OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

LABOR FÜR
DIGITALISIERUNG

MASTER'S THESIS

Simon Thelen

Investigation of the Performance of Different QAOA Variants Under the Influence of Realistic Noise

October 29, 2023

Faculty:	Computer Science and Mathematics
Study Programme:	Master of Computer Science (IM)
Deadline:	October 31, 2023
Supervisor:	Prof. Dr. Wolfgang Mauerer
Secondary Supervisor:	Prof. Dr. Carsten Kern

Erklärung

1. Mir ist bekannt, dass dieses Exemplar der Masterarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.
2. Ich erkläre hiermit, dass ich diese Masterarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum und Unterschrift

Presented by:	Simon Thelen
Student ID:	3301229
Study Programme:	Master of Computer Science (IM)
Time Frame:	May 1, 2023 – October 31, 2023
Supervisor:	Prof. Dr. Wolfgang Mauerer
Secondary Supervisor:	Prof. Dr. Carsten Kern

Abstract

Quantum computers have the potential to solve certain problems much more efficiently than would be possible with conventional computers. In particular, the Quantum Approximate Optimization Algorithm (QAOA) and its various variants have produced promising results on quantum simulators for a variety of optimization problems. In practice, however, noise and other errors are characteristic of computation on real quantum computers. This work addresses the question of how noise affects the performance of the QAOA and three QAOA variants: Warm-Starting QAOA (WSQAOA), WS-Init-QAOA and Recursive QAOA (RQAOA). In particular, we will investigate which QAOA variants are particularly resistant to noise and how much noise can be tolerated. To answer these questions, a noise model simulating the effects of thermal relaxation, gate infidelities and state preparation and measurement (SPAM) errors will be developed based on publicly available noise data for IBM superconducting quantum systems. The numerical simulations performed reveal measurable differences between the QAOA variants, with the RQAOA being especially resistant to noise and achieving good results with only a single QAOA layer. In addition, our results show that thermal relaxation due to circuit depth caused by $CNOT$ gates is the largest source of noise. We can also demonstrate that for moderate noise levels, using more than one QAOA layer can still improve the performance of the standard QAOA, the WS-Init-QAOA and the RQAOA.

Acronyms

AQC adiabatic quantum computation

NISQ noisy, intermediate-scale quantum

QAOA Quantum Approximate Optimization Algorithm

QPU quantum processing unit

RQAOA Recursive QAOA

SDP semidefinite program

SPAM state preparation and measurement

WSQAOA Warm-Starting QAOA

Contents

1. Introduction

Numerous advances in computing, in both software and hardware, have made it possible to solve complex computational problems today that were once deemed impractical mere decades ago. However, despite recent developments in fields such as machine learning, there still exist several limitations to our computational capabilities, whether of a practical or theoretical nature. While current computer processors rely on traditional electronic circuitry, quantum computation aims to utilize quantum mechanical phenomena such as quantum superpositions and entanglement to gain an advantage over classical computers. Specifically, gate-based quantum computers work with quantum bits called *qubits* that can be in a superposition of zero and one. For multiple algorithmic problems, there are algorithms on gate-based quantum computers with a better asymptotic time complexity than the best-known classical algorithm. Shor's algorithm, for instance, can solve the *Integer factorization problem* and the *Discrete logarithm problem* in polynomial time whereas no classical, polynomial-algorithm is known [1]. For the problem of searching an unstructured database, *Grover's algorithm* achieves a asymptotic time complexity of $O\left(\sqrt{n}\right)$, which is a quadratic speedup compared to the best-possible classical time complexity of $O(n)$ [2].

Both of these algorithms require fault-tolerant quantum computers with a large number of qubits in order to provide an advantage over classical computers. However, current quantum systems, such as those developed by IBM, are subject to quantum noise leading to errors and offer only a limited number of qubits [3]. These kinds of quantum systems are referred to as noisy, intermediate-scale quantum (NISQ) devices [4]. While the computational capability of NISQ systems is limited when compared to the best current classical computers, there have been several efforts to develop algorithms which can take advantage of these systems. Of particular interest in recent years has been the Quantum Approximate Optimization Algorithm (QAOA), a hybrid algorithm that uses both a gate-based quantum computer and a classical computer operating in an alternating fashion [5]. The QAOA can be used to find approximate solutions to many hard optimization problems, including optimization variants of all 21 of Karp's original NP-complete problems [6], [7]. It is considered one of the most promising algorithms of the NISQ era [8]. Still, as suggested by various studies, both solution quality and runtime are severely affected by noise experienced on near-term quantum devices [9]–[12]. Many variants of the QAOA have been proposed, such as the ADAPT-QAOA [13], the Warm-Starting QAOA (WSQAOA) [14] or the Recursive QAOA (RQAOA) [15], [16], which have the potential of overcoming at least some of the limitations caused by noise.

In this thesis, we will analyze the impact of noise on the performance of four QAOA variants via numerical simulations on a Qaptiva 800 quantum simulation platform, using two NP-complete problems, *Max-cut* and *Partition*, as examples. We will develop a model that simulates noise caused by gate infidelities, thermal relaxation, and state preparation and measurement (SPAM) errors using realistic noise parameters obtained from data of IBM superconducting quantum systems. The QAOA variants examined are the standard QAOA, the RQAOA, the WSQAOA and a variant of the WSQAOA [17], which we will call WS-Init-QAOA. We will study how much noise the individual variants can handle and which kinds of noise affects their performance the most. Furthermore, we will derive potential lessons for hardware manufacturers and best practices for quantum engineers and software developers.

The rest of this work is structured as follows: Chapter 2 gives a brief introduction on fundamental concepts of gate-based quantum computing. Chapter 3 introduces both the two studied problems, Max-cut and Partition. It also motivates and explains the QAOA as well as the QAOA variants WSQAOA and RQAOA. In Chapter 4, we will explain the mathematical formalisms used to describe quantum noise as well as how many typical types of noise occurring in a quantum circuit can be modeled. Chapter 5 describes how the QAOA performance analysis was performed and discusses several aspects regarding the design of the noise model and the implementation of the simulation. In Chapter 6, we evaluate the results of the performance analysis, highlighting certain interesting findings and discussing possible reasons for the observations. Finally, Chapter 7 summarizes the main results of the analysis and talks about possible implications and future research directions.

2. Fundamental concepts of quantum computation

In a classical computer, a group of n bits is in one of 2^n possible states, which in the context of quantum computing are called the *computational basis states*. The qubits in a *quantum circuit*, however, can be in a superposition of these states, determined by the states' *amplitudes*. *Quantum gates* can manipulate these amplitudes. When measuring the qubits, their superposition collapses and we only get one of 2^n possible results, depending on the amplitudes. This chapter introduces fundamental mathematical concepts (Section 2.1) as well as the key components of quantum circuits: qubits (Sections 2.2 and 2.3), gates (Section 2.4) and measurements (Section 2.5).

2.1. Fundamental notions from linear algebra

A *complex vector space* is a vector space whose scalars are complex numbers instead of real numbers. Throughout this thesis we will be working exclusively in the finite complex vector space $\mathbb{C}^n$. The *bra-ket notation* $|\psi\rangle$ (*"ket psi"*) is used to describe a vector ψ in the complex vector space $\mathbb{C}^n$:

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix}$$

If $|\psi\rangle$ and $|\varphi\rangle$ are vectors in some vector space, such as $\mathbb{C}^n$, then so is $\alpha|\psi\rangle + \beta|\varphi\rangle$ where $\alpha, \beta \in \mathbb{C}$.

A linear operator A maps vectors in $\mathbb{C}^n$ to vectors in $\mathbb{C}^m$ and will be represented as a matrix with complex elements: $A \in \mathbb{C}^{m \times n}$. $\mathbb{1}_n$ is the n-dimensional *identity* operator ($\mathbb{1}_n|\psi\rangle = |\psi\rangle$ for every $|\psi\rangle \in \mathbb{C}^n$). It can be written as a matrix with ones on the diagonal and zeros everywhere else. If the number of dimensions is not important or clear from context, the identity is written as $\mathbb{1}$.

The complex conjugate z^* of a complex number z is defined as $(a + b\mathrm{i})^* = a - b\mathrm{i}$. Further, $|z|^2 = z^* z = a^2 + b^2$. The complex conjugate A^* of a matrix A is defined by taking the complex conjugate element-wise: $(A^*)_{ij} = (A_{ij})^*$. $A^\dagger$ is the *conjugate transpose* or *Hermitian transpose* of A: $A^\dagger = (A^T)^*$, i.e. $(A^\dagger)_{ij} = (A_{ji})^*$. It has the

following property: $(AB)^\dagger = B^\dagger A^\dagger$. As a shorthand, we define the *vector dual* $\langle\psi|$ (*"bra psi"*) as $\langle\psi| = \big(|\psi\rangle\big)^\dagger$. This notation allows us to define the *inner product* $\langle\psi|\varphi\rangle \in \mathbb{C}$ of the two vectors $|\psi\rangle$ and $|\varphi\rangle$ as

$$\langle\psi|\varphi\rangle = \begin{pmatrix}\psi_1^* \psi_2^* \dots \psi_n^*\end{pmatrix} \begin{pmatrix}\varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_n\end{pmatrix} = \sum_i \psi_i^* \varphi_i.$$

The norm of a vector $|\psi\rangle$ is defined as $\sqrt{\langle\psi|\psi\rangle}$. A *normalized vector* is a vector with norm 1 ($\langle\psi|\psi\rangle = 1$). Two vectors $|\psi\rangle$ and $|\varphi\rangle$ are *orthogonal* if $\langle\psi|\varphi\rangle = 0$. A set of vectors $\{|e_i\rangle\} \subseteq \mathbb{C}^n$ forms a *basis* of $\mathbb{C}^n$ if every vector in $|\psi\rangle \in \mathbb{C}^n$ can be written as a linear combination of the $|e_i\rangle$: $|\psi\rangle = \sum_i \alpha_i |e_i\rangle$ where $\alpha_i \in \mathbb{C}$. If the $|e_i\rangle$ are normalized and mutually orthogonal, they form an *orthonormal basis*. For any two states $|e_i\rangle, |e_j\rangle$ of some orthonormal basis, $\langle e_i|e_j\rangle = \delta_{ij}$, where δ_{ij} denotes the *Kronecker delta*

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

From the definition of the vector dual, it follows that $|\varphi\rangle\langle\psi|$ is a linear operator which maps $|\psi\rangle$ to $|\varphi\rangle$ and all vectors that are orthogonal to $|\psi\rangle$ to the zero vector. In particular, if $|i\rangle$ and $|j\rangle$ are standard basis vectors, then $|i\rangle\langle j|$ is matrix A with $A_{ij} = 1$ and zeros everywhere else. On the other hand, if A is a matrix and $|i\rangle$ and $|j\rangle$ are standard basis vectors, then $\langle i|A|j\rangle = A_{ij}$ [18].

2.2. Quantum states and qubits

In the context of quantum computing, quantum states can be represented as normalized vectors in $\mathbb{C}^n$. The two vectors

$$|0\rangle = \begin{pmatrix}1 \\ 0\end{pmatrix}, \quad |1\rangle = \begin{pmatrix}0 \\ 1\end{pmatrix}$$

form an orthonormal basis of $\mathbb{C}^2$. This basis is commonly referred to as the *1-qubit computational basis*.

The smallest unit of information in quantum computing is the qubit. The state of a qubit is represented as a normalized vector $|\psi\rangle \in \mathbb{C}^2$ and can therefore be written as a linear combination of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \begin{pmatrix}\alpha \\ \beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle \text{ with } \langle\psi|\psi\rangle = |\alpha|^2 + |\beta|^2 = 1 \tag{2.1}$$

Whereas a classical bit can either be 0 or 1, a qubit can be in a superposition of $|0\rangle$ and $|1\rangle$, which is determined by its *amplitudes* α and β. If we are given

some qubit, we cannot access the amplitudes directly but we can obtain partial information about them through measurement. Measurements are performed in an orthonormal basis, which usually but not always is the computational basis. When measuring the qubit (2.1) in the computational basis, we will find it in state $|0\rangle$ with probability $|\alpha|^2$ and in state $|1\rangle$ with probability $|\beta|^2$. By measuring the qubit, its superposition collapses. The qubit's state after the measurement will be the computational basis state in which it was measured.

Consider the two states

$$|+\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big), \;\; |-\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big).$$

When measured in the computational basis, we get the same kind of results: $|0\rangle$ and $|1\rangle$ with probability $1/2$ each. They are still considered different states because the relative phase of the two amplitudes is different. Often, it is more useful to think of the complex amplitudes in polar form: $z = r\exp(\mathrm{i}\varphi)$ where r describes the absolute value of the amplitude and φ describes its phase. The relative phase of the amplitudes of $|0\rangle$ and $|1\rangle$ is significant and plays an important role in many quantum algorithms. However, if two state vectors $|\psi\rangle$ and $|\psi'\rangle$ are the same up to a global phase factor, that is $|\psi'\rangle = \exp(\mathrm{i}\theta)|\psi\rangle$, they are physically indistinguishable and are considered to represent the same state. Thus, without loss of generality, we can assume that the phase of $|0\rangle$ is 0. By further using the property $(\sin\theta)^2 + (\cos\theta)^2 = 1$, each qubit can be written in the form

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \exp(\mathrm{i}\varphi)\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.2}$$

with $\varphi \in [0, 2\pi]$ and $\theta \in [0, \pi]$. For example, $\varphi = \pi, \theta = \pi/2$ yields the state $|-\rangle$. This results in a useful visualization known as the *Bloch sphere*, shown in Figure 1. Every single-qubit state can be thought of as a point on the Bloch sphere, where $|0\rangle$ is the "north pole" and $|1\rangle$ is the "south pole" of the sphere. This visualization, however, only works for single-qubit systems [18].

2.3. Multi-qubit systems

Multi-qubit systems are expressed using the *tensor product*. If V and W are vector spaces, then the tensor product $V \otimes W$ is the vector space containing all vectors which can be written as a linear combination of vectors of the form $|v\rangle \otimes |w\rangle$ where $|v\rangle \in V$ and $|w\rangle \in W$. Many useful properties follow from this definition such as associativity and $(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$. As a shorthand notation, we will write $|v\rangle \otimes |w\rangle = |v, w\rangle = |vw\rangle$. States in a two-qubit system are represented as normalized vectors in $\mathbb{C}^2 \otimes \mathbb{C}^2$ and can be written as linear combinations of vectors from the two-qubit computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \text{ with } |\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$$

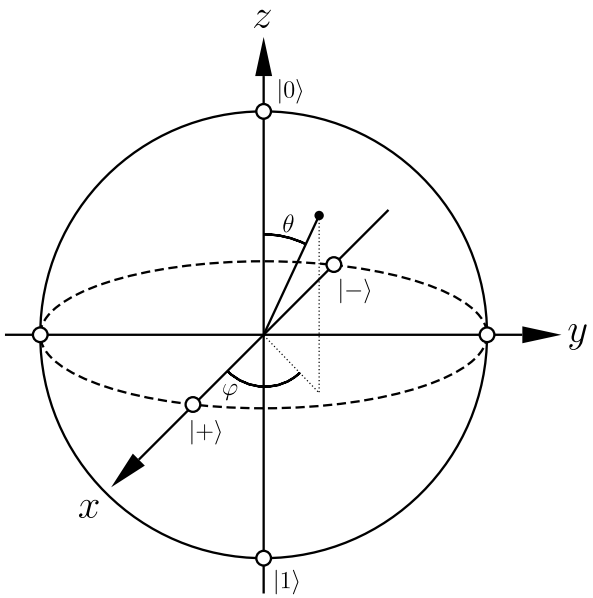


Figure 1.: The Bloch sphere: a useful way to visualize the general single-qubit state $\cos\frac{\theta}{2}|0\rangle + \mathrm{e}^{\mathrm{i}\phi}\sin\frac{\theta}{2}|1\rangle$ [18]

In practice, a vector in $\mathbb{C}^n \otimes \mathbb{C}^m$ is commonly represented as a vector in $\mathbb{C}^{nm}$. This transformation can be done using the *Kronecker product*, which for two qubits is defined as:

$$|v\rangle \otimes |w\rangle = \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} \otimes \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} v_0 \cdot \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \\ v_1 \cdot \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} v_0 w_0 \\ v_0 w_1 \\ v_1 w_0 \\ v_1 w_1 \end{pmatrix} \tag{2.3}$$

By generalizing this definition to higher dimensions, the state of an n-qubit system can be represented as a 2^n-dimensional, complex vector [18].

2.4. Quantum gates

By the postulates of Quantum mechanics, if $|\psi(t)\rangle$ is a state of some closed quantum system at time t, then the evolution of the state from time t_1 to time t_2 can be written as $|\psi(t_2)\rangle = U|\psi(t_1)\rangle$ where U is a *unitary operator*. Quantum gates are therefore represented as unitary operators. An operator/a matrix U is unitary if $UU^\dagger = U^\dagger U = \mathbb{1}$. From this property, it follows that unitary operators preserve inner products since the inner product of $U|\psi\rangle$ and $U|\varphi\rangle$ is

$$\big(U|\psi\rangle\big)^\dagger U|\varphi\rangle = \langle\psi|U^\dagger U|\varphi\rangle = \langle\psi|\mathbb{1}|\varphi\rangle = \langle\psi|\varphi\rangle.$$

In particular, normalized vectors stay normalized and orthogonal vectors stay orthogonal after applying a unitary. Unitary operators can therefore be thought

of as rotations, mapping the vectors of one orthonormal basis $\{|e_i\rangle\}$ to the vectors of some other orthonormal basis $\{|f_i\rangle\}$ of the same dimension. In fact, any unitary can be written as

$$U = \sum_j |f_j\rangle\langle e_j| \text{ with } U|e_i\rangle = \sum_j |f_j\rangle \underbrace{\langle e_j|e_i\rangle}_{\delta_{ij}} = |f_i\rangle.$$

The tensor product can also be used on operators by defining $(A \otimes B)(|v\rangle \otimes |w\rangle) = A|v\rangle \otimes B|w\rangle$ and extending this definition to all vectors by linearity. This includes vector duals: $|vw\rangle^\dagger = \langle vw|$. The Kronecker product can be defined analogously to (2.3) for matrices. For example, for $A \in \mathbb{C}^{2\times 3}$ and $B \in \mathbb{C}^{2\times 2}$:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & a_{13}B \\ a_{21}B & a_{22}B & a_{23}B \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} & a_{13}b_{11} & a_{13}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} & a_{13}b_{21} & a_{13}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} & a_{23}b_{11} & a_{23}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} & a_{23}b_{21} & a_{23}b_{22} \end{pmatrix}$$

The definition of unitary operators implies that if U_1 and U_2 are unitary, then so are U_2U_1 and $U_1 \otimes U_2$. In other words, unitaries can be composed sequentially and in parallel. A quantum circuit can thus be thought of as one large unitary operator which is composed of many smaller unitary operators, the individual quantum gates.

While quantum gates acting on any number of qubits are conceivable, in practice almost all quantum gates either act on a single qubit or two qubits. Three of the most widely used single-qubit gates are the so-called *Pauli gates*

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Each Pauli gate can be interpreted as a $180°$ rotation around its respective axis of the Bloch sphere. As an alternative interpretation, the X gate applies a bit flip and is equivalent to the classical NOT gate ($X|0\rangle = |1\rangle, X|1\rangle = |0\rangle$), Z applies a phase flip by adding a relative phase of π or $180°$ to the amplitude of $|1\rangle$ ($Z|0\rangle = |0\rangle, Z|1\rangle = -|1\rangle$), and Y applies both a bit and a phase flip.

Another important gate is the Hadamard gate H which maps $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$:

$$H = |+\rangle\langle 0| + |-\rangle\langle 1| = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Since qubits are initialized as $|0\rangle$ in many physical quantum computers, the Hadamard is often used to prepare the equal superposition $|+\rangle$.

These four gates all have in common that applying each gate twice restores the original state: $XX = YY = ZZ = HH = \mathbb{1}$. Three other useful gates which generally do not have this property are the rotational gates $RX(\theta), RY(\theta), RZ(\theta)$. Each gate is parameterized by an angle θ and corresponds to a rotation of θ

around the respective Bloch sphere axis. To find the matrix representations of these gates, one needs to compute the matrix exponentiation of the Pauli matrices. By using the Taylor series expansions of $\exp(\cdot)$, $\sin(\cdot)$ and $\cos(\cdot)$, you can show that for every matrix A such that $AA = \mathbb{1}$,

$$\exp(\mathrm{i}\theta A) = \cos(\theta)\mathbb{1} - \mathrm{i}\sin(\theta)A.$$

Using this fact one can derive $RX(\theta)$, $RY(\theta)$ and $RZ(\theta)$:

$$\begin{aligned} RX(\theta) &= \exp(-\mathrm{i}\theta X/2) = \begin{pmatrix} \cos\frac{\theta}{2} & -\mathrm{i}\sin\frac{\theta}{2} \\ -\mathrm{i}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \\ RY(\theta) &= \exp(-\mathrm{i}\theta Y/2) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \\ RZ(\theta) &= \exp(-\mathrm{i}\theta Z/2) = \begin{pmatrix} \exp\left(-\mathrm{i}\frac{\theta}{2}\right) & 0 \\ 0 & \exp\left(\mathrm{i}\frac{\theta}{2}\right) \end{pmatrix} \end{aligned}$$

where the factor $1/2$ is necessary to preserve the Bloch sphere interpretation given by Figure 1. Every possible single-qubit gate can be represented as a combination of RX, RY and RZ. Since chaining multiple rotations results in another rotation, every single-qubit gate is equivalent to some rotation around the Bloch sphere.

The identity gate $\mathbb{1}_2$ is trivially unitary since $\mathbb{1}\mathbb{1}^\dagger = \mathbb{1}\mathbb{1} = \mathbb{1}$. To apply a single-qubit gate to some qubit in a multi-qubit system, we can tensor it with the identity. For example, to apply the X gate to the first qubit in a three-qubit system, we apply the unitary $X \otimes \mathbb{1}_2 \otimes \mathbb{1}_2$ to the system. As a notational shorthand, we will write the unitary applying the single qubit gate U to the i-th qubit as $U^{(i)}$. The unitary of the previous example would therefore be written as $X^{(1)}$.

If a quantum circuit only contains single-qubit gates, every qubit is essentially independent. This changes when we introduce two-qubit gates. The most common two-qubit gate is the controlled-NOT or $CNOT$ gate. It involves a control qubit and a target qubit. The gate flips the target qubit if the control qubit is in state $|1\rangle$ and leaves it untouched if the control qubit is in state $|0\rangle$:

$$CNOT = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10| = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Alternatively, we can interpret $CNOT$ as a gate mapping $|a, b\rangle$ to $|a, a \oplus b\rangle$ where $\oplus$ denotes Boolean XOR. Every possible unitary operation on a multi-qubit system can be implemented using single-qubit gates and the $CNOT$ gate.

$CNOT$ can cause two qubits to become entangled:

$$CNOT(|+\rangle \otimes |0\rangle) = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big) \tag{2.4}$$

If a multi-qubit state $|\psi\rangle$ can be written as a tensor product $|a\rangle \otimes |b\rangle$, we call it *separable*. Otherwise, we call it entangled. Intuitively, if two qubits are entangled, then measuring one gives us additional information about the other which we did not have before. In (2.4), for example, if we measure the first qubit and find it in state $|0\rangle$, we know that the second qubit must also be in state $|0\rangle$ since $|01\rangle$ has amplitude 0 [18].

2.5. Measurements

Measuring a single-qubit state $|\psi\rangle$ in the computational basis can be interpreted as a projection of the state onto the basis vectors $|0\rangle$ and $|1\rangle$ where $\langle\psi|0\rangle\langle 0|\psi\rangle = |\langle 0|\psi\rangle|^2$ is the probability of getting the result $|0\rangle$ when measuring $|\psi\rangle$. In general, these kinds of projective measurements of a quantum system are typically described using *observables*. The same way a quantum gate is represented by a unitary operator, an observable is represented by a *Hermitian operator*. An operator H is Hermitian if $H = H^\dagger$.

Given a square matrix A, a vector $|\psi\rangle$ and a scalar λ, if $A|\psi\rangle = \lambda\psi$, then λ is called an *eigenvalue* of A and $|\psi\rangle$ is called the corresponding *eigenvector* or *eigenstate*. The eigenvalues of an observable are the possible measurement results. It can be shown that eigenvalues of an observable/Hermitian operator are always real. It can be further shown that every observable M has a so-called *spectral decomposition* and can be written as

$$M = \sum_i m_i |v_i\rangle\langle v_i|, \tag{2.5}$$

where the m_i are the eigenvalues of M, the $\{|v_i\rangle\}$ form an orthonormal basis of $\mathbb{C}^n$ and each $|v_i\rangle$ is an eigenvector of M with eigenvalue m_i. $\{|v_i\rangle\}$ can be thought of as the measurement basis of M. If any of the $|v_i\rangle$ is measured with M, the measurement result will be m_i with probability 1. If the m_i are all distinct, the decomposition (2.5) will be unique. In the general case, we need to group terms with the same eigenvalue, giving the decomposition

$$M = \sum_m m P_m. \tag{2.6}$$

Here, the distinct m are the eigenvalues of M and every P_m is a so-called *projector*, which can be written as $P_m = \sum_j |v_j\rangle\langle v_j|$ for a subset of the basis vectors. A projector can be thought of as a Hermitian operator which "projects" vectors from $\mathbb{C}^n$ into the subspace formed by the $|v_j\rangle$.

Measurements of the observable given by (2.6) are defined as follows: When measuring the state $|\psi\rangle$, the probability of getting outcome m is given by

$$\Pr(m) = \langle\psi|P_m|\psi\rangle. \tag{2.7}$$

As discussed earlier, measuring a quantum system changes its state. The state after the measurement is given by

$$\frac{P_m|\psi\rangle}{\sqrt{\Pr(m)}}.$$

Visually, P_m projects $|\psi\rangle$ into the subspace which aligns with the measurement result and the factor $1/\sqrt{\Pr(m)}$ re-normalizes the state. Definition (2.7) justifies the idea that two quantum states which are equal up to a global phase are physically indistinguishable since

$$\big(\exp(\mathrm{i}\theta)|\psi\rangle\big)^\dagger P_m \exp(\mathrm{i}\theta)|\psi\rangle = \underbrace{\exp(-\mathrm{i}\theta)\exp(\mathrm{i}\theta)}_{\exp(-\mathrm{i}\theta+\mathrm{i}\theta)}\langle\psi|P_m|\psi\rangle = \langle\psi|P_m|\psi\rangle,$$

so there is no observable to tell them apart. Using (2.7), the expected measurement result when measuring M is given by

$$\mathrm{E}[M] = \sum_m m\Pr(m) = \sum_m m\langle\psi|P_m|\psi\rangle = \langle\psi|\left(\sum_m mP_m\right)|\psi\rangle = \langle\psi|M|\psi\rangle. \quad (2.8)$$

The most common kind of measurement is the Z measurement with observable Z (as defined in Section 2.4). Z has the eigenvalue/eigenstate pairs $Z|0\rangle = +1|0\rangle$ and $Z|1\rangle = -1|1\rangle$. So, a Z measurement is a measurement in the computational basis, where we associate $|0\rangle$ with result 1 and $|1\rangle$ with result -1. For example, measuring $|+\rangle$ yields

$$\Pr(1) = \langle+|0\rangle\langle0|+\rangle = \frac{1}{2} \text{ and } \Pr(-1) = \langle+|1\rangle\langle1|+\rangle = \frac{1}{2}$$

as expected.

Another common measurement is the X measurement using observable X with $X|+\rangle = +1|+\rangle$ and $X|-\rangle = -1|-\rangle$. It is equivalent to first applying the unitary H and then performing a Z measurement. Finally, $\mathbb{1}_2$ can also be thought of as a measurement. It has only one possible outcome, 1, and leaves the measured state untouched.

Observables can be composed with the tensor product. If a_i are the possible results of observable A and b_j are the possible results of observable B, then $A \otimes B$ is an observable whose results are all products $a_i b_j$. For example, the observable $Z^{(1)}Z^{(3)} = Z \otimes \mathbb{1}_2 \otimes Z$ measures qubits 1 and 3 of a 3-qubit system in the computational while leaving qubit 2 untouched. The result is $+1$ if the qubits are found to be in the same computational basis state and -1 if their computational basis states are different.

We can also add observables, adding their corresponding results, as long as they are diagonal in the same basis, that is they share the same measurement basis $\{|v_i\rangle\}$. For example, $M = Z \otimes \mathbb{1}_2 + Z \otimes Z$ is an observable with

$$\langle00|M|00\rangle = 2, \langle01|M|01\rangle = 0, \langle10|M|10\rangle = -2, \langle11|M|11\rangle = 0$$

since both $Z \otimes \mathbb{1}_2$ and $Z \otimes Z$ can be written as a linear combination of $|v\rangle\langle v|$ with $v \in \{00, 01, 10, 11\}$ [18].

3. Quantum optimization

One of the most promising applications of quantum computers is solving hard optimization problems, with the QAOA being one of the leading candidate algorithms of this field. This chapter motivates and explains the QAOA and two of its variants, which will be investigated in this thesis. Section 3.1 explains how optimization problems are typically represented in the context of quantum computing. In Section 3.2 and Section 3.3, two NP-hard optimization problems are introduced, which will be the main focus of the investigation of noisy QAOA. Section 3.4 introduces adiabatic quantum computation (AQC). This is the key motivation behind the QAOA, which will be explained in Section 3.5. Finally, Section 3.6 explores two QAOA variants: WSQAOA and RQAOA.

3.1. Hamiltonians and the Ising model

In the context of Quantum mechanics, a Hamiltonian is an operator which represents the energy of a particular quantum system. Mathematically, a Hamiltonian H is expressed as a Hermitian operator. The eigenvectors of a Hamiltonian are usually called its *energy eigenstates*. The corresponding eigenvalues are the energy values of their respective eigenstate. Since every Hamiltonian is Hermitian, its eigenvalues are always real (cf. Section 2.5). The eigenstate with the lowest eigenvalue is called the *lowest-energy eigenstate* or *ground state*. We can interpret a Hamiltonian as an observable. With this interpretation $\langle\psi|H|\psi\rangle$ is the average energy value when measuring state $|\psi\rangle$ [18].

Suppose we want to solve some optimization problem where we try to find the optimal input $x \in X$ to some objective function $f(\cdot)$, meaning the goal is to find $\arg\min_{x\in X} f(x)$. We can limit ourselves to minimization problems since any maximization problem is equivalent to the minimization problem that results from identifying $f'(x) = -f(x)$. In quantum computing, there are multiple approaches to solving optimization problems, such as AQC, Variational Quantum Eigensolvers or the QAOA. Conceptionally, they all have in common that they reduce the problem of finding the optimal solution to finding the ground state of a certain Hamiltonian, where the ground state is the state $|\psi\rangle$ that minimizes $\langle\psi|H|\psi\rangle$ [5], [19], [20]. This can be achieved by constructing a Hamiltonian whose eigenstates correspond to the possible inputs $x \in X$ and whose

respective eigenvalues correspond to the $f(x)$. Therefore, the spectral decomposition of this *problem Hamiltonian* H is

$$H = \sum_{x \in X} f(x)|x\rangle\langle x|. \tag{3.1}$$

Often, the inputs to the objective function are binary strings ($X = \{0,1\}^n$). In this case, the $|x\rangle$ are usually represented as the n-qubit computational basis states. From (3.1), we see that H will then be a 2^n by 2^n diagonal matrix with the $f(x)$ as its diagonal elements. Therefore, constructing the Hamiltonian matrix explicitly would be just as difficult as finding the optimal x using brute force.

Fortunately, for many classes of problems, efficient constructions of the problem Hamiltonian exist. One particular example is the *Ising model*. Here, the objective function C is given by [7]:

$$C(s) = C(s_1, s_2, \ldots, s_n) = -\sum_{i<j} J_{ij} s_i s_j - \sum_i h_i s_i$$

The s_i are *spin variables* ($s_i = \pm 1$) and the J_{ij} and the h_i are problem-specific, real constants. Intuitively, the h_i describe the incentive for any individual s_i to be $+1$ or -1, while the J_{ij} describe correlations between the s_i. For example, a negative J_{ij} can be interpreted as: "s_i and s_j should have different values". Simple Ising formulations exist for many classic NP-hard optimization problems, including Max-clique, Knapsack, Vertex cover and Max-cut [7].

