chow, and Jay M Gambetta. Error mitigation extends the computational reach of a noisy quantum processor. *Nature*, 567(7749):491–495, 2019.

[25] William J. Huggins, Sam McArdle, Thomas E. O'Brien, Joonho Lee, Nicholas C. Rubin, Sergio Boixo, K. Birgitta Whaley, Ryan Babbush, and Jarrod R. McClean. Virtual distillation for quantum error mitigation. *Phys. Rev. X*, 11:041036, Nov 2021.

[26] Filip B Maciejewski, Flavio Baccari, Zoltán Zimborás, and Michał Oszmaniec. Modeling and mitigation of cross-talk effects in readout noise with applications to the quantum approximate optimization algorithm. *Quantum*, 5:464, 2021.

[27] Ryan LaRose, Andrea Mari, Sarah Kaiser, Peter J Karalekas, Andre A Alves, Piotr Czarnik, Mohamed El Mandouh, Max H Gordon, Yousef Hindy, Aaron Robertson, et al. Mitiq: A software package for error mitigation on noisy quantum computers. *Quantum*, 6:774, 2022.

[28] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.

[29] Daniel Loss and David P. DiVincenzo. Quantum computation with quantum dots. *Phys. Rev. A*, 57:120–126, Jan 1998.

[30] Craig S Lent and P Douglas Tougaw. A device architecture for computing with quantum dots. *Proceedings of the IEEE*, 85(4):541–557, 1997.

[31] Petar Jurcevic, Ali Javadi-Abhari, Lev S Bishop, Isaac Lauer, Daniela F Bogorin, Markus Brink, Lauren Capelluto, Oktay Günlük, Toshinari Itoko, Naoki Kanazawa, et al. Demonstration of quantum volume 64 on a superconducting quantum computing system. *Quantum Science and Technology*, 6(2):025020, 2021.

[32] John Clarke and Frank K Wilhelm. Superconducting quantum bits. *Nature*, 453(7198):1031–1042, 2008.

[33] Michel H Devoret and Robert J Schoelkopf. Superconducting circuits for quantum information: an outlook. *Science*, 339(6124):1169–1174, 2013.

[34] Göran Wendin. Quantum information processing with superconducting circuits: a review. *Reports on Progress in Physics*, 80(10):106001, 2017.

[35] Juan I Cirac and Peter Zoller. Quantum computations with cold trapped ions. *Physical review letters*, 74(20):4091, 1995.

[36] Rainer Blatt and Christian F Roos. Quantum simulations with trapped ions. *Nature Physics*, 8(4):277–284, 2012.

[37] Martin Ringbauer, Michael Meth, Lukas Postler, Roman Stricker, Rainer Blatt, Philipp Schindler, and Thomas Monz. A universal qudit quantum processor with trapped ions. *Nature Physics*, 18(9):1053–1057, 2022.

[38] Anastasiia S. Nikolaeva, Evgeniy O. Kiktenko, and Aleksey K. Fedorov. Universal quantum computing with qubits embedded in trapped-ion qudits. *Phys. Rev. A*, 109:022615, Feb 2024.

[39] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Bell's theorem, quantum theory, and conceptions of the universe, 1989.

[40] Daniel M Greenberger, Michael A Horne, Abner Shimony, and Anton Zeilinger. Bell's theorem without inequalities. *American Journal of Physics*, 58(12):1131–1143, 1990.

[41] Mark W Krentel. The complexity of optimization problems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 69–76, 1986.

[42] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[43] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[44] Ivan Zelinka, Vaclav Snasael, and Ajith Abraham. *Handbook of optimization: from classical to modern approach*, volume 38. Springer Science & Business Media, 2012.

[45] Ali Mohammadi and Farid Sheikholeslam. Intelligent optimization: Literature review and state-of-the-art algorithms (1965–2022). *Engineering Applications of Artificial Intelligence*, 126:106959, 2023.

[46] Joseph SB Mitchell et al. Geometric shortest paths and network optimization. *Handbook of computational geometry*, 334:633–702, 2000.

[47] Ammar W Mohemmed, Nirod Chandra Sahoo, and Tan Kim Geok. Solving shortest path problem using particle swarm optimization. *Applied Soft Computing*, 8(4):1643–1653, 2008.

[48] Andrew Goldberg and Robert Tarjan. Solving minimum-cost flow problems by successive approximation. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 7–18, 1987.

[49] Delbert R Fulkerson. An out-of-kilter method for minimal-cost flow problems. *Journal of the Society for Industrial and Applied Mathematics*, 9(1):18–27, 1961.

[50] Kurt M Bretthauer and Bala Shetty. The nonlinear resource allocation problem. *Operations research*, 43(4):670–683, 1995.

[51] Kibeom Seong, Mehdi Mohseni, and John M Cioffi. Optimal resource allocation for ofdma downlink systems. In *2006 IEEE international symposium on information theory*, pages 1394–1398. IEEE, 2006.

[52] Fangzhe Chang, Jennifer Ren, and Ramesh Viswanathan. Optimal resource allocation in clouds. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 418–425. IEEE, 2010.

[53] Alejandro Ribeiro. Optimal resource allocation in wireless communication and networking. *EURASIP Journal on Wireless Communications and Networking*, 2012:1–19, 2012.

[54] Victor M Preciado, Michael Zargham, Chinwendu Enyioha, Ali Jadbabaie, and George J Pappas. Optimal resource allocation for network protection against spreading processes. *IEEE Transactions on Control of Network Systems*, 1(1):99–108, 2014.

[55] Michael Patriksson. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research*, 185(1):1–46, 2008.

[56] Pascal Halffmann, Luca E Schäfer, Kerstin Dächert, Kathrin Klamroth, and Stefan Ruzika. Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. *Journal of Multi-Criteria Decision Analysis*, 29(5-6):341–363, 2022.

[57] Angel A Juan, Peter Keenan, Rafael Martí, Seán McGarraghy, Javier Panadero, Paula Carroll, and Diego Oliva. A review of the role of heuristics in stochastic optimisation: From metaheuristics to learnheuristics. *Annals of Operations Research*, 320(2):831–861, 2023.

[58] G. Hadley. *Nonlinear and Dynamic Programming*. Addison-Wesley series in management science and economics. Addison-Wesley Publishing Company, 1964.

[59] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[60] Sera Kahruman, Elif Kolotoglu, Sergiy Butenko, and Illya V Hicks. On greedy construction heuristics for the max-cut problem. *International Journal of Computational Science and Engineering*, 3(3):211–218, 2007.

[61] Marc Pirlot. General local search methods. *European journal of operational research*, 92(3):493–511, 1996.

[62] Frank Olken and Doron Rotem. Random sampling from databases: a survey. *Statistics and Computing*, 5:25–42, 1995.

[63] Shagofah Noor, Omid Tajik, and Jawad Golzar. Simple random sampling. *International Journal of Education & Language Studies*, 1(2):78–82, 2022.

[64] Michel Gendreau, Jean-Yves Potvin, et al. *Handbook of metaheuristics*, volume 2. Springer, 2010.

[65] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied soft computing*, 11(6):4135–4151, 2011.

[66] Sigurdur Ólafsson. Metaheuristics. *Handbooks in operations research and management science*, 13:633–654, 2006.

[67] Amir Hossein Gandomi, Xin-She Yang, Siamak Talatahari, and Amir Hossein Alavi. Metaheuristic algorithms in modeling and optimization. *Metaheuristic applications in structures and infrastructures*, 1:1–24, 2013.

[68] Alexander G Nikolaev and Sheldon H Jacobson. Simulated annealing. *Handbook of metaheuristics*, pages 1–39, 2010.

[69] Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau. Simulated annealing: From basics to applications. *Handbook of metaheuristics*, pages 1–35, 2019.

[70] Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

[71] Fred Glover and Manuel Laguna. *Tabu Search*, pages 2093–2229. Springer US, Boston, MA, 1998.

[72] Fred Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[73] Michel Gendreau and Jean-Yves Potvin. Tabu search. *Search methodologies: introductory tutorials in optimization and decision support techniques*, pages 165–186, 2005.

[74] Colin R Reeves. Genetic algorithms. *Handbook of metaheuristics*, pages 109–139, 2010.

[75] Kenneth Sörensen and Fred Glover. Metaheuristics. *Encyclopedia of operations research and management science*, 62:960–970, 2013.

[76] Iman Ahmadianfar, Omid Bozorg-Haddad, and Xuefeng Chu. Gradient-based optimizer: A new metaheuristic optimization algorithm. *Information Sciences*, 540:131–159, 2020.

[77] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[78] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pages 177–186. Springer, 2010.

[79] Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22. Springer Science & Business Media, 2012.

[80] Magnus Rudolph Hestenes, Eduard Stiefel, et al. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.

[81] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[82] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[83] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80:8091–8126, 2021.

[84] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[85] Michael JD Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[86] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[87] Aleta Berk Finnila, Maria A Gomez, C Sebenik, Catherine Stenson, and Jimmie D Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chemical physics letters*, 219(5-6):343–348, 1994.

[88] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[89] Tameem Albash and Daniel A Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018.

[90] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[91] Arnab Das and Bikas K Chakrabarti. Colloquium: Quantum annealing and analog quantum computation. *Reviews of Modern Physics*, 80(3):1061, 2008.

[92] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):1–7, 2014.

[93] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[94] Sebastian Brandhofer, Daniel Braun, Vanessa Dehn, Gerhard Hellstern, Matthias Hüls, Yanjun Ji, Ilia Polian, Amandeep Singh Bhatia, and Thomas Wellens. Benchmarking the performance of portfolio optimization with qaoa. *Quantum Information Processing*, 22(1):25, 2022.

[95] Harry M Markowitz and Harry M Markowitz. *Portfolio selection: efficient diversification of investments*. J. Wiley, 1967.

[96] Jack S. Baker and Santosh Kumar Radha. Wasserstein solution quality and the quantum approximate optimization algorithm: A portfolio optimization case study, 2022.

[97] Mark Hodson, Brendan Ruck, Hugh Ong, David Garvin, and Stefan Dulman. Portfolio rebalancing experiments using the quantum alternating operator ansatz. *arXiv:1911.05296*, 2019.

[98] Jinjing Shi, Yongze Tang, Yuhu Lu, Yanyan Feng, Ronghua Shi, and Shichao Zhang. Quantum circuit learning with parameterized boson sampling. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[99] Jinjing Shi, Wenxuan Wang, Xiaoping Lou, Shichao Zhang, and Xuelong Li. Parameterized hamiltonian learning with quantum circuit. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–10, 2022.

[100] Stuart Hadfield, Zhihui Wang, Bryan O'gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, 2019.

[101] Zhihui Wang, Nicholas C Rubin, Jason M Dominy, and Eleanor G Rieffel. X y mixers: Analytical and numerical results for the quantum alternating operator ansatz. *Physical Review A*, 101(1):012320, 2020.

[102] Robert H Dicke. Coherence in spontaneous radiation processes. *Physical review*, 93(1):99, 1954.

