Gottfried Wilhelm Leibniz Universität Hannover
Institut für theoretische Physik

Bachelor Thesis
B.Sc. Physics

The Quantum Approximate Optimization Algorithm and the Time-Dependent Variational Principle

by
Leonhard Felix Richter
Student number: 10019579

Supervised by
Prof. Dr. Tobias J. Osborne

Hanover
26.01.2023

Contents

List of Illustrations

Figures

Tables

Summary

The idea of quantum computing was first proposed by Feynman and independently by Benioff in the 1980s [1–3]. Since then, it has been picked up by many scientists and especially over the last years it has also gotten large interest from industrial companies. The current state of the art in the so-called noisy intermediate-scale quantum (NISQ) era are hardware realizations with a low but growing number of qubits that are vulnerable to noise and errors [4]. One field of practical applications is solving optimization problems [5]. Among the more auspicious approaches to such problems are variational quantum algorithms, which are hoped to be beneficial in the near future [6, 7]. A commonly used variational algorithm is the *Quantum Approximate Optimization Algorithm* (QAOA), first introduced by Farhi, Goldstone and Gutmann [8]. The QAOA and variants of it have been tested extensively especially for the Maximum Cut (Max-Cut) problem, which is a prominent example of a combinatorial optimization problem [9–27].

In this thesis, we extend the variants of the QAOA by utilizing the *Time-Dependent Variational Principle* (TDVP) for the parameter optimization of QAOA. This optimization approach recognizes the geometry of the accessible state space and promises provably optimal parameter paths during the optimization process [28]. A method to compute the metric and the gradient of QAOA by quantum circuits is being introduced, a basic complexity result for this method is shown (Chapter 3) and the QAOA with TDVP-optimization is simulated for several Max-Cut-instances (Chapter 4). The simulation results of QAOA with the TDVP-optimization are being compared to those of QAOA with the well-established *Constrained Optimization BY Linear Approximation* (COBYLA) [29–31] and with the standard *gradient-descent* (GD) for parameter optimization.

In summary, the results confirm the TDVP to deliver slightly better results, while the computational resources needed to achieve these results are significantly larger compared to the two established algorithms. In comparison to the gradient descent, the results hint that the TDVP takes a more efficient path and is less likely to shoot over the nearest local minimum due to finite stepsizes.

Chapter 1 gives a short introduction to quantum algorihtms and Chapter 2 explains the two central concepts concerned in this thesis. These concepts get combined in Chapter 3, where a method to compute the metric of QAOA is introduced. Chapter 4 describes the methods and results of numerical simulations.

Chapter 1

Introduction to quantum algorithms

There are different computational models for quantum computing. This thesis sorely relies on the *circuit model* of quantum computation, which is briefly introduced in this chapter. The notions of quantum bits (qubits), gates and measurements, as well as some key differences to classical computation, are presented.

Although this text requires just minimal prior knowledge about quantum computing, it is by far neither a complete nor a coherent introduction to the topic. For more details and further reading, consider for instance Nielsen and Chuang [32] and de Wolf [33].

1.1 The Qubit

The goal of an algorithm is solving some computational problem. In order to perform computations, information needs to be stored in some memory. In classical computation, it is widely common to use *bits* for storing information. Mathematically a bit is not more than a variable with exactly two possible values like $x \in \{0, 1\}$. Variables of this kind are called *binary*, and the terms "bit" and "binary variable" are used equivalently. Given a set of bits, a *register*, an algorithm changes its values according to some predefined rules and their current value. After the algorithm is finished, the given values are its answer.

The quantum analog of a binary variable is a two-dimensional quantum state, called *qubit*, represented as a vector in two-dimensional Hilbert space. Measurements in a two-dimensional Hilbert space give one of two answers with certain probability. These measurement outcomes can also be viewed as the two real eigenvalues of a 2×2 self-adjoint matrix on $\mathbb{C}^2$. Thus, after a measurement was performed, a qubit gives a probability distribution of a binary variable. However, before measurement, the quantum state itself can take more than just two states, as quantum mechanics inherently allow for superposition.

This is one of the main differences between classical and quantum computation. Even if the variables obtained in measurements are classical and binary, the states with which the computation is done are not classical and in a sense non-binary. The goal for the rest of this section is to make this concept more precise.

Definition 1.1. *Consider the two-dimensional Hilbert space $\mathcal{H} = \mathbb{C}^2$, endowed with the usual inner product $\langle\cdot|\cdot\rangle : \ : H \times H \to \mathbb{C}$; $\langle\phi, \psi\rangle = \langle\psi|\phi\rangle = \sum_{j=1}^{2} \bar{\psi}_j \phi_j$. A* qubit *is a two-dimensional quantum state described by some normalized vector $\psi \in \mathbb{C}^2$, $\|\psi\| = 1$.*

Measurements of a qubit are represented by self-adjoint matrices $A \in \mathbb{C}^{2\times 2}$. The two possible measurement outcomes are the two real eigenvalues of A. Here and in the following, we follow Dirac's bra-ket notation [34]. Denote the standard basis vectors of $\mathbb{C}^2$ by $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$. Qubits, being described by vectors in $\mathbb{C}^2$, can be written as linear combination

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \quad |\alpha|^2 + |\beta|^2 = 1, \tag{1.1}$$

where $\alpha, \beta \in \mathbb{C}$. These kinds of linear combinations are often referred to as *superpositions* of the states $|0\rangle$ and $|1\rangle$. An example for a superposition state is $1/\sqrt{3}\,|0\rangle + \sqrt{2}/\sqrt{3}\,|1\rangle$. When expanded in the eigenbasis of the measurement matrix one can directly read of useful information about measurement outcomes. The measurement of the above superposition state will give 0 with a chance of $(1/\sqrt{3})^2 = \frac{1}{3}$ and 1 with a chance of $2/3$. For a general superposition state as in (1.1), the probability for observing $|0\rangle$ is $|\alpha|^2$ and for observing $|1\rangle$ is $|\beta|^2$.[1] It is worth emphasizing again, that the state generally does *not* correspond to either of the two classical states 0 or 1 before measurement.

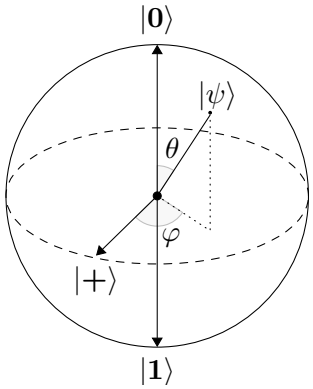


Figure 1.1: The Bloch sphere representation of a qubit

One very useful way to think about qubits is the Bloch sphere representation. As in Equation (1.1) $|\alpha|^2 + |\beta|^2 = 1$ we can write $|\alpha|^2 = \alpha\bar{\alpha} = \cos(\theta/2)$ and $|\beta|^2 = \beta\bar{\beta} = \sin(\theta/2)$. Thus, for $\alpha = |\alpha|e^{i\lambda_1}$ and $\beta = |\beta|e^{i\lambda_2}$ we can write any qubit state as

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\varphi}\sin(\theta/2)\,|1\rangle\,, \tag{1.2}$$

where $\varphi = \lambda_2 - \lambda_1$, and we neglect a global phase factor $e^{i\lambda_1}$ as it is not detectable by measurements. The numbers $\varphi \in [0, 2\pi)$ and $\theta \in [0, \pi)$ can be interpreted as spherical

[1] Here "probability for observing $|0\rangle$" means the expectation value of the projection $|0\rangle\langle 0|$, i.e. $|\langle 0|\psi\rangle|^2$

coordinates of a point on a unit sphere: the *Bloch sphere*. In Figure 1.1 a plot of the Bloch sphere together with some important points is shown. The North Pole corresponds to $|0\rangle$, the South Pole to $|1\rangle$. In between there are superpositions of those. For example for $\theta = \pi/2$ on the equator of the sphere, there are states with equal portions of $|0\rangle$ and $|1\rangle$. They only differ in the phase difference φ. For $\varphi = 0$ in the front, there is the state

$$|+\rangle \coloneqq \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle\right) \tag{1.3}$$

and on the opposite side (not drawn) for $\varphi = \pi$ is the state

$$|-\rangle \coloneqq \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle\right). \tag{1.4}$$

These states form another basis of $\mathbb{C}^2$ known as the *Hadamard basis*.

It is a common misconception that qubits can somehow store more information than classical bits because they have infinitely many possible states. Unfortunately, this is not true. Every information that can be observed lies in the results of measurements, which are classical. "We cannot 'see' a superposition itself, but only classical states."[33, p. 3]

This implies that only one qubit will hardly suffice. Serious algorithms certainly need more than one qubit. While the experimental realization of many connected qubits is one of the biggest challenges quantum computing is facing at the moment [35], the mathematical description of interactions of qubits, is the same as for general composed quantum systems. Multi-qubit systems are mathematically described by the corresponding tensor product of the single-qubit Hilbert spaces. For n qubits we get the Hilbert space $\mathcal{H} = \left(\mathbb{C}^2\right)^{\otimes n} \cong \mathbb{C}^{2^n}$. Note that the complex dimension 2^n grows exponentially with a rising number of qubits.

Take for example $n = 2$. Then, the Hilbert space describing the multi-qubit system is $\left(\mathbb{C}^2\right)^{\otimes 2} \cong \mathbb{C}^4$ and for example one state is given by $|0\rangle \otimes |1\rangle =: |01\rangle$. It describes a state of a two-qubit system, where the first qubit is in state $|0\rangle$ and the second qubit is in state $|1\rangle$. In the standard notation for vectors, it is written as

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{1.5}$$

and builds one of the standard basis vectors of $\mathbb{C}^4$. In general for $\{|e_0\rangle, |e_1\rangle\}$ being a basis of $\mathbb{C}^2$,

$$\{|e_{j_1} \dots e_{j_n}\rangle \mid j_k \in \{0, 1\}\, \forall k \in \{1, \dots, n\}\} \tag{1.6}$$

is a basis of $\left(\mathbb{C}^2\right)^{\otimes n}$. So in the example above $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ is a basis. Note that the labels of the states look very much like binary counting. Indeed, the label of the m-th basis state will be the binary representation of m with $n - m$ leading zeros. This basis is referred to as the *computational basis*.

1.2 Quantum gates

The ability to save and read information does not suffice for computation in the quantum circuit model. Algorithms also need a controllable way to alter qubits. As we have mathematically introduced qubits as vectors in a two-dimensional Hilbert space, the natural objects to describe transformations of qubits are linear mappings defined on this Hilbert space. A transformation of objects should preserve their defining properties, so the selection of valid linear mappings is further restricted to the norm-preserving ones. Norm-preserving linear mappings on finite vector spaces are represented by *unitary* matrices. A square matrix U is unitary if $U^*U = UU^* = \mathbb{I}$, where $(U^*)_{i,j} \coloneqq \overline{U_{j,i}}$ is the adjoint operator. The same applies to multi-qubit systems. Being invertible, unitary matrices are the mathematical realization of reversibly changing the state of a qubit system.

Definition 1.2 (Quantum gates)**.** *A* quantum gate *on n qubits is a unitary $2^n \times 2^n$ complex matrix. Especially for one and two-qubit systems, these matrices are often referred to as* n-qubit gate*.*

Quantum gates replace classical logic gates in quantum circuits. A quantum circuit is essentially a synonym for just any unitary matrix and builds the main ingredient in designing a quantum algorithm. The general procedure of quantum circuit-based algorithms is as follows. Initially, the (multi-)qubit system is prepared in state $|\psi_{\text{in}}\rangle$. Then a quantum circuit U (unitary operator) is applied to change the state of the system to some output state $|\psi_{\text{out}}\rangle = U|\psi_{\text{in}}\rangle$, which is being measured in the last step. The measurement outcome is the result of the algorithm. The key to algorithm design is, to find or construct a circuit that solves the desired computational problem encoded in the initial state of the qubit system. One way to construct circuits is by composing multiple well-known gates into one. In this context, composing quantum gates means simple matrix multiplication of unitaries. The result will again be unitary because

$$(U_1U_2)^*(U_1U_2) = U_2^* \underbrace{U_1^*U_1}_{=\mathbb{I}} U_2 = U_2^*U_2 = \mathbb{I}, \tag{1.7}$$

if U_1 and U_2 are unitary.

Note that in general, there is no difference between a "quantum gate" and a "quantum circuit", as both are defined to be unitary operations. In practice, the difference comes from the way they are constructed. "Quantum gates" are unitary operations that are concretely defined and whose properties and actions on quantum states are well-known. "Quantum circuits" on the other hand are unitary operations that are a priori not well-known or that are designed for a specific purpose. Typically, a quantum circuit is viewed as a sequence of quantum gates chosen from some fixed set. In this perspective, quantum gates are the building blocks of quantum circuits. The distinction also comes from the physical realization of quantum computers, where there is a technical limitation to which unitaries may be applied directly.

1.3 Single qubit gates

In order to make use of the circuit model for quantum algorithms we need to know some quantum gates to begin with.

The simplest gate is the identiy

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1.8}$$

on $\mathcal{H}$, which does only act trivially: $\mathbb{I}\,|\psi\rangle = |\psi\rangle \;\; \forall \psi \in \mathcal{H}$.

Often, quantum gates can be constructed by analogy to classical logic gates. Take for example the logical `NOT` operation. It flips the value of a bit x from 0 to 1 and vice versa: $\texttt{NOT}(0) = 1$, $\texttt{NOT}(1) = 0$. The quantum counterpart to NOT is called the X-gate and is defined by

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{1.9}$$

Note that the matrix representation of quantum gates is basis dependent. In this text we exclusively refer to the computational basis $\{|0\rangle, |1\rangle\}$. Direct computation shows: $X\,|0\rangle = |1\rangle$ and $X\,|1\rangle = |0\rangle$, so on computational basis states X acts like classical `NOT`. For arbitrary qubit states expanded in the computational basis, it swaps the amplitudes of $|0\rangle$ and $|1\rangle$ by linearity:

$$X\left(\alpha\,|0\rangle + \beta\,|1\rangle\right) = \alpha\,|1\rangle + \beta\,|0\rangle\,. \tag{1.10}$$

Two other gates are called very similar: the Y-gate and the Z-gate.

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.11}$$

On computational basis states, Y acts as follows. It adds a phase factor in addition to swapping $|0\rangle$ and $|1\rangle$: $Y\,|0\rangle = i\,|1\rangle$, $Y\,|1\rangle = -i\,|0\rangle$. This also changes the behavior for arbitrary qubit states:

$$Y(\alpha\,|0\rangle + \beta\,|1\rangle) = i(\alpha\,|1\rangle - \beta\,|0\rangle). \tag{1.12}$$

The Z-gate does not act noticeably on computational basis states at all. It will exclusively flip the sign of the amplitude for $|1\rangle$ and leaves $|0\rangle$ invariant. For arbitrary qubit states this results in:

$$Z(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\,|0\rangle - \beta\,|1\rangle\,. \tag{1.13}$$

Note that the Z-gate swaps $|-\rangle$ and $|+\rangle$ just the way as X swaps $|0\rangle$ and $|1\rangle$. This is an example of the basis dependency of the representations we choose. We could choose the Hadamard states $\{|-\rangle, |+\rangle\}$ instead of $\{|0\rangle, |1\rangle\}$ as a basis and the roles of X and Z would swap.