Conventionally, for computational basis states $|x\rangle = |x_1, \ldots, x_1\rangle$ the x_i are interpreted as binary ($x_i = \{0,1\}$). By identifying $s_i = (-1)^{x_i}$, we can write the objective function in terms of binary variables:

$$C'(x) = C'(x_1, \ldots, x_n) = C(s_1, \ldots, s_2) = C(s)$$

The problem Hamiltonian H_C can then be represented as:

$$H_C = -\sum_{i<j} J_{ij} Z^{(i)} Z^{(j)} - \sum_i h_i Z^{(i)}$$

To see that this is the correct Hamiltonian, we can check that $H_C|x\rangle = C'(x)|x\rangle$ for every computational basis state $|x\rangle = |x_1, x_2, \ldots, x_n\rangle$, matching (3.1):

$$\begin{aligned}
H_C|x\rangle &= \left(-\sum_{i<j} J_{ij} Z^{(i)} Z^{(j)} - \sum_i h_i Z^{(i)}\right)|x\rangle \\
&= -\sum_{i<j} J_{ij} Z^{(i)} Z^{(j)}|x\rangle - \sum_i h_i Z^{(i)}|x\rangle && \text{by linearity} \\
&= -\sum_{i<j} J_{ij}(-1)^{x_i}(-1)^{x_j}|x\rangle - \sum_i h_i(-1)^{x_i}|x\rangle && \text{since } Z^{(i)}|x\rangle = (-1)^{x_i}|x\rangle \\
&= \left(-\sum_{i<j} J_{ij} s_i s_j - \sum_i h_i s_i\right)|x\rangle = C'(x)|x\rangle && \text{by linearity}
\end{aligned}$$

Here, we used the fact that $Z^{(i)}|x\rangle = (-1)^{x_i}|x\rangle$ for every computational basis state $|x\rangle$. For example, $(Z \otimes \mathbb{1})|x_1, x_2\rangle = Z|x_1\rangle \otimes |x_2\rangle = (-1)^{x_1}|x_1\rangle \otimes |x_2\rangle$ [7].

3.2. The Max-cut problem

The *Max-cut problem* is a well-known NP-hard optimization problem on undirected graphs [21]. Visually, one can think of the problem as choosing one of two colors for every vertex such that the number of edges between different-color vertices is as large as possible. Max-cut is known to have a particularly simple problem Hamiltonian, which makes it a great choice for analyzing quantum approximation algorithms and QAOA in particular [5].

For the purposes of this thesis, we will formally define Max-cut as follows: A graph $G = (V, E)$ is defined as a set of vertices $V = \{1, 2, \ldots, n\}$ and an edge relation $E \subseteq V \times V$. We pose the additional restriction that edges only go from lower to higher vertices: $(u, v) \in E \Rightarrow u < v$. This avoids double-counting the edges, which makes the problem easier to work with, especially regarding the RQAOA (cf. Section 3.6.2). We can assume that the reverse edge (v, u) is implicitly part of the graph as well. Given a graph $G = (V, E)$, a *cut* is a vertex partition into two sets S and T ($V = S \cup T, S \cap T = \varnothing$). An edge (u, v) *crosses* the cut if $u \in S$ and $v \in T$ or vice versa. The *size* of a cut is defined as the number of edges crossing the cut. The Max-cut problem is the problem of finding the largest-size cut for a given graph.

We can describe a cut using n spin variables $s = \{s_1, s_2, \ldots, s_n\}$:

$$s_i = \begin{cases} 1 & \text{if } i \in S \\ -1 & \text{if } i \in T \end{cases}$$

The contribution of each edge (u, v) to the size of the cut is then given by

$$\frac{1}{2}(1 - s_u s_v) = \begin{cases} 0 & \text{if } s_u = s_v \\ 1 & \text{if } s_u \neq s_v. \end{cases}$$

The objective function $f(s)$ to maximize is the size of the cut and can therefore be written as

$$f(s) = \sum_{(u,v)\in E} \frac{1}{2}(1 - s_u s_v) = \frac{|E|}{2} + \sum_{(u,v)\in E} -\frac{1}{2} s_u s_v.$$

Since we can ignore the constant, cut-independent term $|E|/2$, we can represent Max-cut in the Ising model as

$$\text{minimize } C(s) = -\sum_{i<j} J_{ij} s_i s_j - \sum_i h_i s_i$$

$$\text{with } h_i = 0, \quad J_{ij} = \begin{cases} -\frac{1}{2} & \text{if } (i, j) \in E \\ 0 & \text{if } (i, j) \notin E. \end{cases}$$

Note that we flipped the sign of the J_{ij} since Max-cut is a maximization problem while the Ising model describes minimization problems.

The decision version of Max-cut "Does there exist a cut with size at least k?" is known to be NP-complete. Since computing the size of a given cut takes polynomial time, decision Max-cut is in NP. NP-hardness can be shown via the following reduction: 3-Sat $\leq_P$ Max-2-sat $\leq_P$ Max-cut [21]. Therefore, assuming P $\neq$ NP, there is no polynomial-time algorithm for Max-cut. There exist, however, multiple approaches to approximate the maximum cut.

A surprisingly effective method is to simply flip a coin for every vertex to determine if it is in S or in T. By choosing all s_i randomly, independently with $\Pr\{s_i = -1\} = \Pr\{s_i = 1\} = 0.5$, we get a cut of size $|E|/2$ in expectation. To see this, we can define an indicator random variable X_e for every edge $e \in E$:

$$X_e = \begin{cases} 1 & \text{if } e \text{ is part of the cut} \\ 0 & \text{otherwise} \end{cases}$$

$\mathrm{E}[X_e]$, the expected value of X_e, is

$$\mathrm{E}[X_e] = 1 \cdot \Pr\{X_e = 1\} + 0 \cdot \Pr\{X_e = 0\} = \Pr\{X_e = 1\}.$$

For any given edge $e = (u, v)$, X_e is 1 if and only if $x_u \neq x_v$. Since all x_i are chosen independently, $\mathrm{E}[X_e] = \Pr\{x_u \neq x_v\} = 1/2$. The size of the cut can be described by the random variable $X = \sum_{e \in E} X_e$. Due to the linearity of expectation, $\mathrm{E}[X] = \sum_{e \in E} \mathrm{E}[X_e] = |E|/2$.

The size of the maximum cut is at most $|E|$, so this simple method is a randomized 0.5-approximation algorithm for Max-cut [22].

The best known classical optimization algorithm for Max-cut is by Goemans and Williamson [23]. Like many approximation algorithms, their approach involves solving a relaxed (continuous) version of the problem and rounding the solution to find an approximate solution for the original (discrete) problem. In particular, their idea is to interpret the spin variables $s_i \in \mathbb{R}$ as 1-dimensional unit vectors with $|s_i| = 1$. In this interpretation, a natural generalization is to consider n-dimensional unit vectors $v_i \in \mathbb{R}^n$. The corresponding relaxation is:

$$\begin{aligned} &\text{maximize } \frac{1}{2} \sum_{\{i,j\} \in E} (1 - v_i \cdot v_j) \\ &\text{subject to: } \forall i \in V : v_i \cdot v_i = 1 \end{aligned} \tag{3.2}$$

Here, $\cdot$ denotes the vector dot product. It is easy to see that if we restrict the vectors v_i to be 1-dimensional, this relaxation is equivalent to the original Max-cut problem.

By concatenating the column vectors v_i to an $n \times n$ matrix $B = (\vec{v_1}\, \vec{v_2}\, \ldots\, \vec{v_n})$ and setting $A = B^T B$, A is the matrix of all dot products of the v_i ($A_{ij} = v_i \cdot v_j$). A is positive (semidefinite), meaning $x^T A x \geq 0$ for all vectors $x \in \mathbb{R}^n$, since a

symmetric matrix A is positive if and only if $A = B^T B$ [24]. This means that one can equivalently formulate (3.2) in terms of A as follows:

$$\text{maximize } \frac{1}{2} \sum_{\{i,j\} \in E} (1 - A_{ij})$$
$$\text{subject to: } A \text{ is positive and } \forall i \in V : A_{ii} = 1$$

This kind of formulation is known as a semidefinite program (SDP). There are several efficient algorithms to solve SDPs [25], [26]. Once the optimal matrix A has been found, one can use Cholesky decomposition [24] to retrieve B and therefore the vectors v_i. This means, the relaxation (3.2) can be solved in polynomial time.

The final step in the Goemans-Williamson algorithm is to convert the solution of the relaxed problem into an approximate solution of the discrete Max-cut problem. If v_i and v_j point in roughly the same direction, then $v_i v_j$ is close to 1 and $s_i s_j$ should equal 1 ($s_i = s_j$). On the other hand, if v_i and v_j point in opposite directions, then $v_i v_j$ is close to -1 and $s_i s_j$ should equal -1 ($s_i \neq s_j$). This motivates the procedure of *hyperplane rounding* where a random n-dimensional hyperplane is selected and for all v_i on one side of the plane, the corresponding s_i will be set to 1, whereas the s_i for the vectors on the other side will be set to -1. This causes vectors pointing in similar directions to have a higher probability of receiving the same spin value. To implement hyperplane rounding, one can choose a random normal vector $r \in \mathbb{R}^n$ by sampling r_i from a normal distribution $\mathcal{N}(0, 1)$. Then, $s_i = \operatorname{sign}(r \cdot v_i)$ [23].

Let C be the average size of the cut obtained by applying hyperplane rounding to the solution of the SDP relaxation and let C' be the size of the maximum cut. Then, [23]

$$\frac{C}{C'} \geq \min_{0 \leq \theta \leq \pi} \left\{ \frac{2\theta}{\pi(1 - \cos(\theta))} \right\} \approx 0.878.$$

3.3. The Partition problem

The Partition problem is the problem of partitioning a multiset[1] of positive integers $S = \{c_1, c_2, \ldots, c_n\}$ into two subsets S_1 and S_2 with equal size where the *size* of a set X is defined as $\operatorname{size}(X) = \sum_{c \in X} c$. This problem is among Karp's 21 original NP-complete problems [6]. The corresponding optimization problem can be described as finding the partition which minimizes the absolute difference $|\operatorname{size}(S_1) - \operatorname{size}(S_2)|$. This is equivalent to minimizing $(\operatorname{size}(S_1) - \operatorname{size}(S_2))^2$, so we can formulate this problem in terms of spin variables as [7]

$$\text{minimize } C(s) = \left(\sum_i c_i s_i \right)^2 = -\sum_{i<j} J_{ij} s_i s_j + \underbrace{\sum_i c_i^2}_{\text{constant}} \text{ with } J_{ij} = -2 c_i c_j.$$

[1] A multiset is similar to a set, but it allows for multiple instances of the same element.

Since the constant term $C = \sum_i c_i^2$ is the same for all assignments of the s_i, we can set $C = 0$ to acquire a valid Ising formulation.

Instead of minimizing the absolute difference of the two sizes, it is often more convenient to think of minimizing the size of the larger set. For a given solution, we will call the size of the larger set the solution's *value*. There are many efficient approximation algorithms for the Partition problem. A simple greedy algorithm, known as *list scheduling*, works as follows: Start with $S_1 = S_2 = \varnothing$. Loop over the c_i. For each c_i, add it to the set with the currently smaller size. This algorithm has an approximation ratio of $3/2$ in the sense that the value of the solution given by the algorithm is at most $3/2$ as large as the optimal value.

Proof. Let OPT be the optimal value and let A be the value of the algorithm's solution. Without loss of generality, assume $\text{size}(S_1) \geq \text{size}(S_2)$. Then, $A = \text{size}(S_1)$.

In the perfect scenario, $\text{size}(S_1) = \text{size}(S_2)$. Therefore, $OPT \geq \frac{1}{2}\sum_i c_i$. Let c_l be the last number added to S_1. Let S_1' and S_2' denote the contents of S_1 and S_2, just before c_l was added. Since c_l is not in S_1' or S_2' and since S_1' and S_2' are disjoint, $\text{size}(S_1') + \text{size}(S_2') + c_l \leq \sum_i c_i$. By the greedy behavior of the algorithm, $\text{size}(S_1') \leq \text{size}(S_2')$. Combining the previous three inequalities, we get

$$OPT \geq \frac{1}{2}\sum_i c_i \geq \frac{1}{2}(\text{size}(S_1') + \text{size}(S_2') + c_l) \geq \text{size}(S_1') + \frac{c_l}{2}.$$

Therefore, $A = \text{size}(S_1) = \text{size}(S_1') + c_l \leq OPT + c_l/2$. Clearly, $OPT \geq c_l$. This implies $A \leq OPT + OPT/2 = 3\,OPT/2$, proving the claim [27], [28]. □

3.4. Adiabatic quantum computation

Adiabatic quantum computation (AQC) is a popular quantum computing approach to solve optimization problems. In addition to the problem Hamiltonian H_C whose ground state is to be found, it introduces a so-called mixer Hamiltonian H_M, which is usually a very simple Hamiltonian with a known ground state. The quantum system is prepared in the ground state of H_M. Then, the Hamiltonian of the system slowly transitions from H_M to H_C according to

$$H(t) = (1 - t/T)H_M + t/T H_C. \tag{3.3}$$

Here $H(t)$ describes the Hamiltonian at time t and T describes the total time of the procedure. Intuitively, $H(t)$ interpolates between H_M and H_C: $H(0) = H_M$ and $H(T) = H_C$. According to the adiabatic theorem, if T is large enough, that is the transition happens slowly enough, the system will remain in the ground state for the entire duration of the procedure. In other words, the system will start in the ground state of H_M and ultimately end up in the ground state of H_C, solving the optimization problem [19], [29].

AQC is generally performed on specialized quantum hardware. The rest of this section covers how to simulate the process on a gate-based quantum computer as outlined in Chapter 2 since this is the motivation behind the QAOA. The Schrödinger equation describes how a closed quantum system $|\psi\rangle$ with Hamiltonian H evolves over time:

$$\mathrm{i}\hbar\frac{\mathrm{d}}{\mathrm{d}t}|\psi\rangle = H|\psi\rangle$$

Here, $\hbar$ is the *reduced Planck constant* whose precise value we can ignore. By scaling the time units appropriately, we can assume $\hbar$ to be equal to 1. With this simplification, for a constant Hamiltonian H, the Schrödinger equation has the solution

$$|\psi(t)\rangle = U(H,t)|\psi(0)\rangle, \text{ with } U(H,t) = \exp(-\mathrm{i}Ht).$$

One can show that $U(H,t)$ is unitary, which means that, in theory, we can simulate it on a gate-based quantum computer. However, the Hamiltonian (3.3) is not constant and changes over time. To approximate it, we divide T into discrete time steps $t_0, t_1, \ldots, t_k$ with $t_i - t_{i-1} = \Delta t$ and apply a series of unitaries:

$$U\big(H(t_k),\Delta t\big)\ U\big(H(t_{k-1}),\Delta t\big)\ \ldots\ U\big(H(t_1),\Delta t\big)\ U\big(H(t_0),\Delta t\big) \tag{3.4}$$

The Hamiltonian $H(t)$ is the sum of two Hamiltonians $H_M(t) = (1 - t/T)H_M$ and $H_C(t) = t/TH_C$. As will later be described in Section 3.5, in many cases we can implement the time evolutions $\exp(-\mathrm{i}H_M(t)\Delta t)$ and $\exp(-\mathrm{i}H_C(t)\Delta t)$ using quantum gates. Unfortunately, implementing the time evolution for $H(t)$ is then still non-trivial since, in general, for two matrices A and B,

$$\exp(A+B) \neq \exp(A)\exp(B).$$

However, by using the so-called Trotter formula

$$\lim_{n\to\infty}\big(\exp(A/n)\exp(B/n)\big)^n = \exp(A+B),$$

we can approximate $U(H(t_i),\Delta t)$ as

$$U(H(t_i),\Delta t) = \big(\exp(-\mathrm{i}H_C(t_i)\Delta t/n)\exp(-\mathrm{i}H_M(t_i)\Delta t/n)\big)^n. \tag{3.5}$$

By combining (3.4) and (3.5), we find that we can simulate AQC on a gate-based quantum computer by preparing $|\psi_0\rangle$, the ground state of H_M and then applying time evolutions of H_C and H_M in an alternating fashion:

$$U(\vec{\beta},\vec{\gamma})|\psi\rangle = \exp(-\mathrm{i}\beta_p H_M)\exp(-\mathrm{i}\gamma_p H_C)\ldots\exp(-\mathrm{i}\beta_1 H_M)\exp(-\mathrm{i}\gamma_1 H_C)|\psi_0\rangle \tag{3.6}$$

By selecting a large enough *depth* p and choosing the parameters $\vec{\beta} = (\beta_1,\ldots,\beta_n)$ and $\vec{\gamma} = (\gamma_1,\ldots,\gamma_n)$ appropriately, we can approximate AQC with arbitrary precision [5], [19], [29].

3.5. The Quantum Approximate Optimization Algorithm

The QAOA tries to approximate the ground state of a problem Hamiltonian H_C by preparing the ground state of a mixer Hamiltonian H_M and then applying the circuit (3.6). However, the algorithm adds one additional idea: Instead of using fixed parameters $\vec{\beta}$ and $\vec{\gamma}$, the circuit is run multiple times, updating $\vec{\beta}$ and $\vec{\gamma}$ using a classical optimization algorithm to minimize the expected energy $\langle\psi_0|U(\vec{\beta},\vec{\gamma})^\dagger H_C U(\vec{\beta},\vec{\gamma})|\psi_0\rangle$. The QAOA can be described as follows [5]:

1. Execute the circuit multiple times, sampling $\langle\psi_0|U(\vec{\beta},\vec{\gamma})^\dagger H_C U(\vec{\beta},\vec{\gamma})|\psi_0\rangle$.
2. Change $\vec{\beta},\vec{\gamma}$ according to a classical optimizer.
3. Repeat steps 1 and 2, until some terminating condition is met.
4. Execute the circuit one final time and measure $U(\vec{\beta},\vec{\gamma})|\psi_0\rangle$ in the computational basis.

While the depth parameter p must be large to approximate AQC with any precision, QAOA tends to produce good results, even for much lower depths. For example, on 3-regular graphs, where each vertex has exactly 3 outgoing edges, 1-layer QAOA already achieves an approximation ratio of 0.6924 [5].

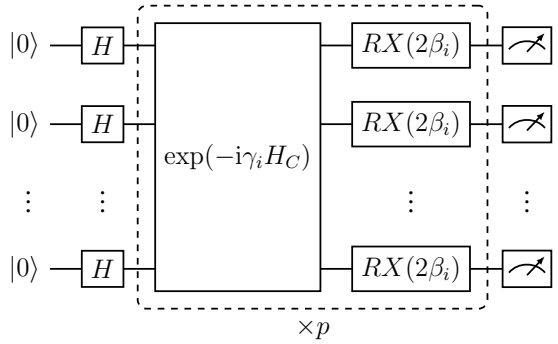
The initial state $|\psi_0\rangle$ of the QAOA circuit is the ground state of the mixer Hamiltonian H_M. The most common choice is $H_M = -\sum_{j=1}^n X^{(j)}$. Since $|+\rangle$ is the ground state of $-X$ with eigenvalue -1, the ground state of H_M is $|\psi_0\rangle = \bigotimes_{i=1}^n |+\rangle$ with eigenvalue $-n$. Qubits are initialized as $|0\rangle$ in most physical implementations, so $|\psi_0\rangle$ is typically prepared by applying the Hadamard gate to each qubit.

As described in Section 2.4, $\exp(-\mathrm{i}\theta X)$ can be implemented using the rotational gate $RX(2\theta)$. To implement $\exp(-\mathrm{i}\theta H_M)$, we can use the fact that if A and B commute, that is $AB = BA$, then $\exp(A+B) = \exp(A)\exp(B)$. If two operators can be written as diagonal matrices in the same basis, they commute. By their spectral decomposition (cf. Section 2.5), the $X^{(i)}$ are diagonal in the n-qubit $|+\rangle$-$|-\rangle$-basis, so they indeed commute. Therefore,

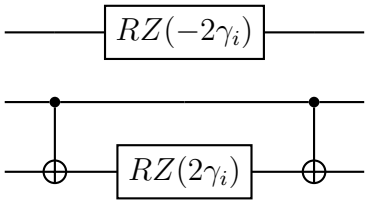
$$\exp(-\mathrm{i}\theta H_M) = \prod_{j=1}^n \exp(-\mathrm{i}\theta X^{(j)}) = \prod_{j=1}^n RX(2\theta)^{(j)} = \bigotimes_{j=1}^n RX(2\theta).$$

This means we can implement the time-evolved mixer Hamiltonian by applying an RX gate to each qubit.

We will assume that H_C is an Ising Hamiltonian. Then, its individual terms are all of the form $Z^{(i)}$ or $Z^{(i)}Z^{(j)}$. This means they commute since they are diagonal in the same basis, namely the computational basis. Therefore, we only need to find quantum circuits implementing $\exp(-\mathrm{i}\theta Z^{(i)})$ and circuits implementing $\exp(-\mathrm{i}\theta Z^{(i)}Z^{(j)})$. Then, we can sequentially compose these circuits in any order



(a) The general QAOA circuit: a Hadamard gate for each qubit, followed by p times alternating time-evolved problem Hamiltonian and mixer Hamiltonian and finally a measurement



(b) $\exp(-\mathrm{i}H_C\gamma_i)$ with $H_C = -Z^{(1)} + Z^{(2)}Z^{(3)}$

Figure 2.: Circuit diagrams for the QAOA

to implement $\exp(-\mathrm{i}\theta H_C)$. As described in Section 2.4, $\exp(-\mathrm{i}\theta Z) = RZ(2\theta)$. To derive $\exp(-\mathrm{i}\theta Z \otimes Z)$, we will use the fact that if $|\psi\rangle$ is an eigenstate of an operator A and λ is the corresponding eigenvalue, then $\exp(A)\psi = \exp(\lambda)\psi$. The 2-qubit computational basis states are the eigenstates of $Z \otimes Z$. Therefore,

$$\exp(-\mathrm{i}\theta Z \otimes Z)|00\rangle = \exp(-\mathrm{i}\theta)|00\rangle, \quad \exp(-\mathrm{i}\theta Z \otimes Z)|01\rangle = \exp(+\mathrm{i}\theta)|01\rangle,$$
$$\exp(-\mathrm{i}\theta Z \otimes Z)|10\rangle = \exp(+\mathrm{i}\theta)|10\rangle, \quad \exp(-\mathrm{i}\theta Z \otimes Z)|11\rangle = \exp(-\mathrm{i}\theta)|11\rangle.$$

Therefore, we can think of $\exp(-\mathrm{i}Z \otimes Z)$ as applying a phase shift whose direction depends on the parity of the two qubits. One can verify that this operation is implemented by $CNOT(\mathbb{1} \otimes RZ(2\theta))CNOT$ where the first $CNOT$ computes the parity, $RZ(2\theta)$ applies the phase shift and the second $CNOT$ restores the state of the second qubit. Figure 2b shows a circuit implementing a simple time-evolved problem Hamiltonian. For the circuit diagram of the complete QAOA, see Figure 2a. In a circuit diagram, qubits are represented as lines to which quantum gates are applied from left to the right. The $CNOT$ gate is denoted by a $\oplus$ (target qubit) and a $\bullet$ (control qubit), connected by a vertical line [5], [18].

3.6. QAOA variants

Recent research on the QAOA has been extensive, exploring many possibilities and variations to improve both its theoretical and empirical performance [8]. In addition to the original QAOA, this thesis will also investigate the performance of two QAOA variants that seem promising in overcoming the challenges of noisy quantum circuits.

3.6.1. The Warm-Starting QAOA

The intuition of QAOA is to simulate the adiabatic evolution from the ground state of a problem-independent mixer Hamiltonian H_M to the ground state of the problem Hamiltonian H_C. The idea of the Warm-Starting QAOA (WSQAOA) is to select a mixer Hamiltonian whose ground state corresponds to a known, very good solution to the given optimization problem or to a relaxation of the problem, giving the algorithm a head start in finding the optimal solution. This shifts the goal from finding the optimal solution from scratch to improving the approximate solution found by a classical algorithm [14].

Let x^* be the solution for a continuous relaxation of the studied optimization problem, where each x_i^* lies in the interval $[0, 1]$. The idea is to then choose a mixer Hamiltonian whose ground state is

$$|\psi\rangle = |\psi_1, \psi_2, \ldots, \psi_n\rangle = \bigotimes_{i=1}^{n} RY(\theta_i)|0\rangle \text{ with } \theta_i = 2 \arcsin\left(\sqrt{x_i^*}\right).$$

For each qubit, the $RY(\theta_i)$ gate performs a rotation around the y-axis of the Bloch sphere between $0°$ and $180°$ based on the corresponding x_i^*. For example, if $x_i^* = 0$, then $|\psi_i\rangle = |0\rangle$, if $x_i^* = 1$, then $|\psi_i\rangle = |1\rangle$ and, if $x_i^* = 0.5$, then $|\psi_i\rangle = |+\rangle$. The mixer Hamiltonian with ground state $|\psi\rangle$ is

$$H_M = \sum_{i=1}^{n} H_{M,i}^{(i)} \text{ with } H_{M,i} = \begin{pmatrix} 2x_i^* - 1 & -2\sqrt{x_i^*(1-x_i^*)} \\ -2\sqrt{x_i^*(1-x_i^*)} & 1 - 2x_i^* \end{pmatrix}.$$

When interpreting each $|\psi_i\rangle$ as a vector on the Bloch sphere, $H_{M,i}^{(i)}$ rotates the i-th qubit $180°$ around this vector. Analogously to the mixer Hamiltonian $\sum_i X^{(i)}$ for the original QAOA, the time evolution of the WSQAOA H_M applies a rotation with angle β around this vector to each qubit and can be implemented as $\bigotimes_{i=1}^{n} RY(\theta_i)RZ(-2\beta)RY(-\theta_i)$. Visually, this operation aligns the axis of rotation with the z-axis, applies the desired rotation and finally restores the original orientation of the Bloch sphere.

Instead of initializing the quantum state with the solution to a relaxation, a natural alternative is to use an approximate solution to the original problem as the initial state. In this case, each x_i^* is either 0 or 1, so the ground state of the mixer Hamiltonian is a computational basis state. This causes a problem, however, since now both the time-evolved mixer Hamiltonian and the time evolved problem Hamiltonian perform phase shifts, not changing the absolute values of the computational basis states' amplitudes. So, measuring the final state after running the circuit will simply return the initial state. To fix this, the authors of [14] propose choosing an $\epsilon \in [0, 0.5]$, setting $x_i' = \epsilon$ for $x_i^* = 0$ and $x_i' = 1 - \epsilon$ for $x_i^* = 1$ and using x' instead of x^* for the ground state $|\psi\rangle$ and the corresponding $\exp(-\mathrm{i}H_M\beta)$. The parameter ϵ can be used to interpolate between $|\psi\rangle = |x^*\rangle$ ($\epsilon = 0$) and $|\psi\rangle = \bigotimes_{i=1}^{n} |+\rangle$ ($\epsilon = 0.5$).

To approximate Max-cut with the WSQAOA, the authors of [14] suggest using the solution of the Goemans-Williamson algorithm (see Section 3.2) for the initial. They also suggest using $\epsilon = 0.25$ as well as a slightly different time-evolved mixer Hamiltonian, $\bigotimes_{i=1}^{n} RY(-\theta_i)RZ(-2\beta)RY(\theta_i)$, where the two RY rotations are swapped. With this change, the initial state is not the ground state of the mixer Hamiltonian anymore. However, the altered mixer Hamiltonian has the benefit of allowing the algorithm to easily restore the cut given by the initial state. This is because for $p = 1, \beta_1 = \frac{\pi}{2}, \gamma_1 = 0$, the state of the j-th qubit after applying the circuit is $-\mathrm{i}|1\rangle$ if $x_j^* = 0$ and $-\mathrm{i}|0\rangle$ otherwise. Measuring this state gives the output y with $y_j = -x_j^*$, which is the same cut as x^*, only the two partitions S and T are swapped. This implies that Max-cut-WSQAOA under ideal conditions yields a cut which is at least as good as the result of Goemans-Williamson. The same argument also applies to the Partition problem whose solutions have the same symmetry as the solutions for the Max-cut problem. This means the WSQAOA with the altered mixer Hamiltonian can reconstruct the Partition solution given by the initial state.

In this thesis, we will investigate two WSQAOA variants. The first variant is the one described in the previous paragraph, which prepares the initial with $\epsilon = 0.25$ as described above and which uses the altered mixer Hamiltonian to ensure that the algorithm can reconstruct the solution of the initial state. In addition, a version of QAOA proposed by [17] will be considered where the initial state is prepared with $\epsilon = 0.25$ as described in the previous paragraph, but the mixer Hamiltonian of the standard QAOA, $H_M = \sum_i X^{(i)}$, is used. This version will be called *WS-Init-QAOA* since it only changes the initial state, compared to the standard QAOA, leaving the time-evolved mixer Hamiltonian unchanged.

3.6.2. The Recursive QAOA

Instead of using the quantum circuit to approximate the optimal value for all variables at once, the main idea of the Recursive QAOA (RQAOA) is to iteratively reduce the problem size by eliminating variables until the problem is trivial to solve or until a predefined threshold is reached and the remaining problem can be solved using a classical optimizer [15], [16].

Suppose we are given an Ising model objective function C_n with n variables where all linear terms h_i are equal to 0:

$$\text{minimize } C_n(s) = -\sum_{i<j} J_{ij} s_i s_j$$

The Ising formulations of both Max-cut and Partition are of this form. Let H_n be the corresponding problem Hamiltonian.

We use the QAOA to find the state $|\psi\rangle = U(\vec{\beta}, \vec{\gamma})|\psi_0\rangle$ that minimizes $\langle\psi|H_n|\psi\rangle$. Then for all non-zero J_{ij}, we will compute $M_{ij} = \langle\psi|Z^{(i)}Z^{(j)}|\psi\rangle$ where $Z^{(i)}Z^{(j)}$ measures the parity of qubits i and j. In practice, this can be done by running the

circuit m times, each time measuring the qubits in the computational basis. This gives results $x^1, x^2, \ldots, x^m \in \{0,1\}^n$. We then, for each non-zero J_{ij}, compute

$$M_{ij} = \frac{1}{m} \sum_{k=1}^{m} (-1)^{x_i^k} (-1)^{x_j^k}.$$

M_{ij} lies in the interval $[-1, 1]$ by definition where $M_{ij} > 0$ can be interpreted as "s_i and s_j should have the same value" and $M_{ij} < 0$ can be interpreted as "s_i and s_j should have opposite values". We now pick the "most correlated" pair (s_k, s_l) which we will define as the pair (s_i, s_j) that maximizes $|M_{ij}|$. The key idea of the RQAOA is to add this correlation as a constraint into the problem Hamiltonian. So if $M_{kl} > 0$, we add the constraint $s_l = s_k$. If $M_{kl} < 0$, we add the constraint $s_l = -s_k$. We achieve this by replacing every instance s_l in the Ising cost function C_n with $\mathrm{sign}(M_{kl}) s_k$. In particular, we apply the following transformation:

$$J_{ij} s_i s_j \to \begin{cases} 0 & \text{if } i = k \text{ and } j = l \\ \mathrm{sign}(M_{kl}) J_{ij} s_i s_k & \text{if } i \neq k \text{ and } j = l \\ \mathrm{sign}(M_{kl}) J_{ij} s_k s_j & \text{if } i = k \text{ and } j \neq l \\ J_{ij} s_i s_j & \text{if } i \neq k \text{ and } j \neq l \end{cases} \tag{3.7}$$

The first case maps to zero since by replacing s_l with $\mathrm{sign}(M_{kl}) s_k$ we get the constant term $\mathrm{sign}(M_{kl}) J_{kl} s_k^2 = \mathrm{sign}(M_{kl}) J_{kl}$ which we can ignore since it is independent of the assignment of the variables. Because we have eliminated all instances of s_l, this new Ising cost function C_{n-1} has $n-1$ variables. Intuitively, if C_n is the cost function for a Max-cut instance, C_{n-1} corresponds to an instance of the weighted Max-cut problem for the graph which results from contracting the edge (k, l) in the original graph. Note that (3.7) might create multiple terms for a single pair of spin variables. For example, if J_{1l} and J_{1k} are non-zero, then $J_{1l} s_1 s_l$ will be transformed into $\mathrm{sign}(M_k l) J_{1j} s_1 s_k$ while the term $J_{1k} s_1 s_k$ stays unchanged. Both terms then need to be combined into a single term $(\mathrm{sign}(M_k l) J_{1j} + J_{1k}) s_1 s_k$ to conform to the Ising model. Since $\mathrm{sign}(M_k l) J_{1j} + J_{1k}$ might be zero, it is actually possible for C_{n-1} to contain fewer than $n-1$ variables.