[103] Ryan LaRose, Eleanor Rieffel, and Davide Venturelli. Mixer-phaser ansätze for quantum optimization with hard constraints. *arXiv:2107.06651*, 2021.

[104] Peter W Shor. Fault-tolerant quantum computation. In *Proceedings of 37th conference on foundations of computer science*, pages 56–65. IEEE, 1996.

[105] Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Physical Review A*, 55(2):900, 1997.

[106] John Preskill. Fault-tolerant quantum computation. In *Introduction to quantum computation and information*, pages 213–269. World Scientific, 1998.

[107] Emanuel Knill, Raymond Laflamme, and Wojciech H Zurek. Resilient quantum computation: error models and thresholds. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):365–384, 1998.

[108] Emanuel Knill. Fault-tolerant postselected quantum computation: Threshold analysis. *arXiv preprint quant-ph/0404104*, 2004.

[109] Peter W Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):R2493, 1995.

[110] Andrew M Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77(5):793, 1996.

[111] Leon Ding, Max Hays, Youngkyu Sung, Bharath Kannan, Junyoung An, Agustin Di Paolo, Amir H. Karamlou, Thomas M. Hazard, Kate Azar, David K. Kim, Bethany M. Niedzielski, Alexander Melville, Mollie E. Schwartz, Jonilyn L. Yoder, Terry P. Orlando, Simon Gustavsson, Jeffrey A. Grover, Kyle Serniak, and William D. Oliver. High-fidelity, frequency-flexible two-qubit fluxonium gates with a transmon coupler. *Phys. Rev. X*, 13:031035, Sep 2023.

[112] Morten Kjaergaard, Mollie E Schwartz, Jochen Braumüller, Philip Krantz, Joel I-J Wang, Simon Gustavsson, and William D Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11:369–395, 2020.

[113] Zhiyuan Li, Pei Liu, Peng Zhao, Zhenyu Mi, Huikai Xu, Xuehui Liang, Tang Su, Weijie Sun, Guangming Xue, Jing-Ning Zhang, et al. Error per single-qubit gate below 10- 4 in a superconducting qubit. *npj Quantum Information*, 9(1):111, 2023.

[114] Rajeev Acharya, Laleh Aghababaie-Beni, Igor Aleiner, Trond I Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Nikita Astrakhantsev, Juan Atalaya, et al. Quantum error correction below the surface code threshold. *arXiv preprint arXiv:2408.13687*, 2024.

[115] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas. High-fidelity quantum logic gates using trapped-ion hyperfine qubits. *Phys. Rev. Lett.*, 117:060504, Aug 2016.

[116] Youngseok Kim, Andrew Eddins, Sajant Anand, Ken Xuan Wei, Ewout Van Den Berg, Sami Rosenblatt, Hasan Nayfeh, Yantao Wu, Michael Zaletel, Kristan Temme, et al. Evidence for the utility of quantum computing before fault tolerance. *Nature*, 618(7965):500–505, 2023.

[117] Miha Papič, Adrian Auer, and Inés de Vega. Error sources of quantum gates in superconducting qubits. *arXiv preprint arXiv:2305.08916*, 2023.

[118] Joel Wallman, Chris Granade, Robin Harper, and Steven T Flammia. Estimating the coherence of noise. *New Journal of Physics*, 17(11):113020, 2015.

[119] Noah Kaufmann, Ivan Rojkov, and Florentin Reiter. Characterization of coherent errors in noisy quantum devices. *arXiv preprint arXiv:2307.08741*, 2023.

[120] Mohan Sarovar, Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. Detecting crosstalk errors in quantum information processors. *Quantum*, 4:321, 2020.

[121] Sebastian Krinner, Stefania Lazar, Ants Remm, Christian K Andersen, Nathan Lacroix, Graham J Norris, Christoph Hellings, Mihai Gabureac, Christopher Eichler, and Andreas Wallraff. Benchmarking coherent errors in controlled-phase gates due to spectator qubits. *Physical Review Applied*, 14(2):024042, 2020.

[122] Matthias Steffen and Roger H. Koch. Shaped pulses for quantum computing. *Phys. Rev. A*, 75:062326, Jun 2007.

[123] Ivo S Mihov and Nikolay V Vitanov. Pulse-shape effects in qubit dynamics demonstrated on an ibm quantum computer. *Physical Review A*, 108(4):042604, 2023.

[124] Andrea Pasquale, Stavros Efthymiou, Sergi Ramos-Calderer, Jadwiga Wilkens, Ingo Roth, and Stefano Carrazza. Towards an open-source framework to perform quantum calibration and characterization. *arXiv preprint arXiv:2303.10397*, 2023.

[125] Arnaud Carignan-Dugas, Dar Dahlen, Ian Hincks, Egor Ospadov, Stefanie J Beale, Samuele Ferracin, Joshua Skanes-Norman, Joseph Emerson, and Joel J Wallman. The error reconstruction and compiled calibration of quantum computing cycles. *arXiv preprint arXiv:2303.17714*, 2023.

[126] Chao Fang, Ye Wang, Shilin Huang, Kenneth R Brown, and Jungsang Kim. Crosstalk suppression in individually addressed two-qubit gates in a trapped-ion quantum computer. *Physical Review Letters*, 129(24):240504, 2022.

[127] Irina Heinz and Guido Burkard. Crosstalk analysis for simultaneously driven two-qubit gates in spin qubit arrays. *Phys. Rev. B*, 105:085414, Feb 2022.

[128] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*, volume 2. Cambridge university press Cambridge, 2001.

[129] Zhenyu Cai, Ryan Babbush, Simon C. Benjamin, Suguru Endo, William J. Huggins, Ying Li, Jarrod R. McClean, and Thomas E. O'Brien. Quantum error mitigation. *Rev. Mod. Phys.*, 95:045005, Dec 2023.

[130] Dean Brand, Ilya Sinayskiy, and Francesco Petruccione. Markovian noise modelling and parameter extraction framework for quantum devices. *Scientific Reports*, 14(1):4769, 2024.

[131] Kevin Young, Stephen Bartlett, Robin J Blume-Kohout, John King Gamble, Daniel Lobser, Peter Maunz, Erik Nielsen, Timothy James Proctor, Melissa Revelle, and Kenneth Michael Rudinger. Diagnosing and destroying non-markovian noise. Technical report, 2020.

[132] Abhishek Agarwal, Lachlan P Lindoy, Deep Lall, François Jamet, and Ivan Rungger. Modelling non-markovian noise in driven superconducting qubits. *Quantum Science and Technology*, 2023.

[133] Alireza Shabani and Daniel A Lidar. Completely positive post-markovian master equation via a measurement approach. *Physical Review A*, 71(2):020101, 2005.

[134] Evgeny Mozgunov and Daniel Lidar. Completely positive master equation for arbitrary driving and small level spacing. *Quantum*, 4:227, 2020.

[135] Adrián A. Budini. Post-markovian quantum master equations from classical environment fluctuations. *Phys. Rev. E*, 89:012147, Jan 2014.

[136] Easwar Magesan, Jay M Gambetta, and Joseph Emerson. Characterizing quantum gates via randomized benchmarking. *Physical Review A*, 85(4):042311, 2012.

[137] Easwar Magesan, Jay M Gambetta, Blake R Johnson, Colm A Ryan, Jerry M Chow, Seth T Merkel, Marcus P Da Silva, George A Keefe, Mary B Rothwell, Thomas A Ohki, et al. Efficient measurement of quantum gate error by interleaved randomized benchmarking. *Physical review letters*, 109(8):080505, 2012.

[138] Ali Shaib, Mohamad Hussein Naim, Mohammed E Fouda, Rouwaida Kanj, and Fadi Kurdahi. Efficient noise mitigation technique for quantum computing. *Scientific Reports*, 13(1):3912, 2023.

[139] Christopher J. Wood. Special session: Noise characterization and error mitigation in near-term quantum computers. In *2020 IEEE 38th International Conference on Computer Design (ICCD)*, pages 13–16, 2020.

[140] James Sud, Jeffrey Marshall, Zhihui Wang, Eleanor Rieffel, and Filip A. Wudarski. Dual-map framework for noise characterization of quantum computers. *Phys. Rev. A*, 106:012606, Jul 2022.

[141] Riddhi Swaroop Gupta, Alistair R Milne, Claire L Edmunds, Cornelius Hempel, and Michael J Biercuk. Autonomous adaptive noise characterization in quantum computers. *npj Quantum Information*, 6(1):53, 2020.

[142] MJ Gullans, Miguel Caranti, AR Mills, and JR Petta. Compressed gate characterization for quantum devices with time-correlated noise. *PRX Quantum*, 5(1):010306, 2024.

[143] Stefano Mangini, Marco Cattaneo, Daniel Cavalcanti, Sergei Filippov, Matteo AC Rossi, and Guillermo García-Pérez. Tensor network noise characterization for near-term quantum computers. *arXiv preprint arXiv:2402.08556*, 2024.

[144] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied physics reviews*, 6(2), 2019.

[145] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta. Validating quantum computers using randomized model circuits. *Phys. Rev. A*, 100:032328, Sep 2019.

[146] Andrew Wack, Hanhee Paik, Ali Javadi-Abhari, Petar Jurcevic, Ismael Faro, Jay M Gambetta, and Blake R Johnson. Quality, speed, and scale: three key attributes to measure the performance of near-term quantum computers (2021). *arXiv preprint arXiv:2110.14108*.

[147] He-Liang Huang, Xiao-Yue Xu, Chu Guo, Guojing Tian, Shi-Jie Wei, Xiaoming Sun, Wan-Su Bao, and Gui-Lu Long. Near-term quantum computing techniques: Variational quantum algorithms, error mitigation, circuit compilation, benchmarking and classical simulation. *Science China Physics, Mechanics & Astronomy*, 66(5):250302, 2023.

[148] Sergey Bravyi, Sarah Sheldon, Abhinav Kandala, David C. Mckay, and Jay M. Gambetta. Mitigating measurement errors in multiqubit experiments. *Phys. Rev. A*, 103:042605, Apr 2021.

[149] Michael R Geller and Mingyu Sun. Toward efficient correction of multiqubit measurement errors: pair correlation method. *Quantum Science and Technology*, 6(2):025009, feb 2021.

[150] Paul D. Nation, Hwajung Kang, Neereja Sundaresan, and Jay M. Gambetta. Scalable mitigation of measurement errors on quantum computers. *PRX Quantum*, 2:040326, Nov 2021.

[151] Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn, Paolo Stornati, and Xiaoyang Wang. Measurement error mitigation in quantum computers through classical bit-flip correction. *Phys. Rev. A*, 105:062404, Jun 2022.

[152] Michael R Geller. Rigorous measurement error correction. *Quantum Science and Technology*, 5(3):03LT01, 2020.

[153] Yanzhu Chen, Maziar Farahzad, Shinjae Yoo, and Tzu-Chieh Wei. Detector tomography on ibm quantum computers and mitigation of an imperfect measurement. *Phys. Rev. A*, 100:052315, Nov 2019.

