OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

LABOR FÜR
DIGITALISIERUNG

BACHELORARBEIT

Vincent Eichenseher

Comparative Analysis of Parameter Selection Heuristics for the Quantum Approximate Optimization Algorithm

March 5, 2024

Faculty: Informatik und Mathematik
Study Programme: Informatik
Supervisor: Prof. Dr. Wolfgang Mauerer
Secondary Supervisor: Prof. Dr. Christian Wolff

Abstract

The Quantum Approximate Optimization Algorithm (QAOA) is a variational quantum algorithm for producing approximations to combinatorial optimization problems. In theory the quality of this approximate solution converges to 1.0, which corresponds to an exact solution, as the circuit depth p goes towards infinity. This circuit depth corresponds directly to the number of QAOA layers, which are each parameterized by two a variational parameters α, β. In practice the quality of the approximation produced on QAOA critically depends on finding good variational parameters for the resulting Ansatz (the parameterized trial state). For this reason, a great amount of research regarding QAOA deals with approaches for finding good variational parameters. This thesis examines two such strategies, FOURIER and INTERP, given some optimal parameters at a certain depth, exploit patterns in the distribution of these parameters to heuristically determine good initial parameters for higher depths. In order to assess the feasibility of using these strategies to improve the basic QAOA, a comparison between heuristically optimized and randomly initialized QAOA was conducted: First, a detailed summary of the currently available research, which is relevant to this work, is given. Second, the heuristic parameter selection strategies were implemented and numerically simulated on random, non-regular instances of MAX-CUT, a NP-hard problem, in order benchmark their performance and compare them to the basic QAOA variant. This comparison showed that for the instances that were examined, the FOURIER strategy produced good results with a considerably greater consistency than INTERP and the basic (random initialization) QAOA method, both of which performed similarly. Moreover, the results indicate that all variants are less sensitive to changes in the problem graph order than to changes in the problem graph size, which leads to a noticeable decrease in solution quality consistency as it increases; further experiments showed that this decrease in consistency may not only be due to the graph size, but instead is affected to a greater extent by other factors, such as high node connectivity, which becomes more probable as the graph size grows. In Summary, the results substantiate the speculation that the FOURIER strategy is a feasible augmentation to the standard QAOA protocol and may be especially advantageous for realistic near term implementations. Conversely, the results do not support such a supposition for the INTERP strategy.

Contents

1. Introduction

In recent years, there has been increasing interest in Quantum Computing, likely due to advances in both quantum hardware and our understanding of the possible applications of the Noisy Intermediate Scale Quantum (NISQ) devices we have access to in the near-term. Many of the most well known quantum algorithms, which have been shown to outperform the best known classical algorithms, such as Shor's factoring algorithm [1] or Grover's search algorithm [2], not only require a considerable number of qubits to be applicable to problem instances that are actually challenging, but also a high number of gates, which will inevitably accumulate a large amount of of errors on NISQ devices, leading to a lower fidelity. We can use quantum error correction to counteract the noise (for an example see [3]; for a more general introduction to error-mitigation see Chapter 10 of [4]), however often this will require extra qubits, scaling the the required resources up to an impractical quantity. Due to this limitation of near term devices, currently the focus lies on finding algorithms which can efficiently solve useful problems on near term devices without requiring a high depth (which would necessitate extensive error correction) and/or a high number of qubits.

One of the most promising group of algorithms, which potentially fulfill these criteria, are Variational Quantum Algorithms (VQA) [5], which make use of a classical optimization routine to iteratively optimize a parameterized trial solution (also commonly referred to as Ansatz), thus utilizing the avaliable quantum resources in a systematic way which allows for the possibility of configuring the number of qubits and the circuit depth to a certain degree, which can be advantageous for computation on NISQ-devices[1]
Two of the best known VQAs are the Variational Quantum Eigensolver (VQE) [6], the first VQA that was proposed, which has been successfully applied to quantum chemistry problems, where it has an inherent advantage over classical hardware due to the complexity of the Hamiltonians involved [7], and the Quantum Approximate Optimization Algorithm (QAOA, sometimes also referred to as Quantum Alternating Operators Ansatz, see [8], [9]), which was first proposed in 2014 by Farhi et al. [10], and produces an approximate solution for combinatorial optimization problems, where both the depth of the circuit and the quality of the approximation are dependent on a parameter p. While QAOA

[1] Since we can configure the algorithm in such a manner that the depth of the resulting circuit is as shallow as possible; in this way we can reduce the number of gates, which operate imperfectly in NISQ-devices, and consequently the capacity for errors, which arise from the flawed operations these gates perform.

can be considered an extension of VQE, the key difference is that for QAOA, a *computational* eigenstate, that is, a quantum state that corresponds to a classical state and that we can physically distinguish from other states, encodes the problem solution.

QAOA is a compelling research subject, because it has been proven to not be efficiently simulatable by classical computers [11], and is an interesting algorithm for exploring the possibility of speedups using near-term quantum devices (as opposed to classical hardware): Seeing that the cost Hamiltonian[2] encodes a classical cost function, we can make a better comparison to known classical algorithms, and thus learn more about the performance of NISQ devices on problems where there is no clear quantum advantage.
For specific problem instances, non trivial performance guarantees can be made at low p (see [10], [12]–[15]), in other words for instances with a low circuit-depth. For a high circuit-depth however, the performance of the algorithm depends on choosing good parameters for the variational circuit [16]. Finding *optimal* parameters is NP-Hard [17]; due to this limiting factor, there has been a a considerable amount of research regarding strategies for finding parameters which can serve as a good starting-point for optimization of a QAOA circuit of arbitrary depth p [10], [16], [18], [19].

This thesis focuses on a comparative analysis of two such strategies, INTERP and FOURIER [18], which take a heuristic approach to selecting good initial parameters for the QAOA Ansatz. Both of these approaches utilize existing patterns in the distribution of optimal parameters, in order to make an educated guess of good initial parameters for QAOA of level p. The thesis provides a general overview of these parameter selection strategies and benchmarks their performance on the NP-hard MAXCUT optimization problem for instances of unweighted non-regular random graphs using numerical simulations with varying problem sizes and orders. A comparison to the standard QAOA variant is conducted in order to assess the feasibility of running these parameter selection strategies on near-term devices.

The information presented in this thesis is organized in the following way: First, in Chapter 2, the necessary background on quantum computation and combinatorial optimization, which is required for understanding the concepts discussed in this work, is presented (it is assumed that the reader is familiar with linear algebra, and has some basic knowledge of calculus). Next, Chapter 3 gives an overview of the related work most relevant to the purposes of this thesis. In the course of this we will cover QAOA and the parameter selection heuristics, which are the primary focus of this thesis. Following this, in Chapter 4, an outline of the numerical symulations, which were conducted in order to benchmark the heuristic strategies and the Basic QAOA variant, as well as the considerations

[2] For the purposes of this introduction the cost Hamiltonian can be considered to be an operator which describes the dynamics of a cost function, a more rigorous definition will be given in Subsection 2.1.5.

that went into the experimental setup, is put forward. Subsequently, the outcome of these experiments and the main observations we can draw from these results are shown in Chapter 5. Given the above, in Chapter 6 these results are discussed in order to make a comparison of the QAOA variants studied in this work and to evaluate how feasible heuristic optimization of QAOA might be in practice. Finally, in Chapter 7, a summary of the conclusions resulting from the research and experiments conducted for this thesis is presented; furthermore an outlook on considerations for future works concerning this topic is given in this section.

2. Background on Quantum Computing and Combinatorial Optimization

Before giving background information on QAOA and introducing the related literature most relevant to this thesis, an overview over the basic fundamentals required to understand the concepts presented in this work is presented in this chapter. If the reader is already familiar with the basics of quantum computing (introduced here in Section 2.1) and combinatorial optimization (introduced here in Section 2.2) , this chapter can be skipped or be referenced as necessary while reading the main text.

2.1. Background on Quantum Computing

This section gives an overview of the basics of quantum computing: Subsection 2.1.1 introduces qubits and the concept of quantum states and superpositions. Following this, in Subsection 2.1.2, the idea of multiple qubit systems and quantum entanglement is presented. Subsequently, Subsection 2.1.3 explains the gate based model of quantum computation and how quantum operations are defined in this model. Next, in Subsection 2.1.4 we examine how to extract information from the quantum states produced by a quantum circuit. Finally, in Subsection 2.1.5 a concise overview of how optimization problems can be represented in the context of quantum computing is given. Unless an explicit citation is given in the text, the source for the statements made throughout Section 2.1 are [4].

2.1.1. Qubits and Quantum States

Similarly to how *bits* are the fundamental components used in classical computation for storing information about the *classical state* the system is in, quantum computing uses *quantum bits*, *qubits* for short, to store information about the *quantum state* of the quantum system. For classical bits, the representation of the state is conceptually simple, as they can only take one of the values 0 or 1, and for multiple bits, the the extent of the possible states the system can be in is

limited to the 2^n possible permutations of 1 and 0 for system size n. For qubits on the other hand, the concept of the system state is less simple: While qubits can exist in the states $|0\rangle$ and $|1\rangle$, which correspond to the classical states 0 and 1, the quantum state can also take other values; a qubit can exist in a *continuum* of states between $|0\rangle$ and $|1\rangle$, until it is observed and determined to be either in the state $|0\rangle$ or the state $|1\rangle$. These two states form the orthonormal basis for the vector space, which the state of the qubit can inhabit, and are often referred to as the *computational basis states*. They are typically defined as follows[1]:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.1}$$

Quantum states that take other values can be written as a linear combination of these computational basis states, that is, as a so called *superposition* of states:

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha|0\rangle + \beta|1\rangle, \text{where } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1 \tag{2.2}$$

As we can see in Equation 2.2, the quantum state is dictated by the complex-valued coefficients α and β, which are often referred to as the *amplitudes* corresponding to the computational basis states. Furthermore, since the condition $|\alpha|^2 + |\beta|^2 = 1$ applies, the vector representing the quantum state, sometimes referred to as *statevector*, is normalized to be of unit length.
A notable property of qubits is that the value of these amplitudes cannot be accessed directly, instead we can only determine restriced information about the state through measurement: When measuring the state that a qubit is in, we will obtain the result 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$. The concept of a measurement changing the quantum state from a pre-measurement superposition of states to a specific computational basis state post-measurement is referred to as the *collapse* of the superposition[2]. More details on Measurement will be given in Subsection 2.1.4. When the probabilities of sampling each computational basis state are equal (in other words: the amplitudes corresponding to the computational basis states are the same), we call this an *equal superposition* (also sometimes called a balanced superposition):

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \ \ |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \tag{2.3}$$

Note that both of the equal superposition states given in Equation 2.3 will yield the state 0 or 1 with probability $1/2$, since $|\frac{1}{\sqrt{2}}|^2 = 1/2$ regardless of the relative

[1] The definition in Equation 2.1 follows the conventional indexing of matrix elements in linear algebra. It would be possible to set up the orthonormal basis such that the definitions of $|0\rangle$ and $|1\rangle$ are inverted, however, when doing this we would also have to redefine the quantum operators accordingly. For the purpose of this thesis, when referring to the computational basis states, the definition given in Equation 2.1 applies.

[2] More formally in Quantum Mechanics this is referred to as the *collapse of the wavefunction*.

phase of the amplitudes; we cannot physically tell these two states apart through measurement. However, we still distinguish between these two states, since operations made on a quantum state may still be affected by the relative phase, even if it has no effect on measurement.
Another way to mathematically represent the amplitudes of a quantum state is by using spherical polar coordinates (r, θ, φ) (where r is the radial distance, θ is the azimuthal angle and φ is the polar angle). Since the quantum state vector is normalized to be of unit length, $r = 1$ and we can formulate Equation 2.2 depending only on the real-valued angles (θ, φ):

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right) \text{, where } \gamma, \theta, \varphi \in \mathbb{R} \tag{2.4}$$

Note that aside from θ and φ, the parameter γ appears[3] in Equation 2.4, however since the term $e^{i\gamma}$ is a global phase factor, which has no observable effects on the quantum state, this factor can be ignored, thus making Equation 2.4 only dependent on θ and φ. The unit 2-sphere on which the points provided by the spherical polar coordinate formulation lie is called the *Bloch sphere*.

2.1.2. Multiple Qubit Systems and Entanglement

For a system with more than one qubit, we will need more amplitudes to represent the system state. This is due to the fact that each computational basis state has an amplitude, which determines the probability of sampling this basis state when measuring the system, corresponding to it; as such number of possible computational basis states increases with the system size.
For expressing quantum states of multiple-qubit systems, we use the tensor product: If V and W are vector spaces, which correspond to the sets of possible states $|v\rangle \in V, |w\rangle \in W$ of two different single qubit systems, we can combine these vector spaces to form a larger vector space, denoted $V \otimes W$ which has linear combinations $|v\rangle \otimes |w\rangle$ as its elements, and $|i\rangle \otimes |j\rangle$ as its orthonormal basis, where $|i\rangle, |j\rangle$ are the orthonormal bases of V and W respectively. The tensor product is often abbreviated as $|v\rangle \otimes |w\rangle = |v, w\rangle = |vw\rangle$.
Consider for example a system of 2 qubits:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle, \text{ where } \alpha_{ij} \in \mathbb{C} \;\; (i, j \in \{0, 1\}) \tag{2.5}$$

As can be seen in Equation 2.5, we need $2^2 = 4$ amplitudes to represent the quantum state of a two qubit system. Also note that since the normalization

[3] This is due to the fact that both of the amplitudes are complex-valued; A complex number of the form $x + iy$ (where $x, y \in \mathbb{R}$) is represented in polar form as $re^{i\varphi} = r(\cos\varphi + i\sin\varphi)$, thus we set $\alpha = r(\cos\gamma + i\sin\gamma)$ and $\beta = r(\cos\varphi + i\sin\varphi)$. Taking into consideration the normalization condition given in Equation 2.2 and the pythagorean trigonometric identity $(\sin(x)^2 + \cos(x)^2 = 1)$ into consideration, we arrive at the formula given in Equation 2.4.

condition given in Equation 2.2 applies, the squared amplitudes $|a_{ij}|^2$ must sum to 1. In order to represent a quantum state of a system with n qubits, the number of complex-valued amplitudes we need is 2^n, which, when scaling up the system size, increases considerably faster than the n real valued binary numbers we need to represent a classical state. Consequently, the amount of information associated with a quantum state is far greater than the amount of information contained in a classical state. According to Nielsen and Chuang, for a system with 500 qubits, the number of amplitudes is larger than the estimated number of atoms in the universe; saving all the complex numbers, which describe these amplitudes, on a classical computer would be infeasible. Quantum computing aims at making use of the potential computational power that comes with this high information volume, in order to possibly gain an advantage over classical computers.

Another phenomenon, that Quantum computing makes use of, is *quantum entanglement*, which is a resource unique to quantum computation that is an essential part of most quantum algorithms. When qubits are entangled with each other, their measurement outcomes are *correlated*, that is, measuring one qubit affects the measurement outcome of the other qubit(s):
Consider for example a two qubit system which is in an equal superposition of two of the four possible computational basis states:

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{2.6}$$

An interesting property of the state given in Equation 2.6 is that when measuring one qubit, we know that the other qubit must be in the same state, without actually measuring it. That is, if the first qubit is measured, there are two possible outcomes, 0 and 1, but for the second qubit there is only one possible outcome, which depends on what state we determined the first qubit to be in: since the quantum state in Equation 2.6 can only collapse into the computational basis states $|00\rangle$ and $|11\rangle$, if we measure the first qubit to be 0, the only possible outcome with the first qubit being 0 is 00, where the second qubit is also 0; analogously, if we measure the first qubit to be 1, the only possible measurement outcome is 11, where the second qubit is also 1. Thus, whatever we measure the first qubit to be will also be the outcome of the second qubit; their outcomes are correlated.
The quantum state shown in Equation 2.6 is one of the so-called *Bell states*, also known as *EPR pairs*, which are the four distinct two-qubit states that arise from entangling[4] the qubits in systems which are initially in one of the four computational basis states; The initial computational basis state corresponding the state in Equation 2.6 is $|00\rangle$.

[4]The procedure for entangeling two qubits is as follows: The first qubit is put into a superposition and then a controlled-NOT gate, with the control on the first qubit, is applied to the system. The concept of quantum gates will be elaborated on in Subsection 2.1.3.

2.1.3. Quantum Operators and Gate-based Quantum Computation

In order to manipulate the quantum information, so that we can actually perform meaningful computations, we will have to apply operations, which modify the amplitudes in coherent way, to the quantum state. A well-known model for representing these operations is the circuit model of quantum computation: Analogously to how a classical computer uses a logic circuit comprised of logic gates to perform logical operations on a classical state, a quantum computer uses *quantum circuit* composed of *quantum gates* to apply operations to a quantum state. Since we represent quantum states with vectors in a complex vector space, the most obvious way of performing a transformation on the amplitudes of the state is by applying a matrix. Thus we can represent the action of a gate U that transforms the state $|\psi\rangle$ to the state $|\phi\rangle$ as:

$$|\phi\rangle = U|\psi\rangle, \text{ where } U^\dagger U = \mathbb{1} \tag{2.7}$$

Note that for the matrix to be a valid transformation of the quantum state, a constraint ($U^\dagger U = \mathbb{1}$) applies, meaning not all matrices are valid quantum operations. Remember the normalization condition given in Equation 2.2: due to the fact that the squared amplitudes of the quantum state must sum to 1, the matrix must preserve this property when transforming the statevector $|\psi\rangle$ to $|\phi\rangle$. The appropriate condition for ensuring that only valid state transformations are made, is that matrix U has to be *unitary*; in other words U matrix-multiplied with $U^\dagger$, which is the conjugate transpose of U, must yield the identity-matrix $\mathbb{1}$.

The most commonly used single qubit gates are the so-called *Pauli gates*, which correspond to the *sigma pauli matrices*:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.8}$$

It can easily be verified that these matrices are unitary by checking if the condition given in Equation 2.7. Furthermore, these operators are not only unitary, but also *Hermitian*, meaning that they are equal to their conjugate transpose: $U = U^\dagger$ (and so $U^\dagger U = U^2 = \mathbb{1}$). As for the specific operations performed by these gates: The X gate acts in a analogous way to how a classical NOT gate performs a negation of classical bits, by taking the state $|0\rangle$ to $|1\rangle$ and vice versa. The Z gate gate adds a relative phase of π to the amplitude corresponding to the $|1\rangle$ basis state, in this way "flipping" the relative phase by transforming $|1\rangle$ to $-|1\rangle$ and leaving the amplitude of $|0\rangle$ unchanged. Finally, the Y gate both performs a bit flip and a phase flip on the state.

The matrix-exponentiation of these Pauli matrices induces a family of unitary single qubit gates that can be used to rotate the statevector around the x, y or z

axes (depending on which Pauli matrix was exponentiated) of the Bloch sphere representation. These *rotational gates* are defined as

$$\begin{aligned} R_x(\theta) = e^{-i\theta X/2} &= \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \\ R_y(\theta) = e^{-i\theta Y/2} &= \begin{bmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \\ R_z(\theta) = e^{-i\theta Z/2} &= \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}, \end{aligned} \tag{2.9}$$

where the parameter θ dictates what the angle, by which we rotate the statevector around an axis, should be. Due to the fact that these gates are parameterized, the rotational gates are especially useful for creating a parameterized quantum trial state (also known as Ansatz); these trial states are used by VQEs and QAOA, the latter of which will be explained in detail in Section 3.1, in order to solve combinatorial optimization problems, by optimizing θ such that a measurement of the circuit gives a optimal solution (or for QAOA an approximation of this solution).

Another important single qubit gate is the *Hadamard* gate, which is also unitary and Hermitian, defined as follows:

$$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.10}$$

This gate is often used for creating superpositions, as it transforms the $|0\rangle$ state to the $|+\rangle$ state and the the $|1\rangle$ state to the $|-\rangle$ state (for the definition of these states see Equation 2.3).

One of the most important two-qubit gates is the *controlled-NOT* or *CNOT* gate, which, among other things, can be used to entangle two qubits, and is given by the following matrix:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.11}$$

Since the first two columns of this matrix determine how the states $|00\rangle$ and $|01\rangle$ are transformed, we can see that if the first qubit is in the state $|0\rangle$, the identity matrix gets applied (in other words we do not change the state). The last two columns correspond to the transformations of the states $|10\rangle$ and $|11\rangle$, where first qubit is $|1\rangle$: if this is the case, a bit-flip is applied to the second bit. We call the fist qubit the *control qubit*, since its value determines whether or not the second qubit, which we refer to as the *target qubit*, will be negated. In general,

we can turn any unitary single qubit gate into a multiple qubit controlled gate by using control qubits to only apply this single-qubit gate when the control qubits are in the state $|1\rangle$.
The CNOT gate together with the single qubit gates is universal[5] for quantum computation.