We now repeat the process iteratively: We define the corresponding problem Hamiltonian H_{n-1} to the cost function C_{n-1}, we find the state $|\psi\rangle = U(\vec{\beta}, \vec{\gamma})|\psi_0\rangle$ that minimizes $\langle\psi|H_{n-1}|\psi\rangle$, we compute all $M_{ij} = \langle\psi|Z^{(i)}Z^{(j)}|\psi\rangle$, find the pair (s_k, s_l) whose $|M_{ij}|$ is maximum, apply (3.7) to define C_{n-2} and so on. We can terminate at the latest when we reach $C_1(s) = 0$. Then, we reconstruct a solution based on the previously encountered $\mathrm{sign}(M_{kl})$. We can interpret the spin variables as vertices and the $\mathrm{sign}(M_{kl})$ as edges of an undirected graph. To find the final solution, one variable in each connected component of this graph is set arbitrarily. The others are then fully determined by the $\mathrm{sign}(M_{kl})$ [15], [16].

4. Modeling noisy quantum circuits

Chapter 2 introduced state vectors, unitary operators and observables which can be used to model idealized, gate-based quantum computation. However, physical quantum computers experience various kinds of errors which distort the results of the final measurement. Some of these errors can be described using unitary operators, such as an incorrectly calibrated RX gate which implements the rotation $RX(\theta + \epsilon)$ instead of $RX(\theta)$. On the other hand, many typical error sources influence the state of the circuit in a non-unitary way. While the formalisms introduced in Chapter 2 assume that the circuit forms a *closed quantum system* without any interactions with the environment, in a physical quantum computer these kinds of interactions can never be eliminated completely resulting in noisy behavior of the circuit's state. Consequently, to analyze the behavior of noisy quantum circuits, we need to model them as *open systems*. Open systems can be viewed as subsystems of a larger, closed system. While a transformation on the closed, total system is always unitary, the same transformation might change the state of subsystems in a non-unitary way, which is then perceived as noise.

This chapter introduces *density matrices* and *quantum channels* which can be seen as the generalization of state vectors and unitary evolutions for open systems. Section 4.1 describes the *trace*, an operation from linear algebra which will be essential for the rest of this chapter, and some of its properties. Section 4.2 introduces the concept of density operators, the noisy equivalent of state vectors. In Section 4.3, the concept of the *partial trace* is introduced which is used to describe (open) subsystems of a closed quantum system. Section 4.4 describes quantum channels, which model the effects unitary transformations in a system have on a subsystem. Section 4.5 introduces the concept of *fidelity* to describe how much a quantum channel alters its inputs. Finally, Section 4.6 provides several examples of quantum channels for realistic noisy phenomena.

4.1. The trace

The trace of a square matrix A is defined as the sum of the diagonal entries

$$\mathrm{Tr}(A) = \sum_i A_{ii} = \sum_i \langle i|A|i\rangle, \tag{4.1}$$

where $\{|i\rangle\}$ is the computational/standard basis. From (4.1), it is easy to verify that the trace is linear, that is $\mathrm{Tr}(\alpha A + \beta B) = \alpha \,\mathrm{Tr}(A) + \beta \,\mathrm{Tr}(B)$, and cyclic: $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$.

For every orthonormal basis $\{|e_i\rangle\}$, there is a unitary U with $|e_i\rangle = U|i\rangle$. Thus,

$$\mathrm{Tr}(A) = \mathrm{Tr}(A\underbrace{UU^\dagger}_{\mathbb{1}}) \overset{\text{cyclicity}}{=} \mathrm{Tr}(U^\dagger A U) = \sum_i \langle i|U^\dagger A U|i\rangle = \sum_i \langle e_i|A|e_i\rangle.$$

In other words, the trace is independent of the basis. This insight leads to a useful identity: For every state $|\psi\rangle$ and every operator A,

$$\mathrm{Tr}(A|\psi\rangle\langle\psi|) = \sum_i \langle e_i|A|\psi\rangle\langle\psi|e_i\rangle \overset{|e_1\rangle=|\psi\rangle}{=} \langle\psi|A|\psi\rangle. \tag{4.2}$$

Here, we chose the orthonormal basis $\{|e_i\rangle\}$ such that $|e_1\rangle = |\psi\rangle$ to ensure that $\langle\psi|e_i\rangle = \delta_{i,1}$ [18], [30]. As a simple corollary to (4.2),

$$\mathrm{Tr}(|\psi\rangle\langle\psi|) = \mathrm{Tr}(\mathbb{1}|\psi\rangle\langle\psi|) = \langle\psi|\mathbb{1}|\psi\rangle = \langle\psi|\psi\rangle. \tag{4.3}$$

4.2. Mixed states and density operators

When the state of a quantum system is not fully known, for example because it was influenced by noise, we say it is in a *mixed state*. A mixed state can be thought of as a statistical mixture of pure states in the sense that the system is in the pure state $|\psi_i\rangle$ with probability p_i, where the p_i must add to 1. We represent mixed states using *density operators* or *density matrices*:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

A pure state $|\psi\rangle$ can be thought of as a special case of a mixed state where the system is in state $|\psi\rangle$ with probability 1. Naturally the density operator for state $|\psi\rangle$ is $|\psi\rangle\langle\psi|$. Density matrices can also represent statistical mixtures of mixed states. For two density operators ρ_1 and ρ_2, the combined operator

$$\rho = p\rho_1 + (1-p)\rho_2 = p\sum_i p_{1,i}|\psi_{1,i}\rangle\langle\psi_{1,i}| + (1-p)\sum_j p_{2,j}|\psi_{2,j}\rangle\langle\psi_{2,j}|$$

is also a density operator since $p\sum_i p_{1,i} + (1-p)\sum_j p_{2,j} = p + (1-p) = 1$.

Unitary evolutions, measurements and tensor products can all be described concisely in terms of density matrices, making them a useful formalism to describe both pure and mixed states:

Unitaries Suppose a unitary evolution U is applied to the mixed state $\rho = \sum_i p_i|\psi_i\rangle\langle\psi_i|$. Intuitively, the system should be in state $U|\psi_i\rangle$ with probability p_i after the unitary is applied. The resulting density operator is therefore

$$\rho' = \sum_i p_i U^\dagger|\psi_i\rangle\langle\psi_i|U \overset{\text{linearity}}{=} U^\dagger\rho U.$$

Measurements Suppose we measure the observable $M = \sum_m mP_m$. By (2.7), the conditional probability of getting the measurement result m, given the system is in pure state $|\psi_i\rangle$, is $\Pr(m|i) = \langle\psi_i|P_m|\psi_i\rangle$. Then, we can compute the probability of getting result m when measuring the mixed state given by the $|\psi_i\rangle$ and the p_i as follows:

$$\begin{aligned}
\Pr(m) &= \sum_i p_i \Pr(m|i) = \sum_i p_i \langle\psi_i|P_m|\psi_i\rangle \\
&= \sum_i p_i \operatorname{Tr}(P_m|\psi_i\rangle\langle\psi_i|) && \text{by (4.2)} \\
&= \operatorname{Tr}\left(P_m \sum_i p_i|\psi_i\rangle\langle\psi_i|\right) && \text{by linearity} \\
&= \operatorname{Tr}(P_m\rho)
\end{aligned}$$

Similarly, one can derive that after the measurement, the system will be in state $P_m\rho P_m^\dagger/\sqrt{\Pr(m)}$. Analogous to (2.8), the expected measurement result is $\operatorname{Tr}(M\rho)$.

Tensor products Suppose system A is in state $\rho^A = \sum_i p_i^A|\psi_i^A\rangle\langle\psi_i^A|$ and system B is in state $\rho^B = \sum_j p_j^B|\psi_j^B\rangle\langle\psi_j^B|$. Then the state of the combined system can be described as

$$\begin{aligned}
\sum_{i,j} p_i^A p_j^B|\psi_i^A,\psi_j^B\rangle\langle\psi_i^A,\psi_j^B| &= \sum_{i,j} p_i^A|\psi_i^A\rangle\langle\psi_i^A| \otimes p_j^B|\psi_j^B\rangle\langle\psi_j^B| \\
&= \sum_i p_i^A|\psi_i^A\rangle\langle\psi_i^A| \otimes \sum_j p_j^B|\psi_j^B\rangle\langle\psi_j^B| = \rho^A \otimes \rho^B.
\end{aligned}$$

Every density operator $\rho = \sum_i p_i|\psi_i\rangle\langle\psi_i|$ has the following properties:

1. It is Hermitian, that is $\rho = \rho^\dagger$. This is because the individual $|\psi_i\rangle\langle\psi_i|$ are Hermitian since

   $$\big(|\psi_i\rangle\langle\psi_i|\big)^\dagger = \big(\langle\psi_i|\big)^\dagger\big(|\psi_i\rangle\big)^\dagger = |\psi_i\rangle\langle\psi_i|,$$

   which extends to the whole operator by linearity.

2. $\operatorname{Tr}(\rho) = 1$:

   $$\operatorname{Tr}(\rho) \overset{\text{linearity}}{=} \sum_i p_i \operatorname{Tr}(|\psi_i\rangle\langle\psi_i|) \overset{(4.3)}{=} \sum_i p_i \underbrace{\langle\psi_i|\psi_i\rangle}_{=1} = \sum_i p_i = 1$$

3. It is positive, that is for every $|\varphi\rangle$, $\langle\varphi|\rho|\varphi\rangle \geq 0$:

$$\langle\varphi|\rho|\varphi\rangle = \langle\varphi| \left(\sum_i p_i |\psi_i\rangle\langle\psi_i| \right) |\varphi\rangle = \sum_i p_i \underbrace{\langle\varphi|\psi_i\rangle\langle\psi_i|\varphi\rangle}_{=|\langle\varphi|\psi_i\rangle|^2 \geq 0} \geq 0$$

Conversely, you can show that every operator satisfying these conditions is a density operator. From properties 1 and 2, it follows that the general density matrix for a single-qubit system is

$$\rho = \begin{pmatrix} a & b \\ b^* & 1-a \end{pmatrix}$$

with $a \in \mathbb{R}, b \in \mathbb{C}$. By convention, the density matrix is represented in the computational basis. Then, the diagonal entries describe the measurement probabilities when measuring the state in the computational basis. For example, the probability of finding the qubit in state $|0\rangle$ is given by $\mathrm{Tr}(|0\rangle\langle 0|\rho) = \langle 0|\rho|0\rangle = a$.

If the off-diagonal entries of a density matrix are zero, then the state is a statistical mixture of computational basis states. On the other hand, if the off-diagonal entries are non-zero, the state is in a superposition of computational basis states. For example, compare $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, the equal superposition of the two vectors of the 1-qubit computational basis, with the statistical mixture of these two vectors, each with probability $\frac{1}{2}$, also known as the *maximally mixed state*:

$$|+\rangle\langle+| = \frac{1}{2}\big(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| + |1\rangle\langle 1|\big) = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$\frac{1}{2}\big(|0\rangle\langle 0| + |1\rangle\langle 1|\big) = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The density matrix of the general, pure, single-qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is

$$|\psi\rangle\langle\psi| = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}.$$

Note that the relative phase information is also only captured by the off-diagonal entries.

Every single-qubit density matrix can be written as

$$\rho = \frac{\mathbb{1}_2 + r_x X + r_y Y + r_z Z}{2},$$

where X, Y, Z are the three Pauli matrices and $\vec{r} = (r_x, r_y, r_z)$ is a real three-dimensional vector with norm $||\vec{r}|| \leq 1$, called the *Bloch vector*. Each Bloch vector corresponds to a point inside the Bloch sphere (cf. Figure 1 on page 6). The points on the Bloch sphere are the pure states, matching the description given by (2.2). The maximally mixed state $\mathbb{1}/2$ is in the center of the Bloch ball [18], [30].

4.3. Subsystems and the partial trace

A mixed state can arise if the state preparation procedure involves randomness. However, even if a system is in a pure state, a subsystem of the system can still be in a mixed state. This fact makes density matrices especially useful for describing open quantum systems. Consider the situation where we have a quantum system made of two (possibly entangled) systems A and B. The state of the combined system is described by ρ^{AB}. Suppose we want to know ρ^A, the state of subsystem A. The operation which allows us to do this is called the *partial trace*, $\mathrm{Tr}_B(\cdot)$, and ρ^A is called the resulting *reduced density operator*: $\rho^A = \mathrm{Tr}_B(\rho^{AB})$.

For $|a\rangle$ and $|a'\rangle$, two vectors in the state space of A, and $|b\rangle$ and $|b'\rangle$, two vectors in the state space of B, the partial trace $\mathrm{Tr}_B(\cdot)$ is defined as follows:

$$\mathrm{Tr}_B(|a\rangle\langle a'| \otimes |b\rangle\langle b'|) = |a\rangle\langle a'| \,\mathrm{Tr}(|b\rangle\langle b'|) = |a\rangle\langle a'|\langle b|b'\rangle$$

By requiring the partial trace to be linear in its input, we can extend the definition to arbitrary operators. This leads to the more explicit definition

$$\mathrm{Tr}_B(\rho_{AB}) = \sum_j (\mathbb{1}_A \otimes \langle e_j|)\rho_{AB}(\mathbb{1}_A \otimes |e_j\rangle). \tag{4.4}$$

Here, $\mathbb{1}_A$ is the identity operator on system A and $\{|e_j\rangle\}$ is an orthonormal basis for system B. Note that the trace is independent of the choice of basis. Tr_A is defined symmetrically. The operation of computing the reduced density operator $\rho^A = \mathrm{Tr}_B(\rho^{AB})$ is called *tracing out system* B.

If ρ^{AB} is separable, that is $\rho^{AB} = \rho^A \otimes \rho^B$, then $\mathrm{Tr}_B(\rho^A \otimes \rho^B) = \rho^A \,\mathrm{Tr}(\rho^B) = \rho^A$ which aligns with the intuition that ρ^A should describe the state of subsystem A. To justify that the partial trace is a sensible definition, even for entangled subsystems, one has to show that ρ^A captures the complete characteristics of any possible measurement.

Let M be any observable for system A. Then $M' = M \otimes \mathbb{1}$ is the corresponding observable for system AB, where we only measure subsystem A, ignoring subsystem B. M' should give the same result in expectation as M. That is, $\mathrm{Tr}(M'\rho^{AB}) = \mathrm{Tr}(M\rho^A)$. It can be shown that the partial trace $\rho^A = \mathrm{Tr}_B(\rho^{AB})$ is the only function satisfying this condition for every observable. Intuitively this is because for computing $\mathrm{Tr}(M'\rho^{AB})$, we need to sum over all basis vectors $\langle a_i|\langle b_j|M'\rho^{AB}|a_i\rangle|b_j\rangle$. In a sense, $\rho^A = \mathrm{Tr}_B(\rho^{AB})$ captures the result of summing over all $|b_j\rangle$ so one only needs to sum over all $|a_i\rangle$ to compute the expected measurement result. A rigorous proof is given, for example, by [31].

As mentioned earlier, if a system is in a pure state, it is possible that a subsystem of this system is in a mixed state. For example, consider the 2-qubit state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, often denoted as $|\Phi^+\rangle$. The density matrix representation of this

state is $|\Phi^+\rangle\langle\Phi^+| = 1/2(|00\rangle + |11\rangle)(\langle 00| + \langle 11|)$. When we trace out the second qubit, we get

$$\begin{aligned}
\mathrm{Tr}_2(|\Phi^+\rangle\langle\Phi^+|) &= \mathrm{Tr}_2\left(\frac{1}{2}(|00\rangle + |11\rangle)(\langle 00| + \langle 11|)\right) \\
&= \frac{1}{2}\big(\mathrm{Tr}_2(|00\rangle\langle 00|) + \mathrm{Tr}_2(|00\rangle\langle 11|) + \mathrm{Tr}_2(|11\rangle\langle 00|) + \mathrm{Tr}_2(|11\rangle\langle 11|)\big) \\
&= \frac{1}{2}\big(|0\rangle\langle 0|\underbrace{\langle 0|0\rangle}_{=1} + |0\rangle\langle 1|\underbrace{\langle 1|0\rangle}_{=0} + |1\rangle\langle 0|\underbrace{\langle 0|1\rangle}_{=0} + |1\rangle\langle 1|\underbrace{\langle 1|1\rangle}_{=1}\big) \\
&= \frac{1}{2}\big(|0\rangle\langle 0| + |1\rangle\langle 1|\big),
\end{aligned}$$

which is the maximally mixed state.

4.4. Quantum channels and Kraus operators

The same way density matrices can be seen as the generalization of state vectors for open systems, *quantum channels* are the generalization of unitary operators for open systems. In particular, the effects unitary evolutions of the total system have on a subsystem can be described as quantum channels. Formally, quantum channels are defined as *completely-positive trace preserving maps* (CPTP maps). CPTP maps can be seen as functions of the form $\mathcal{E} : \mathbb{C}^{n\times n} \to \mathbb{C}^{m\times m}$ mapping density matrices to other density matrices. By definition, a CPTP map $\mathcal{E}$ has the following properties:

1. It is *linear*, that is $\mathcal{E}\left(\sum_i p_i\rho_i\right) = \sum_i p_i\mathcal{E}(\rho_i)$. Intuitively, if ρ is a statistical mixture of states ρ_i, then $\mathcal{E}(\rho)$ should result in the same state as preparing state $\mathcal{E}(\rho_i)$ with probability p_i, which is exactly what the linearity property describes.
2. It is *trace-preserving*, that is $\mathrm{Tr}(\mathcal{E}(\rho)) = \mathrm{Tr}(\rho)$. The resulting operator must have the same trace (1) to be a valid density operator
3. It is *positive*, that is for every positive operator ρ, $\mathcal{E}(\rho)$ is positive. This again is to ensure that $\mathcal{E}$ maps density matrices to density matrices, which are positive operators.
4. It is *completely positive*, that is for any other density operator ρ' with arbitrary dimension, $(\mathcal{I}\otimes\mathcal{E})(\rho'\otimes\rho)$ is positive if $\rho'\otimes\rho$ is positive. Here, $\mathcal{I}$ denotes the identity map. This ensures that if $\mathcal{E}$ is applied to a subsystem, the operator for the total system remains a valid density operator. While any completely positive operator is also positive, the converse does not hold necessarily.

There are multiple possible representations for quantum channels/CPTP maps. In the scope of this thesis, we will focus on one particular representation, called

operator-sum representation, also known as *Kraus representation* where the map is described in the form of a set of matrices $\{K_i\}$, the *Kraus operators*:

$$\mathcal{E}(\rho) = \sum_i K_i \rho K_i^\dagger$$

The K_i must satisfy the constraint $\sum_i K_i^\dagger K_i = \mathbb{1}$. This constraint is also known as the *completeness relation*. It ensures that the map preserves the trace of the density operator:

$$\begin{aligned}\mathrm{Tr}(\mathcal{E}(p)) = \mathrm{Tr}\left(\sum_j K_j \rho K_j^\dagger\right) &\overset{\text{linearity + cyclicity}}{=} \mathrm{Tr}\left(\sum_j K_j^\dagger K_j \rho\right) \\ &\overset{\text{linearity}}{=} \mathrm{Tr}\left(\left(\sum_j K_j^\dagger K_j\right)\rho\right) = \mathrm{Tr}(\mathbb{1}\rho)\end{aligned}$$

Kraus' theorem states that a map is a CPTP map if and only if it can be written in Kraus representation. Note that the Kraus representation of a particular quantum channel is not necessarily unique. For instance, the two sets of Kraus operators $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ and $\left\{\frac{1}{\sqrt{2}}\mathbb{1}_2, \frac{1}{\sqrt{2}}Z\right\}$ both describe the same channel

$$\mathcal{E} : \begin{pmatrix} a & b \\ b^* & 1-a \end{pmatrix} \mapsto \begin{pmatrix} a & 0 \\ 0 & 1-a \end{pmatrix}.$$

Just like unitaries, assuming they have the correct dimensions, two quantum channels, $\mathcal{A}$ with Kraus operators $\{A_i\}$ and $\mathcal{B}$ with Kraus operators $\{B_j\}$, can be composed sequentially

$$(\mathcal{B} \circ \mathcal{A})(\rho) = \mathcal{B}(\mathcal{A}(\rho)) = \sum_j B_j \left(\sum_i A_i \rho A_i^\dagger\right) B_j^\dagger = \sum_{i,j} B_j A_i \rho A_i^\dagger B_j^\dagger \tag{4.5}$$

and in parallel

$$(\mathcal{A} \otimes \mathcal{B})(\rho) = \sum_{i,j} (A_i \otimes B_j)\rho(A_i^\dagger \otimes B_j^\dagger). \tag{4.6}$$

It is easy to check that if $\mathcal{A}$ and $\mathcal{B}$ satisfy the completeness relation then so do $\mathcal{B} \circ \mathcal{A}$ and $\mathcal{A} \otimes \mathcal{B}$.

Noisy, non-unitary behavior of a quantum circuit comes from interactions with the environment. These interactions can be thought of as unitary evolutions on the combined system that includes both the circuit and its environment. Therefore, we can describe interactions with the environment as a quantum channel in three steps: add the environment, apply a unitary to the combined system and trace out the environment. Each of these steps can be represented as a quantum channel in Kraus representation, so their sequential composition is also a valid quantum channel. The three steps are modeled as follows:

1. **Add the environment** Without loss of generality, we can assume that the environment is in some pure state $|e_1\rangle\langle e_1|$. If the environment were not in a pure state, we could think of it as being part of an even larger system. Combine ρ with the environment $|e_1\rangle\langle e_1|$ to get $\rho' = \rho \otimes |e_1\rangle\langle e_1|$. This can be written as a quantum channel with the single Kraus operator $\mathbb{1} \otimes |e_1\rangle$:

$$\begin{aligned}\mathcal{E}(\rho) &= (\mathbb{1} \otimes |e_1\rangle)\rho(\mathbb{1} \otimes \langle e_1|) = (\mathbb{1} \otimes |e_1\rangle)(\rho \otimes \mathbb{1}_1)(\mathbb{1} \otimes \langle e_1|) \\ &= (\mathbb{1}\rho\mathbb{1}) \otimes (|e_1\rangle\mathbb{1}_1\langle\psi|) = \rho \otimes |e_1\rangle\langle e_1|\end{aligned}$$

   The completeness relation is satisfied: $(\mathbb{1} \otimes \langle e_1|)(\mathbb{1} \otimes |e_1\rangle) = \mathbb{1} \otimes \langle e_1|e_1\rangle = \mathbb{1}$.

2. **Apply a unitary to the combined system** Applying the unitary transformation U to system ρ' is a quantum channel with the single Kraus operator U since unitary evolution on density matrices is described as $U\rho'U^\dagger$ and since $U^\dagger U = \mathbb{1}$.

3. **Trace out the environment** By setting $K_j = \mathbb{1} \otimes \langle e_i|$ for the orthonormal basis $\{|e_i\rangle\}$, we can define the partial trace in terms of Kraus operators, matching the definition (4.4). Again, this satisfies the completeness relation:

$$\sum_i (\mathbb{1} \otimes |e_i\rangle)(\mathbb{1} \otimes \langle e_i|) = \mathbb{1} \otimes \left(\sum_i |e_i\rangle\langle e_i|\right) = \mathbb{1} \otimes \mathbb{1} = \mathbb{1}$$

   Here, we have used the fact that $\sum_i |e_i\rangle\langle e_i| = \mathbb{1}$ for any orthonormal basis $\{|e_i\rangle\}$ since

$$\left(\sum_i |e_i\rangle\langle e_i|\right) \sum_j \alpha_i |e_j\rangle = \sum_{i,j} \alpha_j |e_i\rangle \underbrace{\langle e_i|e_j\rangle}_{=\delta_{ij}} = \sum_j \alpha_j |e_j\rangle.$$

According to (4.5), the channel describing the whole sequence of adding the environment, applying a unitary and tracing out the environment can be described by the following set of Kraus operators:

$$K_i = (\mathbb{1} \otimes \langle e_i|)U(\mathbb{1} \otimes |e_1\rangle) \tag{4.7}$$

Conversely, for every set of square Kraus operators K_i, one can find a unitary U that satisfies (4.7).

Single-qubit quantum channels can be interpreted visually in terms of the Bloch ball. Just as single-qubit quantum gates (unitary operators) can be interpreted as rotations of the Bloch sphere, quantum channels can be seen as affine transformations mapping the Bloch ball into itself [18].

4.5. Fidelity

The *fidelity* is a measure of the "distance" between two quantum states. It is also a useful measure to describe how much a noisy quantum channel alters its input compared to an idealized, unitary operation. Given two states represented as density matrices ρ and σ, their fidelity $F(\rho, \sigma)$ is given by:

$$F(\rho, \sigma) = \operatorname{Tr}\left(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2 \tag{4.8}$$

It can be shown that the fidelity between any two states is always between 0 and 1 ($0 \leq F(\rho, \sigma) \leq 1$ for all ρ, σ) with $F(\rho, \sigma) = 1$ if and only if $\rho = \sigma$. It can also be shown that the fidelity is symmetric: $F(\rho, \sigma) = F(\sigma, \rho)$ [32].

Assuming ρ is a pure state, that is $\rho = |\psi\rangle\langle\psi|$, then $\sqrt{\rho} = \rho$ since $|\psi\rangle\langle\psi|\psi\rangle\langle\psi| = |\psi\rangle\langle\psi|$. If at least one of the two states is pure, (4.8) can thus be simplified to

$$\begin{aligned} F(|\psi\rangle\langle\psi|, \sigma) &= \operatorname{Tr}\left(\sqrt{|\psi\rangle\langle\psi|\sigma|\psi\rangle\langle\psi|}\right)^2 = \operatorname{Tr}\left(\sqrt{\langle\psi|\sigma|\psi\rangle}\,|\psi\rangle\langle\psi|\right)^2 \\ &= \left(\sqrt{\langle\psi|\sigma|\psi\rangle}\underbrace{\operatorname{Tr}(|\psi\rangle\langle\psi|)}_{=1}\right)^2 = \langle\psi|\sigma|\psi\rangle \qquad \text{by linearity.} \end{aligned}$$

Some authors use an alternative definition of fidelity: $F'(\rho, \sigma) = \sqrt{F(\rho, \sigma)}$, for example in [18]. In the context of this thesis, however, we will only use the definition (4.8), as given in [32] or [33], for example.

Let $\mathcal{E}$ be a quantum channel which maps d-dimensional density matrices to d-dimensional density matrices. The *average fidelity* of $\mathcal{E}$ is defined as

$$\bar{F}(\mathcal{E}) = \int \langle\psi|\,\mathcal{E}\big(|\psi\rangle\langle\psi|\big)\,|\psi\rangle\,\mathrm{d}\psi.$$

So, informally, the average fidelity of a quantum channel describes how much it changes its input state, averaged over all possible pure input states. A closed formula for computing the average fidelity of a channel exists [34]. In particular, for a single qubit channel [35],

$$\bar{F}(\mathcal{E}) = \frac{1}{2} + \frac{1}{12}\sum_{P\in\{X,Y,Z\}} \operatorname{Tr}(P\mathcal{E}(P)). \tag{4.9}$$

One can also define the average fidelity of a channel with respect to a unitary operator U to describe how well the channel approximates U:

$$\bar{F}(\mathcal{E}, U) = \int \langle\psi|U^\dagger\mathcal{E}\big(|\psi\rangle\langle\psi|\big)U|\psi\rangle\,\mathrm{d}\psi$$

Since $U^\dagger\mathcal{E}\big(|\psi\rangle\langle\psi|\big)U = \mathcal{U}^\dagger \circ \mathcal{E}$ with $\mathcal{U}^\dagger(\rho) = U^\dagger\rho U$,

$$\bar{F}(\mathcal{E}, U) = \bar{F}(\mathcal{U}^\dagger \circ \mathcal{E}). \tag{4.10}$$

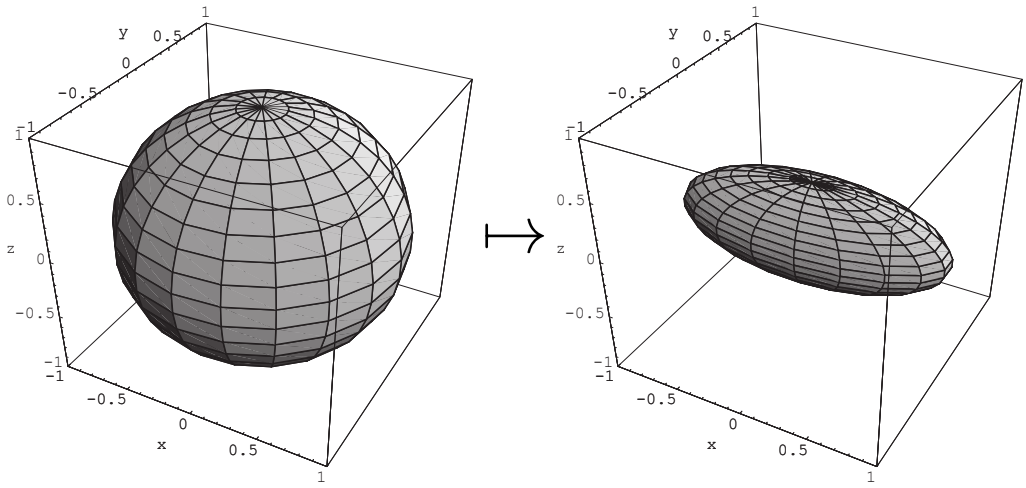


Figure 3.: Visualization of the bit flip channel ($p = 0.3$) as an affine transformation of the Bloch ball [18]

4.6. Noisy quantum channels

This section introduces some common quantum channels that are used to describe typical sources of error in quantum computers.

4.6.1. The bit flip channel

The *bit flip channel* models the situation where the value of a qubit flips with probability p:

$$\mathcal{E}_{BF}(\rho) = (1-p)\mathbb{1}\rho\mathbb{1} + pX\rho X$$

The corresponding Kraus operators are therefore $K_1 = \sqrt{1-p}\mathbb{1}, K_2 = \sqrt{p}X$. The completeness relation is satisfied since

$$(1-p)\underbrace{\mathbb{1}\mathbb{1}}_{\mathbb{1}} + p\underbrace{XX}_{\mathbb{1}} = (1-p+p)\mathbb{1} = \mathbb{1}. \tag{4.11}$$

In terms of the Bloch ball, the bit flip channel can be interpreted as a contraction towards the x-axis, as shown in Figure 3 [18].

4.6.2. The phase flip channel

The *phase flip channel* applies a phase flip, that is a phase shift of $180°$, with probability p:

$$\mathcal{E}_{PF}(\rho) = (1-p)\mathbb{1}\rho\mathbb{1} + pZ\rho Z \tag{4.12}$$

The Kraus operators are $K_1 = \sqrt{1-p}\mathbb{1}, K_2 = \sqrt{p}Z$. Similar to (4.11), the completeness relation is satisfied. The phase flip channel contracts the Bloch ball towards the z-axis (cf. Figure 4) [18].