[154] Ewout van den Berg, Zlatko K. Minev, and Kristan Temme. Model-free readout-error mitigation for quantum expectation values. *Phys. Rev. A*, 105:032620, Mar 2022.

[155] Filip B Maciejewski, Zoltán Zimborás, and Michał Oszmaniec. Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography. *Quantum*, 4:257, 2020.

[156] Benjamin Nachman, Miroslav Urbanek, Wibe A de Jong, and Christian W Bauer. Unfolding quantum computer readout noise. *npj Quantum Information*, 6(1):84, 2020.

[157] Rebecca Hicks, Christian W. Bauer, and Benjamin Nachman. Readout rebalancing for near-term quantum computers. *Phys. Rev. A*, 103:022407, Feb 2021.

[158] Evan Peters, Andy C. Y. Li, and Gabriel N. Perdue. Perturbative readout-error mitigation for near-term quantum computers. *Phys. Rev. A*, 107:062426, Jun 2023.

[159] Dieter Suter and Gonzalo A. Álvarez. Colloquium: Protecting quantum information against environmental noise. *Rev. Mod. Phys.*, 88:041001, Oct 2016.

[160] Mustafa Ahmed Ali Ahmed, Gonzalo A. Álvarez, and Dieter Suter. Robustness of dynamical decoupling sequences. *Phys. Rev. A*, 87:042309, Apr 2013.

[161] Bibek Pokharel, Namit Anand, Benjamin Fortman, and Daniel A. Lidar. Demonstration of fidelity improvement using dynamical decoupling with superconducting qubits. *Phys. Rev. Lett.*, 121:220502, Nov 2018.

[162] Alexandre M Souza, Gonzalo A Álvarez, and Dieter Suter. Robust dynamical decoupling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1976):4748–4769, 2012.

[163] Vinay Tripathi, Huo Chen, Mostafa Khezri, Ka-Wa Yip, EM Levenson-Falk, and Daniel A Lidar. Suppression of crosstalk in superconducting qubits using dynamical decoupling. *Physical Review Applied*, 18(2):024068, 2022.

[164] Bram Evert, Zoe Gonzalez Izquierdo, James Sud, Hong-Ye Hu, Shon Grabbe, Eleanor G Rieffel, Matthew J Reagor, and Zhihui Wang. Syncopated dynamical decoupling for suppressing crosstalk in quantum circuits. *arXiv preprint arXiv:2403.07836*, 2024.

[165] Jiawei Qiu, Yuxuan Zhou, Chang-Kang Hu, Jiahao Yuan, Libo Zhang, Ji Chu, Wenhui Huang, Weiyang Liu, Kai Luo, Zhongchu Ni, et al. Suppressing coherent two-qubit errors via dynamical decoupling. *Physical Review Applied*, 16(5):054047, 2021.

[166] H. Y. Carr and E. M. Purcell. Effects of diffusion on free precession in nuclear magnetic resonance experiments. *Phys. Rev.*, 94:630–638, May 1954.

[167] S. Meiboom and D. Gill. Modified Spin-Echo Method for Measuring Nuclear Relaxation Times. *Review of Scientific Instruments*, 29(8):688–691, 12 2004.

[168] A.A Maudsley. Modified carr-purcell-meiboom-gill sequence for nmr fourier imaging applications. *Journal of Magnetic Resonance (1969)*, 69(3):488–491, 1986.

[169] Gonzalo A. Álvarez, Alexandre M. Souza, and Dieter Suter. Iterative rotation scheme for robust dynamical decoupling. *Phys. Rev. A*, 85:052324, May 2012.

[170] Lorenza Viola, Emanuel Knill, and Seth Lloyd. Dynamical decoupling of open quantum systems. *Phys. Rev. Lett.*, 82:2417–2421, Mar 1999.

[171] Alexandre M. Souza, Gonzalo A. Álvarez, and Dieter Suter. Effects of time-reversal symmetry in dynamical decoupling. *Phys. Rev. A*, 85:032306, Mar 2012.

[172] Alexandre M. Souza, Gonzalo A. Álvarez, and Dieter Suter. Robust dynamical decoupling for quantum computing and quantum memory. *Phys. Rev. Lett.*, 106:240501, Jun 2011.

[173] Götz S. Uhrig. Keeping a quantum bit alive by optimized π-pulse sequences. *Phys. Rev. Lett.*, 98:100504, Mar 2007.

[174] G De Lange, ZH Wang, D Riste, VV Dobrovitski, and R Hanson. Universal dynamical decoupling of a single solid-state spin from a spin bath. *Science*, 330(6000):60–63, 2010.

[175] Jiangfeng Du, Xing Rong, Nan Zhao, Ya Wang, Jiahui Yang, and RB Liu. Preserving electron spin coherence in solids by optimal dynamical decoupling. *Nature*, 461(7268):1265–1268, 2009.

[176] D Farfurnik, A Jarmola, LM Pham, ZH Wang, VV Dobrovitski, RL Walsworth, D Budker, and N Bar-Gill. Improving the coherence properties of solid-state spin ensembles via optimized dynamical decoupling. In *Quantum Optics*, volume 9900, pages 111–120. SPIE, 2016.

[177] Demitry Farfurnik, Andrey Jarmola, Linh M Pham, Zhi-Hui Wang, Viatcheslav V Dobrovitski, Ronald L Walsworth, Dmitry Budker, and Nir Bar-Gill. Optimizing a dynamical decoupling protocol for solid-state electronic spin ensembles in diamond. *Physical Review B*, 92(6):060301, 2015.

[178] Benjamin Merkel, Pablo Cova Fariña, and Andreas Reiserer. Dynamical decoupling of spin ensembles with strong anisotropic interactions. *Phys. Rev. Lett.*, 127:030501, Jul 2021.

[179] J Medford, C Barthel, CM Marcus, MP Hanson, AC Gossard, et al. Scaling of dynamical decoupling for spin qubits. *Physical review letters*, 108(8):086802, 2012.

[180] Jonas Bylander, Simon Gustavsson, Fei Yan, Fumiki Yoshihara, Khalil Harrabi, George Fitch, David G Cory, Yasunobu Nakamura, Jaw-Shen Tsai, and William D Oliver. Noise spectroscopy through dynamical decoupling with a superconducting flux qubit. *Nature Physics*, 7(7):565–570, 2011.

[181] Michael J. Biercuk, Hermann Uys, Aaron P. VanDevender, Nobuyasu Shiga, Wayne M. Itano, and John J. Bollinger. Experimental uhrig dynamical decoupling using trapped ions. *Phys. Rev. A*, 79:062324, Jun 2009.

[182] Michael J Biercuk, Hermann Uys, Aaron P VanDevender, Nobuyasu Shiga, Wayne M Itano, and John J Bollinger. Optimized dynamical decoupling in a model quantum memory. *Nature*, 458(7241):996–1000, 2009.

[183] I-Chi Chen, Benjamin Burdick, Yongxin Yao, Peter P. Orth, and Thomas Iadecola. Error-mitigated simulation of quantum many-body scars on quantum computers with pulse-level control. *Phys. Rev. Res.*, 4:043027, Oct 2022.

[184] André Melo, Nathan Earnest-Noble, and Francesco Tacchino. Pulse-efficient quantum machine learning. *Quantum*, 7:1130, 2023.

[185] Ying Li and Simon C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X*, 7:021050, Jun 2017.

[186] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.*, 119:180509, Nov 2017.

[187] Tudor Giurgica-Tiron, Yousef Hindy, Ryan LaRose, Andrea Mari, and William J. Zeng. Digital zero noise extrapolation for quantum error mitigation. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 306–316, 2020.

[188] Shuaining Zhang, Yao Lu, Kuan Zhang, Wentao Chen, Ying Li, Jing-Ning Zhang, and Kihwan Kim. Error-mitigated quantum gates exceeding physical fidelities in a trapped-ion system. *Nature communications*, 11(1):587, 2020.

[189] Andrea Mari, Nathan Shammah, and William J. Zeng. Extending quantum probabilistic error cancellation by noise scaling. *Phys. Rev. A*, 104:052607, Nov 2021.

[190] Changjun Kim, Kyungdeock Daniel Park, and June-Koo Rhee. Quantum error mitigation with artificial neural network. *IEEE Access*, 8:188853–188860, 2020.

[191] Armands Strikis, Dayue Qin, Yanzhu Chen, Simon C. Benjamin, and Ying Li. Learning-based quantum error mitigation. *PRX Quantum*, 2:040330, Nov 2021.

[192] Jihye Kim, Byungdu Oh, Yonuk Chong, Euyheon Hwang, and Daniel K Park. Quantum readout error mitigation via deep learning. *New Journal of Physics*, 24(7):073009, 2022.

[193] Haoran Liao, Derek S Wang, Iskandar Sitdikov, Ciro Salcedo, Alireza Seif, and Zlatko K Minev. Machine learning for practical quantum error mitigation. *arXiv preprint arXiv:2309.17368*, 2023.

[194] Tao Jiang, John Rogers, Marius S Frank, Ove Christiansen, Yong-Xin Yao, and Nicola Lanatà. Error mitigation in variational quantum eigensolvers using tailored probabilistic machine learning. *arXiv preprint arXiv:2111.08814*, 2021.

[195] Elizabeth R Bennewitz, Florian Hopfmueller, Bohdan Kulchytskyy, Juan Carrasquilla, and Pooya Ronagh. Neural error mitigation of near-term quantum simulations. *Nature Machine Intelligence*, 4(7):618–624, 2022.

[196] Tobias Haug, Chris N Self, and MS Kim. Quantum machine learning of large datasets using randomized measurements. *Machine Learning: Science and Technology*, 4(1):015005, 2023.

[197] Stefan H. Sack and Daniel J. Egger. Large-scale quantum approximate optimization on nonplanar graphs with machine learning noise mitigation. *Phys. Rev. Res.*, 6:013223, Mar 2024.

[198] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T Chong, and Margaret Martonosi. Scaffcc: A framework for compilation and analysis of quantum computing programs. In *Proceedings of the 11th ACM Conference on Computing Frontiers*, pages 1–10, 2014.

[199] Damian S Steiger, Thomas Häner, and Matthias Troyer. Projectq: an open source software framework for quantum computing. *Quantum*, 2:49, 2018.

[200] Tyson Jones and Simon C Benjamin. Robust quantum compilation and circuit optimisation via energy minimisation. *Quantum*, 6:628, 2022.

[201] Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I. Schuster, Henry Hoffmann, and Frederic T. Chong. Optimized compilation of aggregated instructions for realistic quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 1031–1044, New York, NY, USA, 2019. Association for Computing Machinery.

[202] Pranav Gokhale, Ali Javadi-Abhari, Nathan Earnest, Yunong Shi, and Frederic T. Chong. Optimized quantum compilation for near-term algorithms with openpulse. In *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020, Athens, Greece, October 17-21, 2020*, pages 186–200. IEEE, 2020.