A more intuitive and visual description of these three gates becomes clear when exponentiating them. Then we obtain the rotation matrices with angle θ around the x-, y- and z-axis, respectively [32].

$$R_X(\vartheta) = e^{-i\vartheta X/2} = \cos\left(\frac{\vartheta}{2}\right)\mathbb{I} - i\sin\left(\frac{\vartheta}{2}\right)X = \begin{pmatrix} \cos\left(\frac{\vartheta}{2}\right) & -i\sin\left(\frac{\vartheta}{2}\right) \\ -i\sin\left(\frac{\vartheta}{2}\right) & \cos\left(\frac{\vartheta}{2}\right) \end{pmatrix} \tag{1.14}$$

$$R_Y(\vartheta) = e^{-i\vartheta Y/2} = \cos\left(\frac{\vartheta}{2}\right)\mathbb{I} - i\sin\left(\frac{\vartheta}{2}\right)Y = \begin{pmatrix} \cos\left(\frac{\vartheta}{2}\right) & -\sin\left(\frac{\vartheta}{2}\right) \\ \sin\left(\frac{\vartheta}{2}\right) & \cos\left(\frac{\vartheta}{2}\right) \end{pmatrix} \tag{1.15}$$

$$R_Z(\vartheta) = e^{-i\vartheta Z/2} = \cos\left(\frac{\vartheta}{2}\right)\mathbb{I} - i\sin\left(\frac{\vartheta}{2}\right)Z = \begin{pmatrix} e^{-i\vartheta/2} & 0 \\ 0 & e^{i\vartheta/2} \end{pmatrix} \tag{1.16}$$

In fact, the gates $R_X(\vartheta)$, $R_Y(\vartheta)$ and $R_Z(\vartheta)$ map a qubit state with some representation on the Bloch sphere (Figure 1.1) to the qubit state corresponding to the point on the Bloch sphere that is rotated by the angle ϑ about the respective axis [32]. The gates X, Y and Z are special cases of these for $\vartheta = \pi$ and therefore act like 180°-rotations in the Bloch sphere representation.

X, Y and Z are commonly known as the *Pauli matrices* and written as σ_x, σ_y and σ_z in physics and mathematics. Being 180°-rotations means, that applying them twice is just the identity. This can also be seen by their definitions as direct calculations show, that the Pauli matrices are both self-adjoint and unitary. On the other hand, products of two distinct Pauli matrices σ_j, σ_k for $j \neq k$ show another structure. Then, the Pauli matrices do not commute but form the *Pauli algebra* characterized by the commutator relation

$$[\sigma_j, \sigma_k] = 2i\sum_{l=0}^{3} \epsilon_{j,k,l}\sigma_k, \quad j,k \in \{0,1,2,3\}\,, \tag{1.17}$$

where $\sigma_0 \coloneqq \mathbb{I}$ and $\epsilon_{j,k,l}$ is the Levi-Civita symbol. Note the similarity to the commutator relation of rotational momenta L_j, $j = 0,1,2,3$ in classical mechanics, where

$$[L_j, L_k] = \sum_{l=0}^{3} \epsilon_{j,k,l}L_k \quad \text{for} \quad j,k \in \{0,1,2,3\}\,. \tag{1.18}$$

Another very important single qubit gate is the Hadamard gate H. It is defined as

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.19}$$

and acts like a change of basis from the computational basis to the Hadamard basis. For example, when acting on computational basis states, it creates superposition states [32]:

$$H\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \tag{1.20}$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle\,. \tag{1.21}$$

On the other hand, when acting on a Hadamard state it turns back to a computational basis state:

$$H|+\rangle = |0\rangle \tag{1.22}$$
$$H|-\rangle = |1\rangle\,. \tag{1.23}$$

In the same way, as for the Pauli matrices, the Hadamard gate is self-adjoint and unitary so $H^2 = \mathbb{I}$. It can also be visualized on the Bloch sphere as a 90° rotation around the y axis followed by a 180° rotation around the x axis [32].

1.4 Multi-qubit gates and circuits

A single qubit operation U on the j-th qubit of an n-qubit system is given by

$$U_j := \mathbb{I}^{\otimes(n-j)} \otimes U \otimes \mathbb{I}^{\otimes n}. \tag{1.24}$$

However, there is not only the possibility to act on each qubit individually, but there are also quantum gates that act non-trivially on more than one qubit at the time. One of such is the *controlled-not gate* `CNOT`. It acts on two qubits, called the *control qubit* and the *target qubit*. On computational basis states, depending on the state of the control qubit, the target qubit is flipped or not:

$$\texttt{CNOT}\,|c,t\rangle = |c, t \oplus c\rangle\,, \tag{1.25}$$

where $\oplus$ denotes addition $\mod 1$. The corresponding matrix representation

$$\texttt{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.26}$$

is unitary and hence `CNOT` is a quantum gate. It may seem like the control qubit is always left invariant by this gate. However, this is not the case for superposition states. For example in the Hadamard basis, the role of control and target qubit get swapped entirely [32]. By a direct calculation, the action of `CNOT` can be viewed as computing the parity of the qubits. For computational basis states the target qubit is set either to 0 or to 1, depending on whether both qubits are in the same state or not.

At this point, it is convenient to introduce a visual notation for quantum circuits. Each qubit corresponds to a horizontal line.

$$|x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle = \begin{array}{l} |x_1\rangle \text{ ——} \\ |x_2\rangle \text{ ——} \\ |x_3\rangle \text{ ——} \end{array} \tag{1.27}$$

Qubit regesiters, i.e. collections $|x_1\rangle \otimes \cdots \otimes |x_n\rangle$ of a possibly large number of qubits, are often drawn as

$$|x_1\rangle \otimes \cdots \otimes |x_n\rangle \quad \text{⟨thick wire⟩} \,. \tag{1.28}$$

A sequence $U_4U_3U_2U_1 |x\rangle$ of gates applied one after the other is drawn from left to right.

$$|x\rangle \;\text{—}\boxed{U_1}\text{—}\boxed{U_2}\text{—}\boxed{U_3}\text{—}\boxed{U_4}\text{—} \tag{1.29}$$

Controlled gates are drawn by vertical lines between the control and target qubits. For example, the `CNOT`-gate is drawn as

$$\begin{array}{l} |c\rangle \\ |t\rangle \end{array} \quad \text{⟨control dot, target ⊕⟩} \,.$$

Another two-qubit gate is the `SWAP`-gate. As its name suggests, it simply swaps the states of the two qubits it acts on: $\texttt{SWAP}\,|\psi\rangle \otimes |\phi\rangle = |\phi\rangle \otimes |\psi\rangle$. Its matrix representation is

$$\texttt{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.30}$$

and it is drawn as

$$\text{⟨SWAP⟩} \;=\; \text{⟨three CNOTs⟩} \tag{1.31}$$

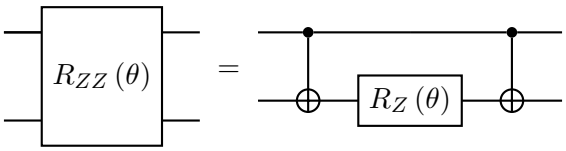
in quantum circuit diagrams, where we have used that SWAP can be decomposed into three `CNOT` gates [32]. While swapping of qubits is possible, creating a copy of an arbitrary unkown qubit state in a unitary way is proven to be impossible. This result is known as the "No-Cloning-Theorem" and demonstrates that quantum computing works fundamentally different compared to classical computation [32].

In Section 2.1, we need a two-qubit version of the single qubit rotation gates (1.14)–(1.16). They are defined as the exponential of a double Pauli matrix like $Z \otimes Z$,

$$R_{ZZ}(\theta) = e^{-i\theta Z\otimes Z/2} = \begin{pmatrix} R_Z(\theta) & 0 \\ 0 & R_Z(-\theta) \end{pmatrix} = \begin{pmatrix} e^{-i\theta/2} & 0 & 0 & 0 \\ 0 & e^{i\theta/2} & 0 & 0 \\ 0 & 0 & e^{i\theta/2} & 0 \\ 0 & 0 & 0 & e^{-i\theta/2} \end{pmatrix}, \tag{1.32}$$

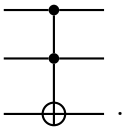
where we used block-matrix notation. R_{XX} and R_{YY} are defined analougusly as the exponential of $X \otimes X$ and $Y \otimes Y$. These gates can be represented by a single qubit

rotation and two CNOT gates as a phase factor is applied only depending on the parity of the two qubits [32].



(1.33)

There are also three-qubit gates. For example the Toffoli-gate



(1.34)

On computational basis states, it flips the third qubit exactly when the first and the second qubit are in state $|1\rangle$. Its most significance lies in its ability to simulate any reversible classical computation, proving that quantum circuits can perform any computation that classical computational models can perform [32].

Note that the SWAP-gate as well as the R_{XX}, R_{YY} and R_{ZZ} gates can be composed by CNOT-gates and single qubit rotations (Equations (1.31) and (1.33)). As discussed in Section 1.2, this kind of decomposition into a set of well-known gates that are available on hardware is an important concept. This lead to the question of how many and which different building blocks are necessary to construct every unitary matrix, i.e. every circuit. In classical computation one single operation, the classical Toffoli gate, already suffices for this task [36]. Such a set of operations is called *universal*. In quantum computation, arbitrary single qubit gates together with the CNOT-gate suffice to construct every quantum circuit [37]. Thus, the set $\{R_X, R_Y, \texttt{CNOT}\}$ forms a universal family of quantum gates as two rotation matrices allow for arbitrary single-qubit unitaries [32]. In this thesis, the available gates are assumed to be at least single qubit rotations R_X, R_Y, R_Z and CNOT.

Chapter 2

Key concepts

Building on the background on quantum algorithms, given in Chapter 1, this chapter discusses the two key concepts considered in this thesis. First, the central quantum algorithm considered in the present work, namely the *Quantum Approximate Optimization Algorithm* (QAOA) is explained in Section 2.1 in detail. Secondly, the *Time-Dependent Variational Principle* is introduced in Section 2.2.

2.1 The Quantum Approximate Optimization Algorithm (QAOA)

This section presents the *Quantum Approximate Optimization Algorithm* (QAOA) which builds the center of this thesis. First, two theorems that serve as motivation for the algorithm are presented in Section 2.1.1. In Section 2.1.2, the algorithm is explained from a general perspective, while its most prominent application for combinatorial optimization problems is discussed in Section 2.1.3. A concrete example is given by the MAX-CUT problem in Section 2.1.4. For further details on the QAOA, we refer to Farhi, Goldstone and Gutmann [8] and Hadfield [12].

2.1.1 Underlying theorems

There are two theorems in quantum theory that are important for understanding QAOA. The first one is about the dynamics of eigenspaces under "slow" time evolution. This theorem is known as the *quantum adiabatic theorem*. The second theorem is called the *Lie product formula* and generalizes the identity $e^{x+y} = e^x e^y$ for $x, y \in \mathbb{R}$ to operators on Hilbert space.

The Adiabatic theorem considers a time dependent hamiltonian $H(s)$ where the time dependency is only due to scaled time $s = t/T$. t is the ordinary time parameter of quantum time evolution, $T > 0$ some number and s is the fraction of time between the initial time and T. The *time-dependent Schrödinger equation* then can be written with respect to scaled time s in the form

$$i\partial_s \psi(s) = TH(s)\psi(s). \tag{2.1}$$

Denote the spectrum of the Hamiltonian $H(s)$ by $\sigma(s)$ for every $s \in [0,1]$. Assume that the set $\left\{(s,\lambda) \in \mathbb{R}^2 \;\middle|\; s \in [0,1], \lambda \in \sigma(s)\right\}$ has a region which is separated from the rest

of the spectrum by some spectral gap $s \mapsto g(s)$. The general statement of the adiabatic theorem is that for large T a system governed by $H(s)$ stays in the eigenspace of that separated band with high probability. This means that the eigenspace of $H(s)$ can be approximated by the corresponding eigenspace of $H(0)$ evolved by the time evolution given by Equation (2.1). In the following, we approximate ground states of $H(1)$ by evolving ground states of $H(0)$.

There are several precise formulations of the adiabatic theorem in quantum mechanics, some being stronger than others. Here, we only state one of these theorems in order to give a motivation for the QAOA. The subsequent theorem follows Jansen, Ruskai and Seiler [38].

Theorem 2.1 (Adiabatic theorem)**.** *Denote the unitary time evolution given by* (2.1) *by $U_T(s)$ and let $H(s)$ be a twice continuously differentiable map from $[0,1]$ to bounded, self-adjoint operators on a fixed domain. Let there be continuous real functions b_+ and b_- and a number $g > 0$ such that*

$$\operatorname{dist}\left(\{b_+(s), b_-(s)\}, \sigma(s)\right) > g \quad \text{for} \quad s \in \{0,1\}, \tag{2.2}$$

where $\sigma(s)$ is the spectrum of $H(s)$ and $\operatorname{dist}(A,B) = \inf_{x\in A, y\in B}(\|y-x\|)$ is the minimal distance between two sets $A, B \subset \mathbb{R}^n$. When $P(s)$ is the projector on the separated band $\sigma(s) \cap [b_-(s), b_+(s)]$ and $P_T(s) = U_T(s)P(0)U_T(s)^*$, then for $\psi \in \operatorname{Im} P(0)$

$$\langle U_T(s)\psi | (\mathbb{I} - P(s))U_T(s)\psi\rangle \in O(1/T^2) \tag{2.3}$$

$$\|P_T(s) - P(s)\| \in O(1/T), \tag{2.4}$$

where $O(h)$ is the set of all functions f s.t. $f(x) < ch(x)$, for $x > x_0$, $c \in \mathbb{R}$.

The main application of the adiabatic theorem in the context of quantum computing is to consider some convex combination $H(s) = (1-s)H_B + sH_P$. It is useful to choose H_B to be some simple Hamiltonian with exact knowledge of the spectrum while H_P is the Hamiltonian of interest with little to no knowledge of the spectrum or eigenspaces. For example, H_P could encode some kind of combinatorial optimization problem (Section 2.1.3). In such scenarios, the adiabatic theorem provides an algorithm to approximately compute the spectrum of H_P and thereby find solutions to a given combinatorial optimization problem [39–41].

The Lie product formula is the second basic result that motivates the QAOA. It provides a way to make use of the *adiabatic theorem* on a universal quantum computer with low circuit depth. Recall that a quantum algorithm is just some unitary matrix U acting on a collection of input qubits and measuring some output qubits. For either implementing this algorithm on quantum hardware or analyzing it in more detail we need to rewrite it as a sequence of basic quantum gates like one or two-qubit Pauli-rotations $R_X, R_Z, R_{ZZ}, \ldots$ (Chapter 1). Generally, this compilation has proven to be a hard problem on its own [42]. In the case that the unitary is generated by a Hamiltonian

$H = H_B + H_P$ and takes the form $U = e^{-iH}$, the Lie product formula may help to compile the circuit.

For the exponential on the reals, there is a simple identity for expanding such expressions: $e^{x+y} = e^x e^y$. In fact, this relation still holds for the matrix exponential in the case of *mutually commuting matrices* A, B. Then $e^{A+B} = e^A e^B$ is true.[1] However, for example the Pauli matrices do not commute (Equation (1.17)), so that the exponential of their sum does not split as nicely. In this case, the *Lie-product formula* provides at least an approximate splitting.