2.1.4. Expectation Value and Measurement of Quantum States

As we already saw in Subsection 2.1.1, the probability of the measurement outcome resulting in a specific computational basis state[6] is given by the squared amplitude corresponding to that basis state. So we can think about measurement as sampling from the probability distribution given by all amplitudes of the quantum state we are measuring, returning a single sample from this distribution each time we measure.
There are different ways of describing a measurement, however we will only focus on the most well-known, measurements made by projecting the quantum state onto the basis vectors $|0\rangle, |1\rangle$. This so-called *projective measurement* is characterized by an *observable* M, which is a Hermitian operator that acts on the vectorspace of the system that is being measured and has the spectral decomposition

$$M = \sum_m m P_m, \tag{2.12}$$

where P_m is the orthogonal projecton-matrix (a projection matrix is a matrix, which fulfills the condition that $P^2 = P = P^\dagger$) onto the eigenspace of M with eigenvalues m. When applying this projective measurement to a normalized quantum state ψ, the probability of the eigenvalue m being sampled, that is, $p(m)$, is given by

$$p(m) = \langle\psi|P_m|\psi\rangle \tag{2.13}$$

and the quantum state after measurement is

$$\frac{P_m|\psi\rangle}{\sqrt{p(m)}}, \tag{2.14}$$

where the term $\langle\psi|$ in equation Equation 2.13 is the conjugate transpose of $|\psi\rangle$.

[5] Meaning that with this set of gates, we can construct any possible quantum circuit.
[6] It is assumed that the measurement is performed with regard to the computational basis, we can also measure in other bases like $|+\rangle, |-\rangle$, however for the purpose of this explanation we will only consider Z-measurement (measurement in the computational basis).

A useful property of projective measurements is that calculating average values of projective measurements is relatively straightforward; The resulting *expectation value*, which we refer to as $\langle M \rangle$, is described by:

$$\langle M \rangle = \langle \psi | M | \psi \rangle, \tag{2.15}$$

which follows trivially from the definition of the average value (the mean) and the spectral decomposition of M given in Equation 2.12.
Note that since we only sample one value from the distribution each time we measure, we will have to run the quantum circuit that produces the state $|\psi\rangle$ multiple times in order to be able to estimate $\langle M \rangle$.

2.1.5. Problem encoding and the Ising Model

In Subsection 2.1.1 and Subsection 2.1.2 we looked at how we can represent a quantum state and in Subsection 2.1.3 explained how to apply transformations to these states. When performing such operations on an arbitrary initial state, it would be convenient to describe the way in which the system changes, that is, how the system evolves under these operations. In other words, given an initial state, what is the state of the system at some other time/position? (more generally: at some other stage in the evolution of the system). We can describe the relationship between the rate of change of a system and its state at a certain time using differential equations. The reader may already be familiar with the notion of describing system dynamics with differential equations from classical mechanics, where we can describe the evolution of physical systems in this way. Analogously, in quantum mechanics, the evolution of a closed quantum system is described by the *Schrödinger equation*,

$$i\hbar \frac{d|\psi\rangle}{dt} = H|\psi\rangle, \tag{2.16}$$

where $\hbar$ is the *Plank's constant*, a physical constant that has to be experimentally determined, and H is the $Hamiltonian$, a fixed hermitian operator that describes the energy dynamics of the system, meaning $H|\psi\rangle$ is the energy of the system in the state $|\psi\rangle$. if H and $\hbar$ is known to us, in principal we understand the system dynamics completely, and given some initial state we can predict the state of the system at an arbitrary time t.
Due to the fact the Hamiltonian is Hermitian, its spectral decomposition is

$$H = \sum_{E} E|E\rangle\langle E|, \tag{2.17}$$

where the normalized eigenvectors $|E\rangle$ are the *energy eigenstates* of the Hamiltonian and their associated eigenvalues E correspond to the *energy value* of

these eigenstates. We call the lowest energy value the *ground state energy* and the corresponding energy eigenstate the *ground state*. many quantum algorithms designed for solving optimization problems, such VQE and QAOA, aim at finding this ground state of a given Hamiltonian. The reasoning behind this is, that since the ground state is by definition the eigenstate with the lowest energy, finding this state is equivalent to finding the state that minimizes the expectation value $\langle H \rangle$. We can use this property for solving optimization problems: consider an optimization problem where we want to find the optimal input $x \in X$ (where X is any set of viable inputs for the optimization problem; in other words, X is the domain of feasible points), which yields the lowest value for some cost function c. Then we can construct a Hamiltonian where the energy eigenstates correspond to to the possible inputs x and where the energy values are associated with the cost $c(x)$ for this input. In other words, following Equation 2.17, the spectral decomposition for this so called *cost Hamiltonian* is

$$H = \sum_{x \in X} c(x)|x\rangle\langle x| \tag{2.18}$$

In this way, the ground state of this Hamiltonian corresponds to the input for which the cost-function is minimal, and thus the solution to the optimization problem. If $c(x)$ is a classical cost function, which takes binary strings $x \in \{0,1\}^n$ (where n is the string length) as input, The cost Hamiltonian acts diagonally in the computational basis, since $H|x\rangle = c(x)|x\rangle$ for each x (see [9]). Thus, the eigenbasis of H is the computational basis, and one of the computational eigenstates corresponds to each of the energy eigenstates of the Hamiltonian.
However, as we can see from Equation 2.18, the Hamiltonian will be a diagonal matrix with $|X|^n$ by $|X|^n$ elements; in the case that the cost function takes binary strings as inputs this would produce a 2^n by 2^n matrix. As such, taking a naive approach to constructing a Hamiltonian, which corresponds to an optimization problem (as in: the ground state of the constructed Hamiltonian encodes the solution to the optimization problem), is a hard task that scales exponentially with the problem size. A more systematic way of encoding optimization problems is using efficient constructions of the cost Hamiltonian, which exist for many problem classes. One such efficiently constructed Hamiltonian is the *Ising model*, which is used to solve optimization problems in statistical physics, and can be used to encode NP-hard optimization problems and NP-complete decision problems with a polynomial number of spins [20]. The definition of the classical Ising model given in [20] is

$$H(s_1, \cdots, s_n) = -\sum_{i<j} J_{ij} s_i s_j - \sum_{i=1}^{n} h_i s_i, \tag{2.19}$$

where the Hamiltonian is formulated as a quadratic function of a set of N spins $s_i = \pm 1$, which are enumerated by the indices i, j; the constants J_{ij} and h_i are

real numbers specific to the problem instance[7].
In order to adapt this model to encode a problem on a quantum computer, we simply replace the spin variables in Equation 2.19 with pauli Z gates:

$$H(Z_1, \cdots, Z_n) = -\sum_{i<j} J_{ij} Z_i Z_j - \sum_{i=1}^{N} h_i Z_i, \tag{2.20}$$

where the subscript i of a pauli operator Z_i indicates the index of the qubit that this gate is acting on.

Now that we have a method for mapping classical optimization problems to a Hamiltonian (more specifically the Ising model) in the next section we will give some background on combinatorial optimization and look at how to formulate such classical optimization problems.

2.2. Background on Combinatorial Optimization

In this section a short overview of combinatorial optimization, with a focus on the concepts relevant to this thesis, is presented: Subsection 2.2.1 provides a general introduction to combinatorial optimization and NP-hard problems. Subsequently, Subsection 2.2.2 explains how to formulate quadratic binary unconstrained optimization problems, and their relation to the Ising model. Finally, due to the fact that graphs are a fundamental combinatorial structure that we can use to define and visualize many combinatorial optimization problems, in Subsection 2.2.3 the basics of graph theory are covered. The main source for the information presented in this section is [21], with the exception of Subsection 2.2.3, for which the main source used is [22]. Statements that do not reference either of these works will be accompanied by an explicit citation.

2.2.1. Combinatorial Optimization and NP-Hard Problems

As previously stated in Subsection 2.1.5, the goal of optimization is to find the optimal input $x \in X$ which yields the optimal value for some cost function c. More formally, we can say an *optimization problem* is a set of *optimization problem instances* which can be defined as a pair (X, c), where X is the domain of feasible inputs which the cost function c takes, mapping

$$c : x \to \mathbb{R}^1 \tag{2.21}$$

[7] We can interpret these terms as follows: J_{ij} is a weight describing the how correlated s_i and s_j are (i.e describing if $s_i = s_j$ or $s_i \neq s_j$). The weight h_i determines if s_i is $+1$ or -1.

The optimization problem associated with these optimization problem instances is finding an $x_{opt} \in X$ for which

$$c(x_{opt}) \geq c(x) \text{ for all } x \in X \tag{2.22}$$

Such a point x_{opt} is called *globally optimal*.
On the other hand, if we find a point $x_{loc} \in X$, which fulfills the condition in Equation 2.22 only for $x \in N(x_{loc})$, where $N(x_{loc})$ is the *neighborhood* of x_{loc}, which contains points x which are *close* to x_{loc} in some sense[8], then we call x_{loc} a *local optimum*. A metric for describing how close an arbitrary point x is to the global optimum, is the *approximation ratio*, which is defined as ratio of the cost for x and the cost for x_{opt}:

$$\frac{c(x)}{c(x_{opt})} \tag{2.23}$$

We can divide optimization problems into two categories: *continuous optimization*, where the variables x are continuous, and *combinatorial optimization*, where the variables are discrete. In continuous optimization, the solution we are looking for is a set of real numbers or a function, while in combinatorial optimization we are looking for an element (e.g an integer, set, permutation or graph) from a finite, or countably infinite[9], set.
For many combinatorial optimization problems a *polynomial-time algorithm* (that is, an algorithm where the number of operations needed to grows polynomially with the size of the input) exists for solving these problems on a *deterministic Turing machine*, and we call the complexity class, which contains these problems, **P**. However, there are some problems for which no polynomial-time algorithms are known. This gives rise to the complexity class **NP**[10], where instead of requiring that a algorithm running on a deterministic Turing machine can *solve* a given problem instance in polynomial time, we require that the algorithm can *verify* the answer to the *decision problem*, given a candidate solution for this problem instance, in polynomial time. If we want to solve such a problem in polynomial time, we would have to use a *non-deterministic Turing machine* instead. In other words, a decision problem is a problem that asks whether a input x fulfills some condition[11], yielding either "yes" or "no" as an answer; given an

[8] More accurately: a point is close to x_{loc} if it is within some fixed euclidean distance of x_{loc}. How we set this distance depends on the structure of X.

[9] That is: a set which is not finite, but denumerable (the elements in the set have a one-to-one correspondence to the set of natural numbers), in other words we can give instructions for identifying any element in the set by specifying an integer that corresponds to it.

[10] NP stands for *nondeterministic polynomial*.

[11] For example, this condition could be "given an input x and a problem instance (X, c), $c(x)$ is greater than or equal to $c(y)$ where $y \in X$" or, in other words, "given a candidate solution for a problem instance this solution is globally optimal". Note that the condition does not necessarily need to be global optimality, any condition that can be answered with either "yes" or "no" is sufficient.

optimization problem, we can formulate a closely related decision problem that is no harder than the original optimization problem.
If a decision problem can be *reduced* to any other decision problem in **NP** (i.e. we can use a polynomial number of calls to this problem to simulate any other decision problem in NP) we call this problem ***NP**-complete*; analogously we call an optimization problem ***NP**-hard* if there is a polynomial time reduction from any optimization problem in NP to this problem; in other words, solving an NP-hard problem is at least as hard as solving any other problem in NP. Despite numerous efforts, no polynomial time algorithms for solving NP-complete or NP-hard problems are known. Algorithms which exactly solve these problems require an *exponential* amount of operations on a deterministic Turing machine in the worst case, making these infeasible for anything but small problem instances. Due to this fact, many existing algorithms for solving these problems settle for producing an approximation instead of an exact solution, or only necessitate that the time requirements are polynomial for *most* problem instances (or problem instances which underlie certain constraints).

2.2.2. QUBO formulation

One way of formulating NP-hard problems is as a Quadratic Unconstrained Binary Optimization (QUBO) model, which is a mathematical formulation for representing combinatorial optimization problems. According to [23], the QUBO model first emerged from research in the fields of quantum annealing and digital annealing, and has also become a subject of study in neuromorphic computing. As given in [23], the QUBO model is expressed as the optimization problem

$$y = \arg\min_x x^T Q x, \tag{2.24}$$

where x is a vector of binary decision variables and Q is a square matrix, where each element q_{ij} is a real valued constant; x^T is the transpose of x. Note that while the optimization problem in Equation 2.24 is a minimization problem, we could also maximize y instead (depending on what we want the optimization to achieve).

Consider for instance an optimization problem where the cost function is

$$c(x_1, x_2) = -ax_1 - bx_2 + cx_1x_2, \tag{2.25}$$

where the variables x_i are binary variables and a, b and c are real numbers. This cost function is a quadratic function with a linear part ($-ax_1 - bx_2$) and a quadratic part (cx_1x_2). Since binary variables satisfy the condition $x_i = x_i^2 = x_ix_i$, we can write the linear part as $-ax_1x_1 - bx_2x_2$, and thus the real factors a, b give the diagonal elements in the matrix Q (since the diagonal entries are of the form

q_{ii} which corresponds to $x_i x_i$ in the cost function). As described in [23], we can then represent the optimization problem as

$$\arg\min y = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} -a & c/2 \\ c/2 & -b \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{2.26}$$

using the quadratic part of the cost function given in Equation 2.25 to obtain the non-diagonal elements. This formulation is equivalent to the notation used in Equation 2.24. Other than the variables being binary variables, there are no constraints affecting the QUBO model, since all problem data is contained in the Q matrix. For this reason, the QUBO model is a useful framework for modeling combinatorial optimization problems.

Furthermore, as explained in [20], the QUBO model is equivalent to the Ising model (see Subsection 2.1.5), since the spin variables used there can be interpreted as pseudo boolean variables, which allows for a straightforward transformation between the two models (by replacing the spin variables with binary variables or vice versa). In this way we can construct an Ising Hamiltonian from any QUBO formulation, allowing us to solve combinatorial optimization problems on a quantum computer.

2.2.3. Graph theory fundamentals

We define a *graph* as a pair of sets $G = (V, E)$, where V is a finite set of *vertices* or *nodes* (these terms are often used interchangably), and E is a set of $edges$, where the elements of E are subsets of V. We call the cardinality of V (in other words, the number of nodes) the *order* of the graph, and the cardinality of E (in other words, the number of edges) the *size* of the graph. An edge e that connects two vertices v_1 and v_2 is referred to as being *incident* to these vertices, and we say that v_1 and v_2 are *adjacent* to each other. The number of edges that are incident to a vertex (in other words, the number of edges that contain this vertex) are what we refer to as the *degree* or *valency* of the vertex. The *maximum degree* of a graph is the maximum value in the degrees of its vertices (for the purposes of this work, we will simply refer to the maximum degree as the "degree of the graph"). A *walk* on a graph $G(V, E)$ is a sequence of edges $e \in E$ that connect a sequence of vertices $v \in V$. If all edges e are distinct, we call this walk a *trail*. If all nodes in a walk are distinct (and thus all edges are also distinct) we refer to this as a *path*. A *cycle*[12] in a graph is a non-trivial (non-empty) trail where only the first and last vertices are equal (non-distinct). If the number of vertices in the cycle is even, we call this an *even cycle*, otherwise we denote this an *odd cycle*. A graph is *connected* if there is a path between any two nodes in V.

[12] Cycles are also sometimes referred to as "simple circuits" (a circuit in the context of graph theory is a walk where the first and last nodes are equal), however for the purpose of this work they will be referred to as cycles, as not to confuse these with circuits in the context of computation.

We call a graph $S = (V_s, E_s)$ a *subgraph* of the graph $G = (V, E)$, if its vertex set V_s is a subset of V, and its edge set E_s is a subset of E; more formally, if $V_s \subseteq V$ and $E_s \subseteq E$. For a visual representation of this concept, see Figure 1.

Figure 1.: A visualization of two graphs. In this example the graph the left can be formalized as $G(V, E)$, where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (1, 4), (2, 3), (3, 4)\}$. The graph on the right can be notated as $G'(V', E')$, where $V' = \{1, 2, 3\}$ and $E' = \{(1, 2), (1, 4)\}$. Note that since G' only contains vertices and edges which are also present in G, we can say the V' is a subset of V and E' is a subset of E. Thus G' is a subgraph of G

In graph theory, we distinguish between different types of graphs; In the remainder of this section we will define some specific graph types which are referred to in the main text of this work:

***d*-regular graphs** A *regular graph* is a graph where each vertex has the same degree. Often we further specify this shared degree by referring to regular graphs as *d-regular graphs*, where d is the degree (e.g. if every vertex of a given graph has three edges incident to it, we call this graph a *3-regular graph*).

Complete graphs A *complete graph* is a graph is a regular graph with degree $n - 1$, where n is the order of the graph. In other words, a complete graph is a graph were every vertex is connected to every other vertex, that is, it is *maximally connected*.

Bipartite graphs A bipartite graph is a graph where the vertex set V can be divided into two disjoint sets A, B such that every edge *must* connect a vertex in A to a vertex in B. This condition holds if (and only if) the graph has no odd cycles. Often bipartite graphs are notated as a tuple of the disjoint sets and the edge set, $G = (A, B, E)$, instead of the standard graph notation $G = (V, E)$.

***k*-tree graphs** a *k-tree graph* is induced by repeatedly adding vertices to a complete graph of order $k+1$, such that each added vertex has exactly k neighbors (neighbors in this context are vertices that are adjacent to one another, that is, they are directly connected by an edge), and each added vertex, together with its k neighbors, forms a *clique* (a clique is a subset of vertices where every vertex is adjacent to every other vertex in this subset).

3. Background on the Quantum Approximate Optimization Algorithm and Related Work

This chapter aims at giving an overview of the concepts relevant to this thesis and the related literature that provides the basis for this work.
In Section 3.1 a high level overview over the basic QAOA is presented, consisting of a short explanation of the procedure, which the algorithm follows, in Subsection 3.1.1 and a short outline of research regarding known performance guarantees and limitations of QAOA under specific conditions (or for certain problem instances) is put forth in Subsection 3.1.2. The paper by Zhou et al. [18], which serves as the cornerstone of this work, is examined in Section 3.2, where Subsection 3.2.1 gives a summary of the INTERP-Strategy while Subsection 3.2.2 covers of the FOURIER-Strategy and its variants. Finally, Section 3.3 gives a concise recapitulation of research concerning the extent to which optimal QAOA parameters can be transferred between problem instances, and what conditions apply to those findings.

3.1. QAOA

The Quantum Approximate Optimization Algorithm was proposed by Farhi, Goldstone and Gutmann in 2014 [10]. The motivation for this algorithm was finding a good approximate solution to an optimization problem, as an alternative to the Quantum Adiabatic Algorithm (QAA) [24], which is designed to determine an optimal solution given enough time, by interpolating between an initial Hamiltonian, which is trivial to construct, and a final Hamiltonian, where the ground state corresponds the solution of the problem.
QAOA obtains a Trotterized[1] approximation of this adiabatic evolution by alternating between the unitary operators $U_c(\gamma)$ and $U_c(\beta)$, where the sum of all angles γ, β is equivalent to the total time evolution of the adiabatic algorithm [24]. In this way QAOA allows for quicker evolution to the target states of otherwise

[1] A Trotterized approximation is an approximation using the Suzuki-Trotter decomposition to approximate the time evolution of the Hamiltonian. This decomposition for arbitrary operators A, B with a commutation-relation $[A, B] \neq 0$ is defined in [25] as: $e^{xA}e^{xB} = e^{x(A+B)+O(x^2)}$, where x is a parameter and O is a correction-term of second order of x (as shown in [25], this can be gereralized to higher order correction terms).

slow adiabatic dynamics. For the purpose of obtaining a good approximation, γ and β must be small, and since we want to the sum of these to equate to a long run QAA time (so as to assure success), this necessitates p, the number of alternations, to be large. As p approaches infinity, QAOA produces an exact solution, since this corresponds to approximating QAA with unlimited run time. Simply put, we can find an approximation arbitrarily close to the optimal solution by choosing p and angles γ, β accordingly, and the approximation improves as p increases.

In 2017, the original QAOA algorithm was generalized through consideration of general parameterized families of unitaries (rather than only MAXCUT-specific Hamiltonians) and applied to diverse set of optimization problems by Hadfield et al. [8], who also coined the term "Quantum Alternating Operators Ansatz" as an alternative interpretation of the QAOA acronym (in order to avoid confusion in contexts other than approximate optimization).
Later in 2021, Hadfield et al. [9] studied the behaviour of QAOA not only for $p = 1$, but also for general depth p, by formulating QAOA circuits according to the Heisenberg picture, where quantum circuits can be seen as acting on quantum observables by conjugation (rather than acting on a quantum state by matrix multiplication). The results they show in this paper will be further elaborated on in Subsection 3.1.2.