[203] Davide Ferrari, Angela Sara Cacciapuoti, Michele Amoretti, and Marcello Caleffi. Compiler design for distributed quantum computing. *IEEE Transactions on Quantum Engineering*, 2:1–20, 2021.

[204] Yanjun Ji, Sebastian Brandhofer, and Ilia Polian. Calibration-aware transpilation for variational quantum optimization. In *IEEE International Conference on Quantum Computing and Engineering, QCE 2022, Broomfield, CO, USA, September 18-23, 2022*, pages 204–214. IEEE, 2022.

[205] Ellis Wilson, Sudhakar Singh, and Frank Mueller. Just-in-time quantum circuit transpilation reduces noise. In *2020 IEEE international conference on quantum computing and engineering (QCE)*, pages 345–355. IEEE, 2020.

[206] Ed Younis and Costin Iancu. Quantum circuit optimization and transpilation via parameterized circuit instantiation. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 465–475. IEEE, 2022.

[207] Fei Hua, Meng Wang, Gushu Li, Bo Peng, Chenxu Liu, Muqing Zheng, Samuel Stein, Yufei Ding, Eddy Z Zhang, Travis Humble, et al. Qasmtrans: A qasm quantum transpiler framework for nisq devices. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 1468–1477, 2023.

[208] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.

[209] Cirq Developers. Cirq. *Zenodo*, December 2023.

[210] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t|ket⟩: a retargetable compiler for nisq devices. *Quantum Science and Technology*, 6(1):014003, nov 2020.

[211] Menghan Dou, Tianrui Zou, Yuan Fang, Jing Wang, Dongyi Zhao, Lei Yu, Boying Chen, Wenbo Guo, Ye Li, Zhaoyun Chen, and Guoping Guo. Qpanda: high-performance quantum computing framework for multiple application scenarios. *arXiv preprint arXiv:2212.14201*, 2022.

[212] Nils Quetschlich, Lukas Burgholzer, and Robert Wille. Compiler optimization for quantum computing using reinforcement learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[213] Robert Wille, Lucas Berent, Tobias Forster, Jagatheesan Kunasaikaran, Kevin Mato, Tom Peham, Nils Quetschlich, Damian Rovara, Aaron Sander, Ludwig Schmid, et al. The mqt handbook: A summary of design automation tools and software for quantum computing. *arXiv preprint arXiv:2405.17543*, 2024.

[214] Wojciech Hubert Zurek. Decoherence, einselection, and the quantum origins of the classical. *Rev. Mod. Phys.*, 75:715–775, May 2003.

[215] Jonathan Wurtz and Peter Love. Maxcut quantum approximate optimization algorithm performance guarantees for p > 1. *Phys. Rev. A*, 103:042612, Apr 2021.

[216] Bochen Tan and Jason Cong. Optimal layout synthesis for quantum computing. In *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, November 2-5, 2020*, ICCAD '20, pages 137:1–137:9, New York, NY, USA, 2020. IEEE.

[217] Matthew Amy and Vlad Gheorghiu. staq—a full-stack quantum processing toolkit. *Quantum Science and Technology*, 5(3):034016, 2020.

[218] Matthew P Harrigan, Kevin J Sung, Matthew Neeley, Kevin J Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C Bardin, Rami Barends, Sergio Boixo, et al. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics*, 17(3):332–336, 2021.

[219] Aleks Kissinger and John van de Wetering. PyZX: Large Scale Automated Diagrammatic Reasoning. In Bob Coecke and Matthew Leifer, editors, Proceedings 16th International Conference on *Quantum Physics and Logic*, Chapman University, Orange, CA, USA., 10-14 June 2019, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 229–241. Open Publishing Association, 2020.

[220] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for nisq-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, pages 1001–1014, New York, NY, USA, 2019. Association for Computing Machinery.

[221] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the qubit routing problem. In Wim van Dam and Laura Mancinska, editors, *14th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2019, June 3-5, 2019, University of Maryland, College Park, Maryland, USA*, volume 135 of *LIPIcs*, pages 5:1–5:32, Dagstuhl, Germany, 2019. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[222] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. Qubit allocation. In Jens Knoop, Markus Schordan, Teresa Johnson, and Michael F. P. O'Boyle, editors, *Proceedings of the 2018 International Symposium on Code Generation and Optimization, CGO 2018, Vösendorf / Vienna, Austria, February 24-28, 2018*, CGO 2018, pages 113–125, New York, NY, USA, 2018. ACM.

[223] Debjyoti Bhattacharjee and Anupam Chattopadhyay. Depth-optimal quantum circuit placement for arbitrary topologies. *arXiv preprint arXiv:1703.08540*, 2017.

[224] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Qubit placement to minimize communication overhead in 2d quantum architectures. In *19th Asia and South Pacific Design Automation Conference, ASP-DAC 2014, Singapore, January 20-23, 2014*, pages 495–500. IEEE, 2014.

[225] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, ASPLOS '19, pages 1015–1029. Association for Computing Machinery, 2019.

[226] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Caroline Collange, and Fernando Magno Quintão Pereira. Qubit allocation as a combination of subgraph isomorphism and token swapping. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):1–29, 2019.

[227] Alwin Zulehner, Alexandru Paler, and Robert Wille. An efficient methodology for mapping quantum circuits to the IBM QX architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(7):1226–1236, 2018.

[228] Andrew M. Childs, Eddie Schoute, and Cem M. Unsal. Circuit transformations for quantum architectures. *CoRR*, abs/1902.09102:3:1–3:24, 2019.

[229] Ze-Tong Li, Fan-Xu Meng, Zai-Chen Zhang, and Xu-Tao Yu. Qubits' mapping and routing for NISQ on variability of quantum gates. *Quantum Information Processing*, 19(10):1–25, 2020.

[230] Siyuan Niu, Adrien Suau, Gabriel Staffelbach, and Aida Todri-Sanial. A hardware-aware heuristic for the qubit mapping problem in the NISQ era. *IEEE Transactions on Quantum Engineering*, 1:1–14, 2020.

[231] Ian D Kivlichan, Jarrod McClean, Nathan Wiebe, Craig Gidney, Alán Aspuru-Guzik, Garnet Kin-Lic Chan, and Ryan Babbush. Quantum simulation of electronic structure with linear depth and connectivity. *Physical review letters*, 120(11):110501, 2018.

[232] Bryan O'Gorman, William J Huggins, Eleanor G Rieffel, and K Birgitta Whaley. Generalized swap networks for near-term quantum computing. *arXiv preprint arXiv:1905.05118*, 2019.

[233] Johannes Weidenfeller, Lucia C Valor, Julien Gacon, Caroline Tornow, Luciano Bello, Stefan Woerner, and Daniel J Egger. Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware. *arXiv preprint arXiv:2202.03459*, 6:870, 2022.

[234] Akel Hashim, Rich Rines, Victory Omole, Ravi K. Naik, John Mark Kreikebaum, David I. Santiago, Frederic T. Chong, Irfan Siddiqi, and Pranav Gokhale. Optimized swap networks with equivalent circuit averaging for qaoa. *Phys. Rev. Res.*, 4:033028, Jul 2022.

[235] `https://github.com/QuantumYanjunJi/AOQMAP`.

[236] Paul D. Nation and Matthew Treinish. Suppressing quantum circuit errors due to system variability. *PRX Quantum*, 4:010327, Mar 2023.

[237] Andrew M Childs, Aaron Ostrander, and Yuan Su. Faster quantum simulation by randomization. *Quantum*, 3:182, 2019.

[238] Paul K Faehrmann, Mark Steudtner, Richard Kueng, Maria Kieferova, and Jens Eisert. Randomizing multi-product formulas for hamiltonian simulation. *Quantum*, 6:806, 2022.

[239] Ernst Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 1909(136):210–271, 1909.

[240] D. Leibfried, D. M. Meekhof, B. E. King, C. Monroe, W. M. Itano, and D. J. Wineland. Experimental determination of the motional quantum state of a trapped atom. *Phys. Rev. Lett.*, 77:4281–4285, Nov 1996.

[241] Marcus Cramer, Martin B Plenio, Steven T Flammia, Rolando Somma, David Gross, Stephen D Bartlett, Olivier Landon-Cardinal, David Poulin, and Yi-Kai Liu. Efficient quantum state tomography. *Nature communications*, 1(1):149, 2010.

[242] David Gross, Yi-Kai Liu, Steven T. Flammia, Stephen Becker, and Jens Eisert. Quantum state tomography via compressed sensing. *Phys. Rev. Lett.*, 105:150401, Oct 2010.

[243] Matthias Christandl and Renato Renner. Reliable quantum state tomography. *Phys. Rev. Lett.*, 109:120403, Sep 2012.

[244] J. F. Poyatos, J. I. Cirac, and P. Zoller. Complete characterization of a quantum process: The two-bit quantum gate. *Phys. Rev. Lett.*, 78:390–393, Jan 1997.

[245] M. Mohseni, A. T. Rezakhani, and D. A. Lidar. Quantum-process tomography: Resource analysis of different strategies. *Phys. Rev. A*, 77:032322, Mar 2008.

[246] Easwar Magesan, J. M. Gambetta, and Joseph Emerson. Scalable and robust randomized benchmarking of quantum processes. *Phys. Rev. Lett.*, 106:180504, May 2011.

[247] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. Randomized benchmarking of quantum gates. *Phys. Rev. A*, 77:012307, Jan 2008.

[248] J. Helsen, I. Roth, E. Onorati, A.H. Werner, and J. Eisert. General framework for randomized benchmarking. *PRX Quantum*, 3:020357, Jun 2022.

[249] Megan L. Dahlhauser and Travis S. Humble. Modeling noisy quantum circuits using experimental characterization. *Phys. Rev. A*, 103:042603, Apr 2021.

[250] Yuki Takeuchi, Yasuhiro Takahashi, Tomoyuki Morimae, and Seiichiro Tani. Divide-and-conquer verification method for noisy intermediate-scale quantum computation. *Quantum*, 6:758, July 2022.

[251] Tom Peham, Lukas Burgholzer, and Robert Wille. Equivalence checking of parameterized quantum circuits: Verifying the compilation of variational quantum algorithms. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ASPDAC '23, page 702–708, New York, NY, USA, 2023. Association for Computing Machinery.

[252] Bojan Mohar and Svatopluk Poljak. Eigenvalues and the max-cut problem. *Czechoslovak Mathematical Journal*, 40(2):343–352, 1990.

[253] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H Booth, et al. The variational quantum eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, 2022.

[254] Dmitry A Fedorov, Bo Peng, Niranjan Govind, and Yuri Alexeev. Vqe method: A short survey and recent developments. *Materials Theory*, 6(1):1–21, 2022.

[255] David Amaro, Carlo Modica, Matthias Rosenkranz, Mattia Fiorentini, Marcello Benedetti, and Michael Lubasch. Filtering variational quantum algorithms for combinatorial optimization. *Quantum Science and Technology*, 7(1):015021, 2022.