Theorem 2.2 (Lie product formula)**.** *Let $A, B \in \mathbb{C}^{n\times n}$ be two matrices. Then*

$$e^{A+B} = \lim_{p\to\infty} (e^{A/p} e^{B/p})^p. \tag{2.5}$$

Proof. A proof of a more general result for self-adjoint operators on Hilbert space is given by Nielsen and Chuang [32, Theorem 4.3, p. 207]. □

The process of splitting a term like e^{A+B} into $(e^{A/p}e^{B/p})^p$ for a finite p is commonly called *Trotterization*, after the *Lie-Trotter product formula*, which is a generalization for semigroups [43].

2.1.2 The general principle

Combining the adiabatic theorem (Theorem 2.1) and the Lie product formula (Theorem 2.2) gives reason to formulate the QAOA. Despite the "Optimization" in the name of the algorithm, we define the algorithm in the sense of finding ground states of a given Hamiltonian first. This slightly more general formulation benefits in a better understanding of the basic idea of the algorithm. In Section 2.1.3, we show how it can be used as a quantum heuristics for solving combinatorial optimization problems.

As the adiabatic theorem formalizes, the main idea is to slowly shift from a well-understood, simple Hamiltonian H_B to the Hamiltonian H_P, we are interested in. The system is initially prepared in a ground state of H_B. By shifting slowly enough towards H_P we hope that the system remains in a ground state ending up in a ground state of H_P, due to the adiabatic theorem.

Assume $H(s) = (1-s)H_B + sH_P$ to have a finite minimal spectral gap $g > 0$ between the instantaneous ground states of $H(s)$ and the rest of its spectrum. We want to evaluate the time evolution of the initial state $|\psi_0\rangle$ under $H(s)$. By the adiabatic theorem, $e^{-isH(s)} |\psi_0\rangle$ has big overlap with the eigenspace of the groundstate energy of $H(s)$ for large enough T. For $s = 1$ i.e. $t = T$, we get high overlap with the desired eigenspace of the groundstate energy of H_P [39, 40, 44].

[1] This is easily shown by using the Cauchy product and the binomial expansion of $(A + B)^n$ for $[A, B] = 0$

This relies on *slowly evolving* under $H(s)$, i.e. on T being large enough. What is regarded as T being "large enuogh" depends on several factors, but is dominated by the minimal spectral gap g [39].

There are so-called *quantum annealers*, which can perform such evolutions for certain H_P [45]. However, evolving continuously under a time-dependent Hamiltonian is not possible on universal quantum computers.[2] In order to implement the quantum adiabatic approach on a universal quantum computer, we need to discretize it into a series of gates. Therefore, we apply the Lie product forumla (Theorem 2.2). It approximates

$$e^{-isH(s)} \approx \left(e^{-i\,(1-s)/p\,H_B} e^{-i\,s/p\,H_P}\right)^p, \tag{2.6}$$

for some $p \in \mathbb{N}$, which serves as a motivation for considering

$$|\beta,\gamma\rangle_p \coloneqq U^{(p)}\,(\beta,\gamma)\,|\psi_0\rangle \coloneqq e^{-i\,\beta_p\,H_B} e^{-i\,\gamma_p\,H_P} \ldots e^{-i\,\beta_1\,H_B} e^{-i\,\gamma_1\,H_P}\,|\psi_0\rangle\,, \tag{2.7}$$

where $\beta,\gamma \in \mathbb{R}^p$. There is a lack of clarity in changing from the same $(1-s)/p$ and s/p for all repetitions to different numbers β_j and γ_j. Equation (2.6) only serves as motivation to consider the following definition.

Definition 2.1. *Let $p \in \mathbb{N}$ and*

$$\mathbb{R}^{2p} \ni (\beta_1,\ldots,\beta_p,\gamma_1,\ldots,\gamma_p) = \delta \cong (\beta,\gamma) \in (\mathbb{R}^p)^2. \tag{2.8}$$

Denote the gates $U_P\,(\gamma_j) \coloneqq e^{-i\gamma H_P}$ and $U_B(\beta_j) \coloneqq e^{-i\beta H_B}$ as QAOA-gates *and one pair $U_B\,(\beta_j)\,U_P\,(\gamma_j)$ as a* QAOA-block *for $1 \le j \le p$. The circuit*

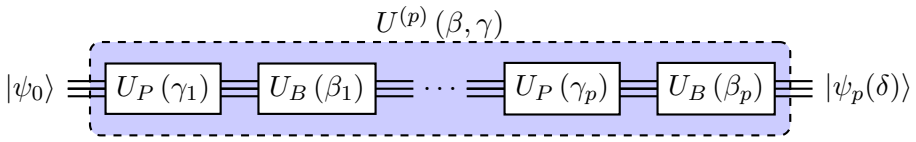
$$U^{(p)}\,(\delta) \coloneqq U^{(p)}\,(\beta,\gamma) \coloneqq U_B\,(\beta_p)\,U_P\,(\gamma_p) \ldots U_B\,(\beta_1)\,U_P\,(\gamma_1) \tag{2.9}$$

is called QAOA-circuit *and δ, β and γ are called* QAOA-parameters *respectively. States of the form*

$$|\psi_p(\delta)\rangle \coloneqq |\beta,\gamma\rangle_p = U^{(p)}\,(\delta)\,|\psi_0\rangle \tag{2.10}$$

are called QAOA-states.

The whole procedure can be illustrated as follows:



(2.11)

The remaining task is to choose suitable β and γ. The intuition is that each β_j and γ_j defines how fast the evolution should be at the given step. Recall that in order for

[2] Of course as $e^{-isH(s)}$ is unitary one can run it on a quantum computer for some fixed s. The problem here is to change this s continuously.

the adiabatic theorem 2.1 to apply, the evolution should take place sufficiently "slow". Wherever the spectral gap is small, the evolution should be especially slow to prevent tunneling in an excited state. So on the one hand, the corresponding QAOA-parameters need to be chosen accordingly. On the other hand, if *all* QAOA-parameters are very small the last QAOA-parameters need to be very high in order to terminate at the equivalence of $s = 1$. This would result in a high probability to tunnel in an excited state. However, the spectrum of $H(s)$ is unkown for $s \neq 0$ so suitable parameters need to be determined by minimizing the energy expectation of the system prepared in the QAOA-state. The optimal parameters in this sense are denoted by $\delta^* = (\beta^*, \gamma^*) \in \mathbb{R}^{2p}$, which are yet to be found. Then, the state $|\beta^*, \gamma^*\rangle$ has ideally large overlap with the eigenspace of the ground state energy of H_P. Measuring it gives knowledge about ground states of H_P.

For the QAOA usually a classical optimization algorithm (*optimizer*) is used. There are several of those, for example, the *Constrained Optimization BY Linear Approximation* (COBYLA) [46]. The concrete function being optimized is the expectation

$$f_p(\beta, \gamma) = {}_p\langle \beta, \gamma | H_P | \beta, \gamma \rangle_p \tag{2.12}$$

of H_P in state $|\beta, \gamma\rangle_p$. In each optimization step of the optimizer the state $|\beta, \gamma\rangle_p$ gets prepared and measured, repeatedly. As the probability for the measurement outcome to be $|x\rangle$ is $|\langle x|\beta, \gamma\rangle_p|^2$, the repeated measurement allows sampling of the QAOA-state $|\beta, \gamma\rangle_p$ [47]. Subsequently, the product ${}_p\langle \beta, \gamma | H_P | \beta, \gamma \rangle_p$ can be estimated easily for suitable H_P. According to the optimizer, β and γ get adjusted and the whole process gets repeated until some condition is met. Classical general-purpose optimization algorithms usually do not respect the restriction of possible states to the ones parametrized by $|\beta, \gamma\rangle_p$ properly. The classical optimizer may be substituted by a partly quantum algorithm based on the *Time-Dependent Variational Principle* (Section 2.2). This is the approach in this thesis.

But for now, define the QAOA in a compact way first. The general procedure is to approximate the states that can be reached by evolving $|\psi_0\rangle$ under $e^{-isH(s)}$ by the parametrization given in Equation (2.7) and to find the ones with the lowest expectation under H_P. This is done by a suitable optimization algorithm. The only role of quantum computing is boosting the evaluation of the expectation values by preparing the parametrized state on the quantum computer instead of doing long matrix multiplication. This procedure will be summarized and properly defined in the subsequent definition, following Hadfield et al. [47].

Definition 2.2 (QAOA)**.** *Given Hamiltonians H_P and H_B such that e^{-isH_P} and e^{-isH_B} can be implemented efficiently, a groundstate $|\psi_0\rangle$ of H_B and some $p \in \mathbb{N}$ define for every $\beta, \gamma \in \mathbb{R}^p$ the* trial state $|\beta, \gamma\rangle_p$ *like in Equation* (2.7)*. Furthermore, let O be some classical optimization algorithm, that given a function $f\colon (\mathbb{R}^p)^2 \to \mathbb{R}$, approximates the input (β^*, γ^*) such that $f(\beta^*, \gamma^*)$ is minimal: $O(\beta, \gamma, f) = (\beta', \gamma')$ s.t. $\|(\beta^*, \gamma^*) - (\beta', \gamma')\| < \epsilon$. Then QAOA is defined as follows.*

1. *Initialize parameters* β, γ.
2. $(\beta', \gamma') = O(\beta, \gamma, f_p)$, *where* f_p *is evaluated as follows:*
   (a) *Prepare initial state* $|\psi_0\rangle$.
   (b) *Apply* $e^{-i\beta_p H_B} e^{-i\gamma_p H_P} \dots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_P}$ *to prepare* $|\beta, \gamma\rangle_p$.
   (c) *Sample* $|\beta, \gamma\rangle_p$ *by repeating 2a, 2b and measurement.*
   (d) *Compute and return expectation value* $f_p(\beta, \gamma)$.
3. *Sample* $|\beta', \gamma'\rangle_p$ *by repeating 2a, 2b and measurement.*

One may ask what benefit may lie in this procedure as in the end a classical optimizer is used. Whether QAOA offers some provable complexity benefit over classical optimization algorithms is still an open research question, but one possible benefit is the following. The QAOA allows the computation of the expectation value ${}_p\langle\psi|H_C|\psi\rangle_p$ of a given cost function C using a quantum circuit. Furthermore, it reduces the domain of the cost function handed over to the optimization algorithm from the Hilbert space $\mathcal{H} = \mathbb{C}^{2^n} \cong \mathbb{R}^{4^n}$ to a parameter set in $\mathbb{R}^{2p}$. For large n or small p respectively, this reduces the dimension of the search space of the optimization algorithm. This comes at the cost of also reducing the state space from $\mathcal{H}$ to some subset of states, that are given by Equation (2.7) for some $\delta \in \mathbb{R}^p$. Only in the limit $p \to \infty$ the Trotterization can generate every vector in $\left\{e^{-iH(s)} \;\middle|\; s \in \mathbb{R}\right\}$. So in general there is a trade-off between more accuracy for higher p and a smaller search space for smaller p.

2.1.3 QAOA for combinatorial optimization problems

In the preceding Section 2.1.2 the general principle of QAOA is introduced in the context of finding ground states of a given Hamiltonian. This section covers how this algorithm can be used for solving combinatorial optimization problems.

Combinatorial optimization problems have a wide variety of applications and many other problems can be reformulated in this form. Famous examples are the *Traveling Salesperson Problem* (TSP), the *Knapsack problem* and the *Maximum Cut problem* (Max-Cut). They all have in common that some function on a discrete domain should be minimized.

Definition 2.3 (Combinatorial optimization problem)**.**

- *For each problem instance let* S *be the discrete set of feasible solutions and* $f\colon S \to \mathbb{R}$ *the cost function. Find* $x \in S$ *such that* $f(x)$ *is minimal.*[3]

[3]Note that maximizing f is equivalent to minimizing $-f$, so maximization problems are also included in this definition.

Problems of this kind are called combinatorial optimization problems. Denote an optimal solution by x^* with value $f^* = f(x^*)$.

- *A* binary combinatorial optimization *problem is a combinatorial optimization problem, where $S \subset \{0,1\}^n$ for some $n \in \mathbb{N}$.*

We focus on the latter. Every binary function f can be expressed in the form

$$f(x) = C + \sum_j C_j x_j + \sum_{j,k} C_{j,k} x_j x_k + \sum_{j,k,m} C_{j,k,m} x_j x_k x_m + \dots \tag{2.13}$$

and the coefficients $C, C_{j,k}, C_{j,k,m}, \dots$ can be computed by Fourier transform [48]. Binary functions that can be expressed as a polynomial like in Equation (2.13) with a degree of at most 2 often suffice to model many interesting optimization problems in diverse contexts and give rise to the following definition.

Definition 2.4 (Quadratic Unconstrained Binary Optimization (QUBO))**.** *A Combinatorial Optimization Problem associated with a binary polynomial of degree at most 2 is called* Quadratic Unconstrained Binary Optimization *(QUBO) problem.*

Sometimes it is more convenient to avoid the binary polynomial in the form of Equation (2.13) and replace it by a symmetric (or upper triangular) matrix $Q \in \mathbb{R}^{n \times n}$ as

$$f(x) = x^T Q x + C \quad (x \in \mathbb{R}^n) \tag{2.14}$$

Thus, diagonal of Q consists of the linear coefficients C_j and the (non-zero) off-diagonal elements correspond to the quadratic coefficients $C_{j,k}$ $(2 \cdot C_{j,k})$.

QAOA approximately solves combinatorial optimization problems with corresponding cost function f. This can be achieved by using the algorithm to find a ground state of a specific Hamiltonian H_f as described in Section 2.1.2. On computational basis states, H_f is defined as the multiplication operator with f that "simulates" the cost function in the sense that it acts like

$$H_f |x\rangle = f(x) |x\rangle \tag{2.15}$$

on computational basis states $|x\rangle$. It is self-adjoint, since f is real valued.

Definition 2.5. *For a function $f\colon S \to \mathbb{R}$ denote the Hamiltonian satisfying Equation* (2.15) *by H_f and say H_f* represents *the function f.*

By definition (2.15) ground states of H_f correspond to solutions x^* minimizing f. For example, for functions of the form (2.13), the Hamiltonian representing it is obtained by exploiting the following Lemma due to Hadfield [12, 48].

Lemma 2.1. *Let f, g be binary functions and $x \in \{0,1\}$ be a binary variable. Then, the following identities hold:*

(a) $H_{x\mapsto x} = \frac{1}{2}(\mathbb{I} - Z)$

(b) $H_{x\mapsto \neg x} = \frac{1}{2}(\mathbb{I} + Z)$

(c) $H_{\lambda_2 f+\lambda_2 g} = \lambda_1 H_f + \lambda_2 H_g, \quad \lambda_1, \lambda_2 \in \mathbb{R}$

Proof. See Hadfield [48]. □

This immediately allows the representation of QUBO functions when given explicitly in terms of constant, linear and quadratic monomials.

Lemma 2.2. *The Hamiltonian, which represents a binary polynomial*

$$f(x) = C + \sum_j L_j x_j + \sum_{j<k} Q_{j,k} x_j x_k \tag{2.16}$$

of degree 2, is given by

$$H_f = (C - L + Q)\mathbb{I} - \frac{1}{2}\sum_j (L_j + Q_j) Z_j + \frac{1}{4}\sum_{j<k} Q_{j,k} Z_j Z_k, \tag{2.17}$$

where $L = \sum_j L_j$, $Q_j = \sum_{k:\, k\neq j} Q_{j,k}$ *and* $Q = \sum_{j,k} Q_{j,k}$.