3.1.1. Algorithm

QAOA produces an approximation of the solution for combinatorial optimization problems by first creating an initial state $|s\rangle$, which is the superposition of all feasible states, and then applying p layers of alternating unitary operators to this state: Each layer first applies a parameterized unitary phase-separation operator $H_C(\gamma)$, which is diagonal in the computational basis and depends on the problem's cost-/objective-function $C(x)$, before applying a parameterized unitary probability amplitude mixing-operator $U_M(\beta)$, which depends on the initial state [2] and does not commute with the with the phase-separation operator.
As a result of the parameterization of the alternating operators, this creates the parameterized quantum state $|\gamma, \beta\rangle$ which depends on the $2p$ real parameters $\gamma = \gamma_1, \gamma_2, \cdots, \gamma_p$ and $\beta = \beta_1, \beta_2, \cdots, \beta_p$:

$$|\gamma, \beta\rangle = U_M(\beta_p)U_C(\gamma_p)\cdots U_M(\beta_1)U_C(\gamma_1)|s\rangle = e^{-i\beta_p H_M}e^{-i\beta_p H_C}\cdots e^{-i\beta_1 H_M}e^{-i\beta_1 H_C}|s\rangle \tag{3.1}$$

[2] Mixing-operators are in general required to 1) preserve the feasible subspace (i.e. for all β, the unitary U_M takes feasible states to feasible states), and 2) provide transitions between all pairs of states corresponding to feasible points/solutions (i.e. for two states a and b that correspond to feasible bitstrings in the computational basis state, there is a β that takes a to b). Due to these requirements, the mixing operator depends on the initial state $|s\rangle$ and there are multiple different mixers we could choose for each initial state. For more information, see Chapter 3 of [8].

Subsequently, we assign parameters γ and β randomly[3] and sample the quantum state by repeatedly running the circuit and making measurements in the computational basis, resulting in $F_p(\gamma, \beta)$, the expectation value of the cost-function C in the state produced by the parameterized quantum state given in Equation 3.1:

$$F_p(\gamma, \beta) = \langle \gamma, \beta | C | \gamma, \beta \rangle, \tag{3.2}$$

where C is the cost-function defined as the number of satisfied clauses (that is, boolean expressions).

The result of this measurement is evaluated by a (classical) optimization routine, which determines new parameters γ^* and β^*, in order to maximize[4] F_p for the next pass though the circuit:

$$(\gamma^*, \beta^*) = \arg\min_{\gamma,\beta} F_p(\gamma, \beta) \tag{3.3}$$

This process is then repeated. A sufficient number of repetitions will yield a good approximation of the solution to the optimization-problem; the quality of this approximation improves as the circuit depth p increases. This general protocol is illustrated in Figure 2.

In the original formulation of the QAOA, the initial state is $|s\rangle$, the uniform superposition of all 2^n computational basis states (where n is the problem size):

$$|s\rangle = |+\rangle^{\otimes n} \tag{3.4}$$

The unitary phase-separation operator U_C is defined by the Cost-Hamiltonian H_C (more generally referred to as the phase-Hamiltonian, e.g in [8], [9]), which encodes the cost-function C and acts diagonally in the computational basis. For the original algorithm this unitary operation is defined as:

$$H_C = e^{-i\gamma C} = \frac{m}{2} I - \frac{1}{2} \sum_{(i,j)\in E}^{m} Z_i Z_j \tag{3.5}$$

for the MAXCUT-problem on a graph $G = (V, E)$ of order n and size m (i.e. $|V| = n$ nodes and $|E| = m$ edges).

The unitary mixing operator U_M is defined as the transverse-field Hamiltonian, which is the sum of single bit Pauli X operators applied to every qubit in the system:

$$H_M = \sum_{i=1}^{n} X_i \tag{3.6}$$

[3] This can be done randomly or based on an educated guess/known good parameters, however for the purpose of this thesis, random initialization will be assumed as the default.

[4] Instead of maximizing F_p, we can also minimize $-F_p$, and this is frequently done in practice, since most numerical optimizers have minimization as their default behaviour.

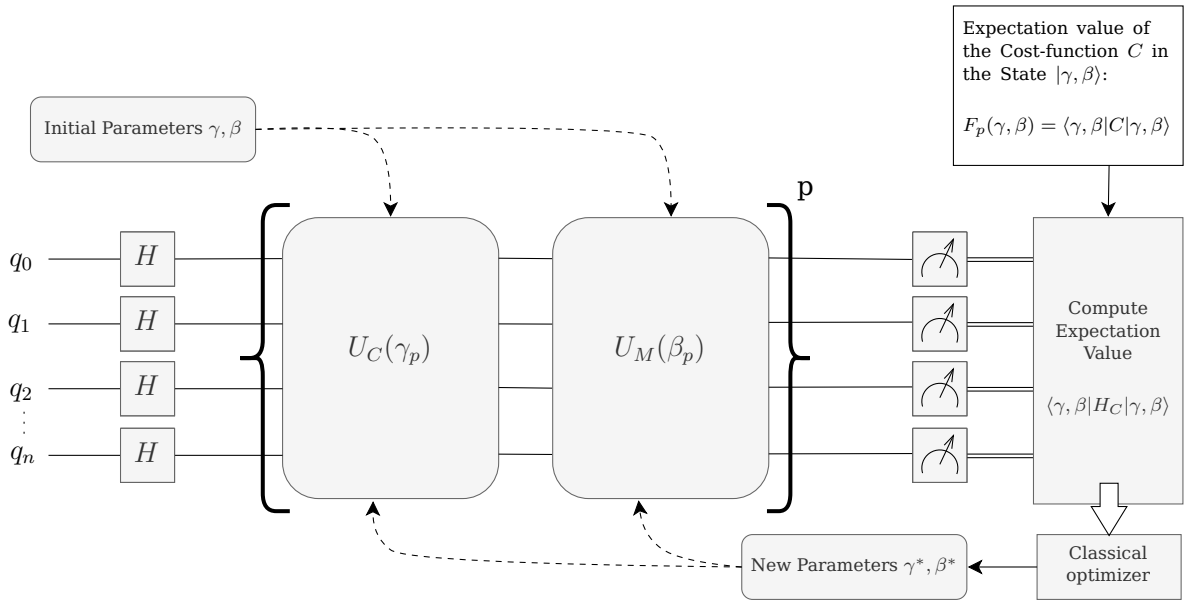


Figure 2.: Basic QAOA Circuit. Note that here the initial state $|s\rangle$ is the superposition of all possible states (see Equation 3.4), as is the case for the original QAOA method [10] as well as multiple other QAOA variants [8]. Due to this circumstance, mixing operator U_M would likely be set to be the transverse field Hamiltonian (see Equation 3.6) for this circuit, assuming measurement is performed in the Z-basis.

This mixing Hamiltonian is not only used for the original algorithm proposed by Farhi et al., but also most other versions of QAOA that have the equal superposition of all possible states as initial state [8].

An important observation here is as follows:

By inserting Equation 3.1 into Equation 3.2 and noting that the the total cost C equals the sum of the cost of each edge C_{ij}, over m where $(i,j) \in E$ (see Equation 3.5), the expectation $F_p(\gamma,\beta)$ can be formulated as

$$F_p(\gamma,\beta) = \sum_{jk}^{m} \langle s|U_C^\dagger(\gamma_1)U_M^\dagger(\beta_1)\cdots U_C^\dagger(\gamma_p)U_M^\dagger(\beta_p)|C_{ij}| \\ U_M(\beta_p)U_C(\gamma_p)\cdots U_M(\beta_1)U_C(\gamma_1)|s\rangle, \tag{3.7}$$

where $C_{ij} = \frac{1}{2}(-Z_i, Z_j + 1)$.

Since the initial state $|s\rangle$ is the product of Pauli X eigenstates ($|+\rangle^{\otimes n}$), we can conclude that each term in the sum in Equation 3.7 depends *exclusively* on the subgraph containing qubit i and j as well as adjacent qubits no further than p away from i or j. Therefore each edge (i,j) is related to the subgraph $g(i,j)$ and contributes

$$f_g(\gamma,\beta) = \langle s, g(i,j)|U^\dagger_{C_g(i,j)}(\gamma_1)\cdots U^\dagger_{M_g(i,j)}(\beta_p)|C_{ij}|U_{M_g(i,j)}(\beta_p)\cdots U_{C_g(i,j)}(\gamma_1)|s, g(i,j)\rangle \tag{3.8}$$

to the total expectation $F_p(\gamma,\beta)$. If two edges induce isomorphic[5] subgraphs, then the functions of (γ,β) that relate to these contributions are the same, and thus we can represent Equation 3.7 as a weighted sum of subgraphs:

$$F_p(\gamma,\beta) = \sum_g w_g f_g(\gamma,\beta), \tag{3.9}$$

where w_g is the number of occurences of a subgraph of type g in Equation 3.7.

Since there are finitely many subgraphs for each p, and the only dependence on n and m arises through the weights w_g. Farhi et al. come to the conclusion that for a fixed p, there exists an efficient classical algorithm capable of evaluating Equation 3.9 [10]. However, as Farhi et. al concede in the conclusion of their paper, this "efficient" algorithm could require space exponential in p. For this reason, the use of an optimization routine which makes repeated calls to the parameterized quantum circuit (in order to evaluate $F_p(\gamma,\beta)$ and update the parameters accordingly) has proven itself to be the more practical option[6].

Furthermore, the formulation of the expectation value in Equation 3.9 as a weighted sum of the expectation value of unique subgraphs is the basis for some of the arguments made in the literature regarding known performance guarantees and limitations of QAOA (see Subsection 3.1.2).

3.1.2. Known Performance Guarantees and Limitations

Few performance guarantees have been made for QAOA, and most of them only concern QAOA at a low depth and/or are restricted to certain groups of problem instances. This is likely due to the fact that if we want to make analytical statements about the performance of the algorithm, we need to be able to understand what outcomes that are possible and be able to identify cases that yield the worst/best possible performance; this becomes exceedingly difficult at high p. This section aims at giving an overview of some of the performance guarantees that were presented so far, with a focus on QAOA for the MAXCUT problem.

The earliest lower bounds for QAOA were put forward by Farhi et al. in the original QAOA paper [10]: The first class of problems they considered was MAXCUT

[5] Two graphs are isomorphic if they have the same structure, that is, they are equivalent.

[6] Another option proposed by Farhi et al. is to try values in a grid on the compact set $[0,\pi]^p \times [0,\pi]^p$, for which the number of points is only polynomial in n and m. This is only feasible if p does not grow with n, and as we will see in Subsection 3.1.2, in many cases we will have to increase p with greater n in order to exceed a certain approximation ratio.

on connected, 2-regular graphs (i.e. cyclic graphs, sometimes referred to as rings or), due to the fact that the simplicity of these problem instances facilitates a straightforward analysis. They found that for every p, the only possible subgraph for these instances is a line graph of length $2p + 2$ with the weight in the sum of subgraphs (i.e. the number of occurrences of this subgraph in the expectation given in Equation 3.8) being n, the number of vertices. Maximizing the function in Equation 3.9, it was deduced that the QAOA applied to these graphs always finds a cut of $n(2p + 1)/(2p + 2) - 1$ or higher for any depth p. They applied similar reasoning to the case of connected 3-regular graphs: For $p = 1$ the maximum of Equation 3.9 was formulated as a function dependent on the number of vertices, isolated triangles[7] and crossed squares[8]. evaluating this function numerically, it was found that the minimum value (i.e the worst case approximation) is 0.6924 and occurs when there are no isolated triangles or crossed squares present. A similar approach[9] was taken to analyze the performance at depth $p = 2$, which showed that the worst case approximation ratio in this case is 0.7559 in the limit of large n.

A comparable approach was taken by Wurtz & Love [14] in 2021, who also analyzed the expectation value of p-level QAOA for MAXCUT on 3-regular graphs formulated as a weighted sum of the expectation value of subgraphs. The difference here is that they not only considered finding the worst possible combinations of subgraphs, but also conjectured that classes of graphs with cycles of a length less than $2p+2$ are the worst case problem instances. Using this so-called large loop conjecture, the authors were able to analytically validate the lower bound of 0.7559 for $p = 2$ that was given by Farhi et al. and determined that the worst approximation ratio for $p = 3$ is 0.7924, and this worst case occurs when the graph is a 3-tree with no loops less than length 8.
Wurtz & Love also conjecture that when fixing the parameters to be optimal for the worst case instances (instead of optimizing them each time), these lower bounds will also hold for any other 3-regular graph instance. In the same year, Wurtz & Lykov [15] provided numerical evidence for this (fixed parameter) conjecture for $p < 12$. They also were able to show that under these assumptions (i.e. if QAOA really performs at least as good as the worst case lower bound for every 3-regular graph), QAOA outperforms the Goemans-Williamson (GW) algorithm [26], which is the best currently know classical algorithm for MAXCUT, for $p \geq 11$.

An analysis of QAOA for the Maximum Independent Set (MIS) problem[10], conducted by Farhi et al. [13] in 2020 and using the Overlap Gap Property[11] as a

[7] Subgraphs of the form $G = (V, E)$ where $V = \{1, 2, 3\}$ and $E = \{(1, 2), (1, 3), (2, 3)\}$.
[8] Subgraphs of the form $G = (V, E)$ where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4)\}$.
[9] The main difference here is that at $p = 2$ the subgraphs in the sum in Equation 3.9 can also contain pentagons; the minimum expectation value here occurs when the graph is bipartite.
[10] Another NP-hard problem, which consists of finding the biggest independent set in a graph. An independent set in the context of graph theory is a set of vertices, where none of these are adjacent to each other; thus we can think of an independant set as a "anti-clique".
[11] The Overlap Gap Property, as explained in [27], is a topological theory of algorithmic hardness

basis for their proof, found that for large problem instances and small values of p, QAOA gives uncorrelated results: Since each qubit will have an influence on d^p other qubits, where d is the degree of the problem instance graph, and qubits at distances larger than $2p$ will be uncorrelated in the output, the QAOA depth p needs to be at least a degree-dependent constant times log n in order to find an independent set at least 0.854 times the maximum possible set size for large degrees d. For the purpose of most pairs of qubits having correlated measurement outcomes, i.e. for QAOA to "cover" the whole graph (and thus find a set bigger than the proposed lower bound), d^{2p} has to be larger than n for random graphs of fixed average degree (assuming a high d).
In another paper by Farhi et al. [12], that was published a month later, they use similar reasoning to show that for MAXCUT on bipartite d-regular graphs, QAOA cannot exceed an approximation ratio of 0.5 if $(d-1)^{2p} < n^A$ for any $A < 1$ (i.e if $(d-1)^{2p}$ is less than n).
These results indicate that QAOA needs to "see" the whole Graph to not have its performance severely limited; considering that two d-regular graphs may locally look the same if the depth p is low enough for QAOA to not cover the whole graph, the algorithm will output the same expectation values for both these graphs, even though one of them may potentially be a bipartite d-regular graph which has a different objective value (than a general d-regular graph), resulting in worse performance for this case.

To sum it up, a large amount of currently available research on the performance guarantees and limitations of QAOA implies that the circuit depth p has to grow with the problem size n if we want to account of worst-case problem instances and be unaffected by the limitations that come with them.

3.2. Parameter Selection Heuristics for QAOA

In 2019, Zhou et al. [18] presented a new approach to carrying out the classical outer optimization loop, alongside performance benchmarks of QAOA for MAXCUT on 3-regular graphs and comparisons to QAA, which were performed to study the adiabatic mechanism of QAOA. In this new approach, existing patterns in the distribution of optimal parameters[12] of (p)-level QAOA are utilized to make an educated guess of quasi-optimal parameters for ($p+1$)-level QAOA, based on

that is based on the disconnectivity of the overlaps of near-optimal solutions. This property is useful, since it allows us to rule out classes of algorithms in a mathematically rigorous way.

[12] Zhou et al. observed that for QAOA for MAXCUT on 3-regular graphs, the shape of the optimal parameters $(\vec{\gamma}^*_{(p+1)}, \vec{\beta}^*_{(p+1)})$ is similar to the shape of $(\vec{\gamma}^*_{(p)}, \vec{\beta}^*_{(p)})$, i.e they follow a smooth distribution, with γ_i increasing smoothly while β_i decreases smoothly with an increasing number of alternating layers, where i is the index of the parameter (i.e the index of the unitary U_C/U_M that the parameter corresponds to). This distribution can also be observed in [15], which is discussed in Subsection 3.1.2 and proposes optimal angles for QAOA on 3-regular graphs which account for the worst case graphs and will yield a performance above a certain lower bound if the fixed angle conjecture is true.

the slowly varying continuous curve underlying γ, β. These quasi-optimal parameters can the serve as a good starting point for optimization. Zhou et al. proposed two different strategies for heuristically optimizing the algorithm, INTERP and FOURIER, which will be covered in the following sections.

3.2.1. INTERP

INTERP, the first of these strategies, uses linear interpolation of the known optimal parameters at the initial depth p for the purpose of finding good initial parameters for depth $p + 1$. This process is then repeated, incrementing p in steps of size 1, until the desired target depth is reached. This general protocol is illustrated in Figure 3 and works as follows:

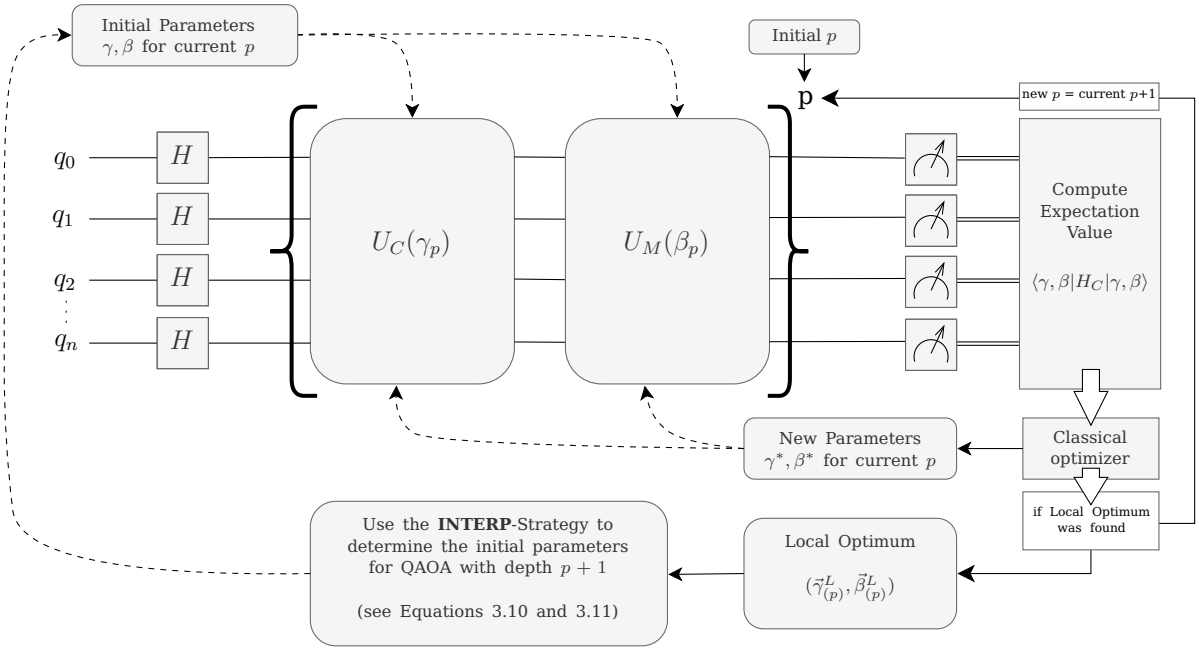


Figure 3.: QAOA-INTERP Circuit: INTERP-Strategy applied to basic QAOA

The basic quantum circuit is identical to the default QAOA method[13] outlined in Section 3.1. Unlike the basic algorithm however, INTERP linearly interpolates the initial parameters from the optimal parameters at a lesser depth. Beginning an initial depth e.g. $p = 1$, either good known parameters for this initial p are provided as a starting point, or the basic QAOA variant with random initialization is run to determine suitable initial parameters for this depth[14]. To find

[13]This can be switched for any other QAOA variant, since the heuristic strategies only concern the variational parameters, and do not require the phase-separation or mixer operators to be of a specific family.