[256] Jonathan Romero, Ryan Babbush, Jarrod R McClean, Cornelius Hempel, Peter J Love, and Alán Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Science and Technology*, 4(1):014008, 2018.

[257] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[258] Dave Wecker, Matthew B Hastings, and Matthias Troyer. Progress towards practical quantum variational algorithms. *Physical Review A*, 92(4):042303, 2015.

[259] Giacomo Nannicini. Performance of hybrid quantum-classical variational heuristics for combinatorial optimization. *Phys. Rev. E*, 99:013304, Jan 2019.

[260] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[261] Johannes S Otterbach, Riccardo Manenti, Nasser Alidoust, A Bestwick, M Block, B Bloom, S Caldwell, N Didier, E Schuyler Fried, S Hong, et al. Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv:1712.05771*, 2017.

[262] C. Monroe and J. Kim. Scaling the ion trap quantum processor. *Science*, 339(6124):1164–1169, 2013.

[263] I. Pogorelov, T. Feldker, Ch. D. Marciniak, L. Postler, G. Jacob, O. Krieglsteiner, V. Podlesnic, M. Meth, V. Negnevitsky, M. Stadler, B. Höfer, C. Wächter, K. Lakhmanskiy, R. Blatt, P. Schindler, and T. Monz. Compact ion-trap quantum computing demonstrator. *PRX Quantum*, 2:020343, Jun 2021.

[264] Joshua Ramette, Josiah Sinclair, Zachary Vendeiro, Alyssa Rudelis, Marko Cetina, and Vladan Vuletić. Any-to-any connected cavity-mediated architecture for quantum computing with trapped ions or rydberg arrays. *PRX Quantum*, 3:010344, Mar 2022.

[265] M. Cetina, L.N. Egan, C. Noel, M.L. Goldman, D. Biswas, A.R. Risinger, D. Zhu, and C. Monroe. Control of transverse motion for quantum gates on individually addressed atomic qubits. *PRX Quantum*, 3:010334, Mar 2022.

[266] Alberto Politi, Martin J. Cryan, John G. Rarity, Siyuan Yu, and Jeremy L. O'Brien. Silica-on-silicon waveguide quantum circuits. *Science*, 320(5876):646–649, 2008.

[267] Jeremy L O'brien, Akira Furusawa, and Jelena Vučković. Photonic quantum technologies. *Nature Photonics*, 3(12):687–695, 2009.

[268] Jianwei Wang, Fabio Sciarrino, Anthony Laing, and Mark G Thompson. Integrated photonic quantum technologies. *Nature Photonics*, 14(5):273–284, 2020.

[269] W. Luo, Lin Cao, Yuzhi Shi, Lingxiao Wan, Hui Zhang, Shuyi Li, Guanyu Chen, Yuan Li, Sijin Li, Yunxiang Wang, Shihai Sun, Muhammad Faeyz Karim, Hong Cai, Leong Chuan Kwek, and Ai Qun Liu. Recent progress in quantum photonic chips for quantum communication and internet. *Light, Science & Applications*, 12, 2023.

[270] Emanuel Knill, Raymond Laflamme, and Gerald J Milburn. A scheme for efficient quantum computation with linear optics. *nature*, 409(6816):46–52, 2001.

[271] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001.

[272] Philip Walther, Kevin J Resch, Terry Rudolph, Emmanuel Schenck, Harald Weinfurter, Vlatko Vedral, Markus Aspelmeyer, and Anton Zeilinger. Experimental one-way quantum computing. *Nature*, 434(7030):169–176, 2005.

[273] Nicolas C. Menicucci, Steven T. Flammia, and Olivier Pfister. One-way quantum computing in the optical frequency comb. *Phys. Rev. Lett.*, 101:130501, Sep 2008.

[274] Felix Zilk, Korbinian Staudacher, Tobias Guggemos, Karl Fürlinger, Dieter Kranzlmüller, and Philip Walther. A compiler for universal photonic quantum computers. In *2022 IEEE/ACM Third International Workshop on Quantum Computing Software (QCS)*, pages 57–67. IEEE, 2022.

[275] Hezi Zhang, Anbang Wu, Yuke Wang, Gushu Li, Hassan Shapourian, Alireza Shabani, and Yufei Ding. Oneq: A compilation framework for photonic one-way quantum computation. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[276] Hezi Zhang, Jixuan Ruan, Hassan Shapourian, Ramana Rao Kompella, and Yufei Ding. Oneperc: A randomness-aware compiler for photonic quantum computing. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 738–754, New York, NY, USA, 2024. Association for Computing Machinery.

[277] Iris Agresti, Koushik Paul, Peter Schiansky, Simon Steiner, Zhengao Yin, Ciro Pentangelo, Simone Piacentini, Andrea Crespi, Yue Ban, Francesco Ceccarelli, et al. Demonstration of hardware efficient photonic variational quantum algorithm. *arXiv preprint arXiv:2408.10339*, 2024.

[278] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Phys. Rev. A*, 98:032309, Sep 2018.

[279] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, nov 2019.

[280] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Phys. Rev. A*, 101:032308, Mar 2020.

[281] Johannes Jakob Meyer, Marian Mularski, Elies Gil-Fuster, Antonio Anna Mele, Francesco Arzani, Alissa Wilms, and Jens Eisert. Exploiting symmetry in variational quantum machine learning. *PRX Quantum*, 4:010328, Mar 2023.

[282] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[283] Konstantinos Georgopoulos, Clive Emary, and Paolo Zuliani. Modeling and simulating the noisy behavior of near-term quantum computers. *Physical Review A*, 104(6):062432, Dec 2021.

[284] Jinfeng Zeng, Zipeng Wu, Chenfeng Cao, Chao Zhang, Shi-Yao Hou, Pengxiang Xu, and Bei Zeng. Simulating noisy variational quantum eigensolver with local noise models. *Quantum Engineering*, 3(4):e77, 2021.

[285] Yanjun Ji, Kathrin F. Koenig, and Ilia Polian. Optimizing quantum algorithms on bipotent architectures. *Phys. Rev. A*, 108:022610, Aug 2023.

[286] Nathan Earnest, Caroline Tornow, and Daniel J. Egger. Pulse-efficient circuit transpilation for quantum applications on cross-resonance-based hardware. *Phys. Rev. Res.*, 3:043088, Oct 2021.

[287] Kaitlin N. Smith, Gokul Subramanian Ravi, Thomas Alexander, Nicholas T. Bronn, Andre Carvalho, Alba Cervera-Lierta, Frederic T. Chong, Jerry M. Chow, Michael Cubeddu, Akel Hashim, Liang Jiang, Olivia Lanes, Matthew J. Otten, David I. Schuster, Pranav Gokhale, Nathan Earnest, and Alexey Galda. Summary: Chicago quantum exchange (cqe) pulse-level quantum control workshop, 2022.

[288] Navin Khaneja, Timo Reiss, Cindie Kehlet, Thomas Schulte-Herbrüggen, and Steffen J Glaser. Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms. *Journal of magnetic resonance*, 172(2):296–305, 2005.

[289] Yanjun Ji. Modellierung der messung von flussqubits mittels quanten-fluss-parametron. *arXiv preprint arXiv:2301.11870*, 2022.

[290] Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I. Schuster, Henry Hoffmann, and Frederic T. Chong. Partial compilation of variational algorithms for noisy intermediate-scale quantum machines. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, page 266–278, New York, NY, USA, 2019. Association for Computing Machinery.

[291] Pranav Gokhale, Teague Tomesh, Martin Suchara, and Frederic T Chong. Faster and more reliable quantum swaps via native gates. *arXiv preprint arXiv:2109.13199*, 2021.

[292] Zhiding Liang, Zhixin Song, Jinglei Cheng, Zichang He, Ji Liu, Hanrui Wang, Ruiyang Qin, Yiru Wang, Song Han, Xuehai Qian, et al. Hybrid gate-pulse model for variational quantum algorithms. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[293] Daniel J. Egger, Chiara Capecci, Bibek Pokharel, Panagiotis Kl. Barkoutsos, Laurin E. Fischer, Leonardo Guidoni, and Ivano Tavernelli. A study of the pulse-based variational quantum eigensolver on cross-resonance based hardware, 2023.

[294] Eric C. Peterson, Lev S. Bishop, and Ali Javadi-Abhari. Optimal synthesis into fixed XX interactions. *Quantum*, 6:696, April 2022.

[295] Akel Hashim, Rich Rines, Victory Omole, Ravi K. Naik, John Mark Kreikebaum, David I. Santiago, Frederic T. Chong, Irfan Siddiqi, and Pranav Gokhale. Optimized swap networks with equivalent circuit averaging for qaoa. *Phys. Rev. Res.*, 4:033028, Jul 2022.

[296] Lingling Lao, Prakash Murali, Margaret Martonosi, and Dan Browne. Designing calibration and expressivity-efficient instruction sets for quantum computing. In *Proceedings of the 48th Annual International Symposium on Computer Architecture*, ISCA '21, page 846–859. IEEE, 2021.

[297] David C. McKay, Stefan Filipp, Antonio Mezzacapo, Easwar Magesan, Jerry M. Chow, and Jay M. Gambetta. Universal gate for fixed-frequency qubits via a tunable bus. *Phys. Rev. Appl.*, 6:064007, Dec 2016.

[298] M. Ganzhorn, G. Salis, D. J. Egger, A. Fuhrer, M. Mergenthaler, C. Müller, P. Müller, S. Paredes, M. Pechal, M. Werninghaus, and S. Filipp. Benchmarking the noise sensitivity of different parametric two-qubit gates in a single superconducting quantum computing platform. *Phys. Rev. Res.*, 2:033447, Sep 2020.

[299] Youngkyu Sung, Leon Ding, Jochen Braumüller, Antti Vepsäläinen, Bharath Kannan, Morten Kjaergaard, Ami Greene, Gabriel O. Samach, Chris McNally, David Kim, Alexander Melville, Bethany M. Niedzielski, Mollie E. Schwartz, Jonilyn L. Yoder, Terry P. Orlando, Simon Gustavsson, and William D. Oliver. Realization of high-fidelity cz and zz-free iswap gates with a tunable coupler. *Phys. Rev. X*, 11:021058, Jun 2021.

[300] Bibek Pokharel, Namit Anand, Benjamin Fortman, and Daniel A. Lidar. Demonstration of fidelity improvement using dynamical decoupling with superconducting qubits. *Phys. Rev. Lett.*, 121:220502, Nov 2018.

[301] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.*, 119:180509, Nov 2017.

[302] Lorenza Viola and Seth Lloyd. Dynamical suppression of decoherence in two-state quantum systems. *Phys. Rev. A*, 58:2733–2744, Oct 1998.

[303] Yanjun Ji and Ilia Polian. Synergistic dynamical decoupling and circuit design for enhanced algorithm performance on near-term quantum devices. *Entropy*, 26(7), 2024.

[304] Yanjun Ji, Kathrin F. Koenig, and Ilia Polian. Improving the performance of digitized counterdiabatic quantum optimization via algorithm-oriented qubit mapping. *Phys. Rev. A*, 110:032421, Sep 2024.