Proof. Substitute $x_j \mapsto \frac{1}{2}(\mathbb{I} - Z_j)$ and expand the arising products $(\mathbb{I} - Z_j)\cdot(\mathbb{I} - Z_k)$. □

Remark 2.5.1. *The same procedure leads to similar expressions*

$$H_f = \tilde{C} + \sum_j \tilde{C}_j Z_j + \sum_{j,k} \tilde{C}_{j,k} Z_j Z_k + \sum_{j,k,m} \tilde{C}_{j,k,m} Z_j Z_k Z_m + \dots \tag{2.18}$$

for general functions as in (2.13)*. But for sake of simplicity, we restrict to QUBO functions.*

For implementing QAOA the unitary evolution generated by H_f needs to be compiled into a sequence of known quantum gates. The form of H_f given in Equation (2.17) allows exactly that.

Lemma 2.3. *Let* $f(x) = C + \sum_j L_j x_j + \sum_{j<k} Q_{j,k} x_j x_k$. *Then*

$$e^{-i\gamma H_f} = \prod_j R_{Z;j}(-\gamma C_j) \prod_{j,k} R_{ZZ;j,k}\left(\frac{1}{4}\gamma Q_{j,k}\right), \tag{2.19}$$

where again $C_j = L_j + Q_j = L_j + \sum_{k:\, k\neq j} Q_{j,k}$, $R_{Z;j}(\omega) = e^{-i\omega Z_j/2}$ *and* $R_{ZZ;j,k}(\omega) = e^{-i\omega Z_j Z_k/2}$ *(Equation* (1.24)*).*

Proof. Note that $e^{A+B} = e^A e^B$ for commuting matrices A and B. With this in mind Equation (2.19) can be computed directly [32]. □

We still lack a concrete choice of H_B (2.7) and $|\psi_0\rangle$. The most basic and commonly used possibility is

$$H_B = \sum_j X_j. \tag{2.20}$$

Its ground state is of the Hadamard states

$$|\psi_0\rangle = |-\rangle = \frac{1}{\sqrt{2}^n} \left(|0\rangle - |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle - |1\rangle\right) \tag{2.21}$$

and the corresponding unitary evolution can be implemented by X-rotations on each qubit:

$$e^{-i\beta H_B} = \prod_j R_X\left(2\beta\right). \tag{2.22}$$

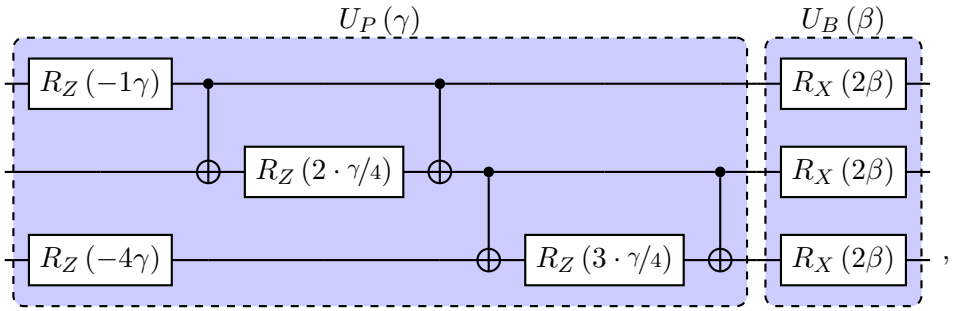
Thus, together with Definition 2.2 we have given complete instructions for implementing QAOA for apprxoimately solving combinatorial optimization problems including QUBO problems. When considering the final state $|\psi_p(\delta')\rangle$ of QAOA as probability distribution

$$\text{prob}(x) = \left|\langle x|\psi_p(\delta')\rangle\right|^2 \tag{2.23}$$

of classical bitstrings, the answer of the QAOA to the given problem is given by the bitstring x_1 with highest probability $\text{prob}(x_1)$. With these definitions and for the QUBO-instance given by

$$Q = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 0 & 3 \\ 0 & 3 & 4 \end{pmatrix} \tag{2.24}$$

one QAOA-block $U_B\left(\beta\right) U_P\left(\gamma\right)$ looks like



(2.25)

where R_{ZZ} gates are implemented according to Equation (1.33).

Having explicit expressions for H_B and H_P another non-formal intuition for the QAOA circuit can be given putting it in the perspective of another more general algorithm. Note

that the R_X rotations create superposition states when acting on computational basis states. Thus, following Hadfield et al. [47], $e^{-i\beta H_B}$ is called a *mixing-operator* as it mixes between $|0\rangle$ and $|1\rangle$. The R_Z rotations on the other hand only change the phases of each basis state in the superposition. By including the coefficients of the problem function the separate rotations get weighted according to the function. Thus, following Hadfield et al. [47], $e^{-i\gamma H_P}$ is called *phase-separation operator* as it separates the phases according to penalty values given by the function of the problem. Then, the algorithm mixes the qubits and weighs their amplitudes according to the cost function repeatedly.

This scheme can be generalized to arbitrary mixing and phase-separation unitaries, possibly not generated by some Hamiltonian. According to Hadfield et al. [47], the mixing unitaries just need to satisfy two conditions. Firstly, they need to "preserve the feasibility" of the state and secondly, they need to "explore the feasible subspace". The phase separators then just include the information about the cost function and not about the constraints of the problem. By this, the algorithm stays within the subspace of feasible solutions and potentially saves resources for problems with hard constraints. This *Quantum Alternating Operator Ansatz* for constructing QAOA-like algorithms was first introduced by Hadfield [12] and generalizes the original QAOA scheme.

To conclude this section, the following remark justifies the assumption that QAOA actually solves problems.

Remark 2.5.2. *When denoting the minimum of* $f_p(\beta, \gamma)$ (2.12) *by*

$$F_p = \min_{\beta,\gamma} \left({}_p\langle \beta, \gamma | H_P | \beta, \gamma \rangle_p \right) = f_p\left(\beta^*, \gamma^*\right) \tag{2.26}$$

it can be shown ([8, 12]) under minor assumptions that QAOA solves many combinatorial optimization problems exactly in the sense that

$$\lim_{p\to\infty} F_p = f^*, \tag{2.27}$$

where $f^* = \min_{x\in S} f(x)$ *However, in practis, only fixed-p circuits can be implemented and QAOA solves combinatorial optimization problems just approximately [12].*

2.1.4 Example: Maximum Cut Problem (Max-Cut)

This section introduces the *Maximum Cut* (Max-Cut) problem, which is proven to be NP-complete [49]. Already in the original paper by Farhi, Goldstone and Gutmann [8], Max-Cut was the main example for demonstrating the performance of QAOA and since then it has been tested extensively for the Max-Cut problem [9–27]. It is the main application considered in this thesis. Beyond defining the problem, we show how the instructions given in the preceding Section 2.1.3 may be applied to this problem.

Roughly speaking, Max-Cut addresses the problem of finding a cut in a graph that passes as many edges as possible (see Figure 2.1). Each edge is allowed to be crossed

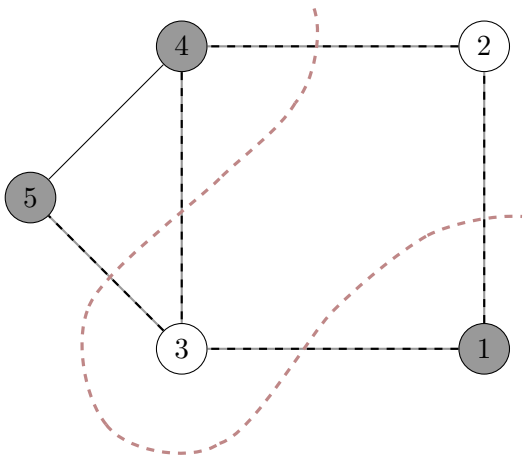


Figure 2.1: An example of a labled graph. The colors of the nodes correspond to a partitioning of the graph solving MAX-CUT. Edges connecting the two subsets are dashed. In red there is an illustration of the "cut" corresponding to this partitioning.

only once and the cut must return to its starting point. Mathematically, an undirected graph $G = (V, E)$ consists of a finite set V of *vertices*[4] and a set E of *edges*. Each vertex is labeled by some number $j \in \mathbb{N}$. Each edge $e \in E$ connects two of such vertices $j, k \in V$ and is given by the labels of those vertices as $e = jk := \{j, k\}$. The graph is called *undirected*, because the order of j and k does not matter and $kj = \{k, j\} = \{j, k\} = jk$ is the same edge. For two disjoint subsets $V_1, V_2 \subset V$, we say an edge $jk \in E$ *connects* V_1 and V_2, if either $j \in V_1$ and $k \in V_2$ or the other way around.

In Figure 2.1 there is an example of a graph G and a solution U to MAX-CUT. With a precise mathematical notion of graphs, we can now define the MAX-CUT problem.

Definition 2.6 (Weighted MAX-CUT)**.** *Let a graph $G = (V, E)$ and weights $w_{j,k} \in \mathbb{R}$ for each $j, k \in V$, with $w_{j,k} = 0$ for $jk \notin E$ be given. Find a partition of V into two complementary sets $V_1 = U \subseteq V$ and $V_2 = V \setminus U$ such that the sum of all weights associated with edges connecting the two subsets is being maximized, i.e. find a subset $U \subseteq V$ that realizes the minimum*

$$\max_{U \in \mathcal{P}(V)} \sum_{\substack{j \in U,\, k \notin U \\ j,k \in V}} w_{j,k}. \tag{2.28}$$

An instance of weighted MAX-CUT is given by a graph $G = (V, E)$ and weights $\{w_{j,k} \mid j, k \in V,\, w_{j,k} = 0 \text{ for } jk \notin E\}$.

The weights of the edges make the problem slightly more general. However, most of the

[4] also called "nodes"

time they are set to $w_{j,k} = 1$ for all edges $jk \in E$. Then, the matrix $(w_{j,k})_{j,k}$, called the adjacency matrix, represents the graph by defining all of its edges. The adjacency matrix of the graph in Figure 2.1 is given by

$$w_{j,k} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}. \tag{2.29}$$

The MAX-CUT problem is an example of binary combinatorial optimization problem. Moreover, we have the following result.

Lemma 2.4 (QUBO formulation of MAX-CUT)**.** *Solving a MAX-CUT instance given by $G = (V, E)$ and weights $\{w_{j,k} \mid j, k \in V,\, w_{j,k} = 0, \text{for } jk \notin E\}$ may be done by solving a corresponding QUBO problem, with QUBO matrix*

$$(Q_{j,k})_{1 \le j,k \le |V|}, \quad Q_{j,k} = \begin{cases} w_{j,k} & (j \neq k) \\ \sum_{l=1}^{|V|} w_{j,l} & (j = k) \end{cases}. \tag{2.30}$$

Proof. See Appendix A. □

For example the QUBO-matrix of the graph in Figure 2.1 is given by

$$Q = \begin{pmatrix} 2 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 3 & 1 & 1 \\ 0 & 1 & 1 & 3 & 1 \\ 0 & 0 & 1 & 1 & 2 \end{pmatrix}. \tag{2.31}$$

The partition illustrated in Figure 2.1 is encoded by the binary string $x = 10011$ and its cost is $C(x) = 10$, which is twice the number of edges connecting the two subsets.

Now, having formulated MAX-CUT in terms of a QUBO cost function as in Equation (2.16), we can immediately make use of Lemma 2.2 and Lemma 2.3 in order to get an appropriate problem Hamiltonian $H_P = H_C$ and QAOA phase-separation unitary, respectively. Thus, Lemma 2.4 gives all information needed to apply the QAOA on MAX-CUT-instances.

2.2 The Time-Dependent Variational Principle (TDVP)

A crucial choice in the QAOA approach is the subroutine optimizer O searching for the optimal parameters β^*, γ^* (Section 2.1.2). Commonly used optimization algorithms like

Constrained Optimization BY Linear Approximation (COBYLA)[50] or *Sequential Least SQuares Programming* (SLSQP)[51] are applicable quite generally. This has the drawback that they do not respect the structure of the QAOA. In particular, the geometry of the QAOA-state space is usually neglected. An example is the standard *gradient descent* (GD), attributed to Cauchy, who introduced the concept in 1847 first, and to Hadamard (1908) [52–55]. It is defined iteravely by

$$\delta_{n+1} = \delta_n - \Delta \cdot \nabla f(\delta_n) \qquad (n \in \mathbb{N} \cup \{0\}), \tag{2.32}$$

where $f(\delta)$ is the function to be minimized, ∇f is the gradient of f and $\Delta > 0$ is some fixed stepsize. It follows the opposite direction of the steepest increase in f, without considering the structure of the QAOA-states producing f. This implies the risk of losing the optimality of the chosen direction when the parameter dependence is restricted in some sense, resulting in non-euclidean geometry. This is the case for the QAOA-manifold

$$\mathcal{M}_p = \left\{ |\beta, \gamma\rangle_p \;\middle|\; \beta, \gamma \in \mathbb{R}^{2p} \right\} \tag{2.33}$$

of QAOA-states (2.7), where the Trotterization (2.6) only approximates arbitrary states for small p.

In this thesis we consider an optimization algorithm that is aware of the geometry of the QAOA-submanifold $\mathcal{M}_p$. This algorithm is based on a variational formulation of the *imaginary time evolution* e^{itH} reducing its evaluation to solving a set of ordinary differential equations (ODEs) in parameter space. In some way, the algorithm presented in this section resembles basic gradient descent (2.32), but respects the geometry of the submanifold and thus is provably optimal [28].

This section introduces the *Time-dependent variational principle* (TDVP) in a general setting as a variational formulation of the real and imaginary time evolution of parametrized quantum states [56, 57]. In Chapter 3 we will show how to apply the algorithm described here for QAOA including the evaluation of its crucial parts by quantum circuits.

We begin in Section 2.2.1 by introducing the TDVP for real time evolution on a parametrized submanifold of the entire Hilbert space first. In Section 2.2.2 we put it into a geometric perspective showing its optimality by the *Mc Lachlan minimal error principle*. This geometric perspective is applied to the imaginary time evolution in Section 2.2.3 deriving a set of ODEs very similar to those of real time evolution. These ODEs give rise to the beforementioned optimization algorithm.

A short introduction to the mathematical notions used in this section is given in Appendix B. For further details, we refer to the literature [28, 56–61]. The proofs of several Lemmas can be found in Appendix A.

2.2.1 Real time evolution

At first, the TDVP offers a variational formulation of real time evolution, which is defined by the *Time-Dependent Schrödinger Equation* (TDSE)[**<empty citation>**]

$$\frac{\mathrm{d}}{\mathrm{d}t}\left|\psi(t)\right\rangle = -iH(t)\left|\psi(t)\right\rangle \qquad (\psi(t) \in \mathcal{H}). \tag{2.34}$$

The TDSE is a partial differential equation given by some self-adjoint operator (the *Hamiltonian*) H acting on the Hilbert space $\mathcal{H}$. The TDVP considers this equation as the solution to a variational problem, minimizing the *action functional*

$$\begin{aligned} S\left[\bar{\psi}(t), \psi(t)\right] &= \int_{t_1}^{t_2} L\left(\bar{\psi}(t), \psi(t)\right) \mathrm{d}t \\ L\left(\bar{\psi}(t), \psi(t), t\right) &= \frac{i}{2}\left\langle\psi(t)|\dot{\psi}(t)\right\rangle - \frac{i}{2}\left\langle\dot{\psi}(t)|\psi(t)\right\rangle - \left\langle\psi(t)\right| H(t)\left|\psi(t)\right\rangle, \end{aligned} \tag{2.35}$$

where L is called the *Lagrangian* and $\dot{\psi}$ denotes the time derivative $\frac{\mathrm{d}}{\mathrm{d}t}\psi$ [28]. The following Lemma formalizes that the TDSE can be derived by minimizing the action functional S.