[14]That is, the strategy at this point is equivalent to running the basic QAOA variant.

good parameters for depth $p+1$, the optimal parameters of depth p are used to interpolate initial parameters for $p+1$ according to:

$$\left[\gamma_{(p+1)}\right]_i = \frac{i-1}{p}\left[\vec{\gamma}^L_{(p)}\right]_{i-1} + \frac{p-i+1}{p}\left[\vec{\gamma}^L_{(p)}\right]_i \tag{3.10}$$

$$\left[\beta_{(p+1)}\right]_i = \frac{i-1}{p}\left[\vec{\beta}^L_{(p)}\right]_{i-1} + \frac{p-i+1}{p}\left[\vec{\beta}^L_{(p)}\right]_i \tag{3.11}$$

The current depth p is then incremented by 1 and the parameters that were obtained through interpolation in the previous step of the protocol are used as initial parameters for QAOA of this new depth. This process is repeated iteratively until the target depth, at which the algorithm is intended to be run, is reached.
According to Zhou et al., for MAXCUT on 3-regular graphs, quasi-optimal parameters can be determined efficiently in $\mathcal{O}(poly(p))$ time. This is a significant improvement compared to the default QAOA variant with random initialization, which requires $2^{\mathcal{O}(p)}$ optimization runs to achieve similar performance. However there is still no guarantee that the parameters found are the best possible ones. Overall, less attention was given to this strategy in the paper by Zhou et al., since the FOURIER method, which will be covered in the next section, was shown to be slightly better than INTERP for MAXCUT[15].

3.2.2. FOURIER

Similarly to the INTERP-strategy, the FOURIER Strategy uses the optimal parameters at the initial depth p for the purpose of finding good initial parameters for depth $p+1$, with the difference being that instead of interpolation, the $p+1$-level parameters are obtained by transforming the sequence of optimal parameters for level p to a frequency domain using the discrete sine/cosine transformation, and then simply reusing the optimized amplitudes of this frequency domain representation at depth p as the initial parameters for depth $p+1$, from where we start optimizing said parameters until a local optimum is found. We then repeat the aforementioned steps, iteratively increasing the current p until the targeted depth is reached. This general protocol is illustrated in Figure 4 and works as follows:

Like the INTERP method, the FOURIER strategy leaves the actual quantum-circuit unmodified, only acting on the parameters of the phase separation and mixer operators. Thus for the FOURIER-strategy applied to the default QAOA method the circuit matches the basic circuit defined in Section 3.1.
Analogously to how the INTERP method alters the classical outer loop as a means to obtain good parameters to start optimization from, the FOURIER

[15] Zhou et al. showed that the FOURIER strategy augmented with random perturbations started outperforming INTERP at $p \approx 20$ for an example instance of a 14-vertex weighted 3-regular graph; the performance of the basic FOURIER strategy did not differ from INTERP.

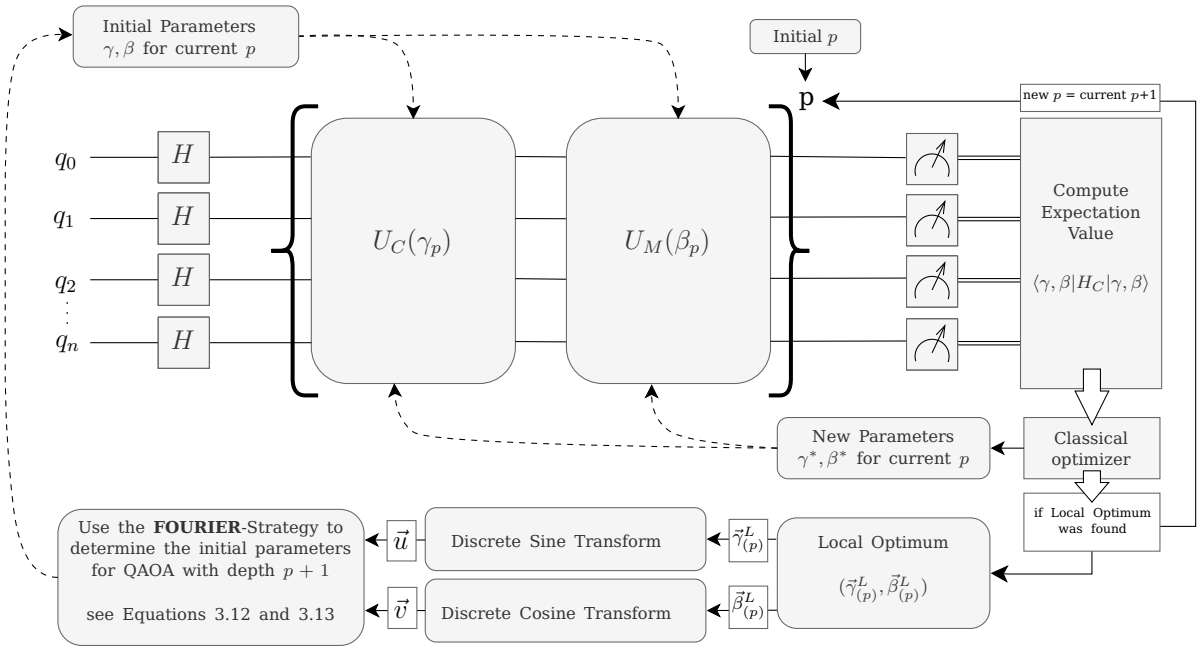


Figure 4.: QAOA-FOURIER Circuit: FOURIER-Strategy applied to basic QAOA

method also makes use of known optimal parameters at depth p to find appropriate initial parameters for a greater depth $p+1$. Beginning with an initial depth e.g. $p=1$ and good known parameters for this depth, which have been either determined beforehand or are identified by running the basic QAOA variant with random initialization at this initial depth. To find good parameters for depth $p+1$, the optimal parameters of depth p are transformed to a frequency domain representation, where the $2p$ parameters (γ, β) are expressed as $2q$ parameters $u = u_1, u_2, \cdots, u_q$ and $v = v_1, v_2, \cdots, v_q$, where the individual elements of (γ, β), γ_i and β_i, are specified as functions of (u, v)[16]. This is accomplished by performing Discrete Sine and Cosine Transformations to γ and β respectively:

$$\gamma_i = \sum_{k=1}^{q} u_k \sin\left[\left(k - \frac{1}{2}\right)\left(i - \frac{1}{2}\right)\frac{\pi}{p}\right] \tag{3.12}$$

$$\beta_i = \sum_{k=1}^{q} v_k \cos\left[\left(k - \frac{1}{2}\right)\left(i - \frac{1}{2}\right)\frac{\pi}{p}\right], \tag{3.13}$$

where u_k and v_k can be read as the amplitude of the k-th frequency component of (u, v)[17]. Initial parameters for depth $p+1$ are generated by reusing the optimal

[16] In essence, the discrete data points of the original sequence of parameters is converted to a sum of sinusoid functions (sine/cosine) which oscillate at different frequencies; in this way a periodic extension of the original domain is created, from which we can obtain data points which lie outside the original parameter sequence.

[17] That is, the amplitude of the k-th frequency component of the frequency domain, which rep-

frequency domain parameters (u, v) of level p[18].
The current depth p is then incremented by 1 and the parameters that were obtained in the previous step of the protocol are used as initial parameters for QAOA of this new depth. This process is repeated iteratively until the desired target depth is reached.

Aside from introducing a new parameterization of the variational parameters that control the operators of the quantum-circuit, the FOURIER method is characterized by two new integer-value parameters, q and R, which affect the generation of initial points in the classical outer loop and give rise to different variants of the FOURIER strategy:
The first parameter, q, gives a limit on the maximum frequency components we allow in the (u, v) representation of the parameters (See Equations 3.12 and 3.13)[19]. If $q \geq p$ the frequency domain representation can describe all possible QAOA protocols at level p. Thus we can use the full power of (p)-level QAOA by setting $q = p$. The notation used to denote this version of the FOURIER-Strategy is FOURIER[$\infty, 0$] (in the context of this thesis it will also be referred to as "FOURIER with unbounded q").
However, according to Zhou et al., the smoothness of the parameter-distribution (γ, β) for increasing p implies that only the low-frequency components are important. For this reason it can make sense to set a fixed q, independent of p, in order to limit the number of parameters, even as the circuit depth p increases. The motivation for restricting the number of parameters is that this may facilitate an easier optimization-process, since the parameter-space that has the classical optimizer has to search over is far smaller than in the case $q = p$. Limiting the maximum frequency component in this way constitutes another version of FOURIER, denoted as FOURIER[$q, 0$][20], where $q < p$ (the designation "FOURIER with fixed q" will be used interchangeably to refer to this strategy within this thesis).
Futhermore Zhou et al. noted that the basic FOURIER strategy with unbounded q is liable to getting stuck in local optima, and considered an additional variant of the FOURIER strategy, which is characterized by generating $R+1$ additional parameter sets[21] based on the optimum at depth p. Subsequently, the best of these $R + 2$ parameter sets is selected and used as the initial parameters for depth $p + 1$. R of these extra initial points are acquired by adding random per-

resents the parameter sequence γ and β respectively; or in simpler terms: The amplitude of the k-th sinusoid function in the sum of sinusoid functions, which we converted the original sequence of parameters to.

[18] Since the frequency domain representations (u, v) is a periodic extension of the original domains, $\gamma = \gamma_1, \gamma_2, \cdots, \gamma_p$ and $\beta = \beta_1, \beta_2, \cdots, \beta_p$, the elements γ_{p+1} and γ_{p+1} can also be expressed as functions of (u, v) using Equations 3.12 and 3.13.

[19] Essentially q determines how many different sinusoid functions we will use to represent the original parameters as a frequency domain; i.e. the number of individual elements of u and v.

[20] For example, if we were to restrict the number of frequency domain parameters in (u, v) to 10 (i.e. 5 parameters each for u and v respectively), we would call this version FOURIER[$5, 0$].

[21] In addition to the parameter set provided by the optimization-routine of the basic variant, thus producing a total of $R + 2$ initial points.

turbations to the optimum which was chosen for depth p, and one is obtained by adding a perturbation of strength zero to the p-level optimum[22]. The motivation for this procedure is that random perturbations of the optimal parameters have been shown to improve the performance of QAOA, since this makes it possible to "escape" from the local optima that cause problems for the basic FOURIER[$\infty, 0$] variant. Following the previously used nomenclature, this improved variant is denoted as FOURIER[q, R], where q is fixed, or FOURIER[∞, R] for the case where p is unbounded[23] (In this thesis, this variant will sometimes be termed "FOURIER with random perturbations"). The FOURIER variant with $R > 0$ is described in detail in Appendix A.
Like the INTERP-strategy, for MAXCUT on 3-regular graphs, quasi-optimal parameters can be determined efficiently in $\mathcal{O}(poly(p))$ time using the FOURIER-strategy, which is a notable improvement compared to the basic QAOA variant, but once again we have no guarantee that the parameters found using this strategy are (globally) optimal.
While examining the difference in performance of the proposed strategies, Zhou et al. found that the FOURIER[$\infty, 10$] started outperforming the FOURIER[$\infty, 0$] and FOURIER[$5, 10$] variants at $p \approx 20$. Moreover, in spite of having the total number of frequency domain parameters restricted to 10, the FOURIER[$5, 10$] variant was able to closely match the performance of the other strategies at low depth and even outperform the FOURIER[$\infty, 0$] variant at high depth.

3.3. On the Transferability of Optimal QAOA Parameters between Problem Instances

An active area of research concerning QAOA that is of particular interest for heuristic parameter selection strategies, is the transferability of optimal parameters between problem instances. As shown in Section 3.2, the FOURIER and INTERP method produce good initial points for $p+1$ from the optimum at depth p, which is either known beforehand or obtained by running the basic QAOA protocol at this depth p. Since running extra evaluations of a quantum-circuit can be a costly task, especially on NISQ-devices, the former would be the more feasible approach for running heuristically optimized QAOA under such conditions. This would however require a reliable method of obtaining good known optima at a low depth; for this reason heuristic parameter selection strategies would greatly benefit from the identification of problem instances classes for which the parameters are transferable between instances without a significant

[22] Essentially this just duplicates the initial point found by the basic FOURIER[$\infty, 0$] variant, however, Zhou et al. found that keeping this initial point (i.e. having a total of $R + 2$ initial points where 2 of these points are equal to the unperturbed initial point) improved the stability of this variant.

[23] For example, if $R = 10$ additional initial points should be generated while restricting the number of frequency domain parameters in (u, v) to 10, we would call this version FOURIER[$5, 10$].

decrease in the approximation ratio. In the following an overview of current results concerning this area of research is given:

In 2018, Brandao et al. [28] demonstrated that if the problem instances are drawn from a reasonable distribution[24], the objective function value is concentrated (in the sense that the objective value of any of these "typical" problem instances closely matches that of the others), when the parameters are fixed values. For this the parameters do not have to be optimal, since the the whole cost-landscape is equivalent for such instances (i.e. the cost-landscape is instance independent, depending on the fixed instance distribution instead). These results are however subject to some further constraint: Brandao et al.'s hypothesis for fixed p requires n to be large and the maximum degree not to grow with n. Furthermore, their conjecture regarding p growing with n requires the assumption, that the terms in the expectation value function F (formulated as a sum of subgraphs, see Equation 3.9) are independent, for the argument to work.

Later in 2021, Lotshaw et al. [29] benchmarked the performance of QAOA with depth $p \leq 3$ for MAXCUT on every non-isomorphic unweighted graph instance with a number of vertices $n \leq 9$. Aside from exemplifying the potential of intermediate depth QAOA to outperform the GW algorithm[25], they also were able to identify consistent patterns in optimal parameters: For depth $p = 1$, they found that generally for all n, optimal parameters concentrated around $\gamma_1 \approx -\frac{\pi}{6}$ and $\beta_1 \approx -\frac{\pi}{8}$ in a very limited range for almost all graphs. For $p > 1$ the majority of the (γ_1, β_1) parameters remained concentrated around this point, but the distribution broadens. For second-layer parameters (γ_2, β_2) the parameters concentrate around $\gamma_1 \approx -\frac{\pi}{4}$ and $\beta_1 \approx -\frac{\pi}{14}$[26], with some concentrations around seemingly random points. These results indicate that at least for small p, optimal parameters are transferable at low depth p between small, unweighted graphs (from the evidence given in this paper alone it cannot be known for certain if these results apply to larger p and n).
In the same year, Akshay et al. [30] took an analytical approach to assessing parameter concentration for QAOA which focused on considering parameter scaling as problem size increases. The outcome of this analysis suggests that optimal parameters concentrate as an inverse polynomial in the problem size: For the purpose of analytically calculating optimal parameters, the conditions which have to apply to achieve zero gradient were evaluated for $p = 1$ and $p = 2$, as well as in the limit of $n \to \infty$. in this framework parameters can be said to concentrate when for any optimal parameters γ_n, β_n at problem size n, there exists at least one set of parameters $\gamma_{n+1}, \beta_{n+1}$ for problem size $n + 1$ which is

[24] For the results presented in the paper, Brandao et al. focused on MAXCUT and chose this distribution to be over all 3-regular graphs, but also remarked that there are other reasonable distributions, e.g. graphs where each edge is included with probability $\frac{3}{(n-1)}$ so the expected average degree is 3.

[25] They found QAOA of depth $p = 3$ outperformed the Goemans-Williamson algorithm for most instances.

[26] Comparing this point to the one at $p = 1$, this brings to mind the observation that γ increases with p while β decreases.

polynomially close in n to γ_n, β_n.[27]. The analysis (as well as numerical evidence up to $p = 5$ and $n = 17$) suggests that the concentrations observed for the optimal parameters scale as $\mathcal{O}(n^{-4})$, which may be an indication for a optimal parameters having a limit as $n \to \infty$[28].
Also in 2021, the paper by Wurtz & Lykov on the fixed angle conjecture [15], which we previously discussed in Sections 3.1.2 and 3.2, was published. As already stated in Subsection 3.1.2, the authors hypothesize that when using fixed parameters, which are optimal for the worst case problem instances, the lower bounds calculated for these instances will also hold for any other 3-regular graph instance. In this way, parameters are transferable between instances while maintaining a performance, which is greater or at worst equal to the lower bound[29]. Wurtz & Lykov provided numerical evidence for this conjecture for $p < 12$ on all 3-regular graphs with $n \leq 16$ vertices, which firstly shows that the fixed angle conjecture holds (at least for small graphs with $n \leq 16$) and secondly indicates that such fixed parameters yield good results for a majority of 3-regular graphs, and serve as a good initial guess for optimizers.

All in all, current research implies that optimal parameters for QAOA *are* generally transferable between problem instances of certain families, however it is not yet fully clear to which problem families this applies and to what extent the QAOA performance is affected, especially at higher depths and problem sizes.

[27] More concretely, the authors formulate this definition of parameter concentration as $\exists l : \forall \gamma_n, \beta_n \exists \gamma_{n+1}, \beta_{n+1} : |\beta_{n+1} - \beta_n|^2 + |\gamma_{n+1} - \gamma_n|^2 = \mathcal{O}(\frac{1}{n^l})$.

[28] Akshay et al. comment that if the parameters concentrate as $\mathcal{O}(n^l)$ where $l \leq 2$, the optimal parameters may not approach any limit, however the existance of instances with such a scaling behaviour is not confirmed.

[29] Note that this differs from the other approaches presented thus far, which try to find parameters which are transferable while maintaining the same performance (or at least maintaining a performance that does not significantly deviate from the performance on other instances).

4. Method

This chapter covers the considerations made for the task of benchmarking the parameter selection heuristics and comparing them with the default (random initialization) QAOA method.
Section 4.1 gives an overview of the problem instances used for benchmarking the algorithms, consisting of a brief summary of the MAXCUT problem in Subsection 4.1.1 and a short outline of the method used in this work for generating problem instances, that are suitable for our experiments, is specified in Subsection 4.1.2. Following these definitions, Section 4.2 explains the components of the setup for the actual benchmarking process: First, in Subsection 4.2.1 we go over the framework used for running these benchmarks, as well as the some of its dependencies which are of particular interest. After that a brief recapitulation of the implementation of the heuristic strategies and their integration into the framework is provided in Subsection 4.2.2. Finally, in Subsection 4.2.3, a detailed description of the experiments (i.e details on the properties of the problem instances used and how often the algorithms are run on a particular problem instance) is presented.

4.1. Problem instance generation

For the purpose of benchmarking the default QAOA method and the heuristic parameter selection strategies outlined in Section 3.2, the algorithms are run for MAXCUT on random unweighted graphs of varying order ($|V|$) and size ($|E|$), which are generated by setting a specific edge-density (as opposed to setting an edge-probability, as is done for Erdös-Rényi graphs). The following two subsections give a more detailed description of generating problem instances for the experiments conducted in this thesis and the reasoning behind these choices.

4.1.1. MAXCUT

The optimization problem instances, which are used for the experiments, are instances of MAXCUT, a graph partitioning problem that is NP-hard to approximate to within a value better than $16/17 - \epsilon$, where $\epsilon > 0$ (this is $\approx 0,941 - \epsilon$ in

decimal floating point representation) [31][1].
While MAXCUT is a difficult problem to solve, it is fairly simple to conceptualize: The idea of MAXCUT is to partition a graph into two complementary sets, such that the number of edges between these two sets is maximal. The resulting cut is necessarily at least as big or bigger than any other possible cut (see Figure 5).

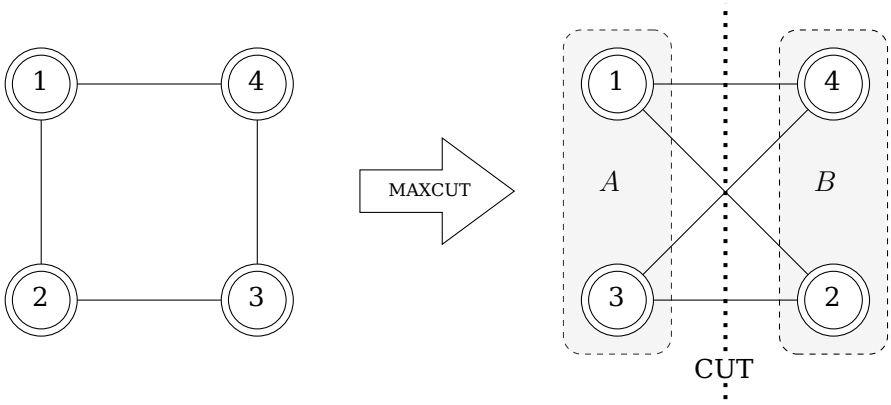


Figure 5.: An illustration of the MAXCUT problem, partitioning the graph on the right into two complementary sets A and B; In this example the graph $G(V, E)$, where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (1, 4), (2, 3), (3, 4)\}$, is partitioned into sets $A = \{1, 3\}$ and $B = \{4, 2\}$, thus yielding a cut of size 4, which is maximal for G.