[305] Masuo Suzuki. Generalized trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Communications in Mathematical Physics*, 51(2):183–190, 1976.

[306] Masuo Suzuki. Decomposition formulas of exponential operators and lie exponentials with some applications to quantum mechanics and statistical physics. *Journal of mathematical physics*, 26(4):601–612, 1985.

[307] Masuo Suzuki. Relationship between d-dimensional quantal spin systems and (d+ 1)-dimensional ising systems: Equivalence, critical exponents and systematic approximants of the partition function and spin correlations. *Progress of theoretical physics*, 56(5):1454–1469, 1976.

[308] Michael Kolodrubetz, Dries Sels, Pankaj Mehta, and Anatoli Polkovnikov. Geometry and non-adiabatic response in quantum and classical systems. *Physics Reports*, 697:1–87, 2017. Geometry and non-adiabatic response in quantum and classical systems.

[309] P. Chandarana, N. N. Hegade, K. Paul, F. Albarrán-Arriagada, E. Solano, A. del Campo, and Xi Chen. Digitized-counterdiabatic quantum approximate optimization algorithm. *Phys. Rev. Res.*, 4:013141, Feb 2022.

[310] Yahui Chai, Yong-Jian Han, Yu-Chun Wu, Ye Li, Menghan Dou, and Guo-Ping Guo. Shortcuts to the quantum approximate optimization algorithm. *Phys. Rev. A*, 105:042415, Apr 2022.

[311] Andrew M. Childs, Yuan Su, Minh C. Tran, Nathan Wiebe, and Shuchen Zhu. Theory of trotter error with commutator scaling. *Phys. Rev. X*, 11:011020, Feb 2021.

[312] Minh C. Tran, Yuan Su, Daniel Carney, and Jacob M. Taylor. Faster digital quantum simulation by symmetry protection. *PRX Quantum*, 2:010323, Feb 2021.

[313] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to variational quantum optimization from symmetry protection. *Phys. Rev. Lett.*, 125:260505, Dec 2020.

[314] Chufan Lyu, Xusheng Xu, Man-Hong Yung, and Abolfazl Bayat. Symmetry enhanced variational quantum spin eigensolver. *Quantum*, 7:899, 2023.

[315] Jin-Guo Liu, Yi-Hong Zhang, Yuan Wan, and Lei Wang. Variational quantum eigensolver with fewer qubits. *Phys. Rev. Res.*, 1:023025, Sep 2019.

[316] N. N. Hegade, P. Chandarana, K. Paul, Xi Chen, F. Albarrán-Arriagada, and E. Solano. Portfolio optimization with digitized counterdiabatic quantum algorithms. *Phys. Rev. Res.*, 4:043204, Dec 2022.

[317] Lingling Lao and Dan E Browne. 2qan: A quantum compiler for 2-local qubit hamiltonian simulation algorithms. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 351–365, 2022.

[318] Andrew Arrasmith, Marco Cerezo, Piotr Czarnik, Lukasz Cincio, and Patrick J Coles. Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558, 2021.

[319] Martin Larocca, Supanut Thanasilp, Samson Wang, Kunal Sharma, Jacob Biamonte, Patrick J Coles, Lukasz Cincio, Jarrod R McClean, Zoë Holmes, and M Cerezo. A review of barren plateaus in variational quantum computing. *arXiv preprint arXiv:2405.00781*, 2024.

[320] Dominik Hangleiter, Martin Kliesch, Matthias Schwarz, and Jens Eisert. Direct certification of a class of quantum simulations. *Quantum Science and Technology*, 2(1):015004, 2017.

[321] Alexandru Gheorghiu, Theodoros Kapourniotis, and Elham Kashefi. Verification of quantum computation: An overview of existing approaches. *Theory of computing systems*, 63:715–808, 2019.

[322] Karla L Hoffman, Manfred Padberg, Giovanni Rinaldi, et al. Traveling salesman problem. *Encyclopedia of operations research and management science*, 1:1573–1578, 2013.

[323] Etienne Barnard et al. Optimization for training neural nets. *IEEE transactions on Neural Networks*, 3(2):232–240, 1992.

[324] Alessia Mammone, Marco Turchi, and Nello Cristianini. Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):283–289, 2009.

[325] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, 7:193–209, 2008.

[326] Giuseppe Buonaiuto, Francesco Gargiulo, Giuseppe De Pietro, Massimo Esposito, and Marco Pota. Best practices for portfolio optimization by quantum computing, experimented on real quantum devices. *Scientific Reports*, 13(1):19434, 2023.

[327] Dylan Herman, Ruslan Shaydulin, Yue Sun, Shouvanik Chakrabarti, Shaohan Hu, Pierre Minssen, Arthur Rattew, Romina Yalovetzky, and Marco Pistoia. Constrained optimization via quantum zeno dynamics. *Communications Physics*, 6(1):219, 2023.

[328] Federico Dominguez, Josua Unger, Matthias Traube, Barry Mant, Christian Ertler, and Wolfgang Lechner. Encoding-independent optimization problem formulation for quantum computing. *Frontiers in Quantum Science and Technology*, 2:1229471, 2023.

[329] P Rabl, D DeMille, John M Doyle, Mikhail D Lukin, RJ Schoelkopf, and P Zoller. Hybrid quantum processors: molecular ensembles as quantum memory for solid state circuits. *Physical review letters*, 97(3):033003, 2006.

[330] Margareta Wallquist, Klemens Hammerer, Peter Rabl, Mikhail Lukin, and Peter Zoller. Hybrid quantum devices and quantum engineering. *Physica Scripta*, 2009(T137):014001, 2009.

[331] Ze-Liang Xiang, Sahel Ashhab, J. Q. You, and Franco Nori. Hybrid quantum circuits: Superconducting circuits interacting with other quantum systems. *Rev. Mod. Phys.*, 85:623–653, Apr 2013.

[332] Philipp Treutlein, Claudiu Genes, Klemens Hammerer, Martino Poggio, and Peter Rabl. Hybrid mechanical systems. *Cavity optomechanics: nano-and micromechanical resonators interacting with light*, pages 327–351, 2014.

[333] Gershon Kurizki, Patrice Bertet, Yuimaru Kubo, Klaus Mølmer, David Petrosyan, Peter Rabl, and Jörg Schmiedmayer. Quantum technologies with hybrid systems. *Proceedings of the National Academy of Sciences*, 112(13):3866–3873, 2015.

[334] Peng-Bo Li, Yong-Chun Liu, S-Y Gao, Ze-Liang Xiang, Peter Rabl, Yun-Feng Xiao, and Fu-Li Li. Hybrid quantum device based on nv centers in diamond nanomechanical resonators plus superconducting waveguide cavities. *Physical Review Applied*, 4(4):044003, 2015.

[335] Peng-Bo Li, Ze-Liang Xiang, Peter Rabl, and Franco Nori. Hybrid quantum device with nitrogen-vacancy centers in diamond coupled to carbon nanotubes. *Physical review letters*, 117(1):015502, 2016.

[336] AA Clerk, KW Lehnert, P Bertet, JR Petta, and Y Nakamura. Hybrid quantum systems with circuit quantum electrodynamics. *Nature Physics*, 16(3):257–267, 2020.

Appendix A

Appendix

A.1 Cloud platform details for Chapter 5

We provide here more details about the two cloud-accessible QPUs ibmq_ehningen and ibm_auckland used for the demonstration presented in Chapter 5. Both QPUs have identical layouts, as shown in Figures 5.1 and 5.12, and employ a basis gate set of {ID, R_Z, SX, X, CX}, where ID represents the identity gate, X represents the Pauli X gate, and SX represents the square root of the X gate. Calibration data at the time of the demonstration of QAOA for PortOpt and MaxCut are summarized in Tables A.1 and A.2, respectively.

A.2 Cloud platform details for Chapter 7

Here we provide details on the IBM QPUs employed in Chapter 7, including ibm_perth with 7 qubits, ibm_cairo, and ibmq_ehningen, both with 27 qubits. Figure A.1 shows the topologies of 7-qubit and 27-qubit QPUs. All QPUs employ a basis gate set of {ID, R_Z, SX, X, CX}. The calibration data at the time of the demonstration are summarized in the following tables. In particular, Tables A.3 and A.4 present the qubits employed in the implementation of DC-QAOA for MaxCut problem on ibm_perth, along with their corresponding qubit properties. Similarly, Tables A.5 and A.6 provide the same information for the MaxCut problem instances executed on ibmq_ehningen. Furthermore, Tables A.7 and A.8 present the qubits utilized and their properties in the implementation of DC-QAOA for portfolio optimization on ibm_cairo, covering depths ranging from one to three. Finally, Tables A.9 and A.10 display the corresponding details for the portfolio optimization instances executed on ibmq_ehningen.