Lemma 2.5. *A curve* $\mathbb{R} \ni t \mapsto \psi(t) \in \mathcal{H}$ *which extremalizes* S *as given in Equation* (2.35)*, solves the TDSE* (2.34)*.*

The above Lemma puts the equation governing time evolution in Quantum Mechanics into the same perspective as in classical mechanics. However, of course, this does not mean Quantum Mechanics is a classical theory. Introducing a Lagrangian has proven worthy just not only in classical mechanics but also in many other theories.

Another reason for putting the TDSE into this perspective becomes clear when restricting to a subset

$$\mathcal{M} = \{\psi(x) \in \mathcal{H} \mid x \in \mathbb{R}^n\} \subset \mathcal{H} \tag{2.36}$$

of parametrized vectors. Requiring that $x \mapsto \psi(x)$ is an embedding makes $\mathcal{M}$ a n-dimensional embedded submanifold of $\mathcal{H}$, called the variational manifold. It is convenient to assume that all vectors in $\mathcal{M}$ are normalized and thus define a unique pure quantum state.

General Assumption. *Assume all vectors in* $\mathcal{M}$ *to be normalized, i.e.*

$$\forall \psi \in \mathcal{M}\colon\ \|\psi\| = 1$$

Note that in contrast to $\mathcal{H}$, $\mathcal{M}$ does in general not carry vector space structure. However, it inherits a *symplectic structure* from the inner product $\langle\cdot|\cdot\rangle$[58]. This has many implications and roughly spoken means that $\mathcal{M}$ is a suitable manifold for many physical theories. For example, classical Hamiltonian mechanics follow naturally from a symplectic structure

[62].
Another thing to point out is that the RHS of (2.34) is not an element of $\mathcal{M}$ as we cannot assume that $\mathcal{M}$ is closed under $H\colon\ \mathcal{H} \to \mathcal{H}$ in general. This leads to the question of how to define time evolution on $\mathcal{M}$ such that the evolved state stays in the variational manifold $\mathcal{M}$. While not being the same, the time evolution on $\mathcal{M}$ should at least approximate the TDSE on $\mathcal{H}$. This can be achieved by the TDVP in the same way as before. By defining the time evolution via minimizing the Lagrangian under small variations inside $\mathcal{M}$ it is assured that the time evolution restricts to $\mathcal{M}$ while following the same principle as the TDSE does on the entire Hilbert space [28].

Lemma 2.6. *Let $\mathcal{M} = \{\psi(x) \in \mathcal{H} \mid x \in \mathbb{R}^n\}$ be an embedded submanifold of $\mathcal{H}$. Curves $\mathbb{R} \ni t \mapsto \psi(x(t)) \in \mathcal{M}$ minimizing the action functional (2.35) when restricted to $\mathcal{M}$ satisfy the following Euler-Lagrange equations:*

$$\omega_{j,k}\dot{x}(t)^k = -2\,\mathrm{Re}\Big(\langle \partial_j \psi(x(t))|\, H\, |\psi(x(t))\rangle\Big), \tag{2.37}$$

where ∂_j denotes $\frac{\partial}{\partial x_j}$ and $\omega_{j,k} = 2\,\mathrm{Im}\,\langle \partial_j\psi(x(t))|\partial_k\psi(x(t))\rangle$ is a symplectic structre on $\mathcal{M}$ [58].

2.2.2 McLachlan minimal error principle

Solving Equation (2.37) for the curve $x(t)$ with initial point $x(0) = x_0$ gives the evolution of $\psi(x(t))$ of the initial vector $\psi_0 = \psi(x_0)$. Although this evolution follows from the same action principle that leads to the ordinary real time evolution (2.34) on the Hilbert space $\mathcal{H}$, it is in general not clear whether there is a "better" choice of time evolution on $\mathcal{M}$. What we really want in the end is that the time evolution on $\mathcal{M}$ gives at any time the change of x that results in a change of $\psi(x)$ closest to the change given by the TDSE of the vector $\psi \in \mathcal{H}$. This change in full Hilbert space is given by $-iH(t)\,|\psi(t)\rangle$ as in Equation (2.34). So the "best" possible time evolution on $\mathcal{M} \subseteq \mathcal{H}$ should give a change of state that is closest to $-iH(t)\,|\psi(t)\rangle$. When speaking of "changes" at a given point x, we mean tangent vectors $\psi \mapsto \partial_j\psi(x)$ in the embedded tangent space $T_{\psi(x)}\mathcal{M} \subset \mathcal{H}$. So the "best" time evolution $\mathbb{R} \ni t \mapsto \psi(x(t)) \in \mathcal{M}$ should minimize the distance

$$\left\| \frac{\mathrm{d}}{\mathrm{d}t}\,|\psi(x(t))\rangle - (-iH)\,|\psi(x(t))\rangle \right\| \tag{2.38}$$

at every given time t under the condition that $\frac{\mathrm{d}}{\mathrm{d}t}\,|\psi(x(t))\rangle \in T_{\psi(x(t))}\mathcal{M}$. This is known as the *McLachlan minimal error principle*, which admits a geometrical perspective [58, 63].

For some vector space V and a subset $M \subset V$ finding the points $y \in M$ that minimize the distance to a given point $x \in V$ is known as the problem of finding *best approximations* in Functional Analysis. In the case of V being a Hilbert space[5] and M being a closed subspace there is a theorem stating that for each $x \in \mathcal{H}$ there exists exactly one best

approximation $y =: P_M(x) \in M \subset \mathcal{H}$ [64, Theorem V.3.2]. Furthermore, $P_M(x)$ must be the orthogonal projection of x onto M , meaning that [64, Lemma V.3.3]

$$\forall z \in M\colon \ \mathrm{Re}(\langle x - P_M(x)|z\rangle) = 0. \tag{2.39}$$

Choose a frame V_j of $\mathcal{M}$ by setting $V_j(x) = \partial_j\psi(x) \in T_{\psi(x)}\mathcal{M} \subset \mathcal{H}$. Observe that the inner product on $\mathcal{H}$ splits into a real and imaginary part,

$$\langle V_j|V_k\rangle = \frac{1}{2}\Big(\underbrace{2\,\mathrm{Re}\,\langle V_j|V_k\rangle}_{=:g_{j,k}} + i\,\underbrace{2\,\mathrm{Im}\,\langle V_j|V_k\rangle}_{=\omega_{j,k}}\Big). \tag{2.40}$$

We already mentioned that $\omega_{j,k}$ gives a symplectic structure to $\mathcal{M}$ in Lemma 2.6. Analogously, its counterpart $g_{j,k} \coloneqq 2\,\mathrm{Re}\,\langle V_j|V_k\rangle$ gives a Riemannian metric on $\mathcal{M}$ as it is a real inner product on the tangent space.[6]

Now having $\mathcal{M}$ equipped with a metric, we can give an explicit form for the orthogonal projection $P_{T_\psi\mathcal{M}}(\phi)$ of $\phi \in \mathcal{H}$ onto $T_\psi\mathcal{M} \subset \mathcal{H}$ following the constructions by Hackl et al. [58]:

$$P_\psi \coloneqq P_{T_\psi\mathcal{M}} = 2\sum_{j,k} |V_j\rangle\, g^{j,k}\,\mathrm{Re}\,\langle V_k|\,, \tag{2.41}$$

where $g^{j,k}$ is the inverse of the metric $g_{j,k}$, such that

$$g^{j,m}g_{m,k} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}. \tag{2.42}$$

A direct computation verifies that indeed $\mathrm{Re}\,\langle \phi - P_\psi(\phi)|\xi\rangle = 0$ for all $\xi \in T_\psi\mathcal{M}$.

Other than the action principle leading to Equation (2.37) the McLachlan minimal error principle is by construction optimal. In some cases both principles are equivalent and hence the real time evolution defined by Equation (2.37) is optimal, too. This equivalence is precisely the case, if the variational manifold $\mathcal{M}$ has the Kähler property, i.e. when $J \coloneqq g^{-1}\omega$ satisfies $J^2 = -\mathrm{id}$ [58]. Another equivalent property is that all tangent spaces $T_\psi\mathcal{M}$ are not only real but also complex subspaces of $\mathcal{H}$ [58]. Haegeman, Mariën, Osborne and Verstraete have shown that the variational manifolds of *Matrix Product States*, of which the QAOA-manifold is a special case, are Kähler manifolds [59].

On the one hand, this justifies the definition of time evolution on a variational manifold by analogy to the TDSE on full Hilbert space as in Section 2.2.1. On the other hand, by putting the TDVP into a geometrical perspective, we can do the same projection (2.41)

[5] More generally, it is sufficient for V to be a uniformly convex Banach space, meaning that the midpoint of two points on the unit sphere has some distance $\epsilon > 0$ to the unit sphere. M needs to be closed and convex in the general case.

[6] As $\mathcal{M}$ is just a real manifold, its tangent spaces $T_{\psi(x)}\mathcal{M}$ are only real vector spaces in general, meaning that an inner product needs to be a mapping to the reals. See Appendix B for more details.

for various objects other than $-iH\left|\psi\right\rangle$. This comes in especially handy in cases where no action principle as in Equation (2.35) exists. This is the case for the imaginary time evolution discussed in the next section [58].

2.2.3 Imaginary time evolution

In this section, we apply the observations about the real time evolution on submanifolds of the Hilbert space, made in Section 2.2.2, to the imaginary time evolution. In its most general form, real time evolution is defined as the action of the unitary one-parameter group

$$U_t \coloneqq e^{-itH} \qquad (t \in \mathbb{R}) \tag{2.43}$$

$$U_t U_s = e^{-itH} e^{-isH} = e^{-i(t+s)H} = U_{t+s} \qquad (t, s \in \mathbb{R}) \tag{2.44}$$

$$U_0 = e^{-i0H} = \mathbb{I}, \tag{2.45}$$

generated by the Hamiltonian H. The imaginary time evolution is defined very similarly as a unitary one-parameter group by

$$U_\tau = e^{-\tau H} \qquad (\tau \in \mathbb{R}). \tag{2.46}$$

In the same way, as the real time evolution U_t applied to pure quantum states results in the TDSE (2.34), imaginary time evolution U_τ results in

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\left|\psi(\tau)\right\rangle = -(H - E(\tau))\left|\psi(\tau)\right\rangle, \tag{2.47}$$

for $\psi(\tau) = e^{-\tau H}\psi$, where $E(\tau) = \left\langle\psi(\tau)\right| H \left|\psi(\tau)\right\rangle$ ensures normalization.

Although real and imaginary time evolution look very similar, they have important differences. While real time evolution preserves the energy expectation value, imaginary time evolution does not. In fact, $\psi(\tau)$ has monotonically decreasing energy expectation converging to the minimal energy state of the system, if the initial state ψ had some finite overlap with it. This can be directly seen by sorting the spectrum $E_0 < E_1 < \ldots$ of H increasingly and expanding the exponential

$$\psi(\tau) = e^{-\tau H} = \sum_j e^{-\tau E_j}\left\langle j|\psi\right\rangle\left|j\right\rangle, \tag{2.48}$$

in the eigenbasis $\left|j\right\rangle$ of H. This expression converges to $e^{-\tau E_0}\left\langle 0|\psi\right\rangle\left|0\right\rangle$ for $\tau \to \infty$, what motivates the usage of imaginary time evolution for finding ground states and minimization problems.

The naive way to utilize this behavior by just computing the evolution $\psi(\tau)$ for large τ is not applicable in practice, as this would require knowledge about the spectrum of H which already is the solution to the problem. Somehow implementing the evolution operator $e^{-\tau H}$ on a universal quantum device does fail, too. Other than real time

evolution, imaginary time evolution is not unitary and thus cannot be implemented on a quantum device. However, it is possible to derive a set of ODEs for the parameters of some parametrized submanifold which approximate the imaginary time evolution on Hilbert space in an optimal way.

Theorem 2.3. *Let* $\mathcal{M} = \left\{\psi(x) \;\middle|\; x \in \mathbb{R}^{2p}\right\}$ *be an embedded submanifold in* $\mathcal{H}$*. Then the curve* $[0, \infty) \ni \tau \mapsto x(\tau)$ *satisfying*

$$\dot{x}^j = -2g^{j,k} \operatorname{Re} \langle \partial_k \psi | \, H \, | \psi \rangle \,, \tag{2.49}$$

minimizes the distance

$$\left\| \frac{\mathrm{d}}{\mathrm{d}\tau} |\psi(x(\tau))\rangle - (E - H) |\psi(x(\tau))\rangle \right\| \tag{2.50}$$

and by the McLachlan minimal error principle gives the optimal approximation

$$[0, \infty) \ni \tau \mapsto \psi(x(\tau)) \in \mathcal{M}$$

to the imaginary time evolution

$$[0, \infty) \ni \tau \mapsto U_\tau \psi \in \mathcal{H}.$$

Furthermore, the solution $x(\tau)$ *to* (2.49) *monotonically decreases the energy expectation* $E(\tau) = \langle \psi(x(\tau)) | \, H \, | \psi(x(\tau)) \rangle$*.*

Proof. A proof following Hackl et al. [58] is given in Appendix A. □

Hence, evaluating the imaginary time evolution of a given initial vector $\psi(x(0)) \in \mathcal{M}$ becomes equivalent to solving the ODEs in Equation (2.49) which is potentially more feasible for practical applications.

Equation (2.49) looks similar to the standard gradient descent (3.17) when noticing that $\operatorname{Re} \langle \partial_k \psi | \, H \, | \psi \rangle$ is the gradient of the energy expectation $\langle \psi | \, H \, | \psi \rangle$. In fact, the standard gradient descent (2.32) looks like a linear approximation to Equation (2.49) when neglecting the metric $g_{j,k}$. Then, imaginary time evolution could be thought of as a non-linear gradient descent weighted by the metric g. However, there is no general justification for treating $x(\tau)$ linearly, so this intuition is to be dealt with carefully. There is a version of the gradient descent called *natural gradient descent* where the metric is also taken into account [65, 66]. However, the *natural gradient descent* is also linear which distinguishes it from the TDVP, even if they are both considering the same differential equations.

For large τ_1, the solution $x(\tau_1)$ yeilds a pure state $\psi(x(\tau_1))$ near the best approximation in $\mathcal{M}$ to the eigenspace of the ground state energy of H. This can be directly utilized as an optimization algorithm. In the remainder of the thesis, we apply the TDVP to find near-to-optimal parameters β, γ for QAOA.