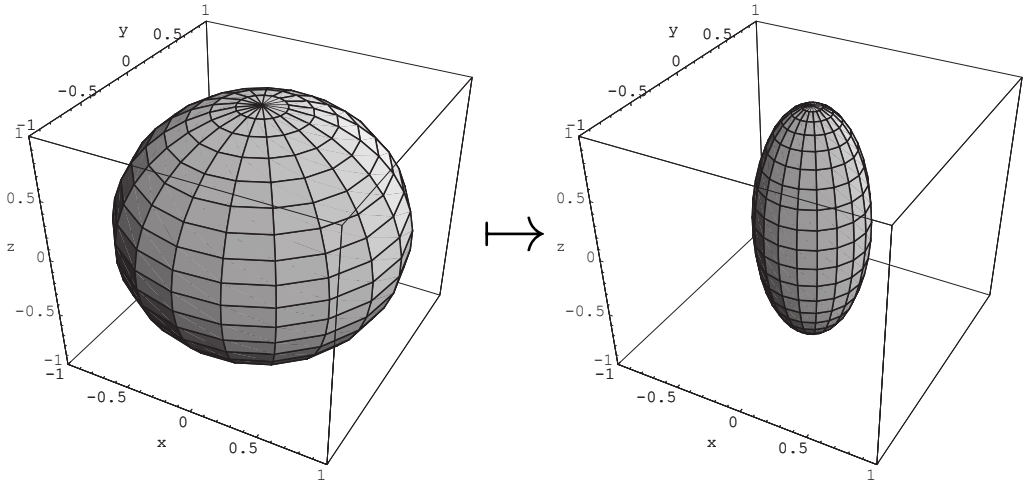


Figure 4.: Visualization of the phase flip channel ($p = 0.3$) [18]

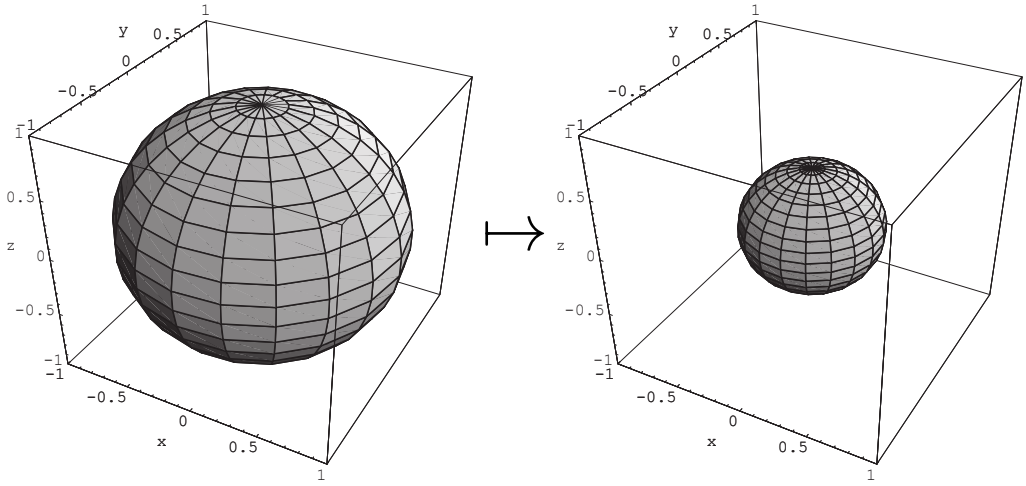


Figure 5.: Visualization of the depolarizing channel ($p = 0.5$) [18]

4.6.3. The depolarizing channel

The *depolarizing channel* describes a situation where, with some probability p, the state of a qubit is replaced with the maximally mixed state $\mathbb{1}/2$:

$$\mathcal{E}_D(\rho) = (1-p)\rho + p \cdot \frac{\mathbb{1}}{2}$$

Unlike the bit flip and the phase flip channels, the depolarizing channel uniformly contracts the Bloch ball towards the maximally mixed state, as visualized by Figure 5. By using the fact that $\mathbb{1} = (\rho + X\rho X + Y\rho Y + Z\rho Z)/2$ for all ρ, we can write $\mathcal{E}_D$ as

$$\mathcal{E}_D(\rho) = (1 - 3p/4)\rho + p/4(X\rho X + Y\rho Y + Z\rho Z).$$

This means, we can represent this channel using the Kraus operators $K_1 = \sqrt{1-3p/4}\mathbb{1}$, $K_2 = \sqrt{p/4}X$, $K_3 = \sqrt{p/4}Y$, $K_4 = \sqrt{p/4}Z$. Since $P^\dagger P = \mathbb{1}$ for every

Pauli matrix P, it is again easy to see that these Kraus operators satisfy the completeness relation $\sum_{i=1}^{4} K_i = \mathbb{1}$. From its Kraus representation, it follows that the depolarizing channel can alternatively be interpreted as a "random change" of the state with probability $3p/4$ where a change refers to applying one of the three Pauli gates with equal probability [18].

The depolarizing channel can be generalized for multi-qubit states:

$$\mathcal{E}_{D,n}(\rho) = (1-p)\rho + p \cdot \frac{\mathbb{1}}{2^n} \tag{4.13}$$

Here, n is the number of qubits. It is possible to find a decomposition of the 2^n-dimensional identity, similar to the one-qubit case above, by using the n-fold tensor products of the Pauli matrices (including the identity) as a basis:

$$\mathcal{B}_n = \bigcup_{(P_1,\ldots,P_n)\in\{\mathbb{1},X,Y,Z\}^n} \left\{ \bigotimes_{j=1}^{n} P_j \right\},$$

$$\mathbb{1} = \frac{1}{2^n} \sum_{B\in\mathcal{B}_n} B^\dagger \rho B$$

Again, this holds for any density operator ρ. The Kraus decomposition of the n-qubit depolarizing channel is therefore

$$\mathcal{E}_{D,n}(\rho) = \left(1 - p + \frac{p}{4^n}\right) \mathbb{1}\rho\mathbb{1} + \sum_{B\in\mathcal{B}_n\setminus\{\mathbb{1}\}} \frac{p}{4^n} B^\dagger \rho B.$$

Given any pure state $|\psi\rangle$, the fidelity of $|\psi\rangle\langle\psi|$ and $\mathcal{E}_{D,n}(|\psi\rangle\langle\psi|)$ is

$$\begin{aligned} F(\mathcal{E}_{D,n}(|\psi\rangle\langle\psi|), |\psi\rangle\langle\psi|) &= \langle\psi| \left((1-p)|\psi\rangle\langle\psi| + p\frac{\mathbb{1}}{2^n} \right) |\psi\rangle \\ &= (1-p) \underbrace{\langle\psi|\psi\rangle^2}_{=1} + \frac{p}{2^n} \underbrace{\langle\psi|\psi\rangle}_{=1} = 1 - p + \frac{p}{2^n}. \end{aligned} \tag{4.14}$$

Since the fidelity does not depend on the state $|\psi\rangle$, the average fidelity $\bar{F}(\mathcal{E}_{D,n})$ is also $\bar{F}(\mathcal{E}_{D,n}) = 1 - p + p/2^n$ [36].

If $\mathcal{E}$ is some quantum channel, we can compute $F(\mathcal{E}_D(\mathcal{E}(|\psi\rangle\langle\psi|)), |\psi\rangle\langle\psi|)$ analogously to (4.14). Doing this, we find that the average fidelity of the composed channel $\mathcal{E}_{D,n} \circ \mathcal{E}$ is [37]

$$\bar{F}(\mathcal{E}_{D,n} \circ \mathcal{E}) = (1-p)\bar{F}(\mathcal{E}) + p/2^n. \tag{4.15}$$

4.6.4. The amplitude damping channel

The *amplitude damping channel* describes a situation where a qubit in state $|0\rangle$ stays unaffected but a qubit in state $|1\rangle$ flips to $|0\rangle$ with some probability p.

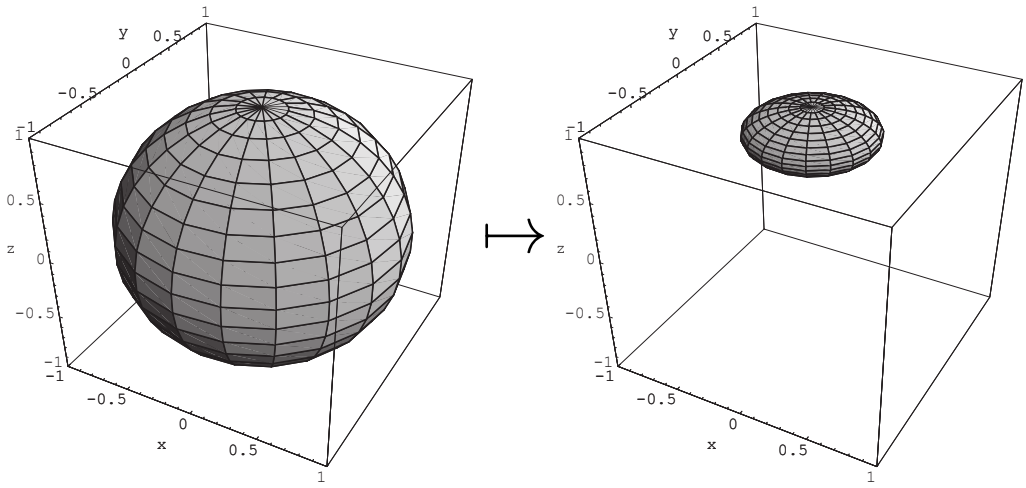


Figure 6.: Visualization of the amplitude damping channel ($p = 0.8$) [18]

This can happen due to the qubit losing energy to its environment in form of a photon. Suppose $|0\rangle$ represents the absence of a photon and $|1\rangle$ represents the presence of a photon. Then, this scenario can be represented by a unitary transformation U with $U|00\rangle = |00\rangle, U|01\rangle = \sqrt{1-p}|01\rangle + \sqrt{p}|10\rangle$. This unitary can be described as

$$U = |00\rangle\langle 00| + \left(\sqrt{1-p}|01\rangle + \sqrt{p}|10\rangle\right)\langle 01| + |\psi\rangle\langle 10| + |\varphi\rangle\langle 11|$$

for some unknown states $|\psi\rangle$ and $|\varphi\rangle$. By (4.7), the Kraus operators describing the effect of this transformation on the second qubit are

$$K_1 = \big(\langle 0| \otimes \mathbb{1}\big)U\big(|0\rangle \otimes \mathbb{1}\big) = |0\rangle\langle 0| + \sqrt{1-p}|1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix},$$
$$K_2 = \big(\langle 1| \otimes \mathbb{1}\big)U\big(|0\rangle \otimes \mathbb{1}\big) = \sqrt{p}|1\rangle\langle 0| = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix}.$$

The resulting quantum channel $\mathcal{E}_{AD}$ acts on a general qubit state as follows:

$$\mathcal{E}_{AD} : \begin{pmatrix} a & b \\ b^* & 1-a \end{pmatrix} \mapsto \begin{pmatrix} (a-1)(1-p)+1 & \sqrt{1-p}\,b \\ \sqrt{1-p}\,b^* & (1-a)(1-p) \end{pmatrix}$$

This causes the Bloch ball to shrink towards the north pole $|0\rangle\langle 0|$, as shown in Figure 6. If probability p is 1, then $\mathcal{E}_{AD}(\rho) = |0\rangle\langle 0|$ for every ρ.

The amplitude damping channel assumes that energy is always transferred from the qubit to its environment as if the environment had zero temperature. The scenario where the environment's temperature is greater than zero can be described using the *generalized amplitude damping channel*, which shrinks the Bloch ball towards the mixed state $a_0|0\rangle\langle 0| + (1-a_0)|1\rangle\langle 1|$:

$$\mathcal{E}_{GAD} : \begin{pmatrix} a & b \\ b^* & 1-a \end{pmatrix} \mapsto \begin{pmatrix} (a-a_0)(1-p)+a_0 & \sqrt{1-p}\,b \\ \sqrt{1-p}\,b^* & (a_0-a)(1-p)+1-a_0 \end{pmatrix}$$

The generalized amplitude damping channel can be represented with the Kraus operators [18]

$$K_1 = \sqrt{a_0}\begin{pmatrix}1 & 0\\ 0 & \sqrt{1-p}\end{pmatrix}, \qquad K_2 = \sqrt{a_0}\begin{pmatrix}0 & \sqrt{p}\\ 0 & 0\end{pmatrix},$$
$$K_3 = \sqrt{1-a_0}\begin{pmatrix}\sqrt{1-p} & 0\\ 0 & 1\end{pmatrix}, \qquad K_4 = \sqrt{1-a_0}\begin{pmatrix}0 & 0\\ \sqrt{p} & 0\end{pmatrix}.$$

4.6.5. The phase damping channel

The *phase damping channel* models an interaction of the qubit with its environment without a transfer of energy. Consider the situation where, with probability p, a qubit in state $|\psi\rangle$ interacts with the environment in state $|0\rangle$. If the qubit's state is $|0\rangle$, then the environment transitions to state $|1\rangle$. If qubit's state it $|1\rangle$, the environment transitions to state $|2\rangle$. The corresponding unitary is

$$U = \Big(\sqrt{1-p}|00\rangle + \sqrt{p}|10\rangle\Big)\langle 00| + \Big(\sqrt{1-p}|01\rangle + \sqrt{p}|21\rangle\Big)\langle 01| + R,$$

where R denotes the remaining terms. The Kraus operators are

$$K_1 = \big(\langle 0|\otimes \mathbb{1}\big)U\big(|0\rangle\otimes\mathbb{1}\big) = \begin{pmatrix}\sqrt{1-p} & 0\\ 0 & \sqrt{1-p}\end{pmatrix},$$
$$K_2 = \big(\langle 1|\otimes \mathbb{1}\big)U\big(|0\rangle\otimes\mathbb{1}\big) = \begin{pmatrix}\sqrt{p} & 0\\ 0 & 0\end{pmatrix},$$
$$K_3 = \big(\langle 2|\otimes \mathbb{1}\big)U\big(|0\rangle\otimes\mathbb{1}\big) = \begin{pmatrix}0 & 0\\ 0 & \sqrt{p}\end{pmatrix}.$$

The corresponding quantum channel can be written as [38]

$$\mathcal{E}_{PD} : \begin{pmatrix}a & b\\ b^* & 1-a\end{pmatrix} \mapsto \begin{pmatrix}a & (1-p)b\\ (1-p)b^* & 1-a\end{pmatrix}.$$

It reduces the non-diagonal entries containing the qubit's relative phase information while keeping the diagonal entries the same.

An alternative definition of the dephasing channel is [18]

$$\mathcal{E}'_{PD}(\rho) = K_1\rho K_1^\dagger + K_2\rho K_2^\dagger$$
$$\text{with } K_1 = \begin{pmatrix}1 & 0\\ 0 & \sqrt{1-\gamma}\end{pmatrix}, \quad K_2 = \begin{pmatrix}0 & 0\\ 0 & \sqrt{\gamma}\end{pmatrix}.$$

By setting, $\gamma = 1-(1-p)^2$, these two definitions are equivalent. The dephasing channel is also equivalent to the phase flip channel (4.12) with probability $p/2$.

4.6.6. The thermal relaxation channel

If a physical qubit is left untouched, over time it will lose energy to its environment. This process is called thermal relaxation or thermalization and it describes the process of a qubit slowly transitioning into the thermal equilibrium state $a_0|0\rangle\langle 0| + (1-a_0)|1\rangle\langle 1|$. The probability of a qubit being in a state which is not the equilibrium state decreases exponentially over time. Thermal relaxation is governed by two parameters: T_1 is the so-called *longitudinal relaxation time*, which describes the rate of motion along the z-axis of the Bloch sphere towards the equilibrium state. T_2 is the *transverse relaxation time*, which describes a shrinking of the x and y components of the Bloch vector. Intuitively, if T_2 is large, then the qubit can hold a superposition of computational basis states for longer. Both T_1 and T_2 can be measured empirically for physical systems [39].

Thermal relaxation over a time span of t can be modeled using the following channel [18], [39]:

$$\mathcal{E}_{TR} : \begin{pmatrix} a & b \\ b^* & 1-a \end{pmatrix} \mapsto \begin{pmatrix} (a-a_0)\mathrm{e}^{-t/T_1} + a_0 & b\mathrm{e}^{-t/T_2} \\ b^*\mathrm{e}^{-t/T_2} & (a_0-a)\mathrm{e}^{-t/T_1} + 1 - a_0 \end{pmatrix} \tag{4.16}$$

By applying a generalized amplitude damping channel with reset probability $p = 1-\mathrm{e}^{-t/T_1}$, we can transform the diagonal elements according to (4.16). It also scales the off-diagonal elements by a factor of $\sqrt{1-p} = \mathrm{e}^{-t/T_1/2}$. So, to achieve the desired off-diagonal scaling factor of e^{-t/T_2}, we will apply a phase damping channel with dephasing probability $p = 1 - \mathrm{e}^{t/T_1/2 - t/T_2}$. Notice that for p to be at least 0, $T_2 \leq 2T_1$. This is the case for the superconducting qubits modeled in this thesis [39]. Therefore, the map (4.16) can be implemented by sequentially composing the generalized amplitude damping channel and the phase damping channel: $\mathcal{E}_{PD} \circ \mathcal{E}_{GAD}$ [18]. If $a_0 = 1$, we can use the regular amplitude damping channel instead of the generalized.

By (4.9), the average fidelity of the thermal relaxation channel with $a_0 = 1$ is

$$\bar{F}(\mathcal{E}_{TR}) = \frac{1}{2} + \frac{1}{6}\exp(-t/T_1) + \frac{1}{3}\exp(-t/T_2). \tag{4.17}$$

4.6.7. The asymmetric bit flip channel

The asymmetric bit flip channel applies a bit flip with probability $p_{0\to 1}$ if the qubit is in state $|0\rangle$ and a bit flip with probability $p_{1\to 0}$ if the qubit is in state $|1\rangle$. To derive the Kraus operators for this channel, one can consider a situation similar to the one described in Section 4.6.4. Instead of one, there are two environment qubits, which initially are in state $|01\rangle$, meaning the second one stores a photon. Like above, if the circuit qubit contains a photon, it may lose it to the first environment qubit with probability $p_{1\to 0}$. In addition, if the circuit

qubit does not contain a photon, it can gain one from the second environment qubit. So, the scenario can be modeled using the unitary

$$U|010\rangle \mapsto (1-p_{0\to1})|010\rangle + p_{0\to1}|001\rangle,$$
$$U|011\rangle \mapsto (1-p_{1\to0})|011\rangle + p_{1\to0}|110\rangle,$$

yielding the Kraus operators

$$K_1 = \begin{pmatrix} \sqrt{1-p_{0\to1}} & 0 \\ 0 & \sqrt{1-p_{1\to0}} \end{pmatrix}, \quad K_2 = \begin{pmatrix} \sqrt{p_{0\to1}} & 0 \\ 0 & 0 \end{pmatrix}, \quad K_3 = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{p_{1\to0}} \end{pmatrix},$$

which satisfy the completeness relation. The fourth Kraus operator is zero.

The asymmetric bit flip channel $\mathcal{E}_{AB}$ can be used to simulate readout errors where measuring the state of a qubit might give incorrect results, even if it is in a computational basis state. For this application, $p_{0\to1}$ describes the probability of incorrectly measuring $|0\rangle$ as $|1\rangle$ and $p_{1\to0}$ is the probability of incorrectly measuring $|1\rangle$ as $|0\rangle$. The diagonal entries of the density matrix correspond to the measuring probabilities for the computational basis states and the channel transforms these according to

$$\operatorname{diag}\left(\mathcal{E}_{AB}\begin{pmatrix} a & b \\ b^* & 1-a \end{pmatrix}\right) = \begin{pmatrix} (1-p_{0\to1})a + p_{1\to0}(1-a) \\ (1-p_{1\to0})(1-a) + p_{0\to1}a \end{pmatrix},$$

which the correct probabilities for this scenario.

5. Design and implementation of the performance analysis

In this thesis, we analyze the performance of the QAOA, WSQAOA, WS-Init-QAOA and RQAOA on random instances of the problems Max-cut and Partition for different levels of noise using numerical simulations. This chapter describes how these simulations were performed and discusses possible limitations and extensions. Sections 5.1 and 5.2 describe which problem instances and algorithms were considered for the analysis. Section 5.3 describes the design and parameter selection of the noise model used for the simulations. Sections 5.4, 5.5 and 5.6 deal with three additional aspects (sampling, gate set transpilation and circuit connectivity) and how they were taken into account for the simulations. Finally, Section 5.7 is about the Qaptiva 800 platform used for the simulation. This section explains how the simulations were implemented and discusses the implementation-relevant features and limitations of the platform.

5.1. Description of the analyzed problem instances

The following problem instances are considered:

- **Max-cut:** For every $n \in \{5, 6, 7, 8, 9, 10\}$, 100 random graphs with n vertices are considered. These graphs are random in the sense that between every pair of vertices an edge is inserted with probability $p = 0.5$ [40]. The graphs are generated using the `gnp_random_graph` function from the Python library NetworkX [41]. For each n, the seeds $\{0, 1, \ldots, 99\}$ are passed into `gnp_random_graph` to generate the graphs.
- **Partition:** For every $n \in \{5, 6, 7, 8, 9, 10\}$, 100 sets of numbers $\{c_1, \ldots, c_n\}$ are considered. Each c_i is drawn uniformly at random from the interval $0 \leq c_i < 1$. The sets are generated using NumPy's `np.random.rand` function, seeded with $\{0, 1, \ldots, 99\}$ [42].

Unfortunately, due to the exponential time complexity of density-matrix-based quantum circuit simulations (cf. Section 5.7.3) and the limited time frame of this thesis, the analysis is limited to circuits with at most 10 qubits.

To measure how well the QAOA variants approximate the optimal solutions with and without noise, we will consider their *performance ratio*. Given a particular problem instance x and an algorithm A, we will define the performance ratio $R_A(x)$ as the ratio between $A(x)$, the approximate solution produced by the algorithm, and the optimal solution $OPT(x)$: $R_A(x) = A(x)/OPT(x)$. For the analysis, we will typically average the performance ratio over all instances $\{x_1, x_2, \ldots, x_{100}\}$ of a particular problem and problem size n. Also note that, since quantum measurements are random, the QAOA and its variants, are randomized algorithms. Therefore, for some problem instance x and some algorithm A, $R_A(x)$ should be averaged over multiple runs of the algorithm. How this is handled for the quantum simulations, will be discussed in Section 5.4.

Max-cut is a maximization problem whose objective, the size of the cut, is always non-negative. Therefore, for Max-cut, the performance ratio always lies in the interval $[0, 1]$ where the trivial cut $S = V, T = \varnothing$ yields the performance ratio 0 and the maximum cut yields the performance ratio 1. Partition, however, is a minimization problem in its typical formulation where the objective is to minimize the absolute difference between the sizes of the two sets. To make it easier to compare the results of Max-cut and Partition, Partition is converted into a maximization problem. The worst possible objective value for Partition is $\text{size}(S_1)+\text{size}(S_2)$, which occurs when all numbers c_i lie in one set ($S_1 = S, S_2 = \varnothing$ or vice versa). The best possible value is 0 when $\text{size}(S_1) = \text{size}(S_2)$. Therefore, the arguably most obvious conversion of Partition into a maximization problem is the linear transformation f which maps $\text{size}(S_1) + \text{size}(S_2)$ to 0 and 0 to 1:

$$f(x) = 1 - \frac{x}{\text{size}(S_1) + \text{size}(S_2)}$$

This is the approach used in this thesis. Note that this transformation is only applied after the fact to compute the performance ratios. The evaluated QAOA variants work with the original problem Hamiltonian, described in Section 3.3.

5.2. Description of the analyzed algorithms

The following algorithms are considered:

- **QAOA** as explained in Section 3.5
- **WSQAOA** with $\epsilon = 0.25$ as explained in Section 3.6.1
- **WS-Init-QAOA** with $\epsilon = 0.25$ as explained in Section 3.6.1
- **RQAOA** as explained in Section 3.6.2

For the main analysis (cf. Section 6.1), we will consider the depths $p \in \{1, 2, 3, 4\}$. For secondary analyses, we will mostly limit ourselves to $p \in \{1, 2, 3\}$ to reduce overall simulation time. If not mentioned otherwise, the RQAOA will be run 5

times on each problem instance, the other algorithms 3 times, averaging the results for each instance. This is explained further in Section 5.4. For all variants, SciPy's COBYLA optimizer with a tolerance of 1 % and 150 maximum iterations is used as the classical parameter optimizer [43]–[46]. In initial tests, using a higher tolerance and more iterations had little effect on the performance ratio for the classic QAOA. So, to speed up the simulations and generate more results within the limited time frame of a Master's thesis, these relatively low parameters are used for the performance analysis. The variants WSQAOA and WS-Init-QAOA require an approximate solution to the problem to prepare the initial state. The WSQAOA's mixer Hamiltonian also depends on this solution. For Max-cut, the approximate solution is generated by the Goemans-Williamson algorithm [23] (cf. Section 3.2). Note that this is a randomized algorithm so the initial state for WSQAOA and WS-Init-QAOA differs from execution to execution. For Partition, the list scheduling algorithm is used [27] (cf. Section 3.3). Even though its approximation ratio is much worse than the one of Goemans-Williamson rounding, list scheduling performs a lot better on problem instances where the numbers are drawn from the interval $[0, 1]$ uniformly at random than in the theoretical worst case [47]. For the analysis, the two algorithms are roughly comparable, although list scheduling has a slightly worse average performance ratio than Goemans-Williamson.

The RQAOA executes the QAOA multiple times, each time with a decreased problem size. For the analysis, we run the RQAOA to the very end, only terminating once all quadratic terms J_{ij} are zero. As the final output of the algorithm, one of the possible solutions given by the edge constraints is selected uniformly at random. This is done by performing a depth-first search traversal on the graph whose vertices are the spin variables s_i and whose edges are the constraints found by the RQAOA. For every connected component of this graph, the first variable encountered by the depth-first search is assigned $s_i = \{-1, +1\}$ uniformly at random. The values of the other variables are then determined by the constraints.

5.3. Noise model

A variety of technologies have been proposed to realize physical quantum computers, including *trapped ions*, *neutral atoms* or *superconducting qubits* [48]. It remains to be seen which of these technologies will prevail to form the basis of future quantum systems. For this thesis, we will implement a model that simulates the behavior of IBM-Q quantum computers, which use so-called superconducting transmons as their qubits. These systems are not only readily available via IBM's cloud service, making them an important tool in modern quantum computing research, IBM also provides detailed information about the noise characteristics of their systems in the form of their so-called *Fake Backends*, which are part of IBM's open source quantum library Qiskit [3], [49].

The model implemented in this thesis is based on Qiskit's own noisy simulator [37] as well as the work of [50]–[52]. The following sources of noise are considered by the model:

Gate infidelities Quantum gates can never be implemented perfectly. This manifests itself in the fact that instead of the desired unitary, a different, unknown unitary or non-unitary operator is applied instead.

Thermal relaxation As explained in Section 4.6.6, even if the qubit is not involved in any quantum gates, its state over time slowly transitions to the thermal equilibrium state $a_0|0\rangle\langle 0| + (1-a_0)|1\rangle\langle 1|$. This equilibrium depends on the temperature. For IBM-Q systems, where the temperature is close to zero Kelvin, the equilibrium state can be approximated as $|0\rangle\langle 0|$ [50].

SPAM errors Consider a qubit that is prepared in its initial state and then immediately measured, meaning the qubit was not affected by thermal relaxation or gate infidelities. Due to errors during the state preparation or the measurement, it is still possible to find the qubit in a different state than the one it was prepared in. Since in this scenario, it is often difficult to tell if state preparation or measurement was the cause of the error, both effects are typically combined under the term *SPAM errors* [53].

Consequently, the implemented model is characterized by the following parameters:

- Longitudinal relaxation time T_1 and transverse relaxation time T_2
- For each gate type (e.g. RZ or $CNOT$), the gate fidelity $\bar{F}(\tilde{U}, U)$ and the gate's duration
- $p_{0\to 1}$, the probability of measuring $|1\rangle$ after preparing $|0\rangle$, and $p_{1\to 1}$, the probability of measuring $|0\rangle$ after preparing $|1\rangle$.

To simplify the model, we will assume that each qubit experiences the same gate fidelities, thermal relaxation and SPAM errors, meaning the parameters above are the same for every qubit.

For idealized simulations, a quantum circuit is represented as the composition of unitary operators, one for each quantum gate, as explained in Chapter 2. For our noisy simulations, we will use the formalisms described in Chapter 4, representing states as density matrices and representing the circuit as a quantum channel composed of many "smaller" quantum channels. To convert an idealized circuit into its noisy equivalent, each gate unitary is replaced with the corresponding quantum channel, meaning the unitary U is replaced with $\mathcal{U}(\rho) = U\rho U^\dagger$. Next, additional (noisy) quantum channels are inserted into the circuit as follows:

Gate noise Due gate infidelities, the qubits involved in a quantum gate may end up in a slightly different state than the one resulting from applying the correct unitary. If we have no information about this altered state, we can

model this scenario by applying the correct unitary U with probability $1-p$ and replacing the state of the involved qubit or qubits with the maximally mixed state $I/2^n$ with probability p: $\tilde{\mathcal{U}}(\rho) = (1-p)U\rho U^\dagger + p\mathbb{1}/2^n$. This is equivalent to inserting a depolarizing channel after the gate: $\tilde{\mathcal{U}} = \mathcal{E}_{D,n} \circ \mathcal{U}$. To compute the average fidelity $\bar{F}(\tilde{\mathcal{U}}, U)$, note that, since $U^\dagger U = \mathbb{1}$,

$$(\mathcal{U}^\dagger \circ \tilde{\mathcal{U}})(\rho) = U^\dagger \left((1-p)U\rho U^\dagger + p\frac{\mathbb{1}}{2^n} \right) U = (1-p)\rho + p\frac{\mathbb{1}}{2^n} = \mathcal{E}_{D,n}(\rho),$$

By using (4.10) and (4.14), we obtain

$$\bar{F}(\tilde{\mathcal{U}}, U) \overset{(4.10)}{=} \bar{F}(\mathcal{U}^\dagger \circ \tilde{\mathcal{U}}) = \bar{F}(\mathcal{E}_{D,n}) \overset{(4.14)}{=} 1 - p + p/2^n.$$

We can solve this for p to find the correct depolarizing probability in order to match the desired fidelity of the gate.

The above approach assumes that the fidelity of the gate can be fully explained by errors due to the application of the gate, ignoring the effects of thermal relaxation. To account for thermal relaxation, after every gate, our model inserts a 1-qubit thermal relaxation channel for each involved qubit, followed by a depolarizing channel on the involved qubit(s). Our model works under the simplified assumption that the thermal equilibrium is $|0\rangle\langle 0|$ and therefore uses $a_0 = 1$, T_1, T_2 and the gate's duration t to parameterize the thermal relaxation channel. The depolarizing probability is computed such that the average fidelity of the composed channel $\mathcal{E}_D \circ \mathcal{E}_{TR}$ (or $\mathcal{E}_{D,2} \circ (\mathcal{E}_{TR} \otimes \mathcal{E}_{TR})$ for a 2-qubit gate) matches the desired gate fidelity. For example, in the single-qubit case, by (4.15) and (4.17), the composed channel's average fidelity is

$$\bar{F}(\mathcal{E}_D \circ \mathcal{E}_{TR}) = (1-p)\left(\frac{1}{2} + \frac{1}{6}\exp(-t/T_1) + \frac{1}{3}\exp(-t/T_2) \right) + \frac{p}{2^n},$$

which we can easily solve for p. This idea of modeling gate noise as the combination of depolarizing and thermal relaxation noise was adapted from the implementation of Qiskit's own noisy circuit simulator [37].

Idle noise It is possible for a qubit to be *idle* for some time during the circuit's execution, meaning it is not involved in any gate. This is due to dependencies between the different qubits caused by multi-qubit gates. For example, if a H gate is applied to qubit 1 and then a $CNOT$ gate is applied to qubits 1 and 2, then qubit 2 remains idle for the duration of the H gate. The thermal relaxation during this idle period is modeled by inserting a thermal relaxation channel for qubit 2 just before the $CNOT$ gate. In general, for every idle period, our model inserts a thermal relaxation channel for the idle qubit right before the following gate. We assume that all qubits are initialized and measured at the same time, meaning there are potential idle periods before the first gate of some qubit and after its last gate. Each idle noise channel is a thermal relaxation channel parameterized by $a_0 = 1$, T_1, T_2 and the idle period's duration t.

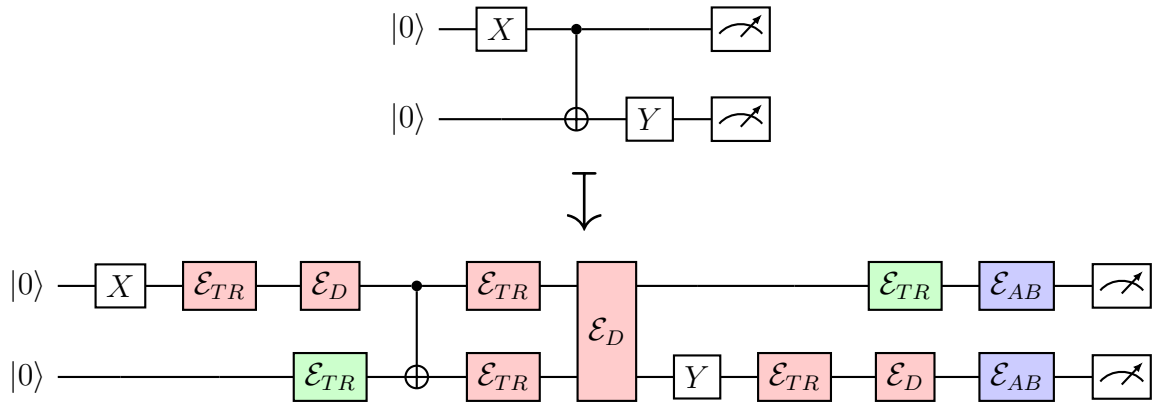


Figure 7.: Transformation of the ideal circuit into its noisy equivalent by inserting noisy quantum channels. Gate noise is shown in red, idle noise in green and readout noise in blue

SPAM errors For simplification, we will model SPAM errors solely as measurement/readout errors. These readout errors can be implemented during post-processing after the circuit simulation. However, it is convenient to include them as part of the circuit anyway. Therefore, our model inserts an asymmetric bit flip channel for each qubit at the very end of the circuit, parameterized by $p_{0\to1}$ and $p_{1\to0}$.