While MAXCUT does have a number of practical applications, such as circuit design/integration and solid-state physics [32], the primary reason why MAXCUT was selected as the problem for benchmarking the different QAOA methods in this work is due to its prevalence in the literature regarding QAOA (see Chapter 3).
Since this work is mainly concerned with a basic comparison of QAOA and the different heuristic parameter selection strategies, rather than identifying specific problem classes for which the heuristic methods outperform the basic variant, it reasonable for the optimization problem, on which we run the benchmarks, to be correspondent to the problem most often considered in related work.
Furthermore, most of the results presented in the paper by Zhou et al. [18], which introduced the FOURIER and INTERP methods, were obtained by running the QAOA variants on MAXCUT. Consequently, by also using this problem, we can compare the performance for the problem instances used in this work (see Subsection 4.1.2 to the performance observed for the 3-regular graphs used by Zhou et al.

[1] While the actual value given in [31] is $17/16 - \epsilon$, they defined the approximation value as $c(x_{opt})/c(x)$, which is the inverse of how we defined the approximation ratio in this thesis (see Equation 2.23). For this reason we give the reciprocal of this value in the text.

Note that such comparisons are somewhat limited by the fact that in this thesis, a different classical optimization routine is used (see Subsection 4.2.3): While Zhou et al. use the gradient-based BFGS optimizer, this work utilizes the gradient-free COBYLA optimization method. Generally, gradient-based methods are more reliable at finding good parameters, but require the gradient of the cost function to be known; gradient-free methods do not rely on gradients to optimize the cost function, but may take longer to converge and (depending on the method used), may lack strong convergence guarantees. Hence, when comparing the results in this work to the results of Zhou et al. keep in mind that the differences in performance observed in this thesis may possibly be less pronounced when using gradient-based optimization, due to the fact that the classical optimization routine may yield better results even for the default QAOA variant in this case.

4.1.2. Generating Graphs by Density

The specific graph instances that MAXCUT optimization will be performed on during the benchmarking process, were chosen to be random graphs, for the reason that the performance of QAOA on such instances is not as well studied as QAOA on MAXCUT for 3-regular graphs (or d-regular graphs in general), see Chapter 3. For this reason it may be more interesting to study the case where the number of neighbors/degree of a vertex may differ between vertices, since we can not necessarily assume that practically interesting problems will be have regular graphs corresponding to them[2]. Another argument against using 3-regular graphs is that the results in the paper by Zhou et al. [18] concern 3-regular graphs, and it is not the goal of this thesis to simply try to reproduce those results, since that would not produce new insights into the performance of heuristically optimized parameter selection for QAOA.
For the purpose of generating random graphs, a density model $G(n, d)$ is used, where graphs are generated by specifying a number of nodes n and an edge density d (a floating point number that describes the ratio of number of edges to the maximum number of edges possible, i.e. $d = \frac{|E|}{max(|E|)}$). The graph is generated by first creating a list of all possible edges between the given nodes and then randomly removing edges from this list and adding them to the graph until the number of edges $|E|$ is such that the given density d is satisfied. For the implementation of this model, the corresponding function in the maxcut module of the ARCS 2022 repository [33] was used.
The reasoning behind using a density model as opposed to the widely used Erdös-Rényi model $G(n, p)$ (where a random graph with n vertices is generated

[2] Some of the related literature suggests that certain results pertaining to 3-regular graphs generalize to other graphs with small bounded degree [14], [28] (e.g for Erdös-Rényi graphs with edge probability $3/(n-1)$ the expected average degree is 3, which, according to Brandao et al [28], means that the parameter concentrations observed for 3-regular graphs also apply to such graphs). However, this is yet another assumption about the graph structure that may not always be true for practically interesting instances.

by adding an edge between two vertices with probability p), is that the size of the Graph ($|E|$) will be invariant between individual instances when using the density model[3]. In this way we can be sure that all graphs that share a density factor also have an identical size, regardless of the sample size (in contrast to this, for Erdös-Rényi graphs the true average will only approach the expected average with an increasing sample size). Since this thesis will only examine a small sample of problem instances (due to the high time and resource requirements for benchmarking the methods), the density model is a more appropriate means of generating graphs for the purposes of this work.
Aside from specifying the number of vertices and the edge density, a numpy random seed is set before generating the graph instances, in order to ensure reproducability of the problem instance generation (the specific seeds used will be covered in Subsection 4.2.3).

4.2. Benchmarking the different strategies

In order to actually run numerical simulations of the QAOA variants on these problem instances, the open-source SDK Qiskit [34], which, aside from providing transpilers for running circuits on actual quantum devices[4], also provides a number of simulator backends that allow for ideal or noisy multi-shot execution of quantum circuits, was used.
In addition, a framework for orchestrating the benchmarks, which ensures that the different variants are run on the same problem instances with the same initial conditions, is needed to make a fair comparison of these variants. The Framework used for the purposes of this thesis will be elaborated upon in the following section.

4.2.1. Benchmarking Framework

The setup used for Benchmarking the variants was the QUARK (QUantum computing Application benchmaRK) framework [35][5], which can be used to run benchmarks of industrally relevant applications on quantum devices. The framework requires an application, an associated mapping, a solver and a device to be specified. While the framework already provides some examples of these

[3] For Erdös-Rényi graphs, the average size can be expected to be $\binom{n}{2}p$, however the actual size will vary between instances due to the fact that the probability of an edge being present is independent of the number of edges already added.

[4] By "re-writing" (transpiling) the qiskit circuit formulation to yield a circuit that matches the topology and gate-set of a specific quantum device.

[5] More specifically, the QUARK v1.0 is used; during the writing of this thesis the version 2.0 was released, which introduces a few extension to the framework, which are useful in the the training and deployment of quantum generative models [36]. However, since this work is not quantum machine learning related (thus the new features would go unused) no switch was made from the v1.0 to the v2.0 version.

modular components, a usefull property of the kit, which was the motivation for using it in this thesis, comes from the fact that it is easily extensible by providing custom implementations of such components, allowing us to add solvers and/or problems not present in the base package.
Since the framework only contains an implementation of the default QAOA method, and no realization of the MAXCUT problem, we need to extend the Framework by writing several components, in order to perform the desired numerical simulations:

1. A MAXCUT module which extends the `MAXCUT` class and allows for generating or importing suitable graph instances, as well as validating and evaluating the solution supplied by the `solver`.

2. A mapping module which extends the `Mapping` class and maps the MAXCUT problem from a graph representation to an Ising model, which can be used by Qiskit to create an Ansatz which encodes the problem for QAOA.

3. A QAOA module which extends the `Solver` class and allows for the use of not only basic QAOA, but also QAOA using the parameter selection strategies proposed by Zhou et al. [18].

4. A means for creating a quantum instance (i.e. a quantum device or statevector simulator) that we will run the benchmark on; this can be done by providing a module which extends the device class, but since we use a Qiskit simulator backend, the functionality needed to create this quantum instance is simple enough that it can be included in a utils module.

5. A module configuration file for dynamically importing these additional components.

The specific details of these implementations are covered in the subsequent section.

4.2.2. Implementation of the Application, Mapping and Solver

For the Implementation of the necessary components, qiskit version 0.40.0 and qiskit.optimization version 0.5.0 were used[6], the specific implementation details are as follows:

[6]These versions are also dependencies of the QUARK v1.0 Framework, so no modifications were necessary in that regard.

Application The MAXCUT class extends the Application class and deals with creating/importing problem instances as well as validating and evaluating solutions to these problems. In order to generate a suitable problem, either graphs can be generated using the $G(n, p)$ Erdös-Rényi model, or existing graphs, which were as a gpickle file can be imported directly by specifying a path to the directory, which contains these files (this is the method used for the experiments, as we generate the graphs outside of QUARK from the density model outlined in Subsection 4.1.2 instead of the using the Erdös-Rényi model). Once a problem has been created or imported, if the MAXCUT class has not seen this problem instance before (the iteration number of the BenchmarkManager is used to keep track of which instances were already seen), it is brute forced, evaluating all possible assignments with the MAXCUT cost-function, to obtain the actual maximum possible Cut and objective value (cost). These results are then saved as a pickle file, so they can be loaded the next time we deal with this particular problem instead of brute forcing it again.
For the purpose of validating a solution, the MAXCUT class checks if the length of the bitstring returned by the solver matches the number of vertices in the problem graph. In order to be a valid solution, the bitstring must assign a group (one or zero) to each vertex. Aside from this, the MAXCUT class tests if all nodes are in one group (i.e. if all bits in the bitstring are the same, e.g. all ones or all zeros). If all vertices are in one group, the graph has not been cut at all, and thus this does not constitute a valid solution. Anything else is considered a valid cut.
After validating the bitstring returned by the solver, the solution is evaluated: The maximum objective value obtained by brute forcing the problem, $c(x_{max})$, and the objective value of the solution which is to be evaluated, $c(x)$, is used to calculate the approximation ratio of this solution to the maximum value, that is, $\frac{c(x)}{c(x_{max})}$. This approximation value is then saved with the other data about the benchmark (time taken for problem generation, validation, evaluation etc.), and used as the measure by which we judge the performance of the solver.

Mapping In order to formulate the problems given by the MAXCUT application class in a way the solver can work with, an ISING class, which ectends the Mapping class was implemented in order to represent the problem in the Ising formulation. This is done by utilizing the Maxcut class from the qiskit.optimization module, which takes a graph and returns a GraphOptimizationApplication object, which can be converted to a QuadraticProgram using one of the methods Qiskit provides for objects of this class. This QuadraticProgram is then converted to a QUBO formulation and from there transformed to the Ising Mapping we need. The ISING class then returns a dictionary containing the Ising model formulation and the time it took to map it.

Solver The QiskitQAOA_Param_Selection module extends the Solver class. Aside from allowing for configuration of the measurement shot number, optimizer method, maximum optimizer evaluations and QAOA depth, it also allows

for choosing a parameter selection strategy: If the "RI(default)" option is chosen, the default Qiskit QAOA is run with the given configurations. However if another option was chosen, starting from a starting depth with also is specified in the parameter options, the `QiskitQAOA_Param_Selection` class iterates through lower depths until the selected depth is reached. In each iteration, QAOA is run at this depth, and the results are used to heuristically determine the initial parameters for the next depth $p+1$ (where p in this case is the depth of the current iteration). The way in which these initial parameters are determined depends on the option selected (either "INTERP", "FOURIER_q_unbounded" or "FOURIER_q_fixed"; these options correspond to the methods with the same name outlined in Section 3.2).
Furthermore, there are options for specifying the FOURIER hyper-parameters q (the maximum frequency component allowed in the frequency domain parameters when using the fixed q FOURIER variant) and R (the number of random permutations added in order to possibly escape local optima)[7]: in the case of a q value being set, this value is passed to a function that determines the next initial parameters, which will only add new higher frequency components until q is reached. If q is bigger than the depth p in the current iteration, q is set to be $q = p$. In the case that a $R > 0$ is chosen, the solver will iterate through $R+1$ additional permutations for every iteration of p between the starting depth and the selected depth. At each of these nested iterations, a random permutation is generated according to the Fourier strategy with random permutations (see Appendix A), evaluated, and the initial parameters are saved if the resulting objective value is better than that of previous iterations[8].
Finally, in order to control the sampling behavior of the statevector simulator, there is an option to include a specific seed for the benchmarks thus allowing the results to be reproducible. This seed is incremented with the iteration number, so different results may still be observed when running QAOA on a problem instance multiple times, but we can still ensure that different QAOA variants will have the identical starting conditions at matching iteration counts of the same problem instance[9]
In addition to returning the bitstring that was sampled with the highest probability when running the the variants, the `QiskitQAOA_Param_Selection` class outputs the final optimal parameters found, the estimated objective value at this point, the number of cost function evaluations taken, the time taken by the optimizer, and the total time taken by the solution process.

[7] This is explained in more detail in Subsection 3.2.2 and Appendix A.
[8] Note that this variant will not be benchmarked in this thesis due to considerations of scope.
[9] For example, we can ensure sure that the FOURIER method at iteration 5 of problem instance 1 will have the same conditions (that is, sampling behavior and random initial QAOA parameters for the first iteration at the starting depth) as the INTERP method at the same iteration of this problem instance.

Device and Utils As mentioned in Section 4.2, since we will be using a Qiskit simulator backend instead of an actual quantum device[10], this functionality doesn't require a whole device class, and was included in a supplementary `QiskitQAOAUtils` module. The simulator used was the Qiskit `QasmSimulator` with the simulation option set to "statevector", that is, a dense statevector simulation which samples measurement outcomes from ideal (noiseless) quantum circuits, and all measurements are made at end of the circuit.
Aside from the function for creating a quantum instance (in this case, creating a `QasmSimulator` backend), a couple of other functions used by the `QiskitQAOA_Param_Selection` class, most notably the functions for heuristically obtaining suitable initial parameters and generating random permutations, were outsourced to this `QiskitQAOAUtils` module.

Module Configuration File In order to run QUARK with the components described in this section, a module configuration file, which contains `json`-formated data that communicates to the `BenchmarkManager`, which modules should be dynamically imported and where these can be found.
When executing the benchmarks, this file can be included in the call to the framework with the -m or --module flag.

4.2.3. Experimental Setup

In the numerical simulations that were run in order to benchmark the QAOA variants the optimizer was chosen to be the COBYLA (Constrained Optimization BY Linear Approximation) optimizer [37] for all variants. COBYLA is a gradient free optimization method, which, similarly to the Nelder-Mead Optimizer, doesn't require the gradient to be computed. COBYLA instead uses a linear approximation of the function, restricted to the neighborhood of the current point, to determine the next point to evaluate; when evaluating a point, COBYLA only considers changes in the cost-landscape within a certain "trust region" around the point. Due to these properties, COBYLA is particularly useful when the cost function is non-differentiable or expensive to evaluate. Despite the name, COBYLA can also solve unconstrained optimization problems [38].
The reason for this choice of optimizer is that the evaluation of a quantum circuit can be a expensive task, especially on imperfect NISQ era devices, and so it would in practice be reasonable to use a gradient-free method, which makes the most of a function evaluation. Another consideration when choosing the optimization-routine was that, since the modifications the heuristic parameter-selection strategies to QAOA occur on the classical side, using a more powerful gradient-based optimization routine, such as BFGS, would mean we have to simulate very high depths or large problem instances for the performance of

[10] So we can perform ideal numerical simulations, and in this way assess the performance of the ideal, noiseless case.

the different variants to diverge (since QAOA will converge on an optimal solution faster if the optimizer already produces near-optimal parameters, even without parameter-selection heuristics), i.e. we would have to perform more computationally expensive simulations to achieve a noticeable difference in approximation ratio[11]. Using the gradient-free COBYLA method, we may observe differences in performance between the methods sooner (at a lower p and problem size/order), since COBYLA lacks strong theoretical convergence guarantees [37], [38].
For the specific parameters of COBYLA, `rhoberg`, that is, the size of the "trust-regions", was kept at the default of 1.0, while `maxiter`, that is, the maximum number of function evaluations, was set to 1000[12].

For the choice of problem instances, which the simulations of the QAOA variants will be run on, the general idea is to examine how QAOA performance varies with changes to two properties of the problem instances: The problem instance order n (i.e the number of vertices) and the problem instance size d (i.e the edge-density, see Subsection 4.1.2). Aside from this, the effect, that starting the heuristic methods at different initial depths has on the performance, will be explored by running the variants with varying initial p on problem instances with a fixed order and size.
Given the above, three sets of experiments are to be run: First, for benchmarking the performance on problem instances with an increasing number of variables, the different variants are run on graphs with increasing order. The specific vertex-counts that were used are $6, 8, 10, 12$ and 14, each with a edge-density of 0.5[13]. Secondly, for assessing how changes in problem instance density affect the performance, the variants were run on graphs with increasing density, starting from $d = 0.5$ and going up to (and including) $d = 0.9$ in increments of 0.1, while the vertex-count is fixed at 8. The case where $d = 1.0$ was not included in the experiments, since in this case, any graph is a fully connected graph, for which finding the maximum cut is trivial[14]. Lastly, for comparing the performance of the variants when starting the heuristic methods with different initial depths, graphs with a fixed order of $n = 10$ and a fixed edge-density of $d = 0.5$ were used, and the different initial depths of $p = 1, 3$ and 5 were simulated.
The variants that are to be benchmarked are:

1. The basic QAOA variant, which will serve as a baseline for comparing the parameter selection heuristics to

[11] In the Paper by Zhou et al. [18], using the BFGS routine, divergence of the variants was only observed at $p \approx 20$ for a 14-vertex weighted 3-regular graph, which is outside of the scope of the simulations made for this thesis.
[12] In practice, none of the problems simulated in this thesis took substantially more than 200 function evaluations.
[13] Initially it was planned to also run the methods on 16-vertex graphs, however this turned out to be infeasible within the limited scope of this work.
[14] Since any balanced assignment of the vertices (i.e any assignment where the sizes of the sets, which we divide the vertices into, is equal or as close to equal as possible) produces the maximum cut.

2. The INTERP method, of which there is only one variant
3. The FOURIER[$\infty, 0$] variant of the FOURIER method, that is, the unbounded variant without random perturbations
4. The FOURIER[$5, 0$] variant of the FOURIER method, which is the fixed q variant without random perturbations, where $q = 5$

For each of these methods, individual simulations are run for every depth $1 \leq p \leq 10$. Unless explicitly stated otherwise, the initial starting depth for the heuristic methods is by default set to 1.
For each of the different problem instance types (as in, types of graphs distinguished by their vertex-counts/edge-densities) examined in the experiments, 3 non-isomorphic graphs are generated, resulting in a total of 30 unique graphs of different problem instance types, and each of the 4 variants is run 10 times on each graph. For the generation of these graphs, the seeds 123456, 246912 and 370368 were used alongside the respective density and vertex-count parameters[15]. For the 10 repetitions of each variant on each unique graph instance, the initial statevector simulator seed was set to 123456$+r$, where r is the number of the current repetition (ranging from 1 to 30, since each variant is run on 3 graphs 10 times; at every depth p between 1 and 10, this totals 300 runs of each QAOA variant per problem instance type in the set of experiments outlined above).

During the evaluation of these simulations, an idea for further interesting experiments arose from the results regarding graph density, the experimental setup for these will be covered together with the presentation of the results of this additional investigation in Subsection 5.3.2.

Finally, the GW algorithm was run on the problem instances that are specified above, in order to give an idea of how well classical algorithms can perform on these instances. The results of these experiments are reported in Appendix B.

[15]The only exception to this are the Graphs with $n = 6, d = 0.5$; for these the given seeds yielded two graphs that were much too similar, differing only in one edge. For this reason, the seeds 123456, 123556 and 123656 were used instead for graphs of this type.

5. Results

In this chapter, the results of the experiments specified in Subsection 4.2.3 are presented and described. First, in Section 5.1, an overview of the way in which the data was plotted and how to interpret this visualization, is given. Next, in Section 5.2, the results of the experiments with increasing graph order will be presented, and subsequently Section 5.3 covers increasing graph density: this section consists of the outcomes of experiments concerning increasing graph density, given in Subsection 5.3.1, and the results of the experiments with an increasing number of fully connected nodes, shown in Subsection 5.3.2. Finally, Section 5.4 reports the results of the experiments starting at an intermediate depths p.

5.1. Plotting the Data

As mentioned in Subsection 2.2.1, a way of describing the quality of a solution to a combinatorial optimization problem is in terms of the ratio between the cost of the solution and the cost of the global optimum, that is, the approximation ratio. This metric is used in the literature discussed in Chapter 3 for evaluating how well QAOA approximates the optimal solution, and will also be the criterion used in this work to gauge the performance of the QAOA variants.
Due to this choice of solution quality metric, plotting the mean and confidence intervals (or variance) would be ill-suited for visualizing the data in an non-misleading way, since the upper bounds of the confidence interval and variance would in some cases exceed a value of 1.0[1]. Statistically, this is unproblematic[2], however conceptually it may be confusing, since the approximation ratio cannot exceed 1.0 (in the context of maximization), as this would only be possible if the cost of the solution is greater than the cost of the global optimum, which is, by definition, the point with the highest cost (and thus, the cost of the solution can at most be equal to the cost of the global optimum).
In order to prevent such confusion, the statistical measures chosen to describe the data are the median and various quantiles. The reasoning behind this is, that

[1] For example, if the mean is very close to 1.0 but there is still some standard deviation such that adding 1.96 standard deviations to the mean (or, for the variance, squaring the standard deviation) results in a value greater than one.