Chapter 3

TDVP-optimization of QAOA

This chapter explains, how the TDVP introduced in Section 2.2 can be used for the QAOA. If the right-hand-side (RHS) of Equation (2.49) is known, solving the resulting differential equations gives a path with monotonically decreasing energy in the variational manifold $\mathcal{M}$. This behavior can be directly used for the parameter optimization of the QAOA (Section 2.1), by using a classical algorithm for solving Equation (2.49). Let

$$\mathcal{M}_p = \left\{ \psi_p(\delta) \;\middle|\; \delta \in (0, 2\pi)^{2p} \right\} \tag{3.1}$$

for some $p \in \mathbb{N}$ and $|\psi_p(\delta)\rangle$ as in Definition 2.1 be the set of all QAOA-states. The map $\psi_{(p)} \colon (0, 2\pi)^{2p} \to \mathcal{M}_p$ is injective, hence bijective. Furthermore, it is an embedding and therefore the set $\mathcal{M}_p$ is an embedded submanifold of $\mathcal{H}$ called the QAOA-manifold. Note that for this manifold the general assumption of all states being normalized holds as unitary operations are norm preserving and $\|\psi_0\| = 1$. After each step of the algorithm that solves the differential equations, the RHS needs to be computed, i.e. a method to compute the metric and the gradient of QAOA is needed. The development of such methods builds the main theoretical content of this thesis. Computing the metric for QAOA is addressed in Section 3.1. A similar method for computing the gradient of the QAOA is demonstrated in Section 3.2.

3.1 The metric of QAOA

A vital part of the ODEs for the imaginary time evolution derived by the TDVP in Section 2.2.3 is the metric of the underlying variational manifold $\mathcal{M}$. Although the metric

$$g^{(p)} \coloneqq 2 \operatorname{Re} \mathfrak{g}^{(p)} \tag{3.2}$$

of $\mathcal{M}_p$ is easily defined as the two times the real part of the Gram matrix

$$\mathfrak{g}^{(p)} = \left(\mathfrak{g}^{(p)}_{j,k} \right)_{1 \le j,k \le 2p} \coloneqq \left(\left\langle \partial_{\delta_j} \psi_p(\delta) \middle| \partial_{\delta_k} \psi_p(\delta) \right\rangle \right)_{1 \le j,k \le 2p}, \tag{3.3}$$

of the tangent space (2.40), it is not that trivial to compute. If we can compute $\mathfrak{g}^{(p)}$, computation of $g^{(p)}$ is trivial. The Gram matrix involves derivatives of the state

parametrization ψ_p. One way to compute those would be to calculate finite differential differences

$$\frac{\psi(x+\epsilon)-\psi(x)}{\epsilon} \qquad (\epsilon > 0), \tag{3.4}$$

but this is not feasible in practice: When measuring the states $\psi(x+\epsilon)$ and $\psi(x)$ after preparing them on a quantum device we are only able to obtain partial information about the quantum state by measuring it periodically and evaluating the resulting probability distribution. Deducing precise information about the difference between both states is at least difficult. A more natural possibility to compute the metric is presented in the following. In the end, the metric is computed similarly to but independently of McArdle et al. [67]. The idea is very similar to the core idea of the QAOA itself. There, the expectation value of the cost function is evaluated by a quantum circuit together with some classical computations.

To see how this concept can be applied to the computation of the metric, recall the QAOA-gates $U_B(\beta) = e^{-i\beta B}$ and $U_P(\gamma) = e^{-i\gamma H_P}$ as in Definition 2.1. In order to compute the Gram matrix $\mathfrak{g}^{(p)}$, we need the partial derivatives of $\psi_p(\delta)$. These have different forms depending on the kind of parameter differentiated.

$$\left|\partial_{\delta_j}\psi_p(\delta)\right\rangle = \begin{cases} U_B(\delta_p)U_P(\delta_{2p})\dots \\ \quad \dots(-iB)U_B(\delta_j)U_P(\delta_{j+p})\dots & (j \le p) \\ \dots U_B(\delta_1)U_P(\delta_{1+p})\left|\psi_0\right\rangle \\ \\ U_B(\delta_p)U_P(\delta_{2p})\dots \\ \quad \dots U_B(\delta_{j-p})(-iH)U_P(\delta_j)\dots & (p < j \le 2p) \\ \dots U_B(\delta_1)U_P(\delta_{1+p})\left|\psi_0\right\rangle \end{cases} \tag{3.5}$$

Note that in both cases besides a global phase $-i$, there is just one operator inserted in the QAOA circuit at a certain position.[1] With this, an element of $\mathfrak{g}^{(p)}$ can be computed by

$$\begin{aligned} \mathfrak{g}^{(p)}_{j,k} = \langle\psi_0|\, & U_P(-\delta_{1+p})U_B(-\delta_1)\dots \\ & \dots U_P(-\delta_{j+p})U_B(-\delta_j)B\dots \\ & \dots U_P(-\delta_{2p})U_B(-\delta_p)U_B(\delta_p)U_P(\delta_{2p})\dots \quad (j \le p,\, p < k \le 2p) \\ & \dots U_B(\delta_{k-p})H_P U_P(\delta_k)\dots \\ & \dots U_B(\delta_1)U_P(\delta_{1+p})\left|\psi_0\right\rangle, \end{aligned} \tag{3.6}$$

with $j \le p$, $p < k \le 2p$ for example. Other cases can be treated equivalently. Now, the goal is to bring this into a form, which can be evaluated on quantum computers just like it was done in Equation (2.19) for the expectation $\langle\psi_p(\delta)|\, H_P\, |\psi_p(\delta)\rangle$. The only problem is that the Hamiltonians B and H_P are not assumed to be unitary and thus the

[1] There might be confusion about the parameter indices. Recall that if $j > p$, $\delta_j = \gamma_j$ and $\delta_{j-p} = \beta_j$.

expressions in (3.5) are not unitary in general. However, this problem can be fixed when considering the explicit form of B and H for QUBO problems given in Equation (2.20) and Equation (2.17).

Using this, we can split (3.6) into multiple circuits. Then, each element of $\mathfrak{g}^{(p)}$ can be computed by evaluating each of those circuits and summing the results (compare with [67]).

Theorem 3.1. *Let B be defined as in Equation (2.20) and let $H_P = H_f$ for a binary polynomial $f(x) = C + \sum_j L_j x_j + \sum_{j<k} Q_{j,k} x_j x_k$ of degree at most 2 as in Equation (2.17). Then, the Gram matrix $\mathfrak{g}^{(p)}$ (3.3) of the QAOA-manifold $\mathcal{M}_p$ (3.1) can be computed by*

$$\mathfrak{g}^{(p)}_{j,k} = \begin{cases} \sum_{l,m} \langle\psi_0| U_j(-\delta, X_l) U_k(\delta, X_m) |\psi_0\rangle & (j,k \le p) \\ -\frac{1}{2} \sum_{l,m} C_m \langle\psi_0| U_j(-\delta, X_l) \tilde{U}_{k-p}(\delta, Z_m) |\psi_0\rangle \\ +\frac{1}{4} \sum_{l,m<n} Q_{m,n} \langle\psi_0| U_j(-\delta, X_l) \tilde{U}_{k-p}(\delta, Z_m Z_n) |\psi_0\rangle & (j \le p, k > p) \\ \overline{\left(\mathfrak{g}^{(p)}_{k,j}\right)} & (j > p, k \le p) \\ \frac{1}{4} \sum_{l,m} C_l C_m \langle\psi_0| \tilde{U}_{j-p}(-\delta, Z_l) \tilde{U}_{k-p}(\delta, Z_m) |\psi_0\rangle \\ -\frac{1}{8} \sum_{l<m,n} Q_{l,m} C_n \langle\psi_0| \tilde{U}_{j-p}(-\delta, Z_l Z_m) \tilde{U}_{k-p}(\delta, Z_n) |\psi_0\rangle \\ -\frac{1}{8} \sum_{l,m<n} C_l Q_{m,n} \langle\psi_0| \tilde{U}_{j-p}(-\delta, Z_l) \tilde{U}_{k-p}(\delta, Z_m Z_n) |\psi_0\rangle & (j > p, k > p) \\ +\frac{1}{16} \sum_{\substack{l<m, \\ n<o}} Q_{l,m} Q_{n,o} \langle\psi_0| \tilde{U}_{j-p}(-\delta, Z_l Z_m) \tilde{U}_{k-p}(\delta, Z_n Z_o) |\psi_0\rangle , \end{cases} \tag{3.7}$$

where $C_k = L_k + Q_k$ and

$$\begin{aligned} U_j(\delta, A) &= U_B(\delta_p) U_P(\delta_{2p}) \dots A U_B(\delta_j) U_P(\delta_{j+p}) \dots U_B(\delta_1) U_P(\delta_{1+p}) \\ \tilde{U}_j(\delta, A) &= U_B(\delta_p) U_P(\delta_{2p}) \dots U_B(\delta_j) A U_P(\delta_{j+p}) \dots U_B(\delta_1) U_P(\delta_{1+p}). \end{aligned} \tag{3.8}$$

are QAOA-circuits with a gate A inserted after or in between the j-th QAOA-block $U_B U_P$. Computing the Gram matrix $\mathfrak{g}^{(p)}$ using the above method runs a total of

$$(p^2 + p)\left(\frac{1}{8}n^4 + \frac{1}{2}n^3 - \frac{5}{8}n^2\right) \tag{3.9}$$

quantum circuits and applies a total of

$$\begin{aligned} \#\left(\mathfrak{g}^{(p)}\right) &= NC_1(n)\left(\frac{7}{3}p^2 + 2p + \frac{1}{3}\right) + (2C_2(n) - NC_1(n))\left(\frac{1}{2}p^2 + \frac{3}{2}p\right) \\ &\in O\left(n^6, p^3\right) \end{aligned} \tag{3.10}$$

single qubit rotations R_X, R_Z and `CNOT`*-gates, where* $C_1(n) = \frac{1}{2}n^4 + 2n^3 + \frac{7}{2}n^2$, $C_2(n) = n^4 + \frac{5}{2}n^3 + \frac{5}{2}n^2$ *and*

$$N = p\,(2n + 3\#(Q))\,. \tag{3.11}$$

$\#(Q)$ is the number of non-zero elements in the QUBO-matrix Q representing f in upper-triangular form.

Proof. By substituting Equations (2.17), (2.20) and (3.5) into Equation (3.3) and writing in terms of $U_j(\delta, A)$ and $\tilde{U}_j(\delta, A)$. Use that the adjoints of U_P and U_B are given by $U_P(-\delta)$ and $U_B(-\delta)$ respectively and that the Pauli operators Z_j and Z_k are self inverse (Section 1.2). For $j > p, k \leq p$ use the fact that $\mathfrak{g}^{(p)}$ is self-adjoint as

$$\mathfrak{g}^{(p)}_{j,k} = \left\langle \partial_{\delta_j}\psi_p \middle| \partial_{\delta_k}\psi_p \right\rangle = \overline{\left\langle \partial_{\delta_k}\psi_p \middle| \partial_{\delta_j}\psi_p \right\rangle} = \overline{\left(\mathfrak{g}^{(p)}_{j,k}\right)}. \tag{3.12}$$

Equation (3.10) is shown in Appendix A. □

Note, that when having computed the Gram matrix $\mathfrak{g}^{(p)}$, computing the metric $g^{(p)} = 2\,\mathrm{Re}\,\mathfrak{g}^{(p)}$ is trivial.

The expressions of the form $\langle\psi_0|\, U_j(\cdot,\cdot)\tilde{U}_k(\cdot,\cdot)\,|\psi_0\rangle$ in Equation (3.7) can be evaluated on quantum devices by running the circuit $U_j(\cdot,\cdot)\tilde{U}_k(\cdot,\cdot)$ and measuring the resulting state

$$U_j(\cdot,\cdot)\tilde{U}_k(\cdot,\cdot)\,|\psi_0\rangle \tag{3.13}$$

in the Hadamard basis. Then, when setting $|\psi_0\rangle = |-\rangle$ as in Section 2.1.3, the measured amplitude of $|-\rangle^{\otimes n}$ corresponds to the desired overlap.

One thing to notice with regard to computational resources is that "in the middle" of each circuit a certain number of gates just cancel each other out and are not needed to be applied. Only when one of the inserted gates is applied, the gates do not cancel. For details see the proof of Theorem 3.1 on p. 63.

3.2 The gradient of QAOA

Similarly to evaluating the metric of the QAOA-manifold using quantum circuits in the previous Section 3.1, we can also compute the gradient

$$\nabla f_p\,(\delta) \coloneqq \Big(\partial_j f_p\,(\delta)\Big)_{1\leq j\leq 2p} = \Big(2\,\mathrm{Re}\,\underbrace{\langle\partial_j\psi_p(\delta)|\, H_P\,|\psi_p(\delta)\rangle}_{\coloneqq\mathcal{E}_j^{(p)}}\Big)_{1\leq j\leq 2p} \tag{3.14}$$

of QAOA where again $H_P = H_f$ is the QUBO Hamiltonian as in Equation (2.17). The procedure is the very same as in Theorem 3.1 and leads to the following Theorem.

Theorem 3.2. *With B and H_P as in Theorem 3.1 the expressions $\mathcal{E}_j^{(p)} = \langle \partial_j \psi_p(\delta)| H |\psi_p(\delta)\rangle$ in the gradient (3.14) of QAOA can be computed by*

$$\mathcal{E}_j^{(p)} = \begin{cases} -\frac{i}{2}\sum_{k,l} C_l \langle\psi_0| U_j(-\delta, X_k)\, Z_l\, U^{(p)}(\delta) |\psi_0\rangle \\ +\frac{i}{4}\sum_{k,l<m} Q_{l,m} \langle\psi_0| U_j(-\delta, X_k)\, Z_l Z_m\, U^{(p)}(\delta) |\psi_0\rangle & (j \le p) \\ \\ \frac{i}{2}\sum_{k,l} C_k C_l \langle\psi_0| \tilde{U}_{j-p}(-\delta, Z_k)\, Z_l\, U^{(p)}(\delta) |\psi_0\rangle \\ -\frac{i}{8}\sum_{k,l<m} C_k Q_{l,m} \langle\psi_0| \tilde{U}_{j-p}(-\delta, Z_k)\, Z_l Z_m\, U^{(p)}(\delta) |\psi_0\rangle \\ -\frac{i}{8}\sum_{k<l,m} Q_{k,l} C_m \langle\psi_0| \tilde{U}_{j-p}(-\delta, Z_k Z_l)\, Z_m\, U^{(p)}(\delta) |\psi_0\rangle & (j > p) \\ +\frac{i}{16}\sum_{\substack{k<l,\\ m<n}} Q_{k,l} Q_{m,n} \langle\psi_0| \tilde{U}_{j-p}(-\delta, Z_k Z_L)\, Z_m Z_n\, U^{(p)}(\delta) |\psi_0\rangle\,, \end{cases} \tag{3.15}$$

where $C_k = L_k + Q_k$ and U_j, $\tilde{U}_j$ are defined as in Equation (3.8). Computing the gradient $\nabla f_p(\delta)$ using the above method and Equation (3.14) runs a total of

$$p\left(\frac{1}{4}n^4 + n^3 - \frac{9}{4}n^2\right) \tag{3.16}$$

quantum circuits and uses

$$\begin{aligned} \#\left(\nabla f_p(\delta)\right) &= 2p^2\left(n^5 + 4n^4 + 3n^3 + \frac{3}{2}n^4\#(Q) + 6n^3\#(Q) + \frac{9}{2}n^2\#(Q)\right) \\ &\quad + p\left(n^4 - 2n^3 + \frac{5}{2}n^2\right) \\ &\le 2p^2\left(\frac{3}{2}n^6 + 7n^5 + \frac{17}{2}n^4 + 3n^3\right) + p\left(n^4 - 2n^3 + \frac{5}{2}n^2\right) \\ &\in O\left(n^6, p^2\right) \end{aligned} \tag{3.17}$$

single qubit roatations R_X, R_Z and `CNOT`-gates, where $\#(Q)$ is the number of non-zero elements in the QUBO-matrix Q representing f in upper-triangular form.