Figure 7 illustrates how quantum channels are inserted by the noise model using a simple example circuit.

5.3.1. Noise parameter selection

As of Qiskit version 0.23.2, the Fake Backends contain the following data points for 46 physical IBM-Q quantum computers relevant to our model [54]:

- T_1 and T_2 relaxation times for each qubit
- SPAM errors, both "prepare $|0\rangle$, measure $|1\rangle$" and "prepare $|1\rangle$, measure $|0\rangle$", per qubit.
- gate errors per gate and qubit (or pair of qubits for two-qubit gates)
- gate durations per gate and qubit(s)

We interpret the gate errors reported by the backends as gate infidelities, where the infidelity of a gate is defined as one minus its average fidelity.

The values for these parameters differ greatly between different systems and sometimes even between individual qubits. As an extreme example, the smallest reported T_1 relaxation time is about 7 µs while the largest is almost 600 µs. To find a representative, unified parameter set, we will use median values. For example, the parameter “duration of the $CNOT$ gate” is determined as follows:

	RZ / $U1$	$\sqrt{X}$ / $U2$	$CNOT$
Gate duration [ns]	0	35	400
Gate fidelity	100 %	99.97 %	99 %
T_1 relaxation time [µs]			100
T_2 relaxation time [µs]			85
Readout: $\|0\rangle \rightarrow \|1\rangle$			1.4 %
Readout: $\|1\rangle \rightarrow \|0\rangle$			3.3 %

Table 1.: Noise parameters used for the performance analysis. Only gates which are used by the QAOA circuits are displayed

For each backend supporting this gate, the median gate duration for that system is computed by taking the median across all qubits. Then, the median of these system medians is computed, which is then rounded to obtain the final parameter. The other parameter values are obtained in the same way. By first taking the median across the qubits for each system and then taking the median across the systems, we ensure that systems with more qubits are not disproportionately over-represented. T_1 and T_2 are rounded to 5 µs, error times to 5 ns. Gate infidelities are rounded to one significant figure and readout errors to two significant figures. The final parameter set is displayed in Table 1. Note that this table does not contain all gates which are supported by the Fake Backends, only the ones which are used to simulate the circuits for the QAOA variants. Also note that the RZ and the $U1$ gate as well as the $\sqrt{X}$ and $U2$ are considered to be the same gate when computing the median gate duration and gate fidelity. This and the choice of gates will be explained in Section 5.5.

5.3.2. Modeling different degrees of noise

In addition to studying the performance of the different QAOA variants under the influence of a realistic noise model, the goal of this thesis is also to analyze how the performance changes when the degree/level of the applied noise changes. What complicates this analysis is the fact that there are different sources of noise, each of which affects the state of the quantum circuit in a different way. From the noise model described above, we can identify three sources of noise: thermal relaxation noise caused by the time needed to apply the quantum gates. Additional gate noise modeled as depolarizing noise. And readout noise modeled using asymmetric bit flip channels. We will study how the performance of the QAOA variants is affected by these individual effects. To do this, we need to adjust the “strength” of the different noise sources. Therefore, three more parameters are added to the model: d_{TR} changes the level of thermal relaxation noise, d_D changes the level of depolarizing noise and d_{AB} changes the level of asymmetric bit flip channels. For each of these parameters, the value 1.0 cor-

responds to the original noise level whereas a value of 0.0 deactivates the error source completely. The three parameters affect the noise sources as follows: d_{TR} scales the time parameter t of each thermal relaxation channel $\mathcal{E}_{TR}$, effectively multiplying the duration of each gate by d_{TR}. d_D scales the depolarizing probability p of each depolarizing channel $\mathcal{E}_D$. Finally, d_{AB} scales the two SPAM error probabilities $p_{0\to1}$ and $p_{1\to1}$ for the asymmetric bit flip channels $\mathcal{E}_{AB}$.

It is reasonable to assume that the noisy circuit's fidelity correlates with the performance ratios of the QAOA variants. The authors of [55] postulate that for certain kinds of noise channels the fidelity of the quantum circuit can be approximated by $(1-p)^{\delta N}$ and the *relative performance ratio*, that is the noisy performance ratio divided by the ideal performance ratio, can be approximated by $(1-p)^{\alpha N}$. Here, N is the circuit depth, p can be thought of as the noise strength and δ and α are noise/architecture-specific constants. This suggests that the relationship between circuit fidelity and relative performance ratio can be approximated by the map $F \mapsto F^{\alpha/\delta}$. These approximations for circuit fidelity and relative performance ratio are further refined in [56]. Still, circuit fidelity and performance ratio indeed seem to be highly correlated for the standard QAOA. Therefore, circuit fidelity will also be part of the analysis.

To get an idea of how the much the different noise sources affect the performance ratios of the QAOA variants, we can consider their effect on the circuit's fidelity. Intuitively, both thermal relaxation noise and depolarizing noise cause a decrease of the fidelity which is exponential in the circuit depth. For the (single-qubit) thermal relaxation channel, (4.17) simplifies to $\bar{F}(\mathcal{E}_{TR}) \approx 1/2 + 1/2\mathrm{e}^{-t/T_1}$ for $T_1 \approx T_2$, which describes an exponential decay, in terms of the circuit depth since the execution time of the circuit is clearly proportional to its depth. Note that the average fidelity of any n-qubit channel is always at least $1/2^n$ [34]. In particular, every single-qubit channel has an average fidelity of at least $1/2$.

The factor d_{TR} linearly scales the decay constant of the thermal relaxation channel's fidelity. Repeated application of the depolarizing channel causes a similar exponential decay of the circuit's fidelity. Let $\mathcal{E}_{D,n}^k$ denote the channel that results from applying the n-qubit depolarizing channel k times, each time with a depolarizing probability of p. Here, k again can be interpreted to be proportional to the circuit depth. Then,

$$\bar{F}\left(\mathcal{E}_{D,n}^k\right) = \frac{1}{2^n} + \frac{2^n-1}{2^n}\mathrm{e}^{k\ln(1-p)}. \tag{5.1}$$

A derivation of this can be found in Section A.1 in the appendix. If the depolarizing probability p is small, we can approximate $\ln(1-p)$ as $-p$ by using the first two terms of the Taylor expansion of $\ln(x)$ about $x=1$: $\ln(x) \approx \ln(1) + \ln'(1)(x-1) = x-1$. Therefore, for small p, d_D also approximately linearly scales the decay constant of the depolarizing channel's fidelity, similar to d_{TR}. This indicates that, if the depolarizing probability is not too large, d_{TR} and d_D influence the effect of their respective channels on the circuit's fidelity in roughly the same way. However, since $\ln(1-p)$ curves downward everywhere ($\frac{\mathrm{d}^2}{\mathrm{d}p^2}\ln(1-p) = -1/(1-p)^p < 0$),

$\ln(1-p)$ is upper bounded by $-p$, which means that, for larger depolarizing probabilities, the effect of d_D increases compared to d_{TR}.

Unlike thermal relaxation and depolarizing noise, readout noise is independent of the circuit depth, making its effect less significant for deeper circuits compared to the other two noise sources. Furthermore, readout noise can be interpreted as a mostly classical effect which can be tackled using classical, statistical methods while thermal relaxation and depolarizing noise seem like more interesting noise sources to study in the context of quantum computing and the QAOA in particular. Therefore, this thesis will mainly focus on the effects of thermal relaxation and depolarizing noise. If not stated otherwise, the model for the evaluation results discussed in Chapter 6 will ignore readout noise, using the parameters $d_D = 1.0, d_{TR} = 1.0, d_{AB} = 0.0$.

5.4. Sampling noisy quantum circuits

An inherent property of quantum circuits is that, unlike classical logic circuits, their output is always governed by a probability distribution. When measuring the final n-qubit state after applying the circuit, one obtains only one of the 2^n possible results. The goal of QAOA and its variants is to adjust the parameters $\vec{\beta}, \vec{\gamma}$ trying to minimize the expected energy $\langle\psi|H_C|\psi\rangle$ of the problem Hamiltonian H_C where $|\psi\rangle = U(\vec{\beta}, \vec{\gamma})|\psi_0\rangle$ is the output of the quantum circuit. On a physical quantum computer, one needs to run the circuit multiple times with the same set of parameters $\vec{\beta}, \vec{\gamma}$ to get an accurate estimate for this expected energy. When simulating the quantum circuit on a classical computer, however, we have a complete characterization of the state $|\psi\rangle$ with all its 2^n amplitudes. This is essentially equivalent to sampling the result of the circuit an infinite number of times. So, we can calculate the expected energy exactly, either by explicitly computing the matrix product $\langle\psi|H_C|\psi\rangle$ or by computing the sum $\sum_i C(s^{(i)})p_i$ where $s^{(i)}$ is the i-th computational basis state and $p_i = |\psi_i|^2$ is the probability of measuring this state. When simulating the noisy circuit $\mathcal{U}(\vec{\beta}, \vec{\gamma})$, the expected energy can be computed as $\mathrm{Tr}\left(H_C\,\mathcal{U}(|\psi_0\rangle\langle\psi_0|)\right)$ or by using the measurement probabilities given by $U(|\psi_0\rangle\langle\psi_0|)$'s diagonal elements.

For the performance analysis, we use the exact values for $\langle\psi|H_C|\psi\rangle$ obtained from the state vectors or density matrices to find the optimal parameters $\vec{\beta}$ and $\vec{\gamma}$. For the final set of parameters, we then use the measurement probabilities for each computational basis state to compute the average performance ratio. This approach allows us to obtain a reasonable estimate of the expected performance ratio while only having to run the algorithm once. It is also easier to implement when using the Qaptiva 800's noisy simulator (cf. Section 5.7.2). However, using the exact value for the expected energy has the downside of using unrealistically accurate values of $\langle\psi|H_C|\psi\rangle$ for the classical optimizer. Therefore, we will also consider the case where only a limited number of samples is used during the

QAOA optimization phase (cf. Section 6.7), although this will not be the main focus of the evaluation.

The RQAOA requires special consideration. Here, the QAOA is run multiple times and after each time the most correlated edge in the graph is contracted. Finding the single edge which maximizes $|\langle\psi|Z^{(i)}Z^{(j)}|\psi\rangle|$ using only a few, say polynomially many, samples is unrealistic. The best one can hope for is finding an edge which is among the most highly correlated. Therefore, for this special scenario of finding the most correlated edge, a limited, finite number of samples is used for the performance analysis. The number of samples will be set to 10. As we will see in Chapter 6, this small number of samples is enough to achieve very good results with the RQAOA. In addition, due to its hybrid approach, we only get a single solution from one execution of the RQAOA while for the other variants, the final state gives us a probability distribution. Consequently, the results for the RQAOA will be less accurate than for the other variants.

5.5. Gate sets and transpilation

While infinitely many conceivable quantum gates exist, in practice most quantum algorithms are expressed using only a few common gates (cf. Section 2.4). However, physical quantum computers typically support even fewer gates natively. These so-called *basis gates* must be combined to simulate all other gates, similar to how every classical logic gate can be implemented using only NAND gates.

Almost all Qiskit Fake Backends support the $CNOT$ gate as well as one of the following two sets of single-qubit gates [54]:

$$G_1 = \{RZ(\lambda), X, \sqrt{X}\},\ G_2 = \{U1(\lambda), U2(\phi,\lambda), U3(\theta,\phi,\lambda)\}$$

Section 2.4 already covered X and RZ. The remaining gates are defined as follows [57], [58]:

$$\sqrt{X} \propto RX\left(\frac{\pi}{2}\right),\ U1(\lambda) = U3(0,0,\lambda) \propto RZ(\lambda),$$
$$U2(\phi,\lambda) = U3\left(\pi/2,\phi,\lambda\right),\ U3(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\mathrm{e}^{\mathrm{i}\lambda}\sin\left(\frac{\theta}{2}\right) \\ \mathrm{e}^{\mathrm{i}\phi}\sin\left(\frac{\theta}{2}\right) & \mathrm{e}^{\mathrm{i}(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

Here, $\propto$ denotes equality up to a global phase. A set of gates is called *universal* if every quantum gate can be implemented using gates from the set. As mentioned in Section 2.4, the set of all possible single-qubit gates plus the $CNOT$ gate form a universal set of gates. All possible single-qubit gates can be implemented using gates from G_1 or by using gates from G_2, making both G_1 and G_2 universal when combined with the $CNOT$ gate.

The $U3$ gate implements a rotation of the Bloch sphere around three Euler angles. Therefore, any 2×2 unitary matrix can be represented, up to a global

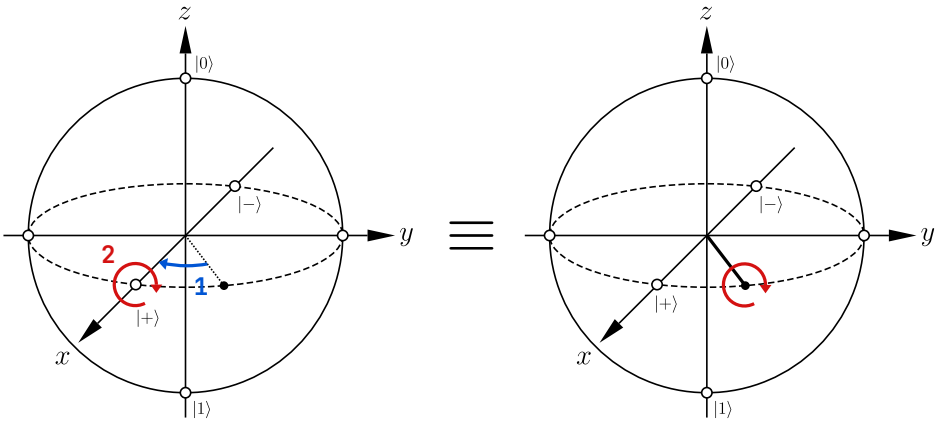


Figure 8.: The virtual RZ gate: The sequence of gates $RX(\phi)RZ(\theta)$ is effectively equivalent to a single rotation of ϕ around the axis $RZ(-\theta)|+\rangle$.

phase, using a single $U3(\theta, \phi, \lambda)$ matrix. Thus, the $U3$ gate alone is enough to implement all single-qubit gates. In particular, for the QAOA variants studied in this work,

$$H = U3(\pi/2, 0, \pi) = U2(0, \pi), \quad RX(\theta) = U3(\theta, -\pi/2, \pi/2), \quad RY(\theta) = U3(\theta, 0, 0).$$

Since

$$U2(\psi, \lambda) \propto RZ(\psi + \pi/2) \cdot \sqrt{X} \cdot RZ(\lambda - \pi/2) \text{ and} \tag{5.2}$$

$$U3(\theta, \phi, \lambda) \propto RZ(\phi + \pi) \cdot \sqrt{X} \cdot RZ(\theta + \pi) \cdot \sqrt{X} \cdot RZ(\lambda), \tag{5.3}$$

the gates from G_1 can also implement all possible single-qubit gates [59], [60].

It turns out that, for the purposes of this thesis, both gate sets are essentially identical in terms of their noise characteristics. To see why, one needs to consider the physical implementation of IBM-Q quantum gates. In simplified terms, for superconducting transmon qubits, as they are used in IBM-Q systems, single-qubit quantum gates are implemented by applying a microwave pulse at a frequency corresponding to the energy difference between the qubit's ground state ($|0\rangle$) and its first excited state ($|1\rangle$). This corresponds to a rotation around the Bloch sphere where the axis of rotation is determined by the phase shift of the pulse and the angle of rotation is determined by the pulse's amplitude. The possible rotations implemented in this way are of the form

$$\exp(-\mathrm{i}\theta/2A) \text{ with } A = \cos(\gamma)X + \sin(\gamma)Y,$$

where the rotation angle θ depends on the pulse's amplitude and γ depends on the pulse's phase. This means, only rotations around an axis in the xy-plane are physically possible. To implement a qubit phase shift, that is a $RZ(\theta)$ rotation, the idea is to simply add an offset of $-\theta$ to all following values for γ, as shown in Figure 8. Intuitively, instead of rotating the Bloch sphere by θ, we instead rotate the reference frame by $-\theta$ and look at the sphere from this angle for all following rotations. If the very last gate for some qubit is an RZ rotation, it can simply

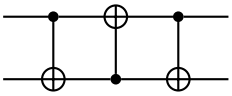


Figure 9.: A *swap* gate built from three $CNOT$ gates

be removed, since it does not affect the final measurement in the computational basis. By keeping track of all the accumulated offsets over the course of the quantum algorithm, for both single-qubit and $CNOT$ gates, this idea allows RZ gates to be implemented virtually, making them essentially error-free since any possible error is already accounted for by the noisy execution of some other gate. This is reflected in Table 1 which assumes a fidelity of 1 and a duration of 0 for $RZ(\theta)$ / $U1(\theta)$ gates [39], [59].

According to the Qiskit documentation, the $U2$ gate is physically implemented as a single $\sqrt{X}$ pulse ($90°$) along with the virtual RZ gate as described in (5.2). The $U3$ gate uses two $\sqrt{X}$ pulses as detailed in (5.3) [60]. In other words, for studying the behavior of noisy circuits, it suffices to consider the $\sqrt{X}$ gate as the only single-qubit noisy gate. Therefore, for the performance analysis, circuits will be transpiled into gate set G_1 (cf. Section 5.7.2).

The fact that only one single-qubit gate and one two-qubit gate are physically implemented on IBM-Q hardware reveals that the noisy quantum channels added by the noise model described in Section 5.3 are extremely similar across all QAOA variants. While the RQAOA uses standard QAOA circuits, the WSQAOA uses a different mixer Hamiltonian, but this mixer Hamiltonian has the same noise behavior since it is also physically implemented using two $\sqrt{X}$ pulses. The only difference in terms of noise characteristics between the QAOA variants is due to the very first gate for each qubit. The standard QAOA and the RQAOA use a Hadamard gate, which is implemented using a single $\sqrt{X}$ pulse, whereas the RY gate used by the WSQAOA and WS-Init-QAOA requires two pulses.

5.6. Circuit connectivity

Throughout the analysis, we assume that $CNOT$ gates can be applied to any pair of qubits. In superconducting quantum computers, however, due to hardware limitations, only certain pairs of qubits actually have a physical connection and can be entangled with $CNOT$ gates. These connections can be described by the coupling graph of the quantum computer. The *connectivity density* describes how strongly the coupling graph of a given quantum system is connected. It is defined as the number of edges of the coupling graph divided by $n(n-1)/2$, the number of edges of the complete coupling graph with the same number of qubits (n). The 46 considered Qiskit Fake Backends have a median connectivity density of about 0.133.

To overcome the problem of non-adjacent qubits, additional gates need to be inserted into the circuit to move qubits around. To swap two qubits, we can apply three $CNOT$ gates as shown in Figure 9, which has the following effect:

$$|a, b\rangle \rightarrow |a, a \oplus b\rangle \rightarrow |\underbrace{a \oplus a}_{=0} \oplus b, a \oplus b\rangle \rightarrow |b, a \oplus \underbrace{b \oplus b}_{=0}\rangle = |b, a\rangle$$

However, $CNOT$ gates typically induce the most noise (cf. Table 1), so adding many swap gates is undesirable. To overcome this problem, one needs to find a circuit which is equivalent to the original circuit, meaning the input and output qubits can be permuted, and which uses the minimum number of $CNOT$ gates. This so-called qubit routing problem is known to be NP-hard. The decision problem "Does there exist a qubit permutation such that all $CNOT$ gates of the circuit respect the coupling graph?" is already NP-complete [61]. Solving the qubit routing problem heuristically is an active topic of research [62]–[64].

Considering qubit connectivity as well as different strategies to handle missing qubit connections increases the complexity of analyzing the performance of the QAOA variants substantially. Therefore, the analysis performed in this thesis assumes a complete coupling graph, ignoring the problem of qubit routing completely. However, higher noise levels than those of the baseline noise model are considered, providing at least some indication of how the variants perform when additional $CNOT$ gates are inserted. In addition, there is some evidence suggesting that above a certain connectivity threshold, the number of necessary swap gates using common qubit routing heuristics decreases dramatically [65], making them less of a problem than one might think. Still, it should be noted that additional swap gates are certainly one of the largest noise factors not captured by the analysis.

5.7. Implementation on the Qaptiva 800 platform

The simulations are performed on a *Qaptiva 800*, formerly known as *Atos Quantum Learning Machine*, using its proprietary QLM library. This library already implements much of the core functionality needed for the simulations, including a density matrix-based noisy simulator. It still comes with some limitations which need to be considered. The following subsections cover the implementation of the simulation pipeline using the QLM library.

5.7.1. Implementation of the hardware model

Through its `NoisyQProc` virtual quantum processing unit (QPU), the QLM library supports the simulation of noisy quantum circuits using density matrices and quantum channels. The noise model described in Section 5.3 is implemented using QLM's `DefaultHardwareModel` class. This class allows the user to specify

gate durations for each gate type and each qubit as well as a list of time-parameterized quantum channels. These channels will then be inserted with the correct time parametrization during qubit idle periods between quantum gates. Implementations for the amplitude damping channel and the phase damping channel are also provided by the QLM library, making it easy to implement the model's idle noise.

The `DefaultHardwareModel` class also allows the user to specify a gate noise channel, parameterized by gate type, qubit and gate parameters like rotation angle. For each quantum gate, the corresponding channel is inserted immediately after the gate. Unlike with idle noise, where multiple channels can be specified, gate noise has to be constructed programmatically as one channel in its complete Kraus representation. This is done by composing the correct amplitude damping, phase damping and depolarizing channels. This naive composition according to (4.5) and (4.6) can result in channels with unnecessarily many Kraus operators, which turns out to have a large impact on the running time of the simulations. In particular, single-qubit gate noise is represented using $(2 \cdot 2) \cdot 4 = 16$ Kraus operators and two-qubit gate noise is represented using $(2 \cdot 2)^2 \cdot 16 = 256$ Kraus operators, even though every single-qubit channel requires at most 4 operators and every two-qubit channel requires at most 16 operators [18]. Therefore, QLM's feature of converting between different quantum channel representations is used to convert from Kraus operator representation to Choi matrix representation [66] and back to Kraus operator representation. The second conversion ensures that the number of Kraus operators is not more than 4 or 16 respectively.

According to our model, the depolarizing probability is set depending on the thermal relaxation channel's average fidelity such that the total fidelity of the combined channel matches the desired gate fidelity. To do this, the function `get_average_process_fidelity` provided by the QLM library is used to compute the fidelity of $\mathcal{E}_{TR}$, or $\mathcal{E}_{TR} \otimes \mathcal{E}_{TR}$ in case of the $CNOT$ gate. Then the depolarizing probability is computed by solving (4.15) for p.

Readout noise is also implemented using the gate noise feature of `DefaultHardwareModel`: For each qubit, a special dummy gate is inserted at the end of the circuit. The gate noise for this dummy gate is configured to always use the asymmetric bit flip channel for the readout error.

5.7.2. Implementation of the QAOA variants

The QLM library already contains much of the functionality needed for simulating QAOA circuits. It can generate Ising formulations of many common NP-hard optimization problems, including Max-cut and Partition. It can also generate (standard) QAOA circuits with depth p for a given Ising Hamiltonian. To generate the QAOA circuit, the $\exp(-\mathrm{i}\gamma Z^{(i)} Z^{(j)})$ gates of the time-evolved problem

Hamiltonian are rearranged using a "greedy coloring heuristic" to reduce circuit depth. Exactly which heuristic is used is not documented. However, it seems likely that it is a technique similar to [67], where the edges of the graph of the non-zero quadratic terms J_{ij} are colored so that no two adjacent edges have the same color. The gates for edges of the same color can be applied in parallel to reduce circuit depth.

Parameterized circuits are also supported: Each circuit can have multiple variables, which are identified by strings. Circuit variables or math expression containing these variables can be used as angles for rotational gates such as RX, RY and RZ. Using the `ScipyMinimizePlugin`, which is a wrapper around the SciPy function `scipy.minimize`, the QLM library can find the optimal set of circuit parameters $\vec{\theta}$ to minimize $\langle\psi_0| U\left(\vec{\theta}\right)^\dagger H_C\, U\left(\vec{\theta}\right)|\psi_0\rangle$ for some problem Hamiltonian H_C. This involves executing the circuit multiple times, each time tuning the variables $\vec{\theta}$. By default, when used together with the `NoisyQProc` virtual QPU, the density matrix representation is used, which means the expected energy of the Hamiltonian is computed with theoretically infinitely many samples during parameter optimization. `NoisyQProc`, however, also supports a *stochastic* mode, which uses the standard state vector representation along with probabilistic sampling to simulate noisy circuits. The number of samples used can be adjusted. As a secondary analysis, we will consider different sample sizes for estimating the expected energy (cf. Section 6.7). For this analysis, the stochastic mode will be used. For the other analyses, we will exclusively use the deterministic mode since it turns out to be much faster than stochastic mode for our simulations. Unfortunately, the `ScipyMinimizePlugin` does not have a documented way of adjusting the number of samples for density-matrix-based simulations.

To implement the WS-Init-QAOA, the circuits produced by QLM's QAOA circuit generator are then modified in code by replacing the H by $RY(\theta)$ gates to prepare the equal superposition. Additionally, for WSQAOA, the $RX(\beta)$ implementing the time evolution of the mixer Hamiltonian are replaced with instances of $RY(-\theta)RZ(-2\beta)RY(\theta)$. The RQAOA is also implemented manually. This involves computing the M_{ij} from the final QAOA parameters, contracting the most correlated edge to generate the new, reduced Ising Hamiltonian, keeping track of which qubit corresponds to which spin variable when the number of qubits decreases during the execution of the algorithm, and reconstructing the final solution using depth-first search.

Circuits are transpiled to conform to the gate set $\left\{RZ(\theta), \sqrt{X}, CNOT\right\}$. The QLM library includes a tool to compile quantum circuits for a given gate set. However, as of library version 1.7, this tool produces unpredictable and often sub-optimal results when applied to the QAOA circuits. Therefore, a custom circuit transpilation implementation is used instead. It involves expressing the H and $RX(\theta)$ gates using $RZ(\theta)$ and $\sqrt{X}$ gates as explained in Section 5.5. For the WSQAOA time-evolved mixer Hamiltonian, one could convert each RY gate individually, ending up with a total of four $\sqrt{X}$ gates. However, to reduce noise, we will instead express it using only two $\sqrt{X}$ gates. In a one-time procedure,

QLM's circuit transpiler is used to express the circuit $RY(-\theta)RZ(-2\beta)RY(\theta)$ in the form $RZ(\theta_3)\sqrt{X}RZ(\theta_2)\sqrt{X}RZ(\theta_1)$. The math expressions for θ_1, θ_2 and θ_3 are then serialized as human-readable strings in prefix notation using library functionality. These serialized expressions are generated once and stored on disk. When transpiling a WSQAOA circuit, these strings are loaded. The correct variable name is inserted by simple string substitution and the math expression is deserialized and inserted as angle parameters into the respective circuit gates. This way, circuit parameters $\vec{\beta}$ are not changed by the transpilation, meaning we can continue to use QLM's `ScipyMinimizePlugin` as before.

5.7.3. Implementation of the simulation pipeline

Quantum simulations are highly computationally intensive. This is especially true for noisy simulations involving density matrices: While an n-qubit pure state can be represented with a 2^n-dimensional state vector, representing an n-qubit mixed state requires a $2^n \times 2^n$ density matrix. The performed performance analysis requires simulating thousands of quantum circuits. Consequently, to limit the time required to perform these simulations, the 192 physical CPU cores of the Qaptiva 800 platform should be utilized as much as possible.

A simulation pipeline built on top the QLM library is implemented. Its main goal is to perform rapid simulations for multiple parameter sets while utilizing the Qaptiva hardware as much as possible. A simulation run performs a QAOA variant on 100 problem instances, as described in Section 5.1. Each simulation run is parameterized by:

- Problem type (Max-cut, Partition)
- Problem size (number of qubits)
- Algorithm type (QAOA, WSQAOA, WS-Init-QAOA, RQAOA)
- Circuit depth p (number of QAOA layers)
- Simulation type (ideal, noisy)
- Noise level parameters d_D, d_{TR}, d_{AB} (only supported for noisy simulations)
- Number of tries per problem instance: The performance ratios are averaged to increase the accuracy of the result.
- Performance indicator: This is usually the performance ratio. However, for noisy simulations, the simulation pipeline can also measure the fidelity of the circuit's output state for the final values of $\vec{\beta}$ and $\vec{\gamma}$, compared with the output of the same, ideal circuit (cf. Section 6.3).
- Number of samples for measuring $\langle\psi|H_C|\psi\rangle$
- Number of samples for measuring $\langle\psi|Z^{(i)}Z^{(j)}|\psi\rangle$ (only RQAOA)

These parameters are specified in a JSON format when starting a simulation run. For one parameter set, simulation time may vary across different problem instances since, for some instances, the optimizer may require fewer iterations than the upper limit of 150. To ensure that all CPU cores are utilized as much as possible, the pipeline supports launching multiple simulation runs. Simulations for the next run can already start without needing to wait for all simulations of the previous run to finish. The parameter sets for multiple runs can be specified in the JSON by using an array instead of a single value for some parameter. The simulation pipeline takes the Cartesian product of the possible combinations to generate all parameter sets. Take the following, simplified, example:

```
{
    "algorithm": ["QAOA", "WSQAOA"],
    "qpu": [
        {"type": "ideal"},
        {"type": "noisy", "d_D": [1.0, 2.0]}
    ]
}
```

This would get expanded to the following parameter sets:

```
{ "algorithm": "QAOA", { "type": "ideal" }}
{ "algorithm": "QAOA", { "type": "noisy", "d_D": 1.0 }}
{ "algorithm": "QAOA", { "type": "noisy", "d_D": 2.0 }}
{ "algorithm": "WSQAOA", { "type": "ideal" }}
{ "algorithm": "WSQAOA", { "type": "noisy", "d_D": 1.0 }}
{ "algorithm": "WSQAOA", { "type": "noisy", "d_D": 2.0 }}
```

The QLM library supports two modes for performing simulations. In *local mode*, simulation scripts are executed directly on the Qaptiva 800, which typically requires access to the platform via SSH. In *server mode*, QLM jobs are created on the user's machine and sent to the Qaptiva 800, which functions as a server. The server mode is well integrated into the QLM library, so only little changes in the user's source code are required to change from local mode to server mode. One notable difference between the two modes is how jobs which require more resources than are currently available are handled. A *resource manager* process is permanently running on the Qaptiva 800 to ensure the fair distribution of resources across multiple users. In local mode, if not enough resources are available for the job according to resource manager, the execution of the job is denied and an exception is thrown. In server mode, jobs are inserted into a work queue. Jobs in this queue are executed as soon as enough resources are available. Since the execution of a job on the user side is asynchronous in server mode, the user's code simply waits for the job to complete.

The asynchronous execution of jobs, which allows the parallel execution of jobs in a single-threaded Python script, and the fact that jobs automatically wait until enough resources are available should make server mode the obvious choice to execute a large batch of simulation. In practice, though, server mode comes with various limitations. First, it requires serialization and deserialization of jobs and their results, since both must be sent over a TCP connection, even if the user's script is running locally on the Qaptiva 800. Additionally, jobs and job results are written to disk in this mode to ensure they can still be accessed in case the connection to the server terminates. Since job dispatching is synchronous, this large additional I/O overhead can cause dispatching a job in server mode to take longer than the entire execution of the same job in local mode, so there is nothing to be gained by being able to execute jobs asynchronously. A possible way to circumvent this problem is QLM's feature to dispatch multiple jobs in a single batch. However, there does not seem to be a way to combine this batching feature with the `ScipyMinimizePlugin` required to simulate QAOA circuits, making this solution impractical.

The alternative to server mode is to use a multi-threaded script in local mode which runs multiple simulations in parallel. When a resource exception occurs, the affected thread waits a random amount of time and then tries again. This approach comes with its own problems: Key parts of QLM library only work correctly in a single-threaded environment. This can be solved by using Python's `multiprocessing.Pool`, which spawns multiple processes instead of threads to execute the submitted tasks. Because the simulation pipeline can perform multiple runs with different numbers of qubits in a single pass, it is difficult to predict the optimal number of threads to maximize throughput and minimize the number of resource exceptions. In addition, the QLM resource manager seems to be overly pessimistic. Scenarios where most processes retry their current job many times without doing any work are hard to avoid.