3Q-QAOA-PortOpt	ECR-circuit	Global-D	Bipotent-circuit
Qubits used	[16, 14, 11]	[18, 21, 23]	[22, 25, 26]
T_1 (µs)	[194.12, 163.4, 150.43]	[214.69, 300.02, 222.14]	[195.66, 257.93, 195.59]
T_2 (µs)	[240.94, 215.18, 174.7]	[333.28, 155.0, 234.21]	[31.47, 354.96, 27.85]
Frequency (GHz)	[5.022, 5.177, 5.119]	[4.996, 4.94, 4.805]	[4.725, 4.95, 5.151]
Anharmonicity (GHz)	[-0.3435, -0.3408, -0.3405]	[-0.343, -0.3456, -0.3471]	[-0.3464, -0.3457, -0.3391]
Prob meas0 prep1 (%)	[0.64, 0.92, 1.66]	[1.14, 1.0, 0.84]	[1.4, 1.16, 0.8]
Prob meas1 prep0 (%)	[0.56, 0.68, 1.4]	[0.86, 0.78, 0.76]	[1.12, 0.66, 0.66]
Readout length (ns)	[846.22, 846.22, 846.22]	[846.22, 846.22, 846.22]	[846.22, 846.22, 846.22]
Readout error (%)	[0.6, 0.8, 1.53]	[1.0, 0.89, 0.8]	[1.26, 0.91, 0.73]
Single-qubit gate error (%)	[0.016, 0.033, 0.022]	[0.026, 0.021, 0.013]	[0.016, 0.024, 0.014]
CX gate error (%)	[0.66, 0.89]	[0.44, 0.52]	[0.62, 0.57]
4Q-QAOA-PortOpt	ECR-circuit	Global-B	Direct-circuit
Qubits used	[16, 14, 11, 8]	[18, 21, 23, 24]	[1, 4, 7, 6]
T_1 (µs)	[194.12, 163.4, 150.43, 145.9]	[214.69, 300.02, 222.14, 244.38]	[243.57, 134.77, 205.84, 105.37]
T_2 (µs)	[240.94, 215.18, 174.7, 157.35]	[333.28, 155.0, 234.21, 225.4]	[172.38, 111.81, 266.71, 183.16]
Frequency (GHz)	[5.022, 5.177, 5.119, 5.174]	[4.996, 4.94, 4.805, 5.074]	[5.182, 5.054, 4.978, 4.89]
Anharmonicity (GHz)	[-0.3435, -0.3408, -0.3405, -0.3399]	[-0.343, -0.3456, -0.3471, -0.3416]	[-0.34, -0.3426, -0.344, -0.3448]
Prob meas0 prep1 (%)	[0.64, 0.92, 1.66, 1.18]	[1.14, 1.0, 0.84, 0.88]	[1.04, 1.02, 0.74, 1.48]
Prob meas1 prep0 (%)	[0.56, 0.68, 1.4, 1.46]	[0.86, 0.78, 0.76, 0.66]	[0.64, 0.58, 0.82, 2.02]
Readout length (ns)	[846.22, 846.22, 846.22, 846.22]	[846.22, 846.22, 846.22, 846.22]	[846.22, 846.22, 846.22, 846.22]
Readout error (%)	[0.6, 0.8, 1.53, 1.32]	[1.0, 0.89, 0.8, 0.77]	[0.84, 0.8, 0.78, 1.75]
Single-qubit gate error (%)	[0.016, 0.033, 0.022, 0.029]	[0.026, 0.021, 0.013, 0.017]	[0.026, 0.025, 0.016, 0.02]
CX gate error (%)	[0.66, 0.89, 1.06]	[0.44, 0.52, 0.49]	[0.68, 0.4, 0.35]
5Q-QAOA-PortOpt	ECR-circuit	Global-B	
Qubits used	[16, 14, 11, 8, 9]	[17, 18, 21, 23, 24]	
T_1 (µs)	[194.12, 163.4, 150.43, 145.9, 178.39]	[143.0, 214.69, 300.02, 222.14, 244.38]	
T_2 (µs)	[240.94, 215.18, 174.7, 157.35, 195.4]	[39.76, 333.28, 155.0, 234.21, 225.4]	
Frequency (GHz)	[5.022, 5.177, 5.119, 5.174, 4.993]	[5.136, 4.996, 4.94, 4.805, 5.074]	
Anharmonicity (GHz)	[-0.3435, -0.3408, -0.3405, -0.3399, -0.3441]	[-0.341, -0.343, -0.3456, -0.3471, -0.3416]	
Prob meas0 prep1 (%)	[0.64, 0.92, 1.66, 1.18, 1.0]	[0.88, 1.14, 1.0, 0.84, 0.88]	
Prob meas1 prep0 (%)	[0.56, 0.68, 1.4, 1.46, 0.74]	[0.56, 0.86, 0.78, 0.76, 0.66]	
Readout length (ns)	[846.22, 846.22, 846.22, 846.22, 846.22]	[846.22, 846.22, 846.22, 846.22, 846.22]	
Readout error (%)	[0.6, 0.8, 1.53, 1.32, 0.87]	[0.72, 1.0, 0.89, 0.8, 0.77]	
Single-qubit gate error (%)	[0.016, 0.033, 0.022, 0.029, 0.02]	[0.018, 0.026, 0.021, 0.013, 0.017]	
CX gate error (%)	[0.66, 0.89, 1.06, 0.94]	[0.47, 0.44, 0.52, 0.49]	
5Q-QAOA-PortOpt	Direct-circuit	Bipotent-circuit	
Qubits used	[6, 7, 4, 1, 2]	[11, 14, 13, 12, 10]	
T_1 (µs)	[42.89, 148.29, 108.74, 154.56, 75.75]	[127.56, 107.65, 147.9, 183.56, 115.74]	
T_2 (µs)	[183.16, 266.71, 111.81, 172.38, 19.75]	[74.74, 196.1, 271.77, 442.33, 57.65]	
Frequency (GHz)	[4.89, 4.978, 5.054, 5.182, 5.127]	[5.119, 5.177, 4.926, 4.725, 4.835]	
Anharmonicity (GHz)	[-0.3448, -0.344, -0.3426, -0.34, -0.3403]	[-0.3405, -0.3408, -0.344, -0.3484, -0.3471]	
Prob meas0 prep1 (%)	[1.48, 0.74, 1.02, 1.04, 1.04]	[1.54, 1.08, 1.74, 1.0, 0.74]	
Prob meas1 prep0 (%)	[2.02, 0.82, 0.58, 0.64, 0.66]	[1.34, 0.44, 0.96, 0.9, 0.46]	
Readout length (ns)	[846.22, 846.22, 846.22, 846.22, 846.22]	[846.22, 846.22, 846.22, 846.22, 846.22]	
Readout error (%)	[1.75, 0.78, 0.8, 0.84, 0.85]	[1.44, 0.76, 1.35, 0.95, 0.6]	
Single-qubit gate error (%)	[0.02, 0.016, 0.025, 0.026, 0.05]	[0.025, 0.031, 0.018, 0.022, 0.017]	
CX gate error (%)	[0.35, 0.4, 0.68, 0.96]	[0.84, 0.59, 0.66, 0.5]	
5Q-QAOA-PortOpt	Bipotent-circuit (ibm_auckland)		
Qubits used	[11, 8, 5, 3, 2]		
T_1 (µs)	[168.1, 177.64, 280.77, 157.44, 126.14]		
T_2 (µs)	[138.88, 97.6, 96.99, 219.48, 196.01]		
Frequency (GHz)	[5.055, 5.204, 4.993, 4.897, 5.006]		
Anharmonicity (GHz)	[-0.3422, -0.3407, -0.3445, -0.3455, -0.3434]		
Prob meas0 prep1 (%)	[0.84, 1.08, 1.14, 1.48, 1.28]		
Prob meas1 prep0 (%)	[0.5, 0.7, 0.74, 0.84, 0.98]		
Readout length (ns)	[757.33, 757.33, 757.33, 757.33, 757.33]		
Readout error (%)	[0.67, 0.89, 0.94, 1.16, 1.13]		
Single-qubit gate error (%)	[0.017, 0.019, 0.049, 0.016, 0.018]		
CX gate error (%)	[0.63, 0.98, 0.59, 0.49]		

Table A.1: Calibration data at the time of the demonstration for portfolio optimization problem presented in Section 5.3.3.

Table A.2: Calibration data at the time of the demonstration for MaxCut problem presented in Section 5.3.3.

5Q-QAOA-MaxCut	ECR-circuit	Global-B
Qubits used	[16, 14, 11, 8, 9]	[6, 7, 4, 1, 0]
T_1 (µs)	[197.21, 207.15, 104.89, 122.12, 135.3]	[186.94, 217.29, 129.35, 143.59, 197.13]
T_2 (µs)	[240.94, 215.18, 174.7, 157.35, 195.4]	[183.16, 266.71, 111.81, 172.38, 201.31]
Frequency (GHz)	[5.022, 5.177, 5.119, 5.174, 4.993]	[4.89, 4.978, 5.054, 5.182, 4.961]
Anharmonicity (GHz)	[-0.3435, -0.3408, -0.3405, -0.3399, -0.3441]	[-0.3448, -0.344, -0.3426, -0.34, -0.3445]
Prob meas0 prep1 (%)	[0.64, 0.92, 1.66, 1.18, 1.0]	[1.48, 0.74, 1.02, 1.04, 1.04]
Prob meas1 prep0 (%)	[0.56, 0.68, 1.4, 1.46, 0.74]	[2.02, 0.82, 0.58, 0.64, 0.88]
Readout length (ns)	[846.22, 846.22, 846.22, 846.22, 846.22]	[846.22, 846.22, 846.22, 846.22, 846.22]
Readout error (%)	[0.6, 0.8, 1.53, 1.32, 0.87]	[1.75, 0.78, 0.8, 0.84, 0.96]
Single-qubit gate error (%)	[0.016, 0.033, 0.022, 0.029, 0.02]	[0.02, 0.016, 0.025, 0.026, 0.018]
CX gate error (%)	[0.66, 0.89, 1.06, 0.94]	[0.35, 0.4, 0.68, 0.51]
5Q-QAOA-MaxCut	Direct-circuit	Bipotent-circuit
Qubits used	[6, 7, 4, 1, 2]	[17, 18, 21, 23, 24]
T_1 (µs)	[42.89, 148.29, 108.74, 154.56, 75.75]	[143.0, 214.69, 300.02, 222.14, 244.38]
T_2 (µs)	[183.16, 266.71, 111.81, 172.38, 19.75]	[39.76, 333.28, 155.0, 234.21, 225.4]
Frequency (GHz)	[4.89, 4.978, 5.054, 5.182, 5.127]	[5.136, 4.996, 4.94, 4.805, 5.074]
Anharmonicity (GHz)	[-0.3448, -0.344, -0.3426, -0.34, -0.3403]	[-0.341, -0.343, -0.3456, -0.3471, -0.3416]
Prob meas0 prep1 (%)	[1.48, 0.74, 1.02, 1.04, 1.04]	[0.88, 1.14, 1.0, 0.84, 0.88]
Prob meas1 prep0 (%)	[2.02, 0.82, 0.58, 0.64, 0.66]	[0.56, 0.86, 0.78, 0.76, 0.66]
Readout length (ns)	[846.22, 846.22, 846.22, 846.22, 846.22]	[846.22, 846.22, 846.22, 846.22, 846.22]
Readout error (%)	[1.75, 0.78, 0.8, 0.84, 0.85]	[0.72, 1.0, 0.89, 0.8, 0.77]
Single-qubit gate error (%)	[0.02, 0.016, 0.025, 0.026, 0.05]	[0.018, 0.026, 0.021, 0.013, 0.017]
CX gate error (%)	[0.35, 0.4, 0.68, 0.96]	[0.47, 0.44, 0.52, 0.49]

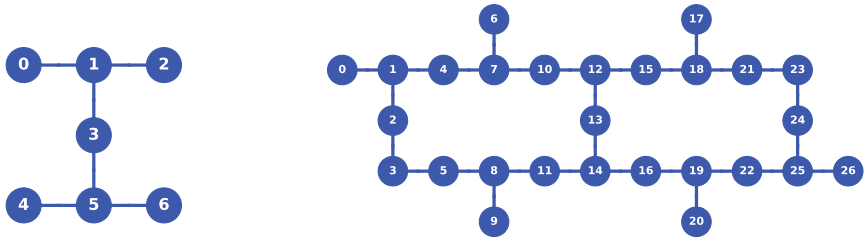

Figure A.1: IBM QPU topology with 7 (left) and 27 (right) qubits. Each qubit is denoted by a circle, and the lines connecting the qubits depict the available two-qubit gates.

Table A.3: Qubits used in the n-qubit DC-QAOA implementation on ibm_perth in Figure 7.17.

n	AOQMAP	Qiskit	Tket
3	{1, 2, 3}	{1, 3, 5}	{1, 2, 3}
4	{1, 2, 3, 5}	{1, 2, 3, 5}	{3, 4, 5, 6]
5	{1, 2, 3, 5, 6}	{1, 3, 4, 5, 6}	{1, 3, 4, 5, 6}

Table A.4: Calibration data at the time of the demonstration on ibm_perth in Figure 7.17.