Proof. By substituting Equations (2.17), (2.20) and (3.5) into Equation (3.14) and writing in terms of U_j and $\tilde{U}_j$. Equations (3.16) and (3.17) are proven in Appendix A. □

Together with Theorem 3.1, we have enabled a way to utilize the TDVP (Section 2.2) on a quantum computer. We compute the QAOA-metric and -gradient by running many quantum circuits with approximately the same depth as QAOA. In this way, for each step in the optimization of the QAOA-parameters, most of the computation of the RHS in (2.49) is performed by quantum circuits. Only basic arithmetic calculations

of the sums are done classically. This makes it feasible to fully utilize the TDVP for optimization of the QAOA-parameters and introduces an optimization algorithm specific to QAOA.

Chapter 4

Numerical simulations

This chapter analyzes the methods introduced in the preceding chapters by numerical simulations. The goal is to compare the performance of QAOA when optimizing its parameters by an ordinary optimizer and by the TDVP-optimizer introduced in Section 2.2.3. Concretely we compare the following three versions of the QAOA.

Definition 4.1 (QAOA variants)**.** *The following three versions of the QAOA are considered.*

(A1) The QAOA with optimizing the QAOA-parameters by the Time-dependent variational principle *(TDVP) for imaginary time evolution (Section 2.2.3).*

(A2) The QAOA with optimizting the parameters by the standard gradient descent GD (2.32)*.*

(A3) The QAOA with optimizing the QAOA-parameters by the Constrained Optimization BY Linear Approximation *(COBYLA) [29–31].*

The performance analysis of these algorithms is split into three research questions aiming at different aspects of algorithmic performance.

Definition 4.2 (Research questions)**.** *We define the following three research questions regarding different aspects of computational performance.*

(Q1) How does the quality of results compare for A1 relative to A2 and A3?

(Q2) How does the amount of computational ressources used by A1 compare to A2 and A3?

(Q3) How does the efficiency of A1 differ to that of A2 and A3?

Section 4.1 introduces the methods that are used for investigating these research questions by simulation of the three algorithms. Results of such simulations are discussed in Section 4.2.

4.1 Methods

This section explains the methods used for investigating the research questions, defined above, by simulations.
A number of fixed problem instances of Max-Cut (Section 2.1.4) of two different problem sizes are chosen for all simulations, such that simulation on ordinary personal computers is convenient. Concretely, Max-Cutinstances for all mutually distinct connected graphs of sizes $n = 4$ (6 graphs in total) and $n = 5$ (10 graphs in total), that are listed in the "Atlas of Graphs", are being considered. The "Atlas of Graphs" is accessed via `networkx.graph_atlas_g()`. For both problem sizes, a different range of p-values is chosen to keep the run time of the simulations reasonable. For instances with $n = 4$ the range from $p = 1$ to $p = 5$ is considered, while for $n = 5$ only values up to $p = 4$ are considered.

For the analysis of the performance, several measures quantifying each performance aspect of the research questions Q1 – Q3 are computed for each simulation result. These measures are explained in Section 4.1.1. The notion of *effect size* from statistics is briefly introduced in Section 4.1.2. In Section 4.1.3, central aspects of the implementations of each algorithm are being summarized.

4.1.1 Performance measures

This section explains the methods and measures used for the analysis of research questions Q1 – Q3. These measures are defined by the author independently of the literature, if not denoted otherwise. Recall that the QAOA outputs a pure quantum state $\psi_p(\delta')$, which may be interpreted as a probability distribution of classical bitstrings. The answer of the algorithm is the bitstring x_1 with the highest probability $|\langle x|\psi_p(\delta')\rangle|$. The set of the optimal solutions to the combinatorial optimization problem is denoted by S^*.

In order to investigate research question Q1, the quality of the output of the algorithm needs to be measured somehow. In the context of approximate combinatorial optimization, three different approaches are considered. The first approach regards the single answer x_1 of the algorithm. A value may be assigned to x_1 by evaluating the cost function $f(x_1)$. The ratio

$$\frac{f(x_1)}{f^*} \tag{4.1}$$

is known as the *approximation ratio* of the approximate answer x_1, where $f^* = \min f(S)$ is the minimal value of the cost function [12]. It gives a "relative" measure of the quality. Here, "relative" refers to the fact that an answer that is not optimal but close to optimal will not be regarded as worthless, but gets a relative quality value assigned to it. Since Max-Cut is formulated as a minimization problem in Section 2.1.4, we have that $\frac{f(x)}{f^*} \leq 1$ with equality if and only if $x \in S^*$ is an optimal solution [12]. Hence, an approximation ratio of 1 is regarded as the best possible quality and lower values are considered worse.

The second approach considers the entire probability distribution given by $\psi_p(\delta') \in$

$\mathcal{H}$. Here, several features of the distribution may be relevant [68]. For example, the expectation of the ground space projector $P_{S^*} = \sum_{x \in S^*} |x\rangle \langle x|$, i.e. the probability

$$\mathbb{E}\left[P_{S^*}\right] \coloneqq \langle P_{S^*} \rangle \coloneqq \left\| P_{S^*} \psi_p\left(\delta^*\right) \right\|^2 = \sum_{x \in S^*} \left| \langle \psi_p\left(\delta^*\right) | x \rangle \right|^2 \tag{4.2}$$

for the output to be one of the optimal solutions $x^* \in S^*$ may be considered as the quality of the algorithm's result. We call this approach *absolute quality* since only optimal answers in S^* are considered to be correct and the entire output of the algorithm is taken into account. This measure is constructed independently from the literature. However, for example Larkin, Jonsson, Justice and Guerreschi [68] follow a similar approach. Moreover, higher momenta of the distribution may be interesting as a quality measure. The centered momentum

$$\mathbb{S}\left[P_{S^*}\right] \coloneqq \left(\left\langle P_{S^*}^2 \right\rangle_{\psi_p(\delta^*)} - \langle P_{S^*} \rangle_{\psi_p(\delta^*)}^2 \right)^{\frac{1}{2}}, \tag{4.3}$$

i.e. the standard deviation gives information about how sharp the output of the algorithm lies in S^*. A higher value indicates higher uncertainty and is considered worse.

The third approach is a hybrid of the first two approaches. In this last approach, the entire output of the algorithm is considered, but not only optimal solutions in S^* are considered as a solution. Concretely, the expectation

$$\mathbb{E}\left(f/f^*\right) = 1/f^* \left\langle H_f \right\rangle_{\psi_p(\delta^*)} = \sum_{x \in S} f(x)/f^* \left| \langle \psi_p\left(\delta^*\right) | x \rangle \right|^2 \tag{4.4}$$

of the approximation ratio (4.1) with respect to the probability distribution (2.23) is considered. Again, a value of $\mathbb{E}\left(f/f^*\right) = 1$ is regarded as the best possible quality. This is precisely the case when *all* x with non-zero probability are optimal solutions to the problem. Smaller values are again considered as outputs of lower quality. The following definition summarizes the considered quality measures.

Definition 4.3 (Quality measures)**.** *The following measures are considered for quantifying the quality of the results of each algorithm.*

(M1) The approximation ratio (4.1) *of the answer* x_1 *with the highest probability.*

(M2) The expectation (4.4) *of the approximation ratio* (4.1) *for the final state* $\psi_p\left(\delta^*\right)$*.*

(M3) The ground state overlap (4.2) *of the final state* $\psi_p\left(\delta^*\right)$ *with the ground states of* H_f*.*

(M4) The uncertainty (4.3) *of the final state* $\psi_p\left(\delta^*\right)$ *in the ground states of* H_f*.*

With the different approaches above, a wide variety of quality measures is available for investigating research question Q1. Regarding research question Q2, there are not as many options for measures. In the end, one is often interested in the resources *time* or

energy. However, for analyzing algorithms such resources are hard to compare as they rely heavily on hardware and exterior circumstances. Thus, usually other quantities that are connected to the algorithm more directly are considered for algorithm analysis. In the case of quantum computing, the number of applied quantum gates is a natural choice of a computational resource [32]. Here, the number of single qubit R_X and R_Z rotations and `CNOT`-gates is being measured by counting the number of how many times each part of the optimizers is run and multiplying by the number of gates necessary for this part. For example, the gate count for A1 is calculated by

$$K\big(\#(g) + \#(\nabla f_p(\delta))\big) + pN, \tag{4.5}$$

where K is the number of optimization steps, p is the number QAOA-blocks, N is the number of gates of one usual QAOA-circuit (3.11) and $\#(g)$ and $\#(\nabla f_p(\delta))$ are defined in Equation (3.10) and Equation (3.17), respectively. The gate count for A3 is being calculated similarly by

$$(K+1)N, \tag{4.6}$$

where here K is the number of function calls of f_p by the optimizer. For A2 the gate count is computed by

$$(4K+1)N, \tag{4.7}$$

as the implementation of the GD computes one QAOA-circuit four times per step (see Section 4.1.3).
The number of necessary gates depends on compilation and on the set of gates available on the hardware. So in addition to the gate count, the total number of quantum circuits that are run is being considered.

Definition 4.4 (Ressource measures)**.** *The following measures are considered for quantifying the resource consumption of each algorithm.*

(M1) The gate count (4.5) *of gates used by the optimization algorithm in total.*

(M2) The circuit count *of the algorithm, i.e. the number of quantum circuits run by the optimization algorithm.*

For investigating the third research question Q3, any combination of quality and resource measures may be considered. Here, the ratio of the expectation of the approximation ratio M1 and the circuit count M2 is considered as an efficiency measure. Differing from this, the total *pathlength* of the path taken in parameter space during the optimization process may be considered for comparing A2 and A1. When compared to the shortest connection between the respective endpoints, it quantifies how directly the optimization algorithm approaches its final parameters. The total *pathlength* is computed by

$$L\big[\big(\delta^{(k)}\big)_k\big] \coloneqq \sum_{k=1}^{K} \left\| \delta^{(k)} - \delta^{(k-1)} \right\|, \tag{4.8}$$

where $\delta^{(k)}$ denotes the QAOA-parameters after the k-th optimization step and $\|\cdot\|$ is the euklidean norm on $\mathbb{R}^{2p}$. The length L^* of the shortest connection in flat $\mathbb{R}^{2p}$ between the intial point $\delta^{(0)}$ and the final parameters $\delta^{(K)} = \delta^*$ is the distance

$$L^* \coloneqq \left\| \delta^{(0)} - \delta^{(K)} \right\| \leq L\big[\big(\delta^{(k)}\big)_k\big]. \tag{4.9}$$

A shorter path length suggests, that the optimizer took fewer unnecessary steps or that the steps were taken in a way leading more efficiently towards the final point. Similar to the approximation ratio, the *relative pathlength*

$$\frac{L^*}{L\big[\big(\delta^{(k)}\big)_k\big]} \leq 1 \tag{4.10}$$

indicates the relative optimality of the path taken in parameter space, where a value of 1 is attained to the shortest path.

In the case of the TDVP-optimizer A1, there theoretically is a continuous parameter path $\delta^{(t)}$ for $t \in [0, T]$, i.e. the analytic solution to the ODEs (2.49). In order to compare this with the discrete optimization steps of the GD optimizer in A2, a finite set of sample points is chosen from the curve. This set is exactly the set of parameter points after each step of numerically solving Equation (2.49).

Definition 4.5 (Efficiency measures)**.** *The following measures are considered for quantifying the efficiency of each algorithm.*

(M1) The ratio of the expectation of the approximation ratio M2 and the circuitcount M2.

(M2) The relative pathlength (4.10) *of the path taken in parameterspace* $\mathbb{R}^{2p}$ *by the optimizer.*

Together, the measures M1-M2, deliver enough information about the algorithms A2-A1 in order to investigate the considered research questions Q1-Q3.

4.1.2 Effect size

The analysis of these measures is based on their *mean values* and *standard deviations* within each instance class given by (n, p). Besides visually interpreting the graphs of those results, the so-called *effect size* is considered for making precise statements about the differences between the algorithms. In statistics, the *effect size* is a measure of the practical relevance of empirical experiments. There are several concretizations of the effect size in the literature [69–71]. Here, we consider a well-refined measure introduced by Hedges [70] that is often referred to as "Hedges g^*" in the literature [72]. One class of standardized effect size measures essentially considers the difference

$$\frac{\mu_1 - \mu_2}{\sigma} \tag{4.11}$$

of the mean values μ_1, μ_2 of two sample groups relative to some uncertainty measure σ [72].

Hedges [70] considers the pooled standard deviation estimate

$$(\sigma^*)^2 := \frac{(n_1 - 1)\sigma_1^2 + (n_2 - 1)\sigma_2^2}{n_1 + n_2 - 2}, \tag{4.12}$$

where n_i is the sample size and σ_i is the standard deviation of sample group $i = 1, 2$. To ensure an unbiased estimator for the effect size, Hedges [70] proposes an additional scaling factor

$$J(a) := \frac{\Gamma\left(\frac{a}{2}\right)}{\sqrt{\frac{a}{2}}\Gamma\left(\frac{a-1}{2}\right)}. \tag{4.13}$$

With those definitions,

$$g^* := J(n_1 + n_2 - 2)\,\frac{\mu_1 - \mu_2}{s^*} \tag{4.14}$$

is the best estimator for the effect size [70, Theorem 3]. Based on the absolute value of the effect size, multiple categories (see Table 4.1) for the practical relevance of the experiment were proposed by Cohen [69, pp. 24–27].

Table 4.1: Sevaral categories for the effect size are defined based on the absolute value of g^*.

Category	Range of effect size
neglectable	$\lvert g^*\rvert < 0,2$
small	$0,2 \leq \lvert g^*\rvert < 0,5$
medium	$0,5 \leq \lvert g^*\rvert < 0,8$
large	$0,8 \leq \lvert g^*\rvert$

4.1.3 Remarks on the implementation

This section briefly describes how the algorithms are implemented in the programming language Python (version `3.10.6`). Implementation of the quantum mechanical parts of the algorithms is based on the *Quantum Toolbox in Python* (QuTiP) library in version `4.7.0` [73, 74]. Classical calculations are done using the *Scientific Python* (SciPy) library in version `1.9.1` [75].

The implementation is centered around a class `QAOA`, which gathers various methods for QAOA-circuit generation. The optimization algorithms are implemented as functions acting on an instance of the `QAOA` class. The entire code can be found in a GitHub repository of the author [76].

The `QAOA` class has properties for all essential parameters of QAOA. For example, there are properties for the p-value, the QUBO-matrix of the given problem in symmetric

form and the Hamiltonian of the problem. The most important method of the class is a constructor of the QAOA-circuit (2.7) for a given parameter vector δ. This is done by utilizing the `qutip.qip.circuit` class and concatenating all gates in (2.7). There is the option to include an arbitrary matrix of correct size at some position as well as the option to delete blocks in a given range. These options are used in separate methods for computing the metric and the gradient of QAOA, where the option to delete gates is utilized to achieve the same gate count when computing the metric as in Theorem 3.2.
Building on that circuit constructor, there are various methods. For example, there is a method for computing the final state after running the circuit using `qutip.qip.circuit.run` or one for the expectation value of the problem Hamiltonian for this state using `qutip.expect(H, final_state)`. This expectation method is then passed as a cost function to the ordinary optimizer.