Therefore, for the performance analysis, the resource manager is turned off entirely via an environment variable and tasks are executed by a `multiprocessing.Pool` with 192 processes, the same as the number of physical CPU cores. Each task corresponds to an execution unit which cannot be split up further. In the context of the performance analysis, a task involves the complete execution of an algorithm (QAOA, WSQAOA, WS-Init, RQAOA) including multiple, sequential simulations of the same circuit (or multiple circuits for the RQAOA).

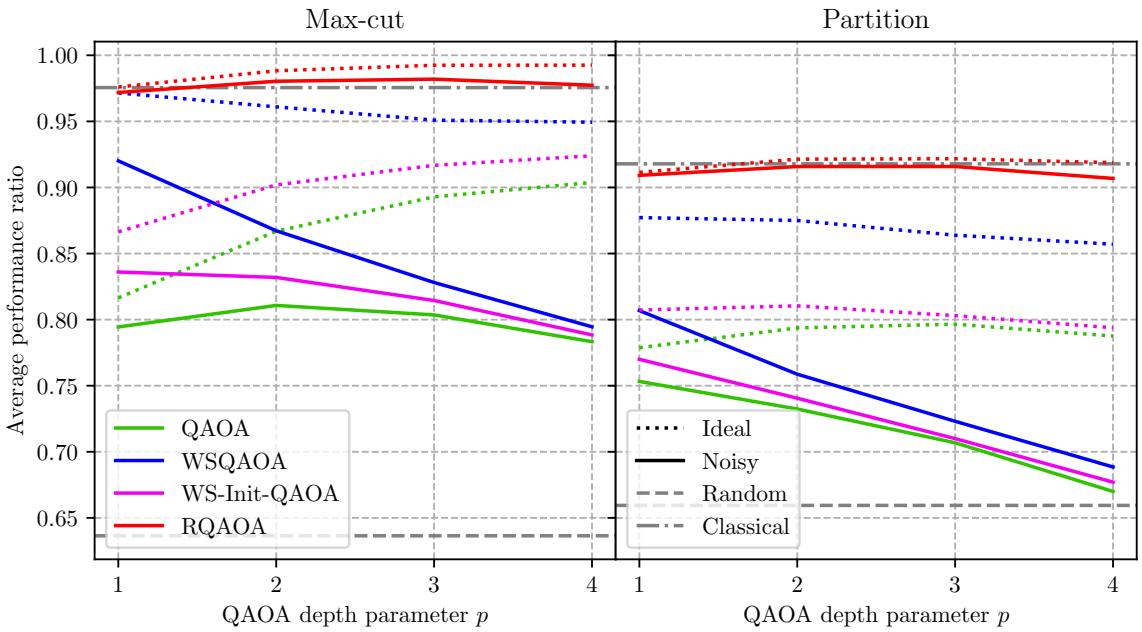
The performance ratios are written into text files. Each line of these files stores the results of a run as a JSON document containing the parameters for that run and the performance ratio/fidelity for each problem instance. This allows the results to be easily filtered and visualized later. When performing multiple runs, once the results for a run are collected, they are immediately written to disk. This way, if the simulation pipeline script needs to be restarted for any reason, it can automatically skip over already completed runs.

6. Evaluation of the results for the noisy performance analysis

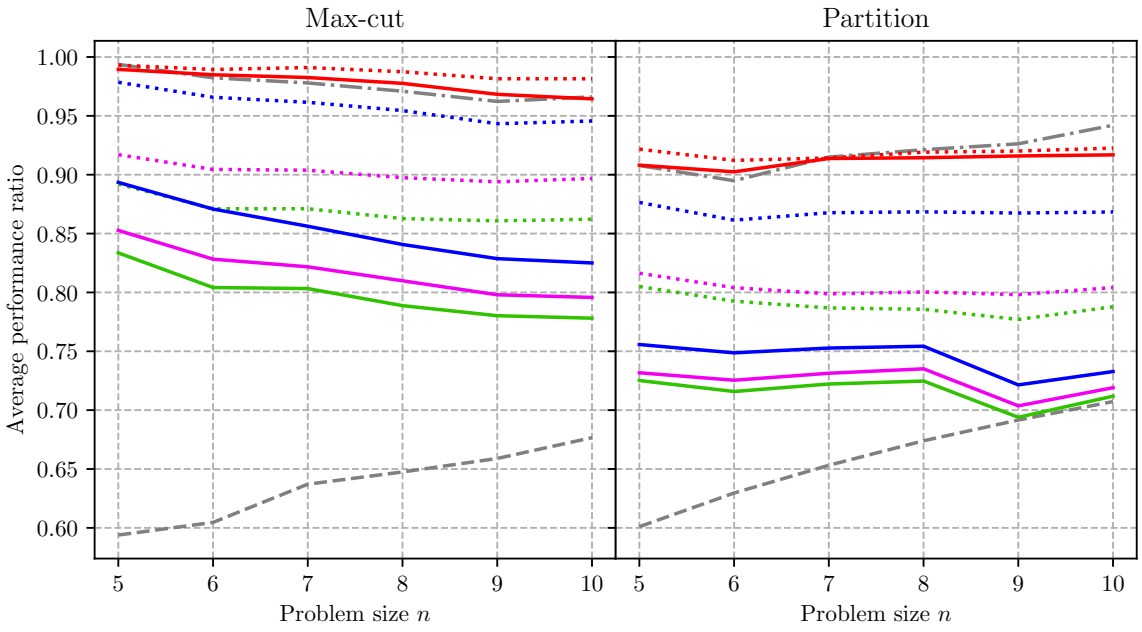
This chapter describes the key findings of the noise performance analysis. Section 6.1 details the results of the main analysis which compares the performance ratios of the QAOA variants when run on ideal and noisy circuits. In Section 6.2, the effects of different noise levels will be analyzed. Section 6.3 examines the effect of noise on circuit fidelity and Section 6.4 evaluates the effect of readout errors on the algorithms' performance. Section 6.5 talks about how the parameter optimization phase of the algorithms is affected by noise, while Section 6.6 compares the effect of single- and two-qubit gate noise. Section 6.7 analyzes the effect of different sample sizes on the performance and Section 6.8 looks at how well the RQAOA can handle higher levels of noise. Finally, Section 6.9 examines the relationship between circuit depth and performance ratio and Section 6.10 attempts to answer the question whether there is an inherent difference in the effects of thermal relaxation noise and gate infidelities on QAOA performance.

6.1. Comparison of the ideal and noisy results

The main result of the performance analysis is summarized in Figure 10a, which compares the average performance ratios of the four QAOA variants with $p \in \{1, 2, 3, 4\}$ layers for the 600 Max-cut instances and 600 Partition instances described in Section 5.1 with problem size $n \in \{5, 6, 7, 8, 9, 10\}$. The results are separated by problem, algorithm and number of layers, and are averaged over the problem sizes (number of qubits). Figure 10b visualizes the same data, but separated by problem size and averaged over the number of layers. The data used for Figure 10, separated by both number of qubits and number of layers, can be found in Section A.2 in the appendix. As justified in Section 5.3.2, the noise model only considers thermal relaxation and depolarizing noise, ignoring SPAM noise ($d_D = 1, d_{TR} = 1, d_{AB} = 0$). The effect of readout errors will be covered in Section 6.4. For orientation, the chart also shows the expected performance ratios when selecting the solution uniformly at random as well as the average performance ratio of the classical approximation algorithm which is also used to initialize the WSQAOA and the WS-Init-QAOA (Goemans-Williamson rounding for Max-cut and list scheduling for Partition).



(a) Separated by number of QAOA layers p



(b) Separated by problem size n

Figure 10.: Average performance ratio across all considered problem instances ($n \in \{5, 6, 7, 8, 9, 10\}$) for the four QAOA variants ($p \in \{1, 2, 3, 4\}$): comparison between the ideal and the noisy circuit model

The two charts reveal a clear order of the algorithms' performance ratios which is the same for both problems and both circuit models (ideal and noisy). The RQAOA performs the best and the standard QAOA the worst, with the WSQAOA and the WS-Init-QAOA being ranked second and third respectively. Although it is not surprising that the algorithms perform worse on noisy circuits than on their ideal counterparts, the difference varies greatly between variants. While noise has a pronounced effect on the performance of the QAOA, WSQAOA and WS-Init-QAOA, the noisy RQAOA seems to perform only slightly worse than the ideal version, even slightly outperforming the Goemans-Williamson algorithm for Max-cut. The RQAOA by design, after each QAOA run, considers only the parity of each pair of spin variables that are part of the Ising Hamiltonian, and selects the pair for which the result is the most conclusive. It seems reasonable to assume that the more conclusive pairs also tend to have a relatively high correlation when the algorithm is executed on a noisy circuit, making the RQAOA much more resistant to noise than the other variants. Considerably less impressive than the performance of the noisy RQAOA is the fact that the noisy QAOA, WSQAOA and WS-Init-QAOA virtually do not benefit from adding more QAOA layers. Only for the standard QAOA and Max-cut, the added computational power of a second layer outweighs the added noise caused by the deeper circuits. It should be noted, however, that for the Partition problem, even in the ideal case, the variants struggle to benefit from adding more layers, showing the limitations of the classical optimizer used. The influence of the classical optimizer will be discussed in Section 6.5.

The noisy RQAOA performs very well, even with only a single layer. While adding more layers does not affect the results much, a slight improvement can still be observed when adding a second and third layer. We can verify this observation by performing a Wilcoxon signed-rank test [68], using the performance ratios of the individual problem instances as data points. While the observation is indeed statistically significant for Max-cut, with the p-value being 10^{-22} for layer 2 and 10^{-11} for layer 3, the situation is not quite clear for Partition where the p-values are 0.039 and 0.051 respectively.

The noise resistance of the RQAOA is further demonstrated in Figure 11, which visualizes the average *relative performance ratio*, separated by problem, problem size, algorithm and number of QAOA layers. For every problem instance, we obtain the relative performance ratio by dividing the noisy performance ratio by the ideal performance ratio. A larger relative performance ratio indicates a larger resistance to noise. Even though the WSQAOA is the second best noisy algorithm overall, it is actually the one most negatively affected by noise while the WS-Init-QAOA is generally only slightly less resistant to noise than the standard QAOA. The bad relative performance of the WSQAOA, compared to the WS-Init-QAOA, might have to do with its comparatively large "absolute" performance ratio. Also, as explained in Section 3.6.1, the WSQAOA has the ability to reconstruct the solution of the initial state. WSQAOA might outperform the standard WS-Init-QAOA for the investigated problem instances mainly to due to this ability, which could be strongly impaired by noise.

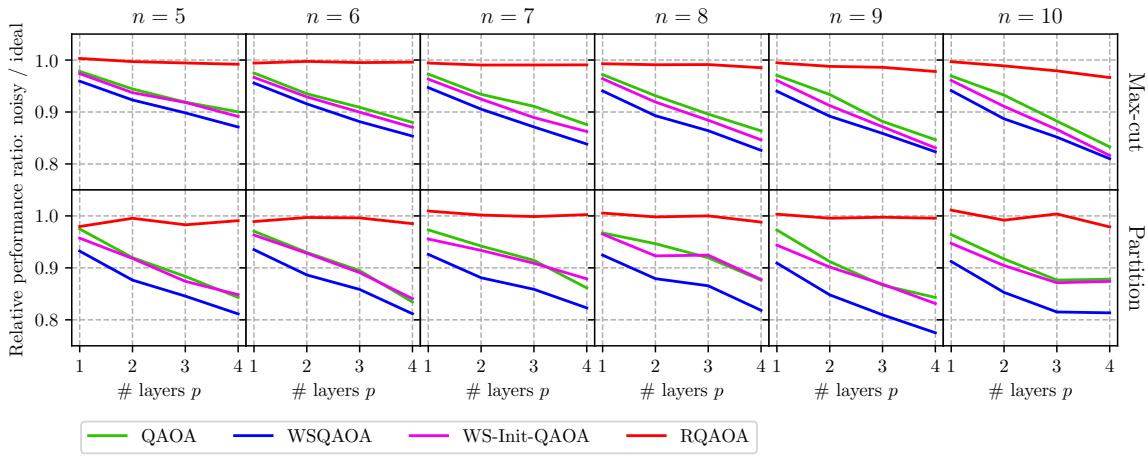


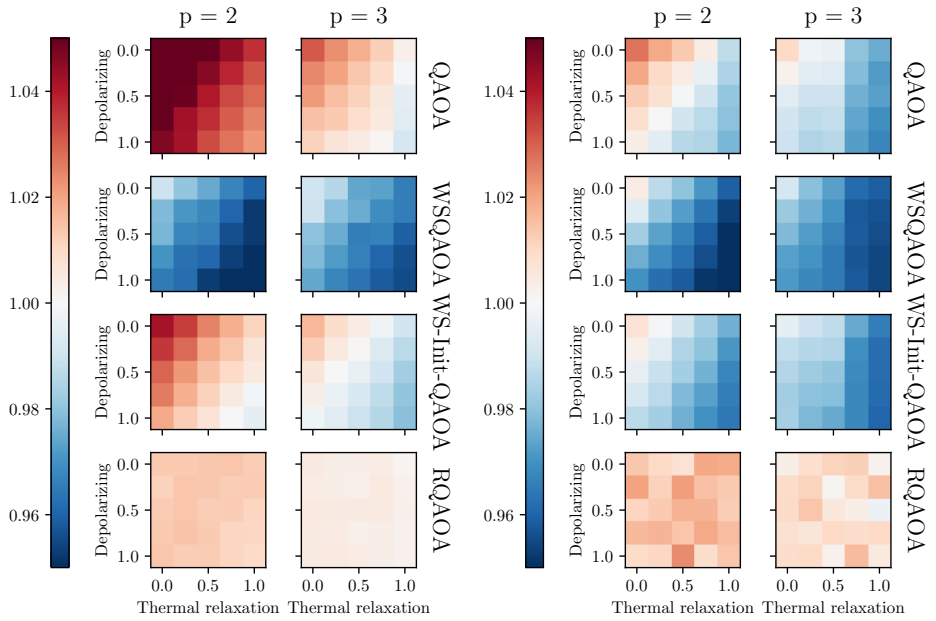
Figure 11.: Comparison of the relative performance ratios (noisy divided by ideal) separated by problem, problem size, algorithm, and number of QAOA layers

Figure 10b and Figure 11 indicate that both the performance ratio of the noisy variants and the relative performance ratio are generally negatively impacted by the problem size. This makes sense since for a larger problem size, the circuits also tend to grow deeper as more $CNOT$ gates are required. Still, the number of QAOA layers seem to have a much larger effect than the problem size. This can be interpreted as a generally good sign when trying to apply the QAOA and its variants to larger problems.

In Figure 10b, the results for the Partition problem with $n \in \{8, 9, 10\}$ and noisy QAOA, WSQAOA and WS-Init-QAOA are particularly noteworthy. Here, a large decrease in performance ratio between $n = 8$ to $n = 9$ can be observed whereas the results for 10 qubits are actually slightly better than the ones for 9 qubits. The dip at $n = 9$ is most likely caused by a large jump in circuit depth whose reason will be explained in Section 6.3. The increase in performance for $n = 10$ is probably due to the fact that the results are nearly completely random at this point and that choosing a solution uniformly at random results in a slightly higher performance ratio for $n = 10$ than for $n = 9$.

6.2. The effect of different noise levels

Since the noise resilience for most of the analyzed variants can be seen as rather disappointing, a question which arises naturally is “How much noise can the algorithms handle?”. More concretely, we will consider the question at which level of noise adding more QAOA layers does not further improve the average performance ratio of the algorithms. The heatmaps depicted in Figure 12 visualize the advantage of 2-layer and 3-layer circuits, compared to circuits with one fewer layer, for different levels of noise ($d_D, d_{TR} \in \{0, 0.25, 0.5, 0.75, 1\}$). The top-left cell in each heatmap is the ideal circuit. The x- and y-axis add more



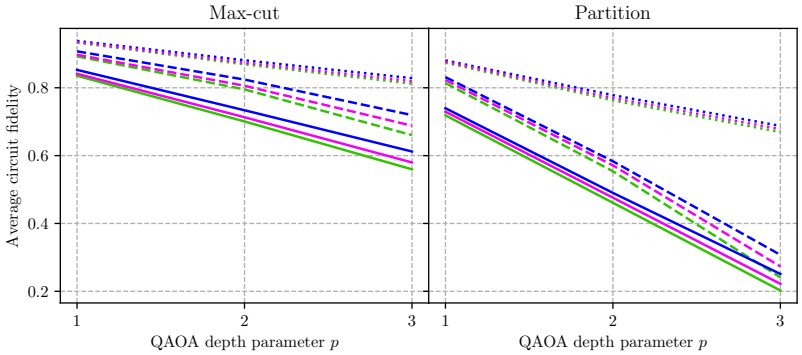
(a) Max-cut **(b)** Partition

Figure 12.: Performance ratio for p divided by performance ratio for $p-1$ with $p \in \{2, 3\}$ for different noise levels: $d_D, d_{TR} \in \{0, 0.25, 0.5, 0.75, 1\}$

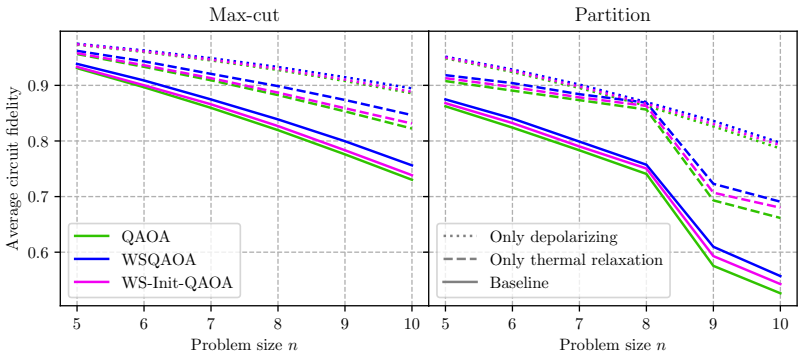
thermal relaxation and depolarizing noise with the bottom-right cell showing the baseline noise model results. A red cell indicates that the algorithm benefits from the added layer whereas a blue cell indicates that the average results are worse. The "raw" performance ratios, separated by problem size and number of layers, are shown in Section A.3 in the appendix for selected values of d_D and d_{TR}. Despite what the baseline noise model might suggest, the standard QAOA and the WS-Init-QAOA show some noise resistance for lower levels of noise when applied to Max-cut. For moderate noise levels, the standard QAOA still benefits from adding a third layer, as does the WS-Init-QAOA when adding a second. For Partition, the results are generally worse, as was already indicated by Figure 11. A likely reason for this is the fact that the QAOA circuits for the considered Partition instances are generally deeper than those for Max-cut. This is because the problem Hamiltonian for Partition contains a quadratic term for each pair of numbers whereas the problem Hamiltonian for Max-cut only contains quadratic terms for roughly half of the pairs since between every pair of vertices an edge is drawn with probability 0.5. Therefore, on average, Partition circuits contain about twice as many $CNOT$ gates as Max-cut circuits.

6.3. The effect of noise on circuit fidelity

The top-right and bottom-left corners of the heatmaps in Figure 12 indicate that for the analyzed model, the effect of thermal relaxation noise seems to be



(a) Separated by number of layers



(b) Separated by problem size

Figure 13.: Comparison of average circuit fidelity if only depolarizing noise, only thermal relaxation noise or both is applied (baseline model)

greater than the effect of depolarizing noise. As explained in Section 5.3.2, circuit fidelity seems to have a great effect on QAOA performance, so we will compare the fidelity of the noisy circuits, with just depolarizing noise ($d_{TR} = 0$), with just thermal relaxation noise ($d_D = 0$) and with both depolarizing and thermal relaxation noise. This is visualized in Figure 13. To compute the circuit fidelity, we execute the noisy QAOA variant as usual, optimizing $\vec{\beta}$ and $\vec{\gamma}$. For the final parameters $\vec{\beta}, \vec{\gamma}$, we compute the fidelity of the output of the noisy circuit and the output of the ideal circuit, with the same parameters. This circuit fidelity is averaged over the problem instances. The charts only visualize the circuit fidelity for the QAOA, WSQAOA and WS-Init-QAOA. The RQAOA executes multiple circuits so there is no clear way to assign a fidelity to the algorithm's execution.

According to Figure 13, thermal relaxation is indeed the main noise source for the considered model, especially for deeper circuits. An interesting observation is the fact that the circuit fidelity differs slightly between algorithms. This can hardly be explained by the extra $\sqrt{X}$ pulse needed to prepare the initial state of the WSQAOA since the WS-Init-QAOA and the WSQAOA circuits have different average fidelities although they use exactly the same noise channels. Nor can it be fully explained by the inherent bias of thermal relaxation noise toward the computational basis favoring the initial states for the WSQAOA and WS-Init-QAOA, since the same difference in fidelity can be observed when only depolarizing noise is considered. In fact, when it comes to circuit fidelity, the order of the algorithms is reversed compared to their relative performance ratios shown in Figure 11. For example, the performance of the WSQAOA is most affected by noise, although the outputs of the noisy WSQAOA circuits are the closest to the ideal outputs among the tested variants. This might indicate that the large impact of noise on the WSQAOA's performance ratio can at least partially be explained by the fact that its performance ratio in the ideal case is higher compared to the standard QAOA and the WS-Init-QAOA, not because it is inherently less resistant to noise.

Figure 13b shows that, at 9 layers, the fidelity of the Partition circuits drops rapidly. This is an artifact of the greedy heuristic used by the QLM to reduce the depth of the time-evolved problem Hamiltonian. For Partition, the graph of the non-zero quadratic terms J_{ij} is always complete, so all Partition circuits with the same number of qubits use the same arrangement of gates. For Partition and $n \in \{5, 6, 7, 8\}$, with the heuristic used, the time-evolved problem Hamiltonian has a depth of 14 $CNOT$ gates. For $n \in \{9, 10\}$, the depth is 30 $CNOT$ gates. Since $CNOT$ gates are the most noisy (cf. Table 1), this explains the drop in fidelity. These circuit depths are definitely sub-optimal: For example, the edges of a complete graph with 10 vertices can be colored with only 9 colors such that no two edges of the same color are adjacent [69]. Therefore, we can apply the $\exp(-\mathrm{i}\gamma Z^{(i)}Z^{(j)})$ gates in 9 layers where each layer corresponds to one of the colors, resulting in a circuit with a depth of only 18 $CNOT$ gates. This shows that minimizing the depth of the time-evolved problem Hamiltonian can greatly improve noisy performance.

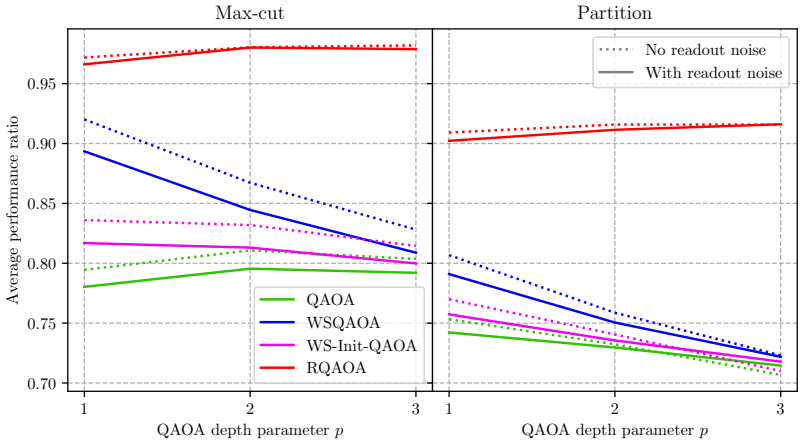


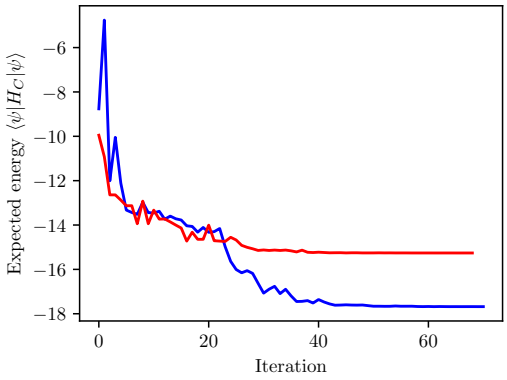
Figure 14.: Performance ratio: comparison between the baseline noise model with and without readout noise

6.4. The influence of readout noise

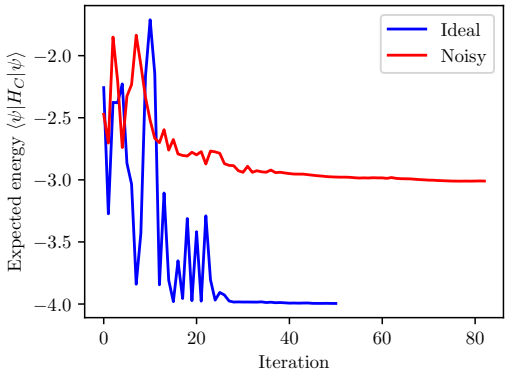
For reasons explained in Section 5.3.2, we ignored the effects of readout noise in the previous evaluations ($d_{AB} = 0$). Figure 14 shows the difference between the baseline noise model and the same model with readout noise ($d_{AB} = 1$). Not surprisingly, adding readout noise has a negative effect on the average performance ratio, where the magnitude of the effect seems to depend on the noise resistance of the variant, with the RQAOA being the least affected and the WSQAOA being the most affected. However, the effect of readout errors appears to be mostly independent of the number of layers, which makes sense, as it is modeled as a one-time effect and not as noise applied over the course of the circuit. In particular, if adding a layer improves the average performance ratio of some QAOA variant for the baseline noise model, then this is generally also true when adding readout noise. If anything, the performance difference actually seems to shrink as the number of layers increases. In fact, the performance for 3-layer QAOA and WS-Init-QAOA is actually slightly better when readout noise is included, although it is possible that this behavior can be explained by unintended side effects due to the specific experimental setup used.

6.5. The effect of noise on the classical optimizer

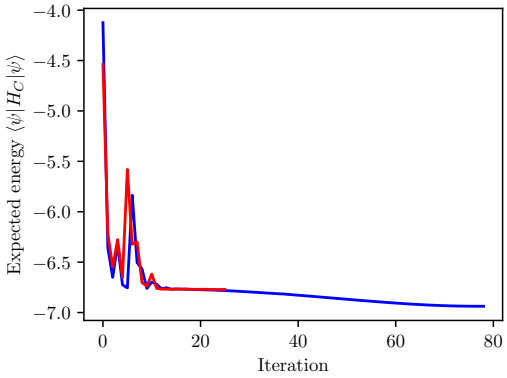
During the execution of the QAOA or one of its variants, the expected energy of the problem Hamiltonian $\langle\psi|H_C|\psi\rangle$ gradually decreases over time while the optimizer tries to find the optimal set of parameters $\vec{\beta}$ and $\vec{\gamma}$. Figure 15 shows how the energy changes over the course of this process for four example instances and for both the ideal and the noisy case. Comparing these and other optimization traces, one can see that the optimization process generally takes



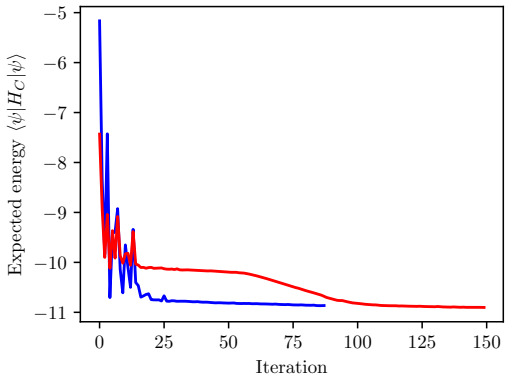
(a) Max-cut QAOA: $n = 10$, $p = 3$



(b) Max-cut WSQAOA: $n = 5, p = 3$



(c) Partition WSQAOA: $n = 5, p = 1$



(d) Partition WS-Init-QAOA: $n = 5, p = 3$

Figure 15.: Effect of circuit noise on the classical optimization for selected problem instances

longer for more circuit layers since having more layers also means having more parameters to optimize. WS-Init-QAOA circuits and especially WSQAOA circuits generally do not require as many iterations as the other variants since the initial state provided by the classical approximation algorithm already gives a head start when trying to find the optimal solution. These observations are not surprising, but it is reassuring that they can be translated almost directly to the noisy case as well, indicating that the optimization procedure operates similarly on noisy circuits as on ideal ones.

When comparing ideal and noisy circuits, for most problem instances the optimization traces look similar to Figure 15a and Figure 15b in the sense that both ideal and noisy optimizations eventually plateau. The difference is that the plateau energy is generally higher in the noisy case than in the ideal case. However, this is not true for all problem instances. In Figure 15c, the optimization trace is very similar between the ideal and the noisy case for the first 20 iterations. However, the SciPy `COBYLA` optimizer terminates much earlier in the noisy case, although the trajectory of the ideal case suggests that a better final state might have been possible. In Figure 15d, both the ideal and the noisy optimization plateau at around iteration 25 but on different energy levels. However, the noisy optimizer can recover within the maximum 150 of iterations and reach a final energy close to the final energy in the ideal case. The examples of Figure 15c and Figure 15d suggest that the gap between the ideal and noisy results could be reduced by improving the classical parameter optimization. To test this, the main evaluation from Section 6.1 was repeated, but with a maximum of 1000 iterations and a tolerance of 10^{-4}. The results, which can be found in Section A.4 in the appendix, show a slight improvement in both ideal and noisy performance. However, the relative performance ratio (noisy ratio divided by ideal ratio) seems to be about the same, indicating that simply increasing the number of optimizer iterations does not bring a substantial improvement.

6.6. Comparing the noise of single-qubit and CNOT gates

As explained in Section 5.5, the studied circuits contain only two types of noisy gates: single-qubit $\sqrt{X}$ gates and two-qubit $CNOT$ gates. As shown in Table 1, $CNOT$ gates are much noisier than single-qubit gates in the sense that they have both a lower fidelity and a higher duration. Therefore, it is reasonable to assume that the gate infidelities of the $CNOT$ gates, as well as the thermal relaxation noise due to their duration, have a greater impact on the performance ratio than the noise caused by $\sqrt{X}$ gates. Figure 16 confirms this assumption. The charts show the same data as Figure 10a, but in addition to the ideal and the baseline noise models, they also display two additional models: one where $CNOT$ gates are assumed to be noiseless, having a fidelity of 1 and a duration of 0, and one where $\sqrt{X}$ gates are assumed to be noiseless.

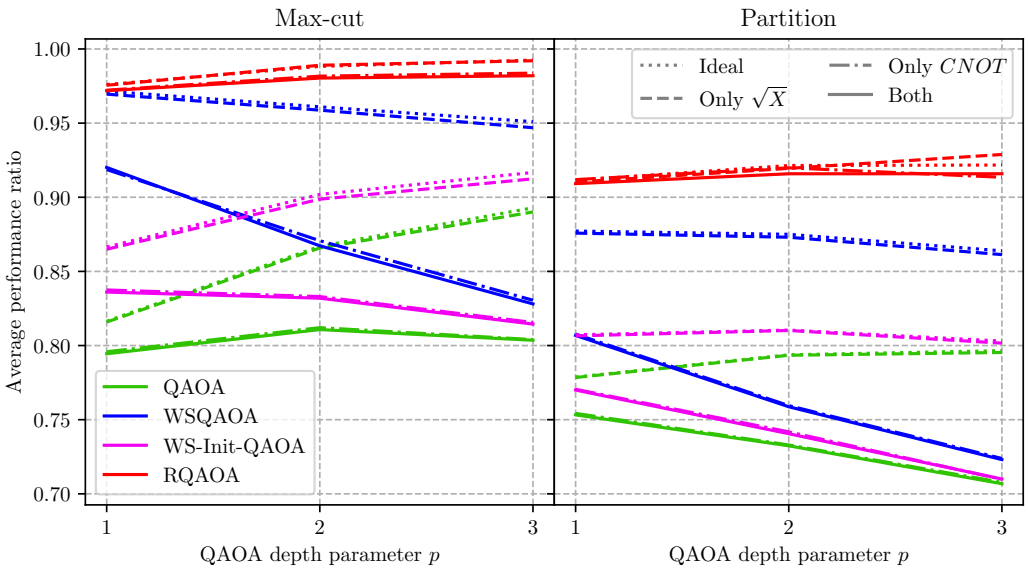


Figure 16.: Performance ratio compared between when no gates induce noise, only single-qubit gates induce noise, only $CNOT$ gates induce noise or both.