[2] Since the confidence interval will still contain the true mean with a probability given by the confidence level, regardless of the fact that its range includes values which the mean cannot take.

since quantiles partition the distribution into multiple subsets of equal sizes, their upper bound cannot exceed the highest value that was observed in the samples. Thus, quantiles will give ranges, which, in our case, have at most 1.0 as the upper limit (since no sample in our data exceeds the approximation ratio of 1.0). The specific quantile ranges to be plotted are: First, the range between the first and third quartile (which can also be regarded as being the 25th and 75th percentile), also known as the interquartile-range (IQR). Second, the range between the 15th and 85th percentile, and lastly the 5th and the 95th percentile. These ranges contain 50%,70% and 90% of the samples respectively, and thus give a more accurate insight into the spread of the data than the median alone provides. The quantiles were calculated using the `quantile` function from the `stats` module in R [39], using the setting `type = 8`. This settings specifies which method to use when calculating the quantiles; type 8 was chosen because it gives quantile estimates that are approximately median-unbiased regardless of the distribution [40].
Given the above, we can interpret the plots given in the subsequent sections as follows: Since the median value, represented in the plots by the black line, is the value that seperates the upper and lower halves of the distribution, when viewing a plot, we can infer that half of all QAOA runs resulted in an approximation ratio equal to or greater than the median, and half of all runs yielded an approximation ratio equal to or less than the median; both halves contain 50% of the samples. The dark gray shaded area is the IQR, ranging from the quarter of the data, which contains the samples with the lowest approximation ratio to the quarter with the highest approximation ratio, thus covering the 50% of the samples that are closest to the median. The intermediate gray shaded area ranges from the 15th to the 85th percentile, in other words from the partition which is composed of the lowest valued 15% of the data to the partition that consists of the highest valued 15%, and so this range contains of 70% of the samples. In the same manner, the light gray shaded area is the range from the 5th to the 95th percentile, that is, the range between the best and worst performing 5% of QAOA all runs; accordingly, this partition spans 90% of the data points.
The median line, together with the shaded percentiles, can interpreted equivalently to a collection of boxplots, where each boxplot corresponds to a certain depth p: the median line and IQR range are consistent with the measures used for plotting the box, and the 70% and 90% ranges correspond to the whiskers[3]
The reason why the data is plotted as a line with shaded areas, rather than as a series of boxplots, is that if we were to plot an individual boxplot for every combination of depth, QAOA variant and graph order/size would negatively impact the readability of the plots, as the boxplots would have to be shrunken down by a significant amount in order to fit the entire series into the plot.

[3]While in most boxplots the whiskers are defined as being ± 1.5 IQR from the median, the choice of these boundaries can vary between representations and is often specified explicitly; for the purposes of this Thesis consider the case where there are two sets of whiskers, containing 50% and 70% of the samples respectively.

5.2. Solution Quality with an increasing Number of Nodes

Shown in Figure 6 are the results for the set of experiments where the edge-density of the problem instances is fixed to 0.5 the order increases from 6 to 14 with a step size of 2.

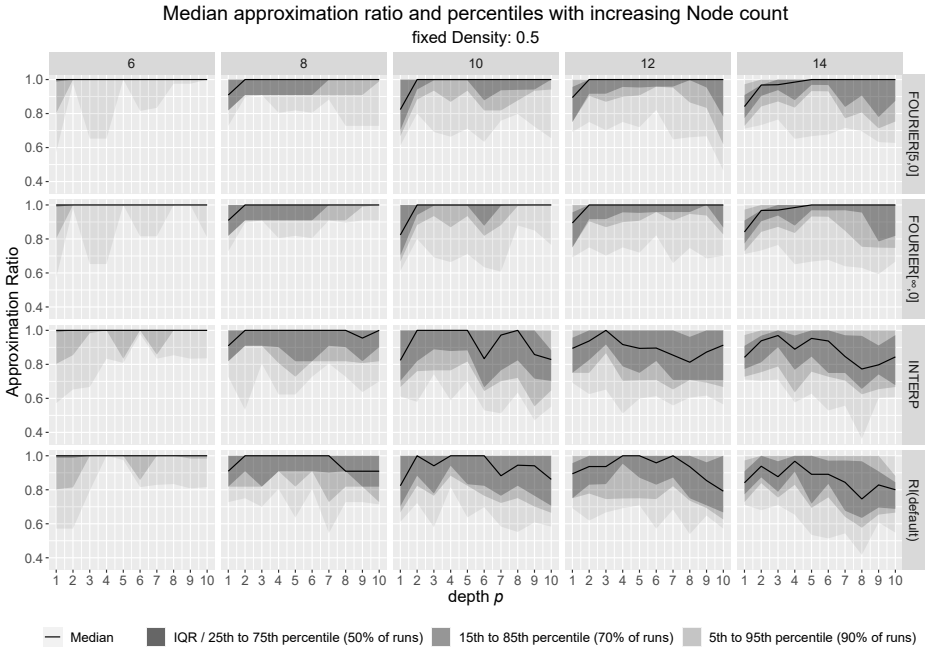


Figure 6.: Median approximation ratio and percentile ranges with increasing problem instance order.

An interesting observation we can make is that in some cases the median solution quality deteriorates with increasing p, which is counter-intuitive to the definition of QAOA given in Section 3.1, where the quality of the approximation improves as p increases. The most likely cause for this is that the number of parameters, which describe the Ansatz, grows as $2p$. This leads to a more expensive classical optimization, as the number of gradient components (or in the case of COBYLA: approximations of gradient components) that have to be computed is linearly dependent on the number of variables being optimized [41]. Consequently, the optimizer might get stuck in a local optimum more easily due to the complexity of the gradient. There has been previous work which suggests that that deep circuits result in exponentially vanishing gradient [42]–[44], which may support this theory.

As can be seen in the figure, for a problem instances with an order of 6, the median approximation ratio of all QAOA variants is 1.0 for every p tested, and their spread is almost identical. No meaningful comparison between the different methods can be drawn here, since solving problem instances of this order is equally uncomplicated for all variants.

For problem instances with an order of 8 however, variants begin to diverge from one another: While all variants first achieve a median approximation ratio of 1.0 at depth $p = 2$, the median of the basic QAOA variant and the INTERP strategy begins to decrease starting at $p = 8$ and $p = 9$ respectively; this decrease is less pronounced for INTERP. In contrast to this, both variants FOURIER variants maintain the approximation ratio of 1.0 for every $p > 2$ they were tested for. Furthermore, the spread of these strategies is confined to a considerably more narrow range than basic QAOA and INTERP, and this range generally decreases further with increasing p.
These differences in solution quality become more evident for a graph order of 10, following the trend we observed so far: Both the basic QAOA and INTERP variants reach a median approximation ratio of 1.0 at $p = 2$, but fluctuate between this value and lower approximation ratios for higher p, with the individual samples being distributed over a fairly large (compared to the case where the order is 8) spread. The FOURIER variants yield an approximation ratio of 1.0 for depths $2 \leq p \leq 10$, and have a comparatively low spread in relation to the other variants. The range of the spread of the FOURIER variants generally decreases up until $p = 6$, where it increases slightly before continuing to decrease as the depth goes towards $p = 10$; this decrease in spread is less pronounced for the FOURIER[$5, 0$] strategy compared to the FOURIER[$\infty, 0$] strategy at high depths.
The results for problem instances of order 12 show that this trend of divergence continues for higher p: While the range of the quantiles only increases slightly compared to the case where the order is 10, the median of the basic QAOA and INTERP variants is generally noticeably lower than than for that case. When examining the FOURIER methods, we can observe that the range of the spread narrows with increasing p up to $p = 9$, after which it broadens, more so for the FOURIER[$5, 0$] strategy than the FOURIER[$\infty, 0$] strategy; the difference between these two variants is more noticeable than in the case with 10 nodes.
The most evident differences in solution quality can be observed for problem instances of order 14, where for the first the basic QAOA and INTERP methods fail to reach a median approximation ratio of 1.0 at any of the tested p and the upper bound of the IQR is below 1.0 in some cases. As with the other problem instances, for these two variants this median generally decreases and the range of spread broadens with increasing p. The FOURIER strategies first reach a median approximation ratio of 1.0 at $p = 5$ and have an overall narrower range of spread compared to the other two variants. This spread broadens somewhat at higher depths, starting at $p = 7$ for the FOURIER[$5, 0$] variant and $p = 9$ the FOURIER[$\infty, 0$] strategy; aside from this fact, the magnitude of the range of spread is comparable for both variants.

In summary, for MAXCUT on random non-regular graphs where the density is fixed and the order increases, the median quality of the solution provided by basic QAOA generally decreases while the range of the spread increases and this effect gets more pronounced with increasing p (possibly due to exponentially vanishing gradients, see the observations made at the beginning of this sec-

tion. While INTERP does in some cases provide better solutions than the base variant, this was often only a minor improvement in this set of experiments, and due to the great variability in the solution quality achieved by INTERP, this strategy often produced worse results than the basic QAOA variant. In comparison to the two preceding variants, the FOURIER methods produced noticeably better solutions: unlike the INTERP and base variants, the median approximation ratio did not decline, and while the range of spread broadens at higher p, this effect is less pronounced and occurs later than for the other variants. In general the FOURIER[$5, 0$] variant closely matches the solution quality of the FOURIER[$\infty, 0$] strategy, with quantile ranges that are slightly broader and/or start to broaden earlier than the unbounded variant.

5.3. Effect of different Graph Densities

This section reports the results pertaining to the experiments with increasing graph density, presented in Subsection 5.3.1 and further experiments that were conducted to examine the effect of different edge assignments while keeping the order and edge-density fixed, given in section Subsection 5.3.2.

5.3.1. Solution Quality with increasing Graph Density

Shown in Figure 7 are the results for the set of experiments where the order of the problem instances is fixed to 8 and the graph-density increases from 0.5 to 0.9 in steps of 0.1.

Similarly to the set of experiments with increasing node count, we can observe that in some cases the the median solution quality declines with increasing QAOA Ansatz depth (see Section 5.2 for an explaination of a possible cause for this behaviour); however, generally, this effect is less prominent and occurs at higher depths compared to those experiments.

As illustrated in the figure, in the case where the graph-density of the problem instances is 0.5, we get the same results as for the experiments for problem instances of order 8 in Section 5.2; this is to be expected, as the edge-density was fixed to 0.5 in that set of experiments and, as explained in Subsection 4.2.3, we set seeds for statevector simulator and the generation of the graphs to ensure reproducibility. To quickly recapitulate the observations made for those problem insances: All variants first achieve a median approximation ratio of 1.0 at $p = 2$, however, the median approximation ratio of the basic QAOA variant and the INTERP strategy begins to decrease at high p while both FOURIER can maintain this solution quality for a p they were tested for. Moreover, the range of the spread of the FOURIER variants is narrower than the other strategies and further decreases with increasing depth.

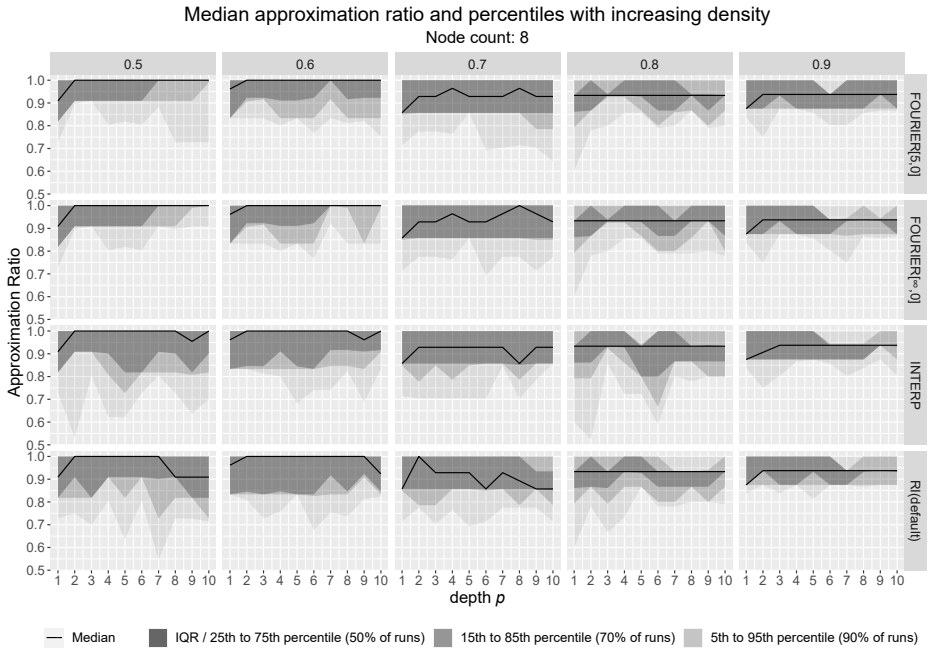


Figure 7.: Median approximation ratio and percentiles with increasing graph density.

For problem instances with a graph-density of 0.6, the results are very similar to the case where the instances have a graph-density of 0.5. Compared to those instances, for graph density 0.6 all strategies start at a somewhat higher median approximation ratio for $p = 1$, but have a slightly broader range of the spread. At $p = 2$, all variants first reach a median approximation ratio of 1.0. For higher depths $p < 6$ we can observe more noticeable differences between the FOURIER variants: The range of the quantiles is somewhat broader for the FOURIER$[5, 0]$ than for the FOURIER$[\infty, 0]$ variant (however, all things considered, the FOURIER$[5, 0]$ strategy still very closely matches the spread in solution quality of the unbounded variant).
When increasing the graph-density from 0.6 to 0.7, the differences between the variants are far more distinct than for the increase from 0.5 to 0.6. For problem instances with a graph density of 0.7, the median approximation ratio is considerably lower than for the previous densities and the range of spread is in general broader. For all methods, the median approximation ratio starts at ≈ 0.8 for $p = 1$. For basic QAOA, the median then increases to 1.0 at $p = 2$ before then decreasing again as p increases, fluctuating between ≈ 0.93 and ≈ 0.8; at high p, the median approximation ratio back at ≈ 0.8, which is the lowest out of all variants at these depths. Furthermore, unlike the other variants, for QAOA of level $p \geq 9$, the upper bound of the IQR is below 1.0. For INTERP, the median solution quality increases somewhat at $p = 2$, reaching ≈ 0.93 and then stays like this for higher depths, with the exception of $p = 8$, where it reverts to the starting value of ≈ 0.8 before rising back to ≈ 0.93 for $p > 8$. The FOURIER variants still generally yield better quality solutions than the other two variants, however the amount by which the median solution quality differs from the other strategies

is far less than in the results that were discussed so far, and the range of the the spread is comparable to that of the other methods. Out of both FOURIER strategies, FOURIER[$\infty, 0$] performed the best, with the median approximation ratio increasing somewhat steadily (though not as smoothly as for the experiments covered in Section 5.2), reaching 1.0 at $p = 8$, before then decreasing somewhat for $p \geq 9$. For depths $7 \leq p \leq 9$ the unbounded FOURIER variant achieves the highest median approximation ratios out of all variants, however, for $p = 10$ the median is identical to the INTERP and FOURIER[$5, 0$] strategies. The fixed q FOURIER variant performs slightly worse, closely matching the median solution quality produced by INTERP, with somewhat better median values at $p = 4$ and $p = 8$.
Increasing the graph density even further, from 0.7 to 0.8, we can observe changes in the spread of the quality of the approximation produced that are similarly significant to those described in the preceding paragraph. For problem instances, the graphs of which have an edge-density of 0.8, the spread of the solution quality is uniform for all of the tested QAOA strategies: They achieve a median approximation ratio of ≈ 0.93, which stays constant for every depth p tested. The range of spread is likewise similar for all strategies, with the upper bound of the IQR fluctuating between 1.0 and the median value, especially for the base QAOA variant, where this upper bound does not rise above the median value for $p \geq 6$; for the fixed q FOURIER variant the upper bound fails to exceed the median starting somewhat later at $p = 8$. The FOURIER strategy with unbounded p performs the best in this regard, maintaining an IQR upper bound of 1.0 for most of the tested depth (with the exception of $p = 1, 3$ and 7). Keep in mind however that these differences in the range of spread are minor, and it may be argued that spread of the solution quality is indistinguishable for the problem instances with graph-density 0.8 that were tested.
Finally, setting the graph-density of the problem instances to 0.9, the performance of the variants barely changes compared to the previous graph-density. The median approximation ratio at depth $p = 1$ is lower than for problem instances with a density of 0.8, increasing to ≈ 0.94 at depth $p = 2$ for the basic QAOA and FOURIER variants, and at $p = 3$ for the INTERP variant. This median value then remains constant for higher p. The range of spread narrows around this median solution quality for the startegies FOURIER[$\infty, 0$] and INTERP as well as the the basic variant. The quantile range of the FOURIER[$0, 5$] strategy generally stays the same. All in all, simliarly to the case where the graph-density is 0.8, the performance of the variants in nearly identical for problem instances with a graph density of 0.9.

In summary, for MAXCUT on random non-regular graphs where the order is fixed and the edge-density increases, the median approximation ratio decreases for all variants, and the higher the graph density is set, the less distinguishable the performance of these variants is. Furthermore, this decrease in median solution quality does not grow consistently with the increase in density: There is barely a difference in approximation quality when comparing problem instances with a graph-density of 0.5 to those with 0.6 or when comparing instances with

density 0.8 to 0.9 dense instances. In contrast to this, the dissimliarities between instances with graph-densities 0.6 and 0.7, as well differences between as 0.7 and 0.8 dense instances, are far more pronounced. Regarding the performance of the different methods relative to each other: For densities 0.5 and 0.6 the FOURIER variants evidently achieve better median approximation ratios and have a narrower spread than the other methods, however for densities ≥ 0.7 the spread of the solution quality more closely matches that of the of the other strategies, with the advantage FOURIER has over the other variants becomes increasingly less noticeable as the density goes toward 1.0. Overall, a high edge-density seems to impact the performance of the FOURIER strategy more negatively than the increasing graph orders examined in Section 5.2.

5.3.2. How Different Edge Assignments Affect the Solution Quality

As stated in the conclusion of the previous section, median solution quality does decrease with increasing density, however this relationship is not linear, with the most significant decreases happening when increasing the density for 0.6 to 0.7 and from 0.7 to 0.8. This may suggest that the magnitude of edge-density of the problem instance graphs is not the only factor affecting performance. Possibly the way in which the edges are assigned may also have an effect on the quality of the approximations produced by QAOA and its variants. In order to investigate this proposition, an additional set of experiments was conducted, to assess if the presence of fully connected nodes (i.e. nodes with degree $n-1$) has an effect on the solution quality.

The problem instances, which were examined for this purpose, have a fixed order of 8 and a fixed density of 0.8. Aside from these properties, an additional constraint we impose on the graph is the maximum degree that we allow for nodes in this graph. The specific values of this bound on the degree are $n-1$, which is equivalent to not limiting the degree at all, and $n-2$, which is equivalent to not allowing fully connected nodes in the graph. In order to modify the problem instances such that no fully connected nodes are present while preserving the original graph structure as much as possible, we perform modify the graph in such a way that the lowest possible number of edges are affected: For every node of degree $n-1$ the edge of v_{adj}, one of the nodes adjacent to this fully connected node, is moved to connect the v_{adj} and a node that is not adjacent to v_{adj}. in this way, the graph-density and the assignment of most of the edges are unaffected by the modifications.

Moreover, in a further set of experiments, the nodes with degree $n-1$ were removed from the problem instance graphs before running the variants on these instances. After a solution, which assigns a group to each of the remaining nodes, is produced, the nodes are reinserted into the graph in a balanced way

(i.e. such that the sizes of the groups are roughly equal); this is done by assigning groups to the removed nodes such that a node always gets added to the group which currently contains less nodes[4].

The results of these supplementary experiments are shown in Figure 8.

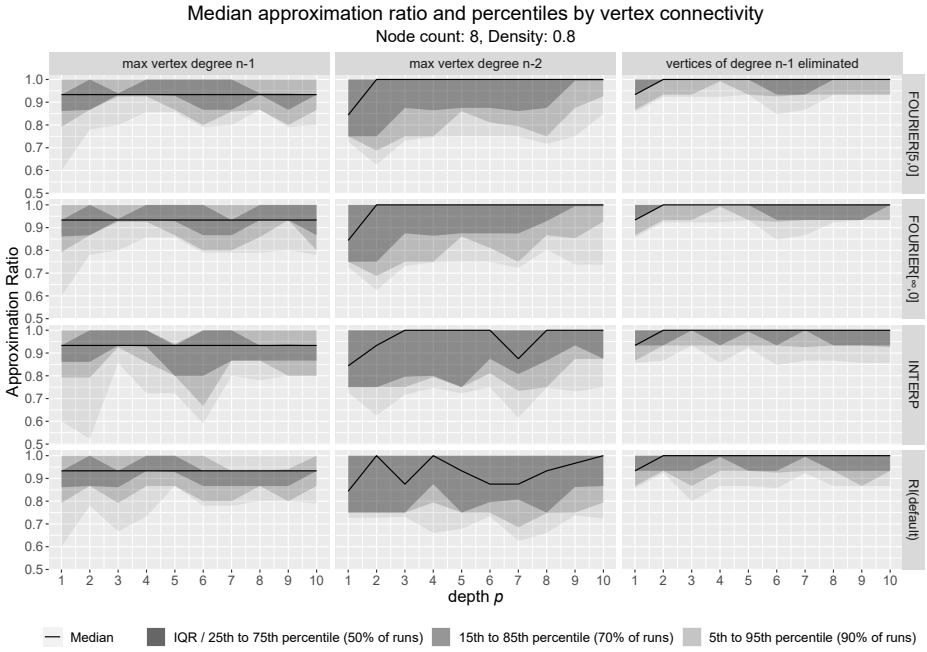


Figure 8.: Median approximation ratio and percentiles by density and presence of nodes with degree n-1.