Qubits	0	1	2	3	4	5	6
T_1 (µs)	119.57	172.46	78.2	223.51	229.34	208.77	123.57
T_2 (µs)	68.66	51.34	98.23	292.47	145.49	236.87	238.67
Frequency (GHz)	5.158	5.034	4.863	5.125	5.159	4.979	5.157
Anharmonicity (GHz)	-0.3415	-0.3444	-0.3473	-0.3404	-0.3334	-0.346	-0.3405
Prob meas0 prep1 (%)	8.9	3.08	2.6	1.78	16.58	3.02	1.2
Prob meas1 prep0 (%)	67.38	3.16	1.88	1.62	11.94	3.02	1.04
Readout length (ns)	721.78	721.78	721.78	721.78	721.78	721.78	721.78
Readout error (%)	38.14	3.12	2.24	1.7	14.26	3.02	1.12
Single-qubit gate error (%)	0.093	0.026	0.023	0.02	0.039	0.035	0.037
CX gate	(0, 1)	(1, 2)	(1, 3)	(3, 5)	(4, 5)	(5, 6)	
CX gate error (%)	1.48	0.75	0.45	0.76	0.89	1.05	

Table A.5: Qubits used in the n-qubit DC-QAOA implementation on ibmq_ehningen in Figure 7.18.

n	AOQMAP	Qiskit	Tket
3	{22, 25, 26}	{22, 25, 26}	{23, 24, 25}
4	{19, 22, 25, 26}	{22, 24, 25, 26}	{22, 24, 25, 26]
5	{16, 19, 22, 25, 26}	{19, 22, 24, 25, 26}	{8, 11, 13, 14, 16}

Table A.6: Calibration data at the time of the demonstration on ibmq_ehningen in Figure 7.18.

Qubits	8	11	13	14	16	19	22	23	24	25	26
T_1 (µs)	147.13	139.39	243.31	118.9	183.36	128.91	156.26	247.34	144.53	300.87	123.47
T_2 (µs)	207.99	225.82	231.39	239.27	250.25	89.89	33.4	399.76	265.3	398.15	23.63
Frequency (GHz)	5.174	5.119	4.926	5.177	5.022	4.784	4.725	4.805	5.074	4.95	5.151
Anharmonicity (GHz)	-0.3399	-0.3405	-0.344	-0.3408	-0.3435	-0.3485	-0.3464	-0.3471	-0.3416	-0.3457	-0.3391
Prob meas0 prep1 (%)	1.4	3.88	1.04	1.4	1.0	1.14	1.62	1.04	1.1	0.84	1.0
Prob meas1 prep0 (%)	1.1	1.76	0.68	0.48	0.58	0.96	1.18	0.76	0.52	0.48	0.6
Readout length (ns)	846.22	846.22	846.22	846.22	846.22	846.22	846.22	846.22	846.22	846.22	846.22
Readout error (%)	1.25	2.82	0.86	0.94	0.79	1.05	1.4	0.9	0.81	0.66	0.8
Single-qubit gate error (%)	0.029	0.031	0.022	0.032	0.023	0.026	0.019	0.017	0.021	0.013	0.028
CX gate	(8, 11)	(11, 14)	(13, 14)	(14, 16)	(16, 19)	(19, 22)	(22, 25)	(23, 24)	(24, 25)	(25, 26)	
CX gate error (%)	1.4	0.81	0.72	0.97	0.52	0.48	0.36	0.38	0.88	0.52	

Table A.7: Qubits used in the n-qubit DC-QAOA implementation with depth p, represented as D_n^p, on ibm_cairo in Figure 7.19.

	AOQMAP	Qiskit	Tket
D_3^1	{12, 13, 14}	{12, 13, 14}	{12, 13, 15}
D_3^2	{3, 5, 8}	{12, 13, 14}	{12, 13, 15}
D_3^3	{13, 14, 16}	{12, 13, 14}	{12, 13, 15}
D_4^1	{10, 12, 13, 14}	{10, 12, 13, 15}	{10, 12, 13, 15}
D_4^2	{3, 5, 8, 9}	{10, 12, 13, 15}	{10, 12, 13, 15}
D_4^3	{3, 5, 8, 9}	{10, 12, 13, 15}	{10, 12, 13, 15}
D_5^1	{10, 12, 13, 14, 16}	{10, 12, 13, 14, 15}	{10, 12, 13, 14, 15}
D_5^2	{10, 12, 13, 14, 16}	{10, 12, 13, 14, 15}	{10, 12, 13, 14, 15}
D_5^3	{10, 12, 13, 14, 16}	{10, 12, 13, 14, 15}	{10, 12, 13, 14, 15}

Table A.8: Calibration data at the time of the demonstration on ibm_cairo in Figure 7.19.

Qubits	3	5	8	9	10	12	13	14	15	16
T_1 (µs)	89.65	105.74	83.87	77.23	95.75	99.99	75.72	77.0	130.63	98.95
T_2 (µs)	134.52	50.84	43.33	30.47	13.92	102.35	155.06	132.19	334.49	207.13
Frequency (GHz)	5.119	5.046	4.969	5.227	5.234	5.115	5.282	5.044	4.963	5.277
Anharmonicity (GHz)	-0.3404	-0.3047	-0.3442	-0.3391	-0.3392	-0.3413	-0.3686	-0.3422	-0.3453	-0.3388
Prob meas0 prep1 (%)	1.72	1.82	2.04	2.76	1.0	1.02	0.5	1.04	0.78	1.92
Prob meas1 prep0 (%)	1.72	1.36	2.6	5.48	0.8	0.98	0.36	0.58	0.42	10.18
Readout length (ns)	732.44	732.44	732.44	732.44	732.44	732.44	732.44	732.44	732.44	732.44
Readout error (%)	1.72	1.59	2.32	4.12	0.9	1.0	0.43	0.81	0.6	6.05
Single-qubit gate error (%)	0.02	0.024	0.016	0.022	0.019	0.019	0.015	0.02	0.011	0.02
CX gate	(3, 5)	(5, 8)	(8, 9)	(10, 12)	(12, 13)	(12, 15)	(13, 14)	(14, 16)		
CX gate error (%)	0.55	0.54	0.8	0.8	1.16	0.93	0.48	0.52		

Table A.9: Qubits used in the n-qubit DC-QAOA implementation with depth p, represented as D_n^p, on ibmq_ehningen in Figure 7.20.

	AOQMAP	Qiskit	Tket
D_3^1	{22, 25, 26}	{22, 25, 26}	{23, 24, 25}
D_3^2	{19, 22, 25}	{22, 25, 26}	{23, 24, 25}
D_3^3	{19, 22, 25}	{22, 25, 26}	{23, 24, 25}
D_4^1	{19, 22, 25, 26}	{22, 24, 25, 26}	{22, 24, 25, 26}
D_4^2	{19, 22, 25, 26}	{19, 22, 25, 26}	{22, 24, 25, 26}
D_4^3	{19, 22, 25, 26}	{22, 24, 25, 26}	{22, 24, 25, 26}
D_5^1	{16, 19, 22, 25, 26}	{16, 19, 20, 22, 25}	{19, 22, 24, 25, 26}
D_5^2	{16, 19, 22, 25, 26}	{16, 19, 20, 22, 25}	{19, 22, 24, 25, 26}
D_5^3	{16, 19, 22, 25, 26}	{16, 19, 20, 22, 25}	{19, 22, 24, 25, 26}

Table A.10: Calibration data at the time of demonstration on ibmq_ehningen in Figure 7.20.

Qubits	16	19	20	22	23	24	25	26
T_1 (µs)	183.36	128.91	43.22	156.26	247.34	144.53	300.87	123.47
T_2 (µs)	250.25	89.89	121.72	33.4	399.76	265.3	398.15	23.63
Frequency (GHz)	5.022	4.784	5.042	4.725	4.805	5.074	4.95	5.151
Anharmonicity (GHz)	-0.3435	-0.3485	-0.3426	-0.3464	-0.3471	-0.3416	-0.3457	-0.3391
Prob meas0 prep1 (%)	1.0	1.14	0.82	1.62	1.04	1.1	0.84	1.0
Prob meas1 prep0 (%)	0.58	0.96	2.18	1.18	0.76	0.52	0.48	0.6
Readout length (ns)	846.22	846.22	846.22	846.22	846.22	846.22	846.22	846.22
Readout error (%)	0.79	1.05	1.5	1.4	0.9	0.81	0.66	0.8
Single-qubit gate error (%)	0.023	0.026	0.038	0.019	0.017	0.021	0.013	0.028
CX gate	(16, 19)	(19, 20)	(19, 22)	(22, 25)	(23, 24)	(24, 25)	(25, 26)	
CX gate error (%)	0.52	0.57	0.48	0.36	0.38	0.88	0.52	

Publications of the Author

In the following, all former publications of the author are listed.

Publications

[1] Yanjun Ji, Kathrin F Koenig, and Ilia Polian. Improving the performance of digitized counterdiabatic quantum optimization via algorithm-oriented qubit mapping. *Phys. Rev. A*, 110:032421, Sep 2024.

[2] Yanjun Ji and Ilia Polian. Synergistic dynamical decoupling and circuit design for enhanced algorithm performance on near-term quantum devices. *Entropy*, 26(7), 2024.

[3] Yanjun Ji, Xi Chen, Ilia Polian, and Yue Ban. Algorithm-oriented qubit mapping for variational quantum algorithms, 2023. (Under third review in *Physical Review Applied*).

[4] Yanjun Ji, Kathrin F. Koenig, and Ilia Polian. Optimizing quantum algorithms on bipotent architectures. *Phys. Rev. A*, 108:022610, Aug 2023.

[5] Sebastian Brandhofer, Daniel Braun, Vanessa Dehn, Gerhard Hellstern, Matthias Hüls, Yanjun Ji, Ilia Polian, Amandeep Singh Bhatia, and Thomas Wellens. Benchmarking the performance of portfolio optimization with QAOA. *Quantum Inf. Process.*, 22(1):25, 2023.

[6] Yanjun Ji, Sebastian Brandhofer, and Ilia Polian. Calibration-aware transpilation for variational quantum optimization. In *IEEE International Conference on Quantum Computing and Engineering, QCE 2022, Broomfield, CO, USA, September 18-23, 2022*, pages 204–214. IEEE, 2022.

[7] Z. H. Wang, Y. J. Ji, Yong Li, and D. L. Zhou. Dissipation and decoherence induced by collective dephasing in a coupled-qubit system with a common bath. *Phys. Rev. A*, 91:013838, Jan 2015.

[8] Yan Xu, Wei Fan, Yan-Jun Ji, Ren-Gang Song, Bing Chen, Zhen-Hua Zhao, and Da Chen. Effective field theory approach to the weakly interacting bose gas. *Acta Phys. Sin.*, 4, 2014.