Figure 16 shows that $CNOT$ gates indeed have a much larger effect on the performance of the QAOA variants to such a degree that the results for the baseline model are nearly the same as the model which assumes $\sqrt{X}$ gates to be noiseless. Similarly, if we ignore $CNOT$ noise, the noise of the single-qubit gates alone only has a small impact on the performance of the QAOA variants. As shown above, RQAOA is the variant most resilient to noise for the tested problem instances. This observation still seems to hold when only considering single-qubit noise although the differences between the variants are very small.

6.7. The effect of sample size on noisy performance

As discussed in Section 5.4, we have assumed that the expected energy $\langle\psi|H_C|\psi\rangle$ can be measured with effectively infinitely many samples. Figure 17 shows how the performance of the QAOA variants decreases for Max-cut when fewer samples are used. The chart shows the relative performance ratios for 10 and 100 samples, compared to the baseline model, where a relative performance ratio of less than 1 indicates that the performance is worse than when using an effective sample size of ∞ and a value of 1 indicates no difference. Due to time constraints (cf. Section 5.7.2), the algorithms were run 5 times for each problem instance, resulting in statistical uncertainty in the data. Still, several trends can be observed: First of all, while the average performance ratio is definitely negatively impacted by finite sample sizes, this impact is comparatively small for $p = 1$ where for each algorithm the decrease is not more than 5 %, even in

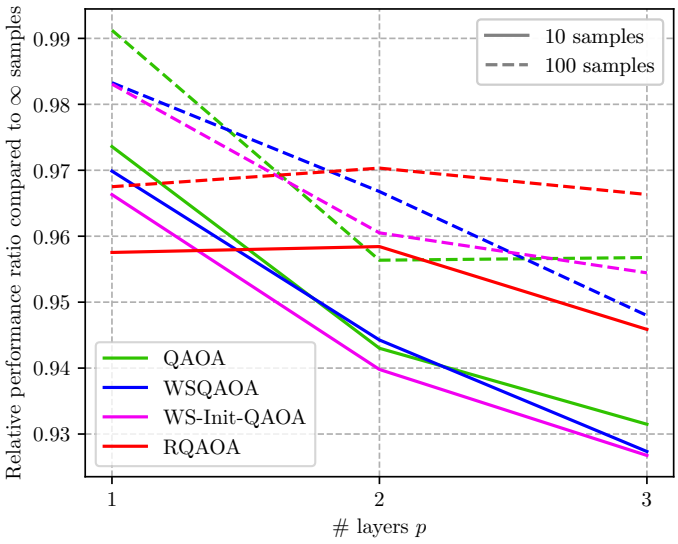


Figure 17.: How does the number of samples for estimating $\langle\psi|H_C|\psi\rangle$ during the QAOA affect the results for Max-cut? The baseline model uses density matrices and can therefore be considered to use infinitely many samples.

the extreme case of only 10 samples. Still, the errors due to sampling increase as the number of layers grows. This is probably because of the difficulty of finding the optimal set of parameters if the expected energy $\langle\psi|H_C|\psi\rangle$ can only be estimated with a very low accuracy. Consequently, the decline in performance ratio from $p = 2$ to $p = 3$ is less severe for 100 samples than for 10 samples.

Interestingly, RQAOA handles a small number of samples worse than the other variants when single-layer QAOA circuits are used. On the other hand, the increase in sampling error due to more layers is not as bad as for the other variants. Therefore, the question of which QAOA variant is most sensitive to using fewer samples cannot be answered conclusively from the data collected.

6.8. The performance of the RQAOA for higher levels of noise

Out of the QAOA variants tested, the RQAOA is clearly the one most resistant to noise. Even though the analyzed noise model accounts for the dominant noise sources in physical quantum computers, namely thermal relaxation and gate infidelities, we should not ignore that there are still some unconsidered aspects, most notably additional swap gates due to limited circuit connectivity. Therefore, the question arises if the RQAOA still performs this well when the noise level increases. Figure 18 visualizes the performance of the RQAOA for higher levels of thermal relaxation and depolarizing noise ($d_D, d_{TR} \in \{1, 2, 3, 4\}$). The RQAOA still performs quite well if the noise level doubles ($d_D = d_{TR} = 2$). For

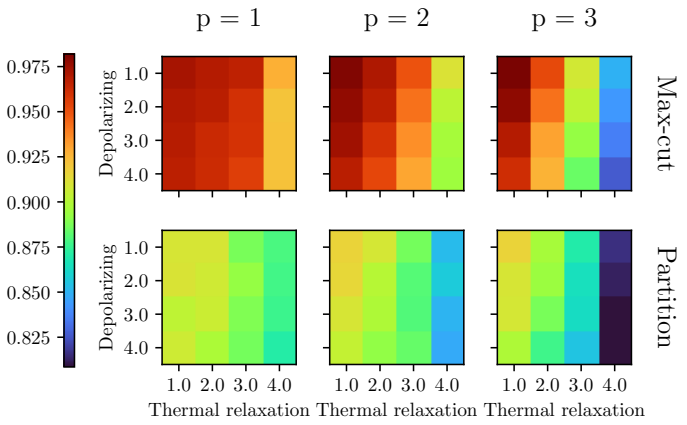


Figure 18.: Average performance ratio of RQAOA for higher levels of noise: $d_D, d_{TR} \in \{1, 2, 3, 4\}$

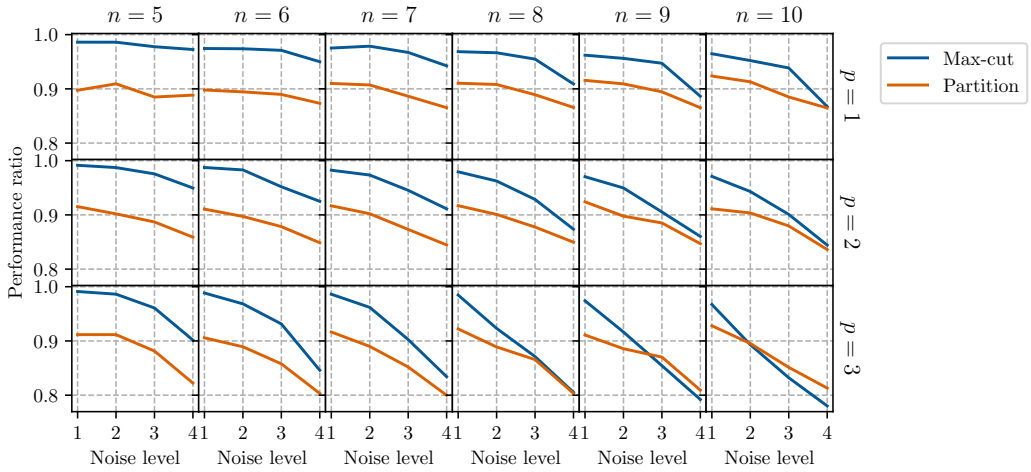


Figure 19.: Average performance ratio of RQAOA for higher levels of noise ($d_D = d_{TR} \in \{1, 2, 3, 4\}$): algorithm comparison

$d_{TR} \geq 3$, however, a noticeable decrease in the performance ratio can be observed, which especially affects circuits with more than one layer. As already indicated by Figure 13, for the noise model considered, thermal relaxation noise has a greater effect on circuit fidelity than depolarizing noise. Since both noise sources cause an approximately exponential decrease in circuit fidelity, this difference is even more pronounced at higher noise levels.

Figure 19 gives a different perspective on the data of Figure 18. Here, we only consider the case where $d_D = d_{TR} \in \{1, 2, 3, 4\}$, but separate the data by number of layers and problem size. While the RQAOA generally performs better for Max-cut than for Partition, large levels of noise seem to affect Max-cut more for larger problem sizes, closing the gap between the two problems. However, Partition, unlike Max-cut, is originally a minimization problem, so the performance ratio might not necessarily be perfectly comparable between the two problems.

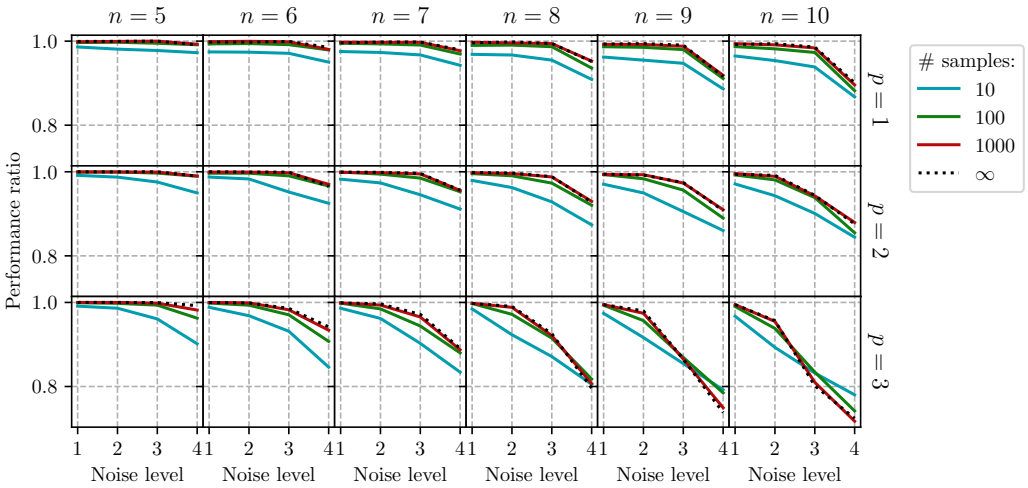


Figure 20.: Performance of Max-cut-RQAOA for higher levels of noise ($d_D = d_{TR} \in \{1, 2, 3, 4\}$) if more samples are used to find the most correlated edge ($\langle\psi|Z^{(i)}Z^{(j)}|\psi\rangle$)

As discussed in Section 5.4, although an effectively infinite number of samples is used to find the optimal parameters $\vec{\beta}$ and $\vec{\gamma}$, we use only 10 samples to find the most correlated edge J_{ij} in order to not give the RQAOA an unfair advantage over the other QAOA variants. It might be possible, however, that 10 samples are simply not enough to overcome higher levels of noise. Figure 20 shows how better sampling of the $\langle\psi|Z^{(i)}Z^{(j)}|\psi\rangle$ improves the performance of Max-cut-RQAOA for high noise levels. The charts also show the results for the effective sample size of ∞, where the exact probabilities for each computational basis state are used to measure $\langle\psi|Z^{(i)}Z^{(j)}|\psi\rangle$, giving an upper bound on the best possible performance. While higher numbers of samples improve the quality of the solutions to some degree, there definitely seems to be a limit to how much better sampling can help against higher levels of noise. In particular, using more than 1000 samples has a negligible effect on the performance of the RQAOA. Even infinitely many samples are not enough to overcome larger noise levels, especially for deeper circuits with more qubits and QAOA layers. More samples actually seem to have a negative effect for $p = 3$ and $n \geq 9$. This might indicate that the RQAOA benefits from more randomization if the noise level grows too large. However, it could also be the result of other undesired side effects, which might be eliminated by, for example, a different classical optimizer.

6.9. The effect of circuit depth on noise resilience

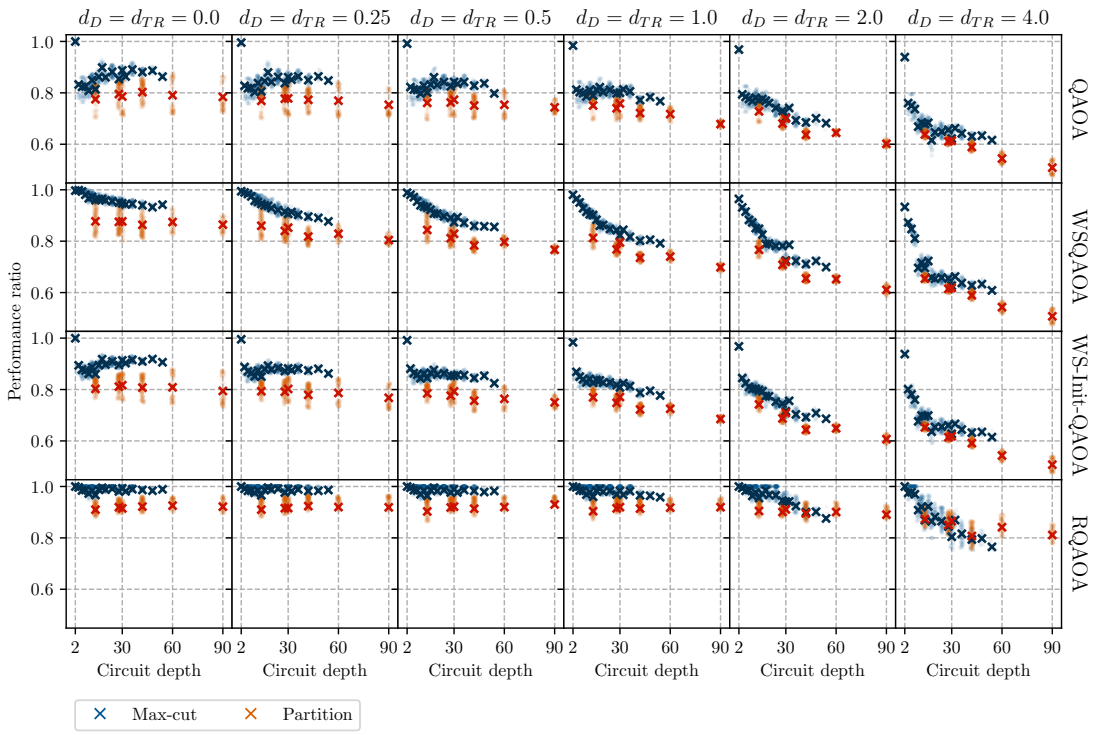
From the previous evaluations, we can deduce that thermal relaxation due to $CNOT$ gates seems to be the largest source of noise, affecting the performance

of the QAOA variants the most. $CNOT$-induced thermal relaxation is directly determined by the depth of the circuit if we define *circuit depth* as the total execution time of the circuit, assuming that $CNOT$ gates have duration 1 and single-qubit gates have duration 0. The idea that circuit depth has the largest effect on performance is further supported by Figure 11 in Section 6.1, which suggests that for Max-cut the effect of noise on the performance increases as the number of qubits, and thus the circuit depth, increases. On the other hand, for Partition, the effect of noise seems to be about the same for $n \in \{5, 6, 7, 8\}$, even though the number of $CNOT$ grows quadratically with the problem size. It is only when the circuit depth increases at $n = 9$ that a decrease in the relative performance ratio can be observed.

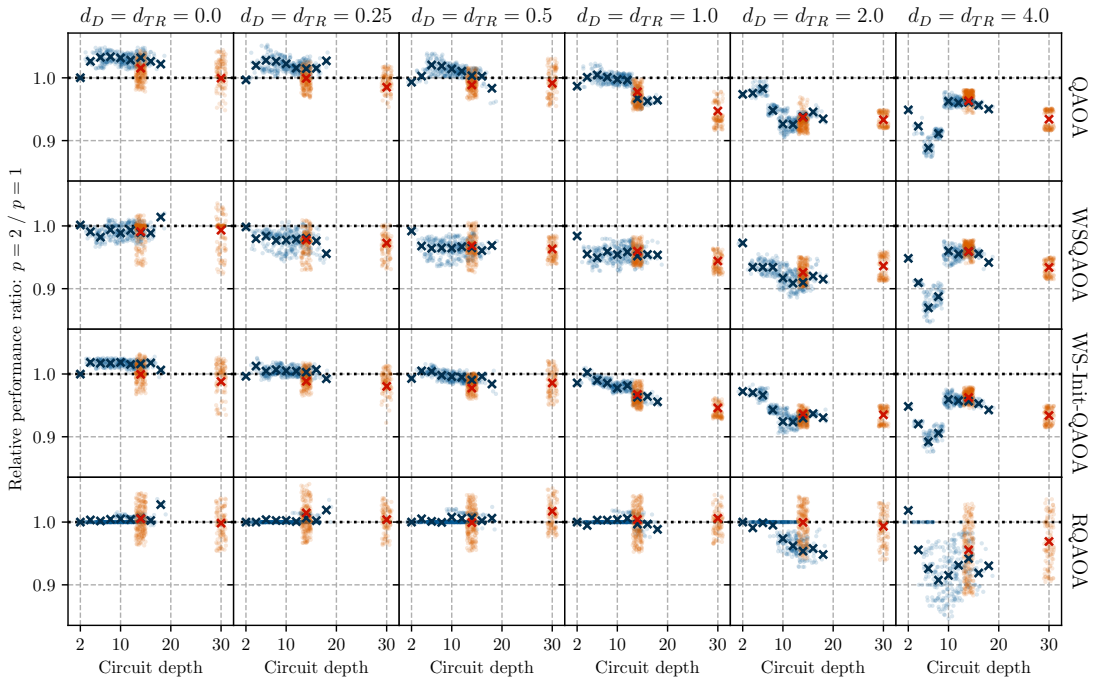
Figure 21a shows the performance ratio for $p \in \{1, 2, 3\}$ depending on circuit depth for different levels of noise in a jitter plot. To help readability, for every algorithm, noise level, problem and circuit, only data points between the 25th and the 75th percentile are included in the plot. The $\times$ show the average performance ratio for each depth. For Max-cut and small noise levels, deeper circuits generally produce better results due to more QAOA layers, plateauing at around depth 30. For larger noise levels, on the other hand, deeper circuits almost always have a negative effect on the performance. The results for the Partition problem are less conclusive since there are fewer possible circuit depths and the data points have a larger variance for circuits of the same depth. Nevertheless, the general trends for Partition seem to be similar to those for Max-cut.

Figure 21b shows the performance ratio for $p = 2$, divided by the performance ratio for $p = 1$, depending on the single-layer circuit depth. If we define an algorithm to be *noise-resilient* if the performance ratio for $p = 2$ is larger than the performance ratio for $p = 1$, we can say that for the baseline noise model ($d_D = d_{TR} = 1$), the standard QAOA is noise-resilient up to a circuit depth of approximately 10. For noise level 0.5, the QAOA is resilient up to at least circuit depth 15, while for noise level 2, noise resilience can only be achieved for very small circuit depths. The WS-Init-QAOA manages to be noise-resilient at similar circuit depths for noise level at most 0.5, for higher noise levels it does not achieve noise resilience reliably. In general, the effect of circuit depth on noise resilience seems to be less for the WS-Init-QAOA compared to the standard QAOA, as indicated by the smaller slope of the plots. For the RQAOA, circuit depth has an even less noticeable effect. Nevertheless, for noise levels above 1, noise resilience seems difficult to achieve.

Comparing the two problems Max-cut and Partition, while the deeper circuits of Partition definitely impact the noise resilience negatively, for the QAOA and WS-Init-QAOA, the relative performance of the 2-layer circuits is still slightly worse on Partition instances with $n \in \{5, 6, 7, 8\}$ when compared to Max-cut instances with the same circuit depth. On the other hand, for the WSQAOA and RQAOA, the Partition instances show similar or slightly better relative performance than the comparable Max-cut instances.



(a) Performance for $p = \{1, 2, 3\}$



(b) The relative performance for $p = 2$, compared to $p = 1$

Figure 21.: Algorithm performance depending on the circuit depth counted in terms of $CNOT$ gates. The data presented in each chart displays the 25th to 75th percentile for each depth. To enhance legibility, the x-position of each data point is randomized. The $\times$ signify the average values for each depth.

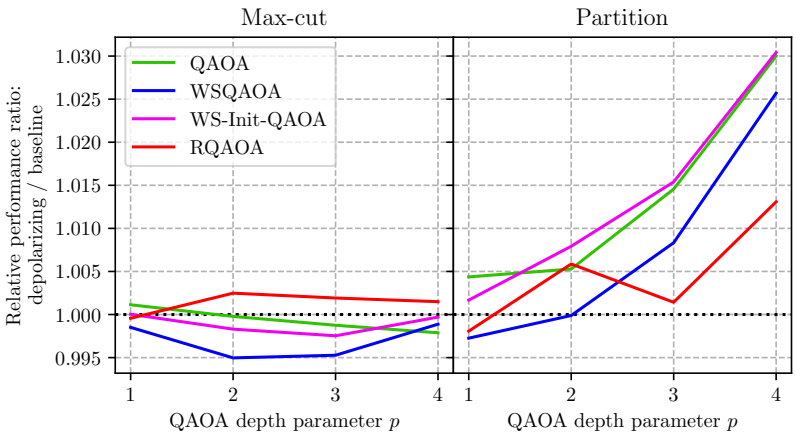


Figure 22.: Relative performance ratio, comparing the baseline noise model with a model where each noise channel is replaced with a depolarizing channel of the same fidelity.

6.10. Difference of thermal relaxation noise and depolarizing noise with the same fidelity

As discussed in the previous sections of this chapter, thermal relaxation affects the performance of the QAOA variants more than depolarizing noise since it has a greater effect on circuit fidelity for the investigated noise parameters. For the final analysis of this evaluation, we want to consider the question: Is this difference in performance only due to a difference in fidelity, or does thermal relaxation inherently have a stronger negative effect on the QAOA than depolarizing noise? To answer this question, we will replace every thermal relaxation channel with a depolarizing channel with the same fidelity. Specifically, for gate noise, instead of sequentially composing thermal relaxation and depolarizing noise, a single depolarizing channel is used whose fidelity matches the desired gate fidelity from Table 1. For $\sqrt{X}$ gates, a one-qubit depolarizing channel is used, for $CNOT$ gates we will use a two-qubit depolarizing channel. Thermal relaxation channels representing idle noise are replaced with one-qubit depolarizing channels with identical fidelity.

Figure 22 shows the relative performance ratios of the four QAOA variants in the sense that for each problem instance, the average performance ratio for the depolarizing model is divided by the average result for the baseline noise model, and these quotients are averaged. The first thing that stands out is that the differences between the baseline and depolarizing models are relatively small in most cases, suggesting that the amount of noise, characterized by the average channel fidelity, plays a larger role than the type of noise applied. However, one can observe that for Partition and $p \geq 3$ the results of the depolarizing model are a few percentage points better than the results of the original noise model. A closer look at the data shows that 9- and 10-qubit problem instances are more

affected, suggesting that this difference is larger for deeper circuits in general. Note that, as explained in Section 6.3, Partition circuits for $n \geq 9$ are much deeper than Partition circuits with $n \in \{5, 6, 7, 8\}$. While the difference between the two models is much smaller for the Max-cut instances, with the baseline model actually performing slightly better in most cases, we can still notice a similar trend of the relative performance ratio increasing for $p = 4$.

The differences between the results for the baseline and the depolarizing model can probably at least partly be explained by properties of the analyzed optimization problems Max-cut and Partition. The kind of thermal relaxation noise considered by the evaluation drives the state of each qubit towards $|0\rangle\langle 0|$ over time which corresponds to setting each spin variable s_i of the Ising Hamiltonian to 1. However, this particular assignment results in the worst possible solution, both for Max-cut and Partition, with a performance ratio of 0. On the other hand, depolarizing noise drives each qubit towards the maximally mixed state $\mathbb{1}/2$, which corresponds to choosing $s_i = \pm 1$ with equal probability. This assignment of variables leads to solutions with performance ratios between roughly 0.6 and 0.7 as shown by Figure 10, possibly giving the depolarizing model an advantage.

When looking at which variants are most affected by the type of noise applied, there does not seem to be a clear trend for the RQAOA, perhaps because of insufficient data. Of the remaining variants, the WSQAOA has the smallest relative performance ratio, indicating that it is slightly more affected by depolarizing noise than the other variants. This might be due to the fact that both the amplitude damping and the phase damping components of the thermal relaxation channel are defined in terms of the computational basis. For the WSQAOA, the initial state and the ground state of the mixer Hamiltonian are closer to the computational basis states than $|+\rangle$, which is why thermal relaxation might have a slight advantage over depolarizing noise for the WSQAOA, compared to the other variants.

7. Conclusion

In this thesis, we conducted simulations to investigate the performance of the QAOA, the WSQAOA, the WS-Init-QAOA and the RQAOA for the problems *Max-cut* and *Partition* when run on noisy quantum circuits with 5 to 10 qubits. For this purpose, a realistic unified noise model based on real data from IBM-Q superconducting quantum computers was developed which uses quantum channels to simulate the effects of thermal relaxation, gate infidelities and SPAM errors. The simulations were performed on an Eviden Qaptiva 800 quantum simulation platform. In order to take full advantage of the platform's computational capabilities and to perform the computationally intensive density-matrix-based simulations quickly, a custom simulation pipeline was implemented on top of the platform's proprietary QLM library.

As the main result of the thesis, we conclude that noise has by far the least effect on the RQAOA, which in its noisy version actually outperforms the ideal versions of the other three algorithms. This appears to indicate that the RQAOA's idea of assigning values to the spin variables incrementally instead of all at once is much less disturbed by noise. The remaining variants are similarly impacted by noise, although the WSQAOA is slightly more affected than the standard QAOA while the WS-Init-QAOA is slightly less affected. The reasons why the WSQAOA is less noise-resistant than the standard QAOA are not clear. What can be said, however, is that it is probably not due to the WSQAOA mixer Hamiltonian being more susceptible to noise than one of the standard QAOA. In fact, the fidelity of noisy WSQAOA circuits actually seems to be slightly better when compared to the standard QAOA.

Of the noise sources analyzed, thermal relaxation has the greatest impact on performance. The data suggests that this is mainly because it has the largest effect on the circuit fidelity for the studied noise parameters. However, there is also some evidence that thermal relaxation noise has an inherently larger negative impact on QAOA performance than depolarizing noise, which was used to model gate infidelities, although this evidence is not conclusive. SPAM errors also degrade the performance of the QAOA variants, but this decrease is much smaller than the one caused by thermal relaxation and gate infidelities whose effect on circuit fidelity is approximately exponential in the circuit depth. Therefore, from a quantum hardware manufacturer's perspective, improving qubit coherence times and reducing gate durations generally benefit the tested QAOA variants more than improving gate fidelities. Compared to single-qubit gates, $CNOT$ gates not only have much larger gate infidelities, but also take

much longer to execute, thereby inducing more thermal relaxation noise. Thus, all QAOA variants would benefit greatly from improving the fidelity of $CNOT$ gates and especially from reducing their duration.

However, the data collected suggest that the number of $CNOT$ gates is less critical than the increased circuit depth they cause. For the analyzed noise model, the standard QAOA manages to achieve better results with two layers than with one layer as long as the depth of the time-evolved problem Hamiltonian does not exceed 10 $CNOT$ gates. For twice the noise level the limit is much lower at about 5 $CNOT$ gates. For the RQAOA, adding a second layer does improve the results at the baseline noise model, even though the improvement is very small. For higher noise levels, adding more layers is generally not worth the additional noise caused by thermal relaxation and gate infidelities. Still, for a depth of about 10 $CNOT$ gates, the RQAOA achieves very good results with only single layer even if the noise level is doubled. At four times the noise level, RQAOA still achieves an average performance ratio of more than 0.8, while the other variants do not perform much better than random guessing at circuit depth 5 and above.

For most evaluations, we assumed that the expected energy of the problem Hamiltonian $\langle\psi|H_C|\psi\rangle$ can be sampled with perfect accuracy, while only 10 samples are used for the RQAOA edge correlations $\langle\psi|Z^{(i)}Z^{(j)}|\psi\rangle$. Decreasing the number of samples for the expected energy during the optimization of the QAOA parameters negatively impacts the performance. This impact grows for higher number of QAOA layers, indicating that more noisy circuits might generally require more samples. Still, even for sample size 10, all variants reach levels of performance within 90 % of their performance with infinitely many samples. Which variants are more affected by fewer samples, can not be answered conclusively from the collected data.

Regarding the sampling of the edge correlations for the RQAOA, using more than 1000 samples does not improve the results for the considered problem instances. For this many samples, the considered problem instances can be approximated with essentially perfect accuracy in the ideal case. For large noise levels, even using an infinite number of samples will result in greatly degraded performance compared to the ideal case, indicating that there appears to be a limit to how much the effects noise can be mitigated by better sampling of the RQAOA edge correlations.

To summarize, when considering the studied problem instances, it can be said that the performance of the QAOA variants is most affected by circuit depth while other aspects like readout noise, the number of qubits, the number of samples and the studied problem are secondary factors with a significant but smaller effect. When trying to achieve good results on noisy circuits, it is therefore essential to consider the depth of the problem Hamiltonian and possible strategies to reduce it, such as using a Hamiltonian with fewer quadratic terms or using a Hamiltonian whose terms can be rearranged in a way that minimizes circuit depth.

Although many aspects have been considered in the analysis performed in this thesis, there are several factors that could not be addressed within the limited time frame. The largest source of unaccounted noise is the extra swap gates required due to the limited connectivity of physical superconducting quantum computers since the noise model studied assumes a complete coupling graph. Even though a performance evaluation for higher noise levels was performed which provides some insight into how the variants might perform with additional $CNOT$ gates, more experiments are needed. There are also other potential noise sources which were not considered [70]–[72]. For the analysis, the SciPy `COBYLA` optimizer with a relatively low tolerance and number of iterations was used to speed up the performed simulations. The optimization traces for some problem instances suggest that using a different classical optimizer might reduce the gap between ideal and noisy QAOA. There has been a great deal of research into improving QAOA parameter optimization. A good overview of the various techniques is given in Section 3.2 of [8]. Finally, this work only considers two problems, Max-cut and Partition. Although most findings of the analysis apply to both problems, there are still differences, most notably that the performance ratios for Partition are a bit lower in general, both in the ideal and in the noisy case. It would be interesting to know to what degree the results of the evaluation can be generalized to other problems.

To conclude, the QAOA and its variants are among the most promising quantum algorithms of the NISQ era. Still, the level of noise experienced in current quantum systems poses a significant threat to their ability to gain an advantage over classical approximation algorithms and heuristics. The results for the RQAOA show that there are techniques to mitigate the effects of noise. However, these results also suggest that many of these techniques have limitations. There are certainly many challenges to overcome before quantum approximate optimization algorithms are a practical option for solving large-scale real-world optimization problems.

A. Appendix

A.1. The average fidelity of the repeated depolarizing channel

Let $\mathcal{E}^k = \underbrace{\mathcal{E} \circ \mathcal{E} \circ \cdots \circ \mathcal{E}}_{\times k}$ denote the k-fold repeated application of channel $\mathcal{E}$. Then, the average fidelity of the quantum channel describing for the k-fold application of the n-qubit depolarizing channel with probability p is

$$\bar{F}\left(\mathcal{E}_{D,n}^k\right) = \frac{1}{2^n} + \frac{2^n - 1}{2^n} \mathrm{e}^{k \ln(1-p)}. \tag{5.1}$$

Proof. We will begin by showing that $\mathcal{E}_{D,n}^k$ is equivalent to a single n-qubit depolarizing channel with depolarizing probability $1 - (1-p)^k$, that is,

$$\mathcal{E}_{D,n}^k(\rho) = (1-p)^k \rho + (1 - (1-p)^k) \frac{\mathbb{1}}{2^n}. \tag{A.1}$$

We can prove this by induction: By definition of the depolarizing channel (4.13), (A.1) is true for $k = 1$. Inductively, assume (A.1) is true for some k. Then, it is also true for $k+1$ since

$$\begin{aligned}
\mathcal{E}_{D,n}^{k+1}(\rho) = \mathcal{E}_{D,n}^k(\mathcal{E}_{D,n}(\rho)) &= (1-p)\mathcal{E}_{D,n}(\rho) + (1 - (1-p)^k)\frac{\mathbb{1}}{2^n} && \text{by ind. hypo.} \\
&= (1-p)^k \left((1-p)\rho + p\frac{\mathbb{1}}{2^n} \right) + (1 - (1-p)^k)\frac{\mathbb{1}}{2^n} \\
&= (1-p)^{k+1}\rho + \left(p(1-p)^k + 1 - (1-p)^k\right)\frac{\mathbb{1}}{2^n} \\
&= (1-p)^{k+1}\rho + \left((1-(1-p))(1-p)^k + 1 - (1-p)^k\right)\frac{\mathbb{1}}{2^n} \\
&= (1-p)^{k+1}\rho + (1 - (1-p)^{k+1})\frac{\mathbb{1}}{2^n}.
\end{aligned}$$

Therefore, by (4.14), the average fidelity of $\mathcal{E}_{D,n}^k$ is

$$\begin{aligned}
\bar{F}(\mathcal{E}_{D,n}^k) &= 1 - (1 - (1-p)^k) + \frac{1 - (1-p)^k}{2^n} = (1-p)^k + \frac{1}{2^k} - \frac{(1-p)^k}{2^n} \\
&= \frac{1}{2^n} + \frac{2^n - 1}{2^n}(1-p)^k = \frac{1}{2^n} + \frac{2^n - 1}{2^n}\mathrm{e}^{k\ln(1-p)},
\end{aligned}$$

proving the claim. □

A.2. Ideal and noisy results, separated by number of layers and qubits

Figures 23 and 24 show the data from the main evaluation (cf. Section 6.1), separated by ideal/noisy, problem, problem size, algorithm and number of layers.