Since setting the maximum allowed degree of the graphs to $n-1$ corresponds to not restricting the degree at all, the results for these problem instances are identical to those presented in Section 5.3 with a graph-density of 0.8. For these instances the median approximation ratio is identical for all methods and the range of the spread is very similar between variants, concentrating in a narrow range around the median. In other words, the approximation ratios provided by the different methods are barely distinguishable from one another.
If we restrict the maximum degree to be $n-2$, that is, disallow fully connected nodes, we can observe results, which are considerably more distinct from each other: In general, the methods have a higher median approximation quality, with the exception of low depths p and some higher depths of the basic variant. Unlike the case where the maximum degree is $n-1$, the methods are able to produce a median approximation ratio of 1.0, and this median is achieved by the heuristic strategies more consistently than by the basic QAOA variant, for which the median frequently fluctuates between 1.0 and lower values. This fluctuation can be described as being seemingly random between $p=1$ and $p=4$, before steadily declining form $p=5$ to $p=7$, after which the median increases continuously up to a value of 1.0 at $p=10$. The INTERP strategy has less variation

[4]Note that this is a very naive approach to eliminating and reinserting nodes, for a more sophisticated method see RQAOA, see [45] for details on this approach.

in its solution quality, steadily increasing to a median of 1.0 at $p = 3$, and maintaining this value for all higher depth except $p = 7$. The range of spread of the basic and INTERP variants is comparable, and overall broader than the spread for the maximum degree $n - 1$ case. The FOURIER variants on the other hand yield distinctly more narrow quantile ranges, which get progressively narrower as p increases. Furthermore, the median approximation value is varies less for these strategies, reaching 1.0 at $p = 2$ and conserving this value for all higher depths that they were tested for. There are no apparent differences between the FOURIER[$5, 0$] and the FOURIER[$\infty, 0$] strategy.
When excluding nodes of degree $n-1$ from the solution process and later assigning groups to these fully connected way in a balanced way, all variants perform similarly, reaching a median of 1.0 at $p = 2$ and then maintaining this solution quality for higher p. The range of the spread is exceptionally narrow for all strategies, which the FOURIER strategies achieving a slightly more narrow spread than the other strategies.

In summary, the results imply that for MAXCUT on random non-regular graphs, the median approximation is lower if fully connected nodes are present in the graph compared to when there are no nodes of degree $n - 1$; conversely, the range in which the individual samples are spread narrows in the presence of fully connected nodes for the problem instances examined in these experiments. Moreover, is seems that the FOURIER strategies can better maintain their advantage in approximation quality over the other methods if there are no fully connected nodes present in the the graph. Finally, the results demonstrate that, when augmenting QAOA to solve a constrained version of the problem, such as removing fully connected nodes from the problem instance graph, the heuristic strategies may not yield noticeably better approximation qualities compared to the basic QAOA variant.

5.4. Solution Quality Starting with an Intermediate Initial p

Figure 9 shows the results for the set of experiments where the edge-density of the problem instances is fixed to 0.5 and the order is fixed to 10, while the initial p, the optimum of which the is used by the heuristic strategies for determining good initial parameters for higher level QAOA, is increased, starting at the default of $p = 1$, and subsequently being set to the intermediate depths $p = 3$ and $p = 5$.

Due to the fact that an initial depth of $p = 1$ is the default used in the previous experiments, the results for the case where the initial p is 1 are identical to those presented in Section 5.2 for problem instances with with a graph order of 10 (and a graph-density of 0.5, since the density was fixed to that value for the experiments that were reported on in that section). To briefly recapitulate the

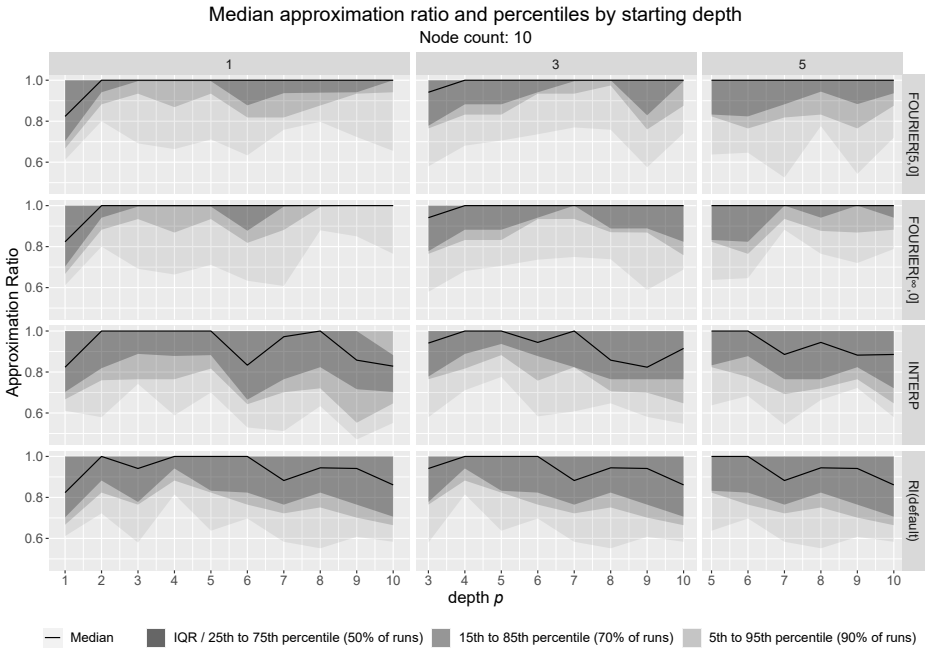


Figure 9.: Median approximation ratio and percentiles by initial depth p

findings for those instances: Both the basic QAOA and INTERP variants reach a median approximation ratio of 1.0 at $p = 2$, but do not always conserve this median for higher depths, with the solution quality decreasing slightly in general; the range of the spread for these variants is fairly broad when compared to the FOURIER strategies. The FOURIER variants yield an median approximation ratio of 1.0 consistently for depths $2 \leq p \leq 10$, and have a comparatively low spread in relation to the other variants, with the range of spread being somewhat narrower for the FOURIER$[\infty, 0]$ strategy compared to the FOURIER$[5, 0]$ strategy at high depths $p > 6$.
As seen in the figure, if we make the heuristic methods start optimization at a higher depth by setting the initial depth to be $p = 3$, these results only change slightly. The approximation quality of the basic QAOA variant unsurprisingly does not change at all, as its starting parameters are randomly initialized[5] and thus it does not use lower depth optima for producing a good initial point to start from. The INTERP variant is the only method where the median does not remain unchanged as the initial depth increases, however in general the magnitude of the shifts of the median is similar to the fluctuations we can already observe for the initial $p = 1$ case. The ranges of the quantiles are also similar to those observed when starting INTERP from $p = 1$, generally broadening while the depth increases. Overall the spread of the solution quality is comparable to the previously tested depth. The median approximation ratio of the FOURIER variants does not change for the initial $p = 3$ case, aside from the different value

[5] As stated in Subsection 4.2.3, this randomness is controlled by an initial seed, and so we get exactly the same results when running the basic variants with different initial p.

at depth $p = 3$, where the median matches that of the basic QAOA variant[6]. The range of the spread on the other hand is somewhat broader compared to the case where the initial depth is $p = 1$, though it is still narrower than for the other strategies at any of the initial p that were tested. This variation in distribution of the individual samples continuously decreases as p grows, up to $p = 7$, where it starts increasing somewhat steadily. All things considered, the results so far suggest that the FOURIER variants are less sensitive to changes to the initial depth than the INTERP strategy.
The last starting depth that was tested is the initial depth $p = 5$, which, in the case of these experiments, is half that of the final depth $p = 10$; This corresponds to the intermediate starting p suggested by Zhou et al. in Section VI. of their paper (see [18]), which covers some considerations for experimental implementation of the heuristic strategies. As in the initial $p = 3$ case, the results of the basic QAOA method are invariant, and again, the only only variant for which the median solution quality slightly changes is INTERP, which nevertheless is comparable (in regard to its median and quantiles) to the results we observed for lower initial depth. Both FOURIER strategies achieve a median approximation ratio of 1.0 for all depths $5 \leq p \leq 10$, which is identical to their median for the previously tested initial p levels. The range of the spread is somewhat broader than when starting the methods at $p = 1$, to a similar extent as in the initial $p = 3$ case, and generally narrows as the final p increases. This decrease in the quantile ranges of the quality of the approximation produced by the FOURIER variants get increasingly more pronounced for the FOURIER[$\infty, 0$] strategy (compared to the FOURIER[$5, 0$] strategy) as the target depth grows.

In summary, for MAXCUT on random non-regular graphs, the effect that starting the heuristic parameter selection strategies at a higher initial p has on the spread of the quality of the resulting approximations seems to be minor for the intermediate initial depths tested. Furthermore, the results indicate that the FOURIER variants are less affected by changes to the starting depth than the INTERP variant. While individual samples of the FOURIER variant with unbounded q lie within a somewhat narrower distribution the fixed q variant, the FOURIER[$5, 0$] strategy nonetheless closely matches its performance, with the spread for the fixed q case being only moderately broader at higher p.

[6]This is of course to be expected, as we have not supplied any known optimal parameters to start from in these experiments. Thus, the heuristic variants do not have access to any lower depth optima at the starting depth, and the basic variant has to be run once to obtain an optimum to start from.

6. Discussion

The series of experiments, which are specified in Chapter 4 and reported on in Chapter 5, indicate that the FOURIER variants that were examined yield a median solution quality that is considerably higher than that of the basic variant. Furthermore, the solution qualities of the individual samples concentrate in a narrower range around this median: In more simple terms, the midpoint of the ordered dataset of approximation ratios produced by the FOURIER variants is higher than that of the basic QAOA variant, and as such we can say that solution qualities produced the by the best 50% of FOURIER runs have a greater lower bound than those of the best 50% basic QAOA runs. Furthermore, the range around this value, in which most of the other samples of solution quality data lie, is in general more narrow for FOURIER than for basic QAOA. In essence, the solution quality produced by the FOURIER variant is more consistent than that of the basic variant.
At higher depths, where finding optimal parameters is more challenging and the median solution quality decreases for basic QAOA optimized by COBYLA, the FOURIER variant is often unaffected by such a decrease for the depths and instances that were tested. The increase in the range of the spread, which can be observed for the basic QAOA variant, in general sets in at a higher depth for the FOURIER strategy or is not present altogether for the depths that were examined. This demonstrates the potential for the FOURIER variants to improve the outer-loop classical optimization of QAOA parameters, finding good parameters more consistently than optimization starting from randomly initialized parameters.

In contrast to this, the INTERP strategy did not produce noticeably different median approximation ratios and quantile ranges compared to the basic variant. In some cases INTERP was able to outperform the default variant for some specific depths, but in other cases or for other depths the default variant performed similarly or better, with no discernable pattern underlying this variation. All in all, the INTERP produces solution qualities with a similar consistency as basic QAOA, and yielded comparable median values and quantile ranges as this variant.

In regard to how certain properties of the problem instances, which were specified in Subsection 4.2.3, affect the solution quality of the different variants, we made the following observations:
Generally for all methods the median decreases and the quantile ranges broaden with increasing problem graph order at all depths, and the magnitude of this decrease/broadening appears to be proportional to the magnitude of the increase

of the order for the problem instances that were tested. In other words, there seems to be a negative linear relationship between solution quality and problem order. Notably, for the depths the variants were tested for, the FOURIER strategies were affected to a noticeably lesser degree than the other strategies, to it is not known if these effects start to set in at higher (untested) p.
When increasing the problem instance graph density (see Subsection 4.1.2), which corresponds to increasing the problem size, the median also decreases, however this decrease is less correlated with the increase in density compared to the experiments where we increased to order: the most significant changes occurred when increasing the density from 0.6 to 0.7 and 0.7 to 0.8, with the other increases barely affecting the solution quality. This may be an indication that the problem size is not the only factor affecting the solution quality here. Further experiments, which placed a limit on vertex connectivity found that graphs with equal density but lower maximum vertex connectivity yielded consistently better results. Consequently, the presence of nodes with high connectivity may have a greater negative effect on the solution quality than just the number of edges or the order of the problem instance graph.
The results of the experiments, where heuristic optimization is started at an intermediate p instead of setting the initial depth to $p = 1$, suggest that starting at intermediate depths does not significantly affect the median solution quality and the range of the quantiles is is only slightly affected. In general, starting heuristically optimized QAOA at an intermediate p seems to be promising for experimental or practical implementations of the FOURIER and INTERP strategies.

Given the above, what have we learned about heuristically optimized QAOA?
On one hand, some observations made by Zhou et al. in [18] could be reproduced for the small sample of random non-regular graphs that was examined in this thesis:
To begin with, we were able to confirm that the FOURIER strategies can improve the outer-loop optimization compared to random initialization. As mentioned above, the qualities of the approximations they produce are more consistent, and often have a higher median than those of the basic variant. Thus, since the default QAOA strategy produces less consistent results, we may often need more runs of the basic variant to find a solution as good those produced by the FOURIER variants. This coincides with the observations made by Zhou et al in Chapter VI of [18], where they showed that the median number of random initialization runs needed to match the performance of the heuristic strategies scales exponentially in p (for MAXCUT on random 3-regular graphs). Moreover, we were able to validate that the FOURIER variant with fixed q, i.e the FOURIER[$q, 0$] strategy was closely able to match the solution quality of the unbounded q variant (FOURIER[$\infty, 0$]), which substantiates the conjecture, that only the low-frequency components of the QAOA parameters in the frequency domain representation are important, as made by Zhou et al. in Chapter IV of [18]. Furthermore, we were able to see that when starting heuristic optimization at an intermediate level p, this did not have a noticeable detrimental

effect on the solution qualities or the consistency with which these are produced. Since starting at a higher depth lets us skip the the QAOA runs at lower depths, this allows us to find a solution faster, and since that consistency of the solution quality does not change significantly, this would be an overall improvement for practical implementations. This supports the consideration made by Zhou et al. in Chapter VI of [18], that starting the heuristic strategies at an intermediate depth may be advantageous of realistic implementations of these.
On the other hand, some observations, which Zhou et al. were able to make for regular graphs, could not be made for the sample of non-regular graphs examined in this thesis:
Most notably, we could not observe an improvement of the outer-loop optimization for the INTERP strategy, which is comparable to the basic QAOA variant in terms of the consistency of the approximation qualities produced. This is in stark contrast to the results for regular graphs, where the INTERP strategy performs similarly well as the FOURIER strategies. This deviation from these observations implies that the INTERP strategy may be less suitable for MAX-CUT on problem instances where we cannot be assured of regularity of the corresponding graphs.
Finally, aside from confirming that an increase in graph order or density has a detrimental effect on the approximation ratios achieved by the strategies, most likely to an increase in the difficulty of the QAOA parameter optimization, we conjectured that there might be a further factor influencing the performance of QAOA, which is the connectivity of the nodes. The results for the small sample of non-regular graphs suggest that the presence of nodes with high connectivity may considerably impact the approximation quality of QAOA and the heuristic strategies[1]. However there still has to be some investigation to be carried out in order to confirm or deny if this conjecture holds, as this effect may not only be due to the nodes having a high connectivity, but instead due to certain other structural properties of the graphs, and thus, this observation may not be present for larger samples of graphs. If however it turns out that high connectivity is an obstacle to the performance for heuristically optimized QAOA, this would motivate using the heuristic strategies in conjunction with a QAOA variant, which iteratively reduces the problem, such as RQAOA [45].

Taking these results and the observations made by Zhou et al. in [18] into consideration, heuristic optimization of QAOA using one of the FOURIER strategy variant seems to be feasible, especially in the context of realistic implementations on NISQ devices:
Considering that we need to run the basic QAOA variant more often to produce an approximation quality that matches that of the FOURIER variants, and that the evaluation of a quantum circuit can be an expensive task on the imperfect NISQ-era devices we currently have access to, the FOURIER strategy seems promising for near-term implementations of QAOA. Out of the various modifications we can make to the FOURIER strategy, setting a fixed q and starting

[1] with the exception of complete graphs of course, as these are trivial to solve, since any balanced assignment of nodes yields the maximum cut

heuristic optimization at an intermediate depth q appear to be especially favorable for realistic implementations. The first of these, setting a fixed $q < p$ (i.e. restricting the maximum number of frequency components we allow in the FOURIER parameterization), simplifies the optimization, as the number of frequency domain parameters is bounded even as the circuit depth grows. The second modification, using a intermediate level starting p, can further reduce the number of times we have to run QAOA, as the effect this has the quality of the approximation is unsubstantial.
A further modification, which was not tested in this thesis but which may be advantageous for near-term implementations, is the use of known good initial parameters (or an educated guess) to start heuristic optimization from, such as the fixed parameters proposed in [14] (see Sections 3.1.2 and 3.3), as in this way we could account for worst case instances and would not have to rely on the initial parameters produced by running the random initialization QAOA variant at the initial p. However, further investigation is required to assess how well this works in practice.

7. Conclusion and Outlook

in summary, this work studied the QAOA parameter optimization heuristics INTERP and FOURIER with a focus on NISQ-era devices by first giving a comprehensive overview of QAOA and the heuristic strategies as proposed by Zhou et al. [18], as well a review of further related research, especially regarding performance guarantees and limitations of QAOA and transferability of optimal parameters. Next, numerical simulations were performed in order to benchmark the heuristic strategies on a small sample of instances of the MAXCUT problem, which vary in graph order and density and are not constrained to being regular graphs.
the results give insight to the performance of heuristically optimized QAOA, and show that the FOURIER strategy variants appear to be a promising modification to the outer-loop optimization of QAOA parameters for realistic near-term implementations of QAOA. Especially the fixed q variant in conjunction with using an intermediate starting p (instead of setting the initial depth $p = 1$) appears to be advantageous for such implementations.
While benchmarking the strategies for different problem instance graph orders and densities, it became apparent that the decrease in median solution quality, which comes with an increase in density, is not linearly proportional to this increase, instead increasing more for some density levels than for others. This may suggest that other factors, such as the connectivity of the individual nodes, may also affect the performance of the QAOA variants. A further comparison of problem instances with differing maximum node connectivity substantiates this inference somewhat, however further investigations need to be carried out to confirm or disprove this assumption. If a high node connectivity proves significantly detrimental to the performance heuristically optimized QAOA, the use of QAOA extentions which iteratively reduce the problem, such as RQAOA, together with the heuristic strategies may be advantageous for near-term QAOA implementations.

Future research into heuristic QAOA parameter optimization strategies could build on the numerical simulations made in this thesis, and examine the performance of the strategies at higher p or for problem instances with greater graph orders, to determine if the FOURIER maintains the consistency, that was observed in the experiments performed in this work, at higher depths or for bigger problem instances.
Furthermore, one could examine the effect that different optimizers have on the performance of the heuristic strategies, since the advantage, which the

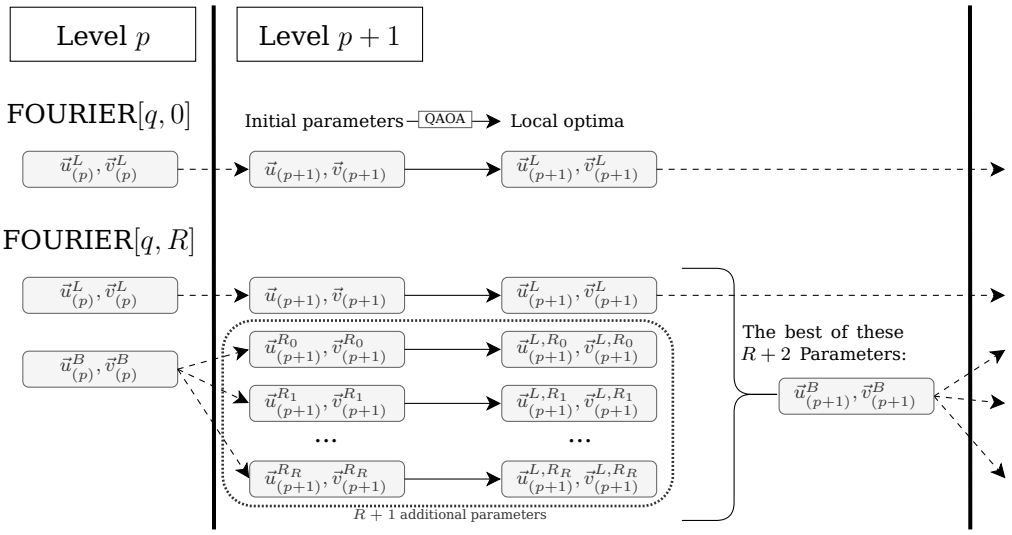
FOURIER strategy provides, may be less substantial when using a more expensive optimization method. Performing such a comparison could give insights into trade-off between solution quality and the number of function evaluations (that is, evaluations of the quantum circuit) required for convergence when using different optimizers with or without heuristic optimization. This knowledge could help us when configuring QAOA to solve a problem of a certain size, since we would have a better idea of what solution quality we can expect from certain configurations.
Additionally, an examination of different problems may be interesting, as this would give us a better idea of how the performance of the heuristic strategies differs from the observations we made for MAXCUT and could possible identify problems or problem instance types for which the use of INTERP is advantageous.
Finally, two interesting research directions, which directly continue where we left off in this thesis, are studies concerning the effect of providing good known parameters, such as the fixed parameters discussed in [14], and further investigations into the the effect of structural properties of the problem instance graphs, such as vertex connectivity, as was mentioned in the above paragraph. The latter of these considerations may motivate the study of heuristically optimized RQAOA: Not only could we overcome possible structural obstacles to the performance of QAOA by recursively eliminating variables and replacing them with problem constraints, but possibly even achieve better approximations in a smaller number of QAOA runs than when using the heuristic strategies or RQAOA independently. The reasoning behind this is as follows: It was shown that when warm-starting RQAOA with an initial state that corresponds to the solution of a relaxation of the problem, this can outperform standard RQAOA [46]. While the warm-start extension of QAOA is not equivalent to the heuristic strategies, both work by modifying QAOA based on some initial solution; in the case of the heuristic strategies, this is knowledge of optimal parameters at a lower depth, while for warm-start QAOA this is the knowledge of a solution to a relaxed version of the problem. Due to these similarities, heuristically optimizing the RQAOA parameters may prove to be favorable for near-term implementations of QAOA.

A. FOURIER method with random perturbations

An improved variant of the FOURIER-Strategy that sets $R \geq 0$; The aim of this version is to circumvent a limitation of the basic version, namely the problem that this variant sometimes gets stuck in sub-optimal local optima. Random perturbations of the initial parameters have been demonstrated to improve the performance of QAOA, since this makes it possible to "escape" from these flawed local optima.

This improved variant determines initial parameters for depth $p+1$ by optimizing over $R+1$ additional parameter sets (in addition to the parameter set provided by the optimization-routine of the basic variant), where R of these sets are generated by adding random perturbations to the best local optimum that was found for depth p using the FOURIER strategy (i.e. a random permutation is added to the optimal frequency domain parameters (u, v) at level p a total of R times, resulting in R extra parameter sets; the one additional set in the $R+1$ randomly perturbed initial points are the best parameters for depth p with a perturbation of zero added, effectively duplicating the point found by the basic variant). Subsequently the best of these $R+2$ parameter sets (consisting of the optimal parameters of the basic version and the $R+1$ additional parameters generated in the previous step) is chosen to be used as initial parameters for $(p+1)$-level QAOA.
Aside from $(u^B_{(p)}, v^B_{(p)})$, which is the best of these $R+2$ initial points, $(u^L_{(p)}, v^L_{(p)})$, which is the original optimum found at level p, is kept in parallel for optimization at level $p+1$; when determining an initial point for $p+2$ based on the optimal parameters at level $p+1$, we will again generate $R+1$ random extra points from $(u^B_{(p+1)}, v^B_{(p+1)})$, while $(u^L_{(p+1)}, v^L_{(p+1)})$ will be kept as is. From these $R+2$ additional parameter sets we once again select the best parameters and repeat this process iteratively until the targeted depth is reached[1].
A comparison between this variant (FOURIER-QAOA augmented with random permutations) and the basic FOURIER variant is made in Figure 10:

[1] by keeping the $(u^L_{(p)}, v^L_{(p)})$ optimum we can improve the stability of this heuristic, avoiding erratic/non-smooth optimal parameters; Since we are thus essentially performing the basic variant in parallel to the actual augmentation of the FOURIER strategy and including those results in the pool of optima to select from, we can ensure that this variant will perform at least as good as the basic strategy (i.e. if the random perturbations lead to a suboptimal performance at the target depth, the initial point obtained by the basic strategy can still be selected, resulting in the same performance as the base variant



Where q is the maximum frequency component allowed for the amplitude parameters $\vec{u}, \vec{v}$; R is the Number of random perturbations added to the best parameters found at level p, with the goal of escaping possible local optima in order to find a better one

Figure 10.: Comparison between the basic QAOA-FOURIER method (FOURIER$[q,0]$) and QAOA-FOURIER strategy with R random perturbations (FOURIER$[q,R]$), based on Figure 10. in [18]

For the purpose of adding random permutations to the optimal parameters at level p, Zhou et al. followed this procedure: For the previously determined best p-level optimum $(u^B_{(p)}, v^B_{(p)})$, a randomly perturbated initial point $(u^{R_r}_{(p)}, v^{R_r}_{(p)})$, where R_r is a the random perturbation with the index $r = 0, 1, \cdots, R$, is generated by adding a random permutation such that

$$(u^{R_0}_{(p)}, v^{R_0}_{(p)}) = (u^B_{(p)}, v^B_{(p)}) \tag{A.1}$$

$$(u^{R_r}_{(p)}, v^{R_r}_{(p)}) = (u^B_{(p)} + \alpha u^{P_r}_{(p)}, v^B_{(p)} + \alpha v^{P_r}_{(p)}) \tag{A.2}$$

where $u^{P_r}_{(p)}$ and $v^{P_r}_{(p)}$ are random numbers drawn from the following normal distributions:

$$\left[u^{P_r}_{(p)}\right]_k \sim \text{Normal}\left(0, \left[u^B_{(p)}\right]^2_k\right) \tag{A.3}$$

$$\left[v^{P_r}_{(p)}\right]_k \sim \text{Normal}\left(0, \left[v^B_{(p)}\right]^2_k\right) \tag{A.4}$$

i.e distributions with a mean of zero and a variance that is determined by $u^B_{(p)}$ and $v^B_{(p)}$.

Note that this introduces a new parameter, α, which controls the strength of the perturbations (See Equation A.2). According to Zhou et al., setting $\alpha = 0.6$ consistenly yields good results, however this value was determined by trial and error, so better assignments of α may be possible.

B. Comparison to the Goemans-Willamson Optimizer

This thesis primarily focuses on drawing a comparison between the heuristic parameter selection strategies and the standard, randomly initialized, QAOA method; in other words the goal is evaluating the performance of these strategies relative to the basic variant. However, the reader may also be interested in how these results compare to those produced by classical algorithms, in order to better contextualize the problem instances for which the QAOA strategies were benchmarked.
For this reason, the maximum cuts for these problem instances were computed using the Goemans-Willamson (GW) algorithm [26] which was previously mentioned in Subsection 3.1.2 as the best currently known classical approximation algorithm for MAXCUT. The implementation used for computing these results was the `GoemansWillamsonOptimizer` of the `qiskit.optimization` module[1]. The `num_cuts` parameter GW Optimizer was set to 1, and `sort_cuts` was left in the default setting `True`. What this means is that the GW optimizer was configured to return only the highest cut found[2].
The results of these supplementary benchmarks are presented in the following sections.

B.1. Results for the Problem Instances with Increasing Order

Table B.1 shows the approximation ratios produced for the problem instances with increasing graph order; the corresponding QAOA results can be found in Section 5.2.

When comparing the results for the GW optimizer with the results for QAOA, we find that the QAOA variants can sometimes match the performance of the GW-Optimizer and, in cases where the GW algorithm produces approximation ratios less than 1.0, produce better approximations. However, the majority of QAOA runs produce a lower approximation ratio, meaning we would have to

[1] As with the implementation of the QAOA variants, qiskit.optimization version 0.5.0 was used.
[2] More accurately: only *one* of the highest cuts found, since there are always multiple cuts with the highest cut value (at least two, since we could reverse the assignment of the nodes and get the same cut value).

	6 Nodes	8 Nodes	10 Nodes	12 Nodes	14 Nodes
Graph 1	1.0	1.0	1.0	$0.91\overline{6}$	≈ 0.968
Graph 2	1.0	1.0	1.0	$0.91\overline{6}$	1.0
Graph 3	1.0	$0.\overline{81}$	1.0	≈ 0.957	≈ 0.969

Table B.1.: Goemans-Willamson approximation ratio for the problem instances with increasing number of nodes, which are reported on in Section 5.2

run QAOA mutiple times in order to match/exceed the approximation quality of the GW algorithm. The FOURIER strategy performs better than the basic QAOA and INTERP variants in that regard, since the spread of the individual approximation ratios is narrower, meaning we can match (or even surpass) the GW approximation ratio in a fewer amount of QAOA runs compared to the basic variant or INTERP.

B.2. Results for the Problem Instances with Increasing Size

Table B.2 reports the approximation ratios produced for the problem instances with increasing graph density (see Subsection 4.1.2 for details on how the density relates to graph size); the corresponding QAOA results can be found in Section 5.3.

	0.5 Dense	0.6 Dense	0.7 Dense	0.8 Dense	0.9 Dense
Graph 1	1.0	≈ 0.923	1.0	$0.9\overline{3}$	≈ 0.938
Graph 2	1.0	1.0	1.0	1.0	≈ 0.938
Graph 3	$0.\overline{81}$	$0.91\overline{6}$	1.0	$0.9\overline{3}$	≈ 0.938

Table B.2.: Goemans-Willamson approximation ratio for the problem instances with increasing number of nodes, which are reported on in Section 5.3

When comparing the results for the GW optimizer with the results for QAOA, we find that, similarly to the results presented in Section B.2, the QAOA variants can sometimes match or exceed the approximation quality of the GW algorithm, but more often produce worse approximations, requiring us to run QAOA multiple times if we want to find a cut that is at least the size of the cut produced by GW. This effect is more pronounced here than for the problem instances with increasing graph order, since the QAOA approximation ratio decreases considerably with increasing density, while approximation quality of the GW algorithm appears to be unaffected by such a negative correlation. In other words, high graph densitiess further increase the average number of QAOA runs needed to match or surpass the approximation qualities of GW.

List of Figures

Bibliography

[1] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, arXiv:quant-ph/9508027, ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172.

[2] L. K. Grover, *A fast quantum mechanical algorithm for database search*, arXiv:quant-ph/9605043, Nov. 1996. DOI: 10.48550/arXiv.quant-ph/9605043.

[3] R. Sagastizabal, X. Bonet-Monroig, M. Singh, *et al.*, “Experimental error mitigation via symmetry verification in a variational quantum eigensolver,” *Physical Review A*, vol. 100, no. 1, p. 010 302, Jul. 2019, Publisher: American Physical Society. DOI: 10.1103/PhysRevA.100.010302.

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

[5] M. Cerezo, A. Arrasmith, R. Babbush, *et al.*, “Variational Quantum Algorithms,” *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, Aug. 2021, arXiv:2012.09265 [quant-ph, stat], ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9.

[6] A. Peruzzo, J. McClean, P. Shadbolt, *et al.*, “A variational eigenvalue solver on a quantum processor,” *Nature Communications*, vol. 5, no. 1, p. 4213, Jul. 2014, arXiv:1304.3061 [physics, physics:quant-ph], ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[7] N. Moll, P. Barkoutsos, L. S. Bishop, *et al.*, “Quantum optimization using variational algorithms on near-term quantum devices,” *Quantum Science and Technology*, vol. 3, no. 3, p. 030 503, Jul. 2018, arXiv:1710.01022 [quant-ph], ISSN: 2058-9565. DOI: 10.1088/2058-9565/aab822.

[8] S. Hadfield, Z. Wang, B. O’Gorman, *et al.*, “From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz,” *Algorithms*, vol. 12, no. 2, p. 34, Feb. 2019, arXiv:1709.03489 [quant-ph], ISSN: 1999-4893. DOI: 10.3390/a12020034.

[9] S. Hadfield, T. Hogg, and E. G. Rieffel, “Analytical Framework for Quantum Alternating Operator Ans\"atze,” *Quantum Science and Technology*, vol. 8, no. 1, p. 015 017, Jan. 2023, arXiv:2105.06996 [quant-ph], ISSN: 2058-9565. DOI: 10.1088/2058-9565/aca3ce.

[10] E. Farhi, J. Goldstone, and S. Gutmann, *A Quantum Approximate Optimization Algorithm*, arXiv:1411.4028 [quant-ph], Nov. 2014.

[11] E. Farhi and A. W. Harrow, *Quantum Supremacy through the Quantum Approximate Optimization Algorithm*, arXiv:1602.07674 [quant-ph], Oct. 2019.

[12] E. Farhi, D. Gamarnik, and S. Gutmann, *The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph: Worst Case Examples*, arXiv:2005.08747 [quant-ph], May 2020.

[13] E. Farhi, D. Gamarnik, and S. Gutmann, *The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph: A Typical Case*, arXiv:2004.09002 [quant-ph], Apr. 2020.

[14] J. Wurtz and P. Love, "MaxCut quantum approximate optimization algorithm performance guarantees for $p>1$," *Physical Review A*, vol. 103, no. 4, p. 042 612, Apr. 2021, Publisher: American Physical Society. DOI: 10.1103/PhysRevA.103.042612.

[15] J. Wurtz and D. Lykov, *The fixed angle conjecture for QAOA on regular MaxCut graphs*, arXiv:2107.00677 [quant-ph], Jul. 2021. DOI: 10.48550/arXiv.2107.00677.

[16] X. Lee, Y. Saito, D. Cai, *et al.*, "Parameters Fixing Strategy for Quantum Approximate Optimization Algorithm," en, in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, arXiv:2108.05288 [quant-ph], Oct. 2021, pp. 10–16. DOI: 10.1109/QCE52317.2021.00016.

[17] L. Bittel and M. Kliesch, "Training Variational Quantum Algorithms Is NP-Hard," *Physical Review Letters*, vol. 127, no. 12, p. 120 502, Sep. 2021, Publisher: American Physical Society. DOI: 10.1103/PhysRevLett.127.120502.

[18] L. Zhou, S.-T. Wang, S. Choi, *et al.*, "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices," *Physical Review X*, vol. 10, no. 2, p. 021 067, Jun. 2020, Publisher: American Physical Society. DOI: 10.1103/PhysRevX.10.021067.

[19] S. Khairy, R. Shaydulin, L. Cincio, *et al.*, "Learning to Optimize Variational Quantum Circuits to Solve Combinatorial Problems," en, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2367–2375, Apr. 2020, Number: 03, ISSN: 2374-3468. DOI: 10.1609/aaai.v34i03.5616.

[20] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014, ISSN: 2296-424X.

[21] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, en. Courier Corporation, Jan. 1998, ISBN: 978-0-486-40258-1.

[22] J. L. Gross and J. Yellen, *Handbook of Graph Theory*, en. CRC Press, Dec. 2003, Google-Books-ID: mKkIGIea_BkC, ISBN: 978-0-203-49020-4.

[23] F. Glover, G. Kochenberger, and Y. Du, *A Tutorial on Formulating and Using QUBO Models*, arXiv:1811.11538 [quant-ph], Nov. 2019. DOI: 10.48550/arXiv.1811.11538.

[24] E. Farhi, J. Goldstone, S. Gutmann, *et al.*, *Quantum Computation by Adiabatic Evolution*, arXiv:quant-ph/0001106, Jan. 2000. DOI: 10.48550/arXiv.quant-ph/0001106.

[25] N. Hatano and M. Suzuki, "Finding Exponential Product Formulas of Higher Orders," in vol. 679, arXiv:math-ph/0506007, Nov. 2005, pp. 37–68. DOI: 10.1007/11526216_2.

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

[27] D. Gamarnik, "The Overlap Gap Property: A Geometric Barrier to Optimizing over Random Structures," *Proceedings of the National Academy of Sciences*, vol. 118, no. 41, e2108492118, Oct. 2021, arXiv:2109.14409 [cs, math], ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2108492118.

[28] F. G. S. L. Brandao, M. Broughton, E. Farhi, *et al.*, *For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances*, arXiv:1812.04170 [quant-ph], Dec. 2018. DOI: 10.48550/arXiv.1812.04170.

[29] P. C. Lotshaw, T. S. Humble, R. Herrman, *et al.*, "Empirical performance bounds for quantum approximate optimization," *Quantum Information Processing*, vol. 20, no. 12, p. 403, Dec. 2021, arXiv:2102.06813 [physics, physics:quant-ph], ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-021-03342-3.

[30] V. Akshay, D. Rabinovich, E. Campos, *et al.*, "Parameter concentrations in quantum approximate optimization," *Physical Review A*, vol. 104, no. 1, p. L010401, Jul. 2021, Publisher: American Physical Society. DOI: 10.1103/PhysRevA.104.L010401.

[31] J. Håstad, "Some optimal inapproximability results," *Journal of the ACM*, vol. 48, no. 4, pp. 798–859, Jul. 2001, ISSN: 0004-5411. DOI: 10.1145/502090.502098.

[32] F. Barahona, M. Grötschel, M. Jünger, *et al.*, "An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design," *Operations Research*, vol. 36, pp. 493–513, May 1988. DOI: 10.1287/opre.36.3.493.

[33] *GitHub - lfd/arcs2022*, Sep. 2022.

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

[35] "Quark: A framework for quantum computing application benchmarking," version 1.1.0, *IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 226–237, 2022. DOI: 10.1109/qce53715.2022.00042.

[36] F. J. Kiwit, M. Marso, P. Ross, *et al.*, *Application-Oriented Benchmarking of Quantum Generative Learning Using QUARK*, arXiv:2308.04082 [quant-ph], Aug. 2023. DOI: 10.48550/arXiv.2308.04082.

[37] M. J. D. Powell, "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation," en, S. Gomez and J.-P. Hennart, Eds., Book Title: Advances in Optimization and Numerical Analysis, Dordrecht: Springer Netherlands, 1994, pp. 51–67. DOI: 10.1007/978-94-015-8330-5_4.

[38] M. Powell, "On trust region methods for unconstrained minimization without derivatives," en, *Mathematical Programming*, vol. 97, no. 3, pp. 605–623, Aug. 2003, ISSN: 1436-4646. DOI: 10.1007/s10107-003-0430-6.

[39] R Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2022.

[40] R. Hyndman and Y. Fan, "Sample Quantiles in Statistical Packages," *The American Statistician*, vol. 50, pp. 361–365, Nov. 1996. DOI: 10.1080/00031305.1996.10473566.

[41] R. Herrman, P. C. Lotshaw, J. Ostrowski, *et al.*, *Multi-angle Quantum Approximate Optimization Algorithm*, arXiv:2109.11455 [quant-ph], Sep. 2021. DOI: 10.48550/arXiv.2109.11455.

[42] J. R. McClean, S. Boixo, V. N. Smelyanskiy, *et al.*, "Barren plateaus in quantum neural network training landscapes," en, *Nature Communications*, vol. 9, no. 1, p. 4812, Nov. 2018, Number: 1 Publisher: Nature Publishing Group, ISSN: 2041-1723. DOI: 10.1038/s41467-018-07090-4.

[43] M. Cerezo, A. Sone, T. Volkoff, *et al.*, "Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits," *Nature Communications*, vol. 12, no. 1, p. 1791, Mar. 2021, arXiv:2001.00550 [quant-ph], ISSN: 2041-1723. DOI: 10.1038/s41467-021-21728-w.

[44] C. Ortiz Marrero, M. Kieferová, and N. Wiebe, "Entanglement-Induced Barren Plateaus," *PRX Quantum*, vol. 2, no. 4, p. 040 316, Oct. 2021, Publisher: American Physical Society. DOI: 10.1103/PRXQuantum.2.040316.

[45] S. Bravyi, A. Kliesch, R. Koenig, *et al.*, "Obstacles to State Preparation and Variational Optimization from Symmetry Protection," *Physical Review Letters*, vol. 125, no. 26, p. 260 505, Dec. 2020, arXiv:1910.08980 [cond-mat, physics:quant-ph], ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.125.260505.

[46] D. J. Egger, J. Marecek, and S. Woerner, "Warm-starting quantum optimization," *Quantum*, vol. 5, p. 479, Jun. 2021, arXiv:2009.10095 [quant-ph], ISSN: 2521-327X. DOI: 10.22331/q-2021-06-17-479.

Erklärung

1. Mir ist bekannt, dass dieses Exemplar der Bachelorarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.
2. Ich erkläre hiermit, dass ich diese Bachelorarbeit 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:	Vincent Eichenseher
Student ID:	2061182
Study Programme:	Informatik
Supervisor:	Prof. Dr. Wolfgang Mauerer
Secondary Supervisor:	Prof. Dr. Christian Wolff