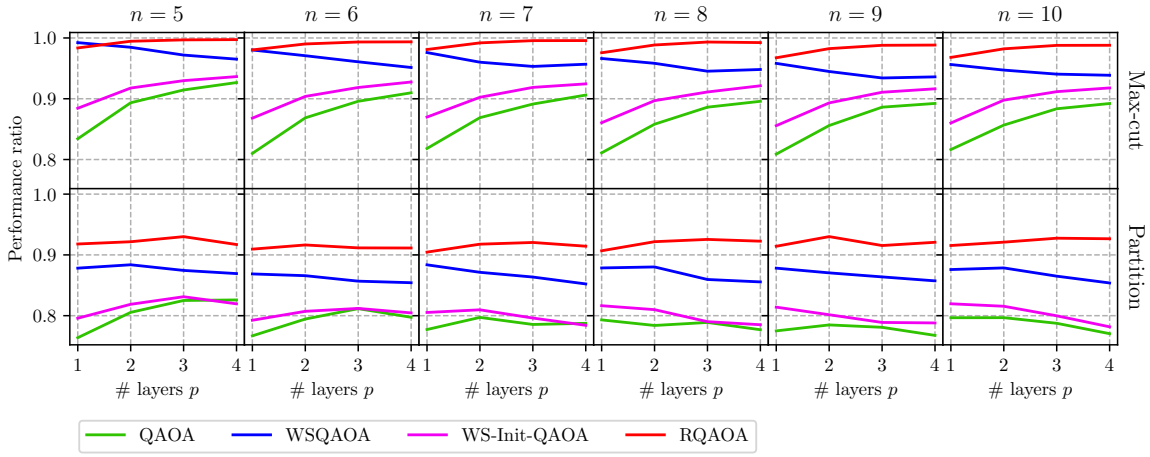


Figure 23.: Ideal circuits

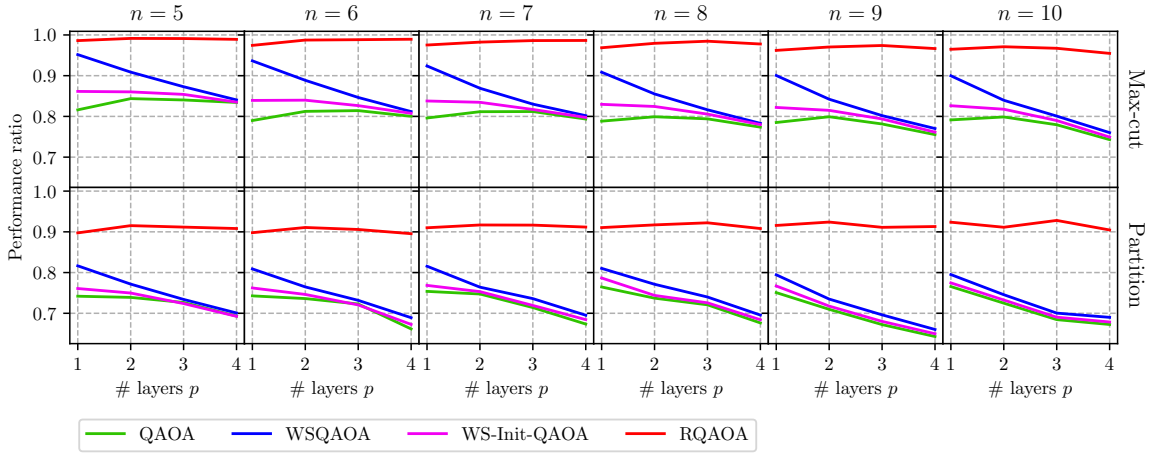


Figure 24.: Baseline noise model ($d_D = 1, d_{TR} = 1, d_{AB} = 0$)

A.3. Results for selected noise levels, separated by number of layers and qubits

Figures 25, 26, 27 and 28 show the performance ratios for the noise level analysis results with $d_D = d_{TR} \in \{0.25, 0.5, 2.0, 4.0\}$ and $d_{AB} = 0$, separated by ideal/noisy, problem, problem size, algorithm and number of layers.

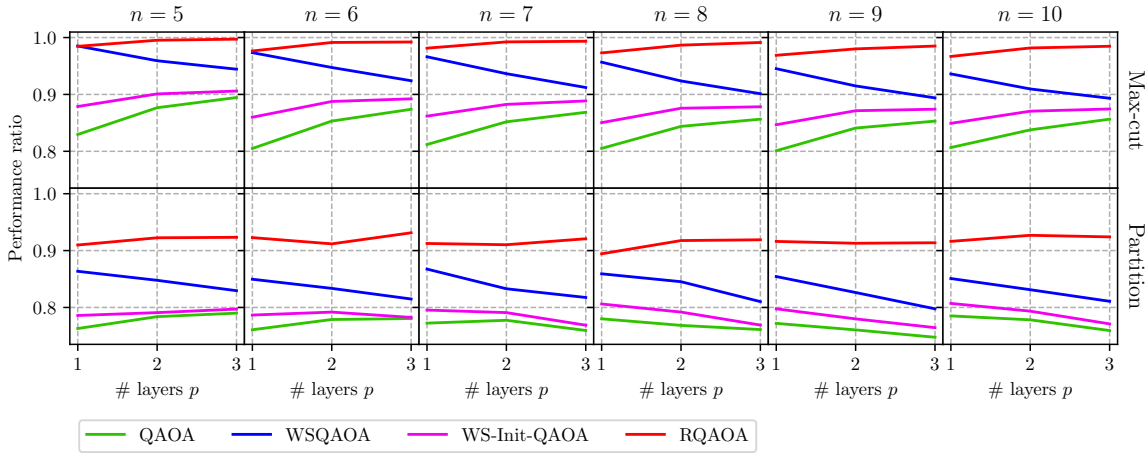


Figure 25.: $d_D = 0.25, d_{TR} = 0.25, d_{AB} = 0$

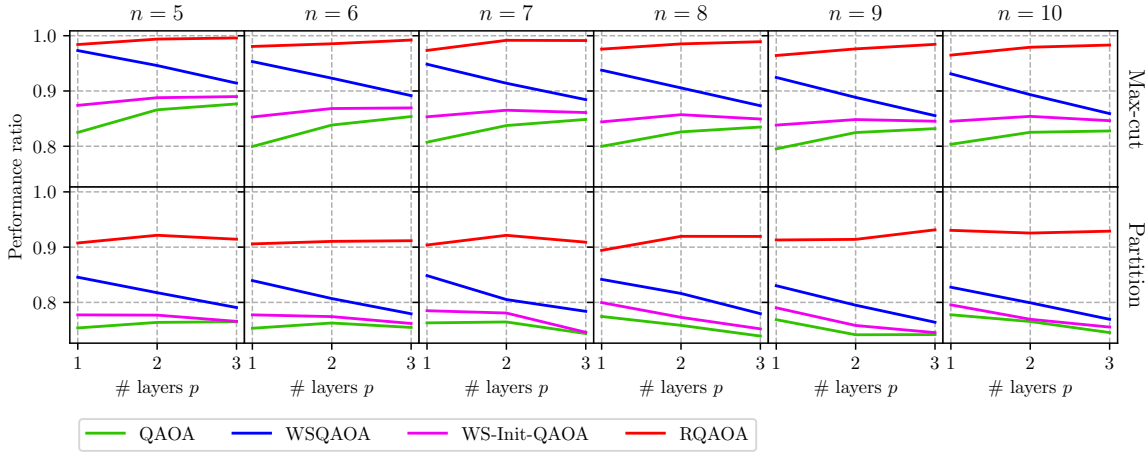


Figure 26.: $d_D = 0.5, d_{TR} = 0.5, d_{AB} = 0$

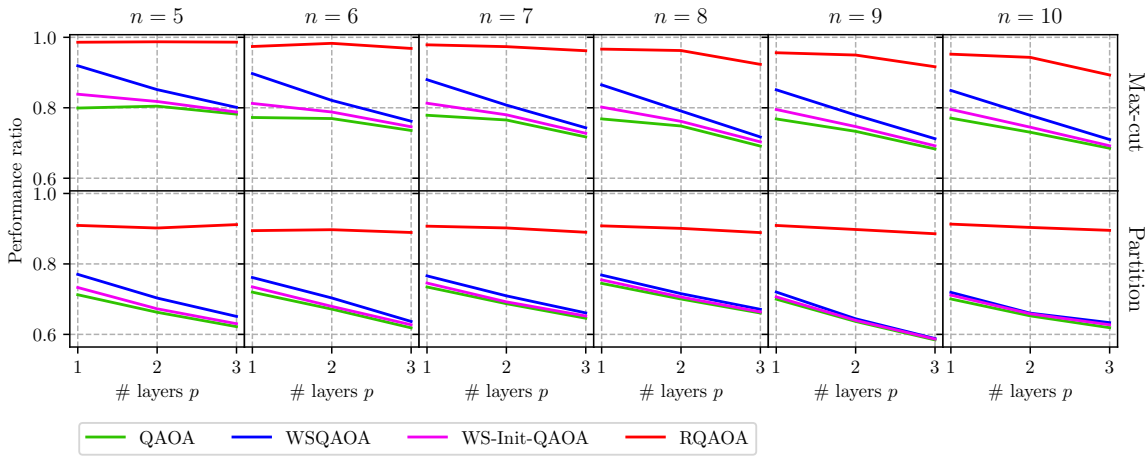


Figure 27.: $d_D = 2, d_{TR} = 2, d_{AB} = 0$

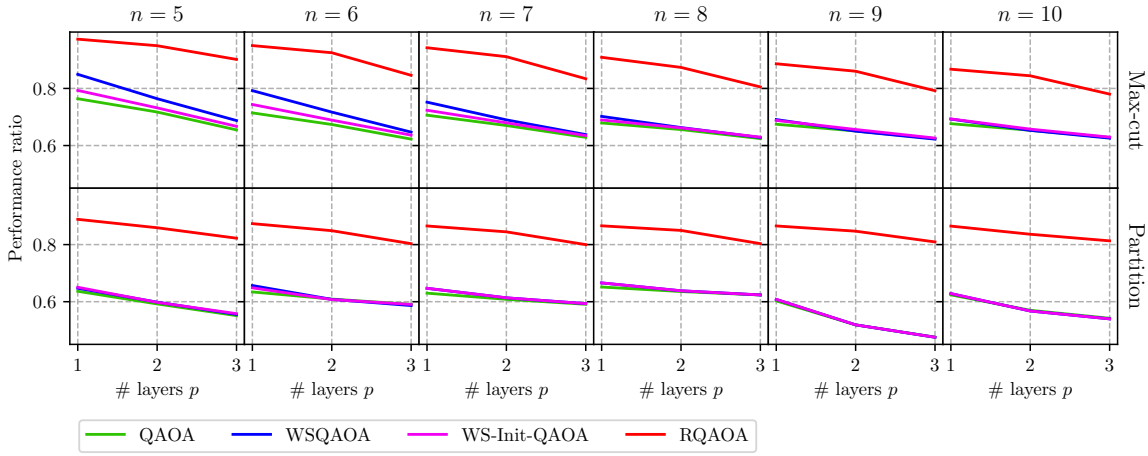


Figure 28.: $d_D = 4, d_{TR} = 4, d_{AB} = 0$

A.4. Results when using more QAOA iterations

For the evaluation in Chapter 6, the SciPy `COBYLA` optimizer with a tolerance of 10^{-2} and a maximum of 150 iterations was used. Figures 30, 30 and 31 visualize the main results from the main evaluation in Section 6.1, but when using a tolerance of 10^{-4} and a maximum of 1000 iterations.

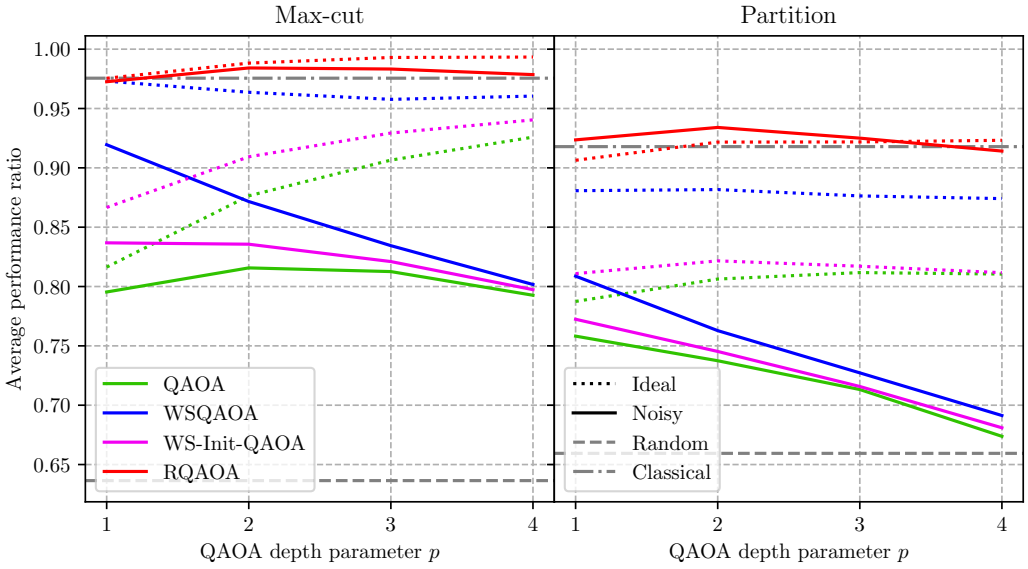


Figure 29.: Average performance ratio, separated by ideal/noisy, problem, algorithm and number of layers (see also Figure 10a)

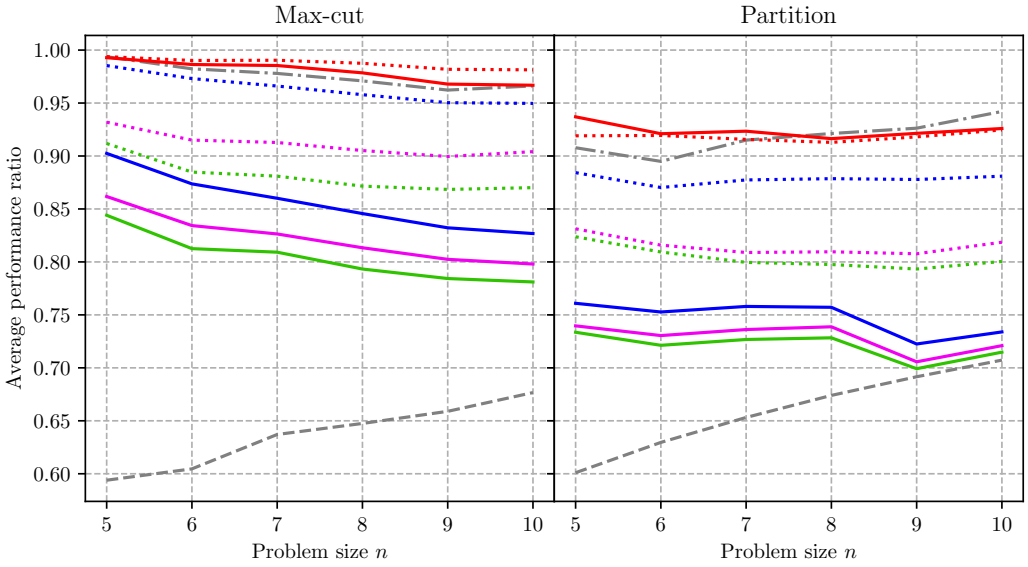


Figure 30.: Average performance ratio, separated by ideal/noisy, problem, problem size and algorithm (see also Figure 10b)

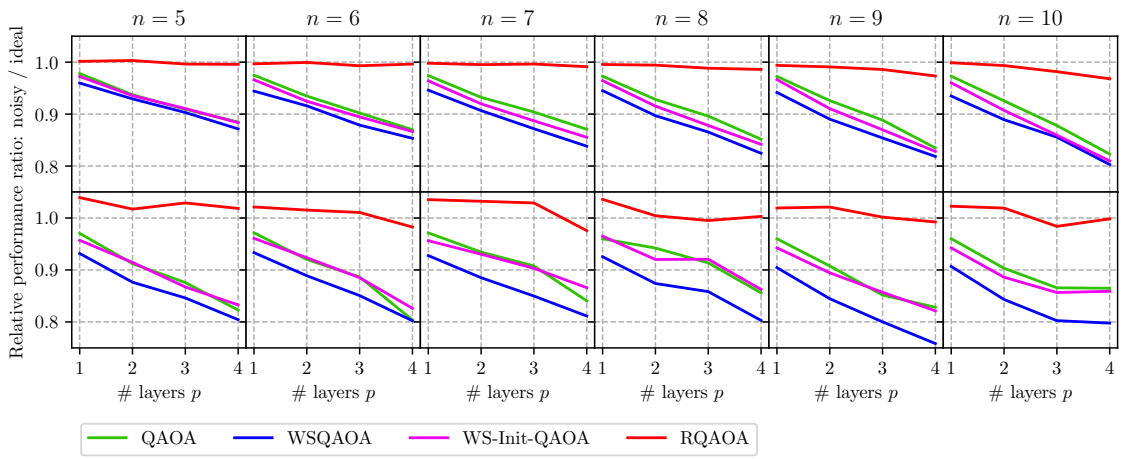


Figure 31.: Relative Average performance ratio (noisy/ideal), separated by problem, problem size, algorithm and number of layers (see also Figure 11)

List of Figures

List of Tables

Bibliography

[1] P. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, IEEE Comput. Soc. Press, 1994, pp. 124–134, ISBN: 9780818665806. DOI: 10.1109/SFCS.1994.365700.

[2] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *STOC ’96*, ACM Press, 1996, pp. 212–219, ISBN: 9780897917858. DOI: 10.1145/237814.237866.

[3] IBM Corporation. “Qiskit API Reference (v0.25.1), IBM Quantum Documentation: Fake Provider.” (2023), [Online]. Available: https://docs.quantum-computing.ibm.com/api/qiskit/providers_fake_provider (visited on 10/01/2023).

[4] J. Preskill, “Quantum computing in the NISQ era and beyond,” *Quantum*, vol. 2, p. 79, Aug. 2018. DOI: 10.22331/q-2018-08-06-79.

[5] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm*, 2014. arXiv: 1411.4028 [quant-ph].

[6] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds. Springer US, 1972, pp. 85–103, ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9.

[7] A. Lucas, “Ising formulations of many NP problems,” *Frontiers in Physics*, vol. 2, 2014, ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005.

[8] K. Blekos, D. Brand, A. Ceschini, *et al.*, *A Review on Quantum Approximate Optimization Algorithm and its Variants*, arXiv:2306.09198 [quant-ph], Jun. 2023. DOI: 10.48550/arXiv.2306.09198.

[9] M. P. Harrigan, K. J. Sung, M. Neeley, *et al.*, “Quantum approximate optimization of non-planar graph problems on a planar superconducting processor,” *Nature Physics*, vol. 17, no. 3, pp. 332–336, Feb. 2021. DOI: 10.1038/s41567-020-01105-y.

[10] M. Alam, A. Ash-Saki, and S. Ghosh, “Design-Space Exploration of Quantum Approximate Optimization Algorithm under Noise,” in *2020 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, Mar. 2020, pp. 1–4, ISBN: 9781728160313. DOI: 10.1109/CICC48029.2020.9075903.

[11] G. Pagano, A. Bapat, P. Becker, *et al.*, "Quantum approximate optimization of the long-range ising model with a trapped-ion quantum simulator," *Proceedings of the National Academy of Sciences*, vol. 117, no. 41, pp. 25 396–25 401, Oct. 2020. DOI: 10.1073/pnas.2006373117.

[12] P. C. Lotshaw, T. Nguyen, A. Santana, *et al.*, "Scaling quantum approximate optimization on near-term hardware," *Scientific Reports*, vol. 12, no. 1, Jul. 2022. DOI: 10.1038/s41598-022-14767-w.

[13] L. Zhu, H. L. Tang, G. S. Barron, *et al.*, *An adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer*, 2022. arXiv: 2005.10258 [quant-ph].

[14] D. J. Egger, J. Mareček, and S. Woerner, "Warm-starting quantum optimization," *Quantum*, vol. 5, p. 479, Jun. 2021. DOI: 10.22331/q-2021-06-17-479.

[15] S. Bravyi, A. Kliesch, R. Koenig, *et al.*, "Obstacles to variational quantum optimization from symmetry protection," *Phys. Rev. Lett.*, vol. 125, p. 260 505, 26 Dec. 2020. DOI: 10.1103/PhysRevLett.125.260505.

[16] E. Bae and S. Lee, *Recursive qaoa outperforms the original qaoa for the max-cut problem on complete graphs*, 2023. arXiv: 2211.15832 [quant-ph].

[17] A. Awasthi, F. Bär, J. Doetsch, *et al.*, *Quantum computing techniques for multi-knapsack problems*, 2023. arXiv: 2301.05750 [quant-ph].

[18] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[19] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, Jan. 2018. DOI: 10.1103/revmodphys.90.015002.

[20] J. Tilly, H. Chen, S. Cao, *et al.*, "The variational quantum eigensolver: A review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, Nov. 2022. DOI: 10.1016/j.physrep.2022.08.003.

[21] M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoretical Computer Science*, vol. 1, no. 3, pp. 237–267, Feb. 1976, ISSN: 03043975. DOI: 10.1016/0304-3975(76)90059-1.

[22] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge University Press, 1995, ISBN: 9780521474658.

[23] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, Nov. 1995, ISSN: 0004-5411. DOI: 10.1145/227683.227684.

[24] R. A. Horn and C. R. Johnson, "Matrix analysis," in Second edition. Cambridge University Press, 2017, pp. 440–441, ISBN: 9780521548236.

[25] F. A. Potra and S. J. Wright, “Interior-point methods,” *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 281–302, Dec. 2000, ISSN: 03770427. DOI: 10.1016/S0377-0427(00)00433-7.

[26] Z. Wen, D. Goldfarb, and W. Yin, “Alternating direction augmented Lagrangian methods for semidefinite programming,” *MPC (Mathematical Programming Computation)*, vol. 2, no. 3-4, pp. 203–230, Dec. 2010, ISSN: 1867-2949, 1867-2957. DOI: 10.1007/s12532-010-0017-1.

[27] R. L. Graham, “Bounds on multiprocessing timing anomalies,” *SIAM Journal on Applied Mathematics*, vol. 17, no. 2, pp. 416–429, 1969, ISSN: 00361399.

[28] L. Epstein, “List scheduling,” in *Encyclopedia of Algorithms*, M.-Y. Kao, Ed. Springer US, 2008, pp. 455–457, ISBN: 978-0-387-30162-4. DOI: 10.1007/978-0-387-30162-4_205.

[29] E. Farhi, J. Goldstone, S. Gutmann, *et al.*, *Quantum computation by adiabatic evolution*, 2000. arXiv: quant-ph/0001106 [quant-ph].

[30] W. Scherer, *Mathematik der Quanteninformatik*. Springer, 2016.

[31] J. Maziero, “Computing partial traces and reduced density matrices,” *International Journal of Modern Physics C*, vol. 28, no. 01, p. 1 750 005, Jan. 2017. DOI: 10.1142/s012918311750005x.

[32] R. Jozsa, “Fidelity for mixed quantum states,” *Journal of Modern Optics*, vol. 41, no. 12, pp. 2315–2323, 1994. DOI: 10.1080/09500349414552171.

[33] I. Bengtsson and K. Zyczkowski, “Geometry of quantum states: An introduction to quantum entanglement,” Jan. 2006. DOI: 10.1017/CBO9780511535048.

[34] M. A. Nielsen, “A simple formula for the average gate fidelity of a quantum dynamical operation,” *Physics Letters A*, vol. 303, no. 4, pp. 249–252, Oct. 2002. DOI: 10.1016/s0375-9601(02)01272-0.

[35] M. D. Bowdrey, D. K. Oi, A. J. Short, *et al.*, “Fidelity of single qubit maps,” *Physics Letters A*, vol. 294, no. 5-6, pp. 258–260, Mar. 2002. DOI: 10.1016/s0375-9601(02)00069-5.

[36] E. Magesan, “Depolarizing behavior of quantum channels in higher dimensions,” *Quantum Info. Comput.*, vol. 11, no. 5, pp. 466–484, May 2011, ISSN: 1533-7146.

[37] C. J. Wood and Qiskit development team. “Qiskit Aer 0.12.2 documentation, Source code for qiskit_aer.noise.device.models.” (2023), [Online]. Available: https://qiskit.org/ecosystem/aer/_modules/qiskit_aer/noise/device/models.html (visited on 10/02/2023).

[38] J. Preskill, *Lecture notes for ph 219 / cs 219 : Quantum information and computation chapter 3*, 2018.

[39] P. Krantz, M. Kjaergaard, F. Yan, *et al.*, “A quantum engineer’s guide to superconducting qubits,” *Applied Physics Reviews*, vol. 6, no. 2, Jun. 2019. DOI: 10.1063/1.5089550.

[40] E. N. Gilbert, “Random Graphs,” *The Annals of Mathematical Statistics*, vol. 30, no. 4, pp. 1141–1144, 1959. DOI: 10.1214/aoms/1177706098.

[41] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring network structure, dynamics, and function using networkx,” in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., 2008, pp. 11–15.

[42] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2.

[43] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[44] SciPy development team. “SciPy API reference (v1.11.3): scipy.optimize.minimize.” (2023), [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#rdd2e1855725e-11 (visited on 10/04/2023).

[45] M. J. D. Powell, “A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation,” in *Advances in Optimization and Numerical Analysis*, S. Gomez and J.-P. Hennart, Eds., Springer, 1994, pp. 51–67, ISBN: 9789048143580 9789401583305. DOI: 10.1007/978-94-015-8330-5_4.

[46] M. J. D. Powell, “Direct search algorithms for optimization calculations,” *Acta Numerica*, vol. 7, pp. 287–336, Jan. 1998, ISSN: 0962-4929, 1474-0508. DOI: 10.1017/S0962492900002841.

[47] E. G. Coffman and E. N. Gilbert, “On the Expected Relative Performance of List Scheduling,” *Operations Research*, vol. 33, no. 3, pp. 548–561, Jun. 1985, ISSN: 0030-364X, 1526-5463. DOI: 10.1287/opre.33.3.548.

[48] Committee on Technical Assessment of the Feasibility and Implications of Quantum Computing, Computer Science and Telecommunications Board, Intelligence Community Studies Board, *et al.*, *Quantum Computing: Progress and Prospects*, E. Grumbling and M. Horowitz, Eds. National Academies Press, Mar. 2019, ch. 5, ISBN: 9780309479691. DOI: 10.17226/25196.

[49] Qiskit contributors, *Qiskit: An open-source framework for quantum computing*, 2023. DOI: 10.5281/zenodo.2573505.

[50] K. Georgopoulos, C. Emary, and P. Zuliani, “Modeling and simulating the noisy behavior of near-term quantum computers,” *Physical Review A*, Dec. 2021, ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.104.062432.

[51] C. Blank, D. K. Park, J.-K. K. Rhee, *et al.*, “Quantum classifier with tailored quantum kernel,” *npj Quantum Information*, vol. 6, no. 1, p. 41, May 2020, ISSN: 2056-6387. DOI: 10.1038/s41534-020-0272-6.

[52] F. Greiwe, T. Krüger, and W. Mauerer, *Effects of imperfections on quantum algorithms: A software engineering perspective*, 2023. arXiv: 2306.02156 [cs.ET].

[53] T. Harty, D. Allcock, C. Ballance, *et al.*, "High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit," *Physical Review Letters*, vol. 113, no. 22, Nov. 2014. DOI: 10.1103/physrevlett.113.220501.

[54] Qiskit development team. "Qiskit Terra: Release 0.23.2." (2023), [Online]. Available: https://github.com/Qiskit/qiskit/releases/tag/0.23.2 (visited on 10/02/2023).

[55] C. Xue, Z.-Y. Chen, Y.-C. Wu, *et al.*, *Effects of quantum noise on quantum approximate optimization algorithm*, 2019. arXiv: 1909.02196 [quant-ph].

[56] J. Marshall, F. Wudarski, S. Hadfield, *et al.*, "Characterizing local noise in QAOA circuits," *IOP SciNotes*, vol. 1, no. 2, p. 025 208, Aug. 2020. DOI: 10.1088/2633-1357/abb0d7.

[57] H. Norlén, *Quantum Computing in Practice with Qiskit® and IBM Quantum Experience®: Practical recipes for quantum computer coding at the gate and algorithm level with Python*. Packt Publishing Ltd, 2020.

[58] C. P. Williams, *Explorations in quantum computing* (Texts in computer science), 2nd ed. Springer, 2011, ISBN: 9781846288876.

[59] D. C. McKay, C. J. Wood, S. Sheldon, *et al.*, "Efficient Z-Gates for Quantum Computing," *Physical Review A*, vol. 96, no. 2, Aug. 2017. DOI: 10.1103/physreva.96.022330.

[60] IBM Corporation. "Qiskit API Reference (v0.25.1), IBM Quantum Documentation: Circuit Library." (2023), [Online]. Available: https://docs.quantum-computing.ibm.com/api/qiskit/circuit_library (visited on 10/03/2023).

[61] M. Y. Siraichi, V. F. D. Santos, C. Collange, *et al.*, "Qubit allocation," in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, ACM, Feb. 2018, pp. 113–125, ISBN: 9781450356176. DOI: 10.1145/3168822.

[62] A. Cowtan, S. Dilkes, R. Duncan, *et al.*, "On the Qubit Routing Problem," 32 pages, 2019. DOI: 10.4230/LIPICS.TQC.2019.5.

[63] G. Li, Y. Ding, and Y. Xie, "Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices," in *Proceedings of ASPLOS XV*, ACM, Apr. 2019, pp. 1001–1014, ISBN: 9781450362405. DOI: 10.1145/3297858.3304023.

[64] S. Niu, A. Suau, G. Staffelbach, *et al.*, "A Hardware-Aware Heuristic for the Qubit Mapping Problem in the NISQ Era," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–14, 2020, ISSN: 2689-1808. DOI: 10.1109/TQE.2020.3026544.

[65] H. Safi, K. Wintersperger, and W. Mauerer, *Influence of hw-sw-co-design on quantum computing scalability*, 2023. arXiv: 2306.04246 [quant-ph].

[66] M.-D. Choi, "Completely positive linear maps on complex matrices," *Linear Algebra and its Applications*, vol. 10, no. 3, pp. 285–290, Jun. 1975, ISSN: 00243795. DOI: 10.1016/0024-3795(75)90075-0.

[67] R. Majumdar, D. Madan, D. Bhoumik, *et al.*, *Optimizing ansatz design in qaoa for max-cut*, 2021. arXiv: 2106.02812 [quant-ph].

[68] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, p. 80, Dec. 1945, ISSN: 00994987. DOI: 10.2307/3001968.

[69] A. Soifer, *The mathematical coloring book: mathematics of coloring and the colorful life of its creators*. Springer, 2009, ISBN: 9780387746425.

[70] F. B. Maciejewski, F. Baccari, Z. Zimborás, *et al.*, "Modeling and mitigation of cross-talk effects in readout noise with applications to the quantum approximate optimization algorithm," *Quantum*, vol. 5, p. 464, Jun. 2021. DOI: 10.22331/q-2021-06-01-464.

[71] A. H. Karamlou, W. A. Simon, A. Katabarwa, *et al.*, "Analyzing the performance of variational quantum factoring on a superconducting quantum processor," *npj Quantum Information*, vol. 7, no. 1, Oct. 2021. DOI: 10.1038/s41534-021-00478-z.

[72] G. Quiroz, P. Titum, P. Lotshaw, *et al.*, *Quantifying the impact of precision errors on quantum approximate optimization algorithms*, 2021. arXiv: 2109.04482 [quant-ph].