Two additional methods are defined for computing the metric and gradient of QAOA. Instead of directly implementing the compiled metric (Equation (3.7)) and the compiled gradient (Equation (3.15)), a trick is used for faster simulation. The reason to expand the general expressions (3.3) and (3.14) is that H_P and B are not necessarily unitary and hence do not operate as quantum gates. However, the simulation of quantum circuits is done by simple matrix multiplication, which does not require the matrices to be unitary. Therefore, implementing Equation (3.3) and (3.14) in a simulation is possible. Exact simulations will yield the same results as for the implementation of Equation (3.7) and (3.15) by construction, as no noise is being simulated. This benefits the need of simulating fewer quantum circuits and enables the simulation on ordinary hardware.
In the implementation of the metric and gradient, the decision of which matrices at which position to insert in the QAOA-circuit is done by multiple **`if ... else`** statements checking whether each index lies in $\{1, \ldots, p\}$ or $\{p+1, \ldots, 2p\}$, i.e. whether a β or γ parameter is being differentiated.
Besides this non-unitary implementation for purposes of exact simulation, the compiled versions (3.7) and (3.15) are also implemented directly.

The optimization algorithms of the QAOA parameters are implemented as functions taking objects of the `QAOA` class and an initial parameter vector as arguments. Each one of them tracks the parameter vector during the optimization and counts the number of steps taken. Together with the number of circuits and Pauli gates for one step, this gives the measures M2 and M1.
The `scipy.minimize` implementation of the *Constrained Optimization BY Linear Approximation* (COBYLA) algorithm A3 serves as a commonly considered reference optimizer [29–31, 75]. Implementation of this algorithm is straightforward by handing the expectation method of the QAOA class to the `scipy.minimize` API. All steps compute exactly one circuit compiled into N gates (Equation (3.11)). A maximal number of iterations is set to 1000.

The standard gradient descent A2 is implemented by fixing a stepsize $\Delta = 0,1$ and computing the gradient $\nabla f_p(\delta)$ of f_p at δ by a finite difference quotient

$$\nabla_j f_p(\delta) = \frac{f_p(\delta + \epsilon e_j) - f_p(\delta)}{\epsilon}, \quad 1 \leq j \leq 2p, \tag{4.15}$$

where $\epsilon = 10^{-10}$ and e_j is the j-th vector of the euclidean standard basis in $\mathbb{R}^{2p}$. The algorithm terminates when the norm $\|\nabla f_p(\delta)\|$ of the gradient becomes smaller than some fixed precision goal (10^{-2}) or a maximal number of steps is reached. This maximal number of steps is set equal to the average number of steps within the given class of instances (n,p) it took for the TDVP (A1) to reach its precision goal (see below). This is done because the simulation of the standard gradient descent exceeds reasonable run times for $p \geq 3$ on the hardware available to the author. Each step of the GD calls the QAOA-circuit $4p$ times.

The implementation of the TDVP-optimization A1 is backed by the methods defined in the QAOA class and by the `scipy` library. A local function for computing the right-hand side of Equation (2.49) is defined. This function computes the QAOA metric and gradient (Chapter 3) using the methods of the QAOA-class, inverts the metric using `scipy.linalg.inv`, selects the real parts using `numpy.real` and finally multiplies the metric with the gradient. The algorithm counts the number of calls of this function to track the number of steps taken. The number of circuits and gates can then be computed by Equation (3.9) and (3.16) or Equation (3.10) and (3.17), respectively. For solving the ODEs in Equation (2.49) `scipy.integrate.solve_ivp` is used with the default method `'RK45'`, which is an implementation of the Runge-Kutta algorithm [77]. The solver is set to terminate whenever $\left\| g^{j,k} \operatorname{Re} \langle \partial_k \psi_p(\delta) | \, H \, | \psi_p(\delta) \rangle \right\|$ becomes smaller than some given precision goal (10^{-2}) or when a maximal number of steps is reached. When integrating with `scipy.integrate.solve_ivp`, a finite integration interval must be set. In the implementation, it is ensured that the algorithm continues in case the end of this integration interval is reached, without one of the two terminating events occurring.

4.2 Results

This chapter discusses the results of numerical simulations as described above, with a special focus on the research questions Q1 – Q3. In Section 4.2.1 the results of the quality measures are discussed and analyzed. Section 4.2.2 continues with the resource measures before the efficiency measures are discussed in Section 4.2.3.

4.2.1 Quality comparison (Q1)

In this section, the results for the quality measures M1 to M4 are being discussed with regard to research question Q1.

First, the ratio of successful simulations differs between the three algorithms, in the sense that the respective algorithm reached its termination goal within the given number of

Table 4.2: The success ratios of each algorithm. A simulation is counted as a success if the algorithm is terminated due to reaching its set precision goal. The counts are shown as $^{\text{sucessfull}}/_{\text{total}} = \text{ratio}$

p	TDVP	COBYLA	GD
1	$^{18}/_{18} = 1$	$^{18}/_{18} = 1$	$^{0}/_{18} = 0$
2	$^{17}/_{18} = 0.94$	$^{18}/_{18} = 1$	$^{0}/_{18} = 0$
3	$^{12}/_{16} = 0.75$	$^{16}/_{16} = 1$	$^{0}/_{16} = 0$
4	$^{10}/_{16} = 0.63$	$^{16}/_{16} = 1$	$^{0}/_{16} = 0$
5	$^{3}/_{6} = 0.5$	$^{6}/_{6} = 1$	$^{0}/_{6} = 0$

steps (Table 4.2, p. 42). Recall that the maximal number of steps for the GD is set to the average number of steps it took the TDVP to reach its precision goal (the same as for GD). Hence, the 0 success rate of GD means that it never reached the same precision as TDVP within a comparable number of steps. For equal convergence speed the same success rate is expected, so this result already indicates faster convergence of the TDVP. However, especially in the case $p = 5$ the TDVP shows a worse success rate compared to COBYLA. Here, it only terminated successfully for half of the instances. The reason for these failures is (in some cases) a `scipy.LinAlgError` during the inversion of the metric, but in most cases an inbuild `ValueError`. This could be a bug in the implementation, but the reason for this could not be found. All such failures due to errors during the simulation are ignored in the following results. In the case of GD all failures are due to not reaching a gradient with length below the set precision goal (10^{-2}) and hence are considered in the following results.

Some particular instances show significantly slower convergence speed with the TDVP (A1) and therefore fail to reach the precision goal within reasonable computation time.

TDVP vs. GD. The results for the approximation ratio (M1) (Figure 4.1, p. 48) clearly show a benefit of TDVP (A1) and COBYLA (A3) over the GD (A2) in all considered cases. Not only the mean values for GD are significantly lower, but also the standard deviation is much higher. The effect size (Table 4.3) indicates an overall large effect of the approximation ratio when comparing the GD and the TDVP. This means that the answer of GD (the highest probability bitstring) is way less reliable than the answer of the other algorithms.

The same tendency continues, when considering the expectation of the approximation ratio (M2) (Figure 4.2, p. 49). Although the standard deviation of GD is not as extreme as in the approximation ratio, the mean values are even worse. The effect size shows in all cases a large value, with especially high values in the cases $(n = 4, p = 2)$ and $(n = 5, p = 3)$. Another point, only noticeable when considering the expectation of the

Table 4.3: The effect sizes of the quality measures. We compare the categorical magnitudes (Table 4.1) of the effect sizes and Hedges g^* (4.14) [70]. The sign of the effect size indicates whether the TDVP (positive) or the respective comparison algorithm (negative) has a higher mean value.

		Approximation ratio M1		Expected approx. ratio M2	
		COBYLA	GD	COBYLA	GD
n	p				
4	1	none (−0.07)	large (1.21)	none (−0.08)	large (1.79)
	2	none (0.00)	large (4.62)	medium (0.58)	large (6.59)
	3	none (0.00)	medium (0.66)	medium (0.73)	large (3.03)
	4	small (0.47)	large (1.17)	large (0.86)	large (4.55)
	5	none (0.00)	large (1.18)	small (0.38)	large (4.62)
5	1	medium (−0.69)	large (1.53)	small (−0.49)	large (1.54)
	2	none (0.20)	large (1.19)	large (1.02)	large (4.13)
	3	medium (0.57)	large (1.89)	large (0.87)	large (6.16)
	4	medium (0.56)	large (1.09)	none (−0.01)	large (4.48)
		Groundstate overlap M3		Groundstate sharpness M4	
		COBYLA	GD	COBYLA	GD
n	p				
4	1	small (0.21)	large (1.18)	none (−0.08)	none (0.17)
	2	medium (0.60)	large (18.44)	small (−0.46)	large (1.32)
	3	medium (0.57)	large (5.38)	small (−0.40)	none (−0.17)
	4	medium (0.78)	large (4.53)	medium (−0.57)	small (−0.38)
	5	small (0.43)	large (6.40)	small (−0.35)	medium (−0.60)
5	1	none (−0.16)	medium (0.75)	medium (−0.56)	medium (0.59)
	2	large (1.09)	large (3.50)	none (0.01)	large (0.87)
	3	large (0.81)	large (6.08)	none (0.11)	large (1.35)
	4	none (0.12)	large (3.86)	medium (0.53)	large (1.53)

approximation ratio (M2), especially in Figure 4.2a, is that in contrast to (A1) and (A3) the GD (A2) tends to a slightly decreasing expectation of the approximation ratio with growing p. This indicates that the GD is more vulnerable to the increasing dimension of the search space.

As expected, the bad performance of the GD (A2) is also reflected in the results for the overlap with the ground states (Figure 4.3, p. 50). It is worth mentioning, that the performance of GD in the instance class ($n = 4, p = 2$) and ($n = 5, p = 1$) with almost no overlap with the ground states is especially bad compared to the other algorithms. While the first case shows an extremely high effect size ($g^* \geq 18$), the latter only shows a medium effect size ($g^* = 0,75$).

Regarding M4 the TDVP shows higher uncertainty of the state to lie in the ground states in the cases ($n = 4, p = 2$) and ($n = 5, p \geq 2$) with large effect size. In all other cases, only differences with none to medium effect size can be observed.

TDVP vs. COBYLA. Differences between the TDVP (A1) and the COBYLA (A3) are more subtle. First, both algorihtms show worst perfomance in the case $p = 1$ with respect to the approximation ratio (M1) (Figure 4.1, p. 48), the expectation of the approximation ratio (M2) (Figure 4.2, p. 49) and the overlap with the ground states (M3) (Figure 4.3, p. 50). Here, both algorithms are not as reliable as in the other cases, since the standard deviation is relatively high. For $n = 4$ the TDVP (A1) finds the optimal answer every time for each $p \geq 2$, while the COBYLA (A3) sometimes failes to do so for $p = 4$ (Figure 4.1a, p. 48). However, the effect size indicates a low practical relevance of these differences. Other for $n = 5$, where the TDVP shows higher approximation ratios than the COBYLA for $p \geq 2$ with medium effect size in the cases $p > 2$. The TDVP finds the optimal answer every time when $p = 3$ or $p = 4$ are simulated. Only for $p = 1$ does the COBYLA show a better approximation ratio.

When taking the entire final QAOA-state into account, the TDVP (A1) shows slightly better results compared to the COBYLA (A3) (Figure 4.2, p. 49). The effect sizes (Table 4.3) indicate better results in the cases ($n = 4$, $p = 4$), ($n = 5$, $p = 2, 3$) with large relevance, but only none to medium relevant differences in the other cases. Overall, the TDVP shows a higher expectation of the approximation ratio for most cases with medium to large relevance.

Similarly, the overlap with the ground states (Figure 4.3, p. 50) generally shows higher values for the TDVP with medium to large effect sizes in most cases. For $n = 4$, the effect sizes are lower than for $n = 5$. However, the uncertainty of the results is higher for both algorithms.

The uncertainty (M4) of the final state to lie in the ground states is almost the same in every case for all algorithms (Figure 4.4, p. 51). Differences only arise with mostly none to medium effect size. Overall, the uncertainty of the state in the ground states is not

Table 4.4: The effect sizes of the resource measures. This table shows the categorical magnitude (Table 4.1) of the effect sizes and Hedges g^* (4.14) [70]. The sign of the effect size indicates whether the TDVP (positive) or the respective comparison algorithm (negative) has a higher mean value.

		Gatecount M1		Circuit count M2	
		COBYLA	GD	COBYLA	GD
n	p				
4	1	large (6.27)	large (6.02)	large (13.47)	large (13.14)
	2	large (1.10)	large (1.09)	large (1.32)	large (1.29)
	3	large (0.90)	large (0.90)	large (0.85)	large (0.84)
	4	large (4.32)	large (4.32)	large (5.06)	large (5.02)
	5	large (1.92)	large (1.92)	large (1.71)	large (1.70)
5	1	large (2.75)	large (2.67)	large (4.10)	large (4.06)
	2	large (1.62)	large (1.61)	large (2.06)	large (2.04)
	3	large (3.25)	large (3.24)	large (4.46)	large (4.44)
	4	large (1.06)	large (1.06)	large (1.06)	large (1.06)

that high.

Regarding a final answer to research question Q1, a benefit of the TDVP (A1) (and the COBYLA (A3)) over the standard gradient descent (A2) can be observed with overall large effect size. A slight tendency of the TDVP delivering results with better quality compared to those of COBYLA becomes visible however, this tendency shows lower effect sizes. Depending on the parameters (n, p), TDVP (A1) proves to deliver higher quality results compared to COBYLA (A3).

4.2.2 Computational resources (Q2)

This section presents the results for the resource measures M1 (Figure 4.5, p. 52) and M2 (Figure 4.6, p. 53) addressing question Q2. For $(n = 4, p = 2)$, $(n = 4, p = 3)$ (Figure 4.5a, p. 52) and $(n = 5, p = 4)$(Figure 4.5b, p. 52) for the gate count and $(n = 4, p = 3)$ (Figure 4.6a, p. 53) and $(n = 5, p = 4)$ (Figure 4.6b, p. 53) for the circuit count, the standard deviation of the results of the TDVP higher than their mean value, so those results must be treated with caution. In all cases, the number of gates as well as the number of circuits increases with growing p for each algorithm. In addition, the TDVP has the highest values and COBYLA the lowest among the three algorithms in every case. The difference between the GD and the COBYLA seems to have a constant factor, while the factor between TDVP and the other two algorithms seems to be increasing in p. The lowest factor in the number of gates between the GD and TDVP is around 100, while the highest factor is around 1000. All effect sizes show large relevance to the results.

Table 4.5: The effect sizes of the efficiency measures. The categorical magnitude (Table 4.1) of the effect sizes and Hedges g^* (4.14) are shown [70]. The sign of the effect size indicates whether the TDVP (positive) or the respective comparison algorithm (negative) has a higher mean value.

		Expected approx. ratio per circuit M1		Relative path length M2
		COBYLA	GD	GD
n	p			
4	1	large (−5.96)	large (−0.91)	large (1.59)
	2	large (−4.80)	none (0.17)	large (7.13)
	3	large (−2.67)	none (−0.04)	large (4.05)
	4	large (−21.16)	small (0.34)	large (7.58)
	5	large (−2.09)	small (0.36)	large (4.16)
5	1	large (−5.45)	large (−1.01)	large (2.14)
	2	large (−14.12)	small (−0.44)	large (3.77)
	3	large (−30.04)	small (0.30)	large (5.46)
	4	large (−5.82)	none (0.02)	large (7.05)

Answering Q2 it can be said that the TDVP (A1) needs extensively more quantum circuits and gates compared to both, the GD A2 and COBYLA (A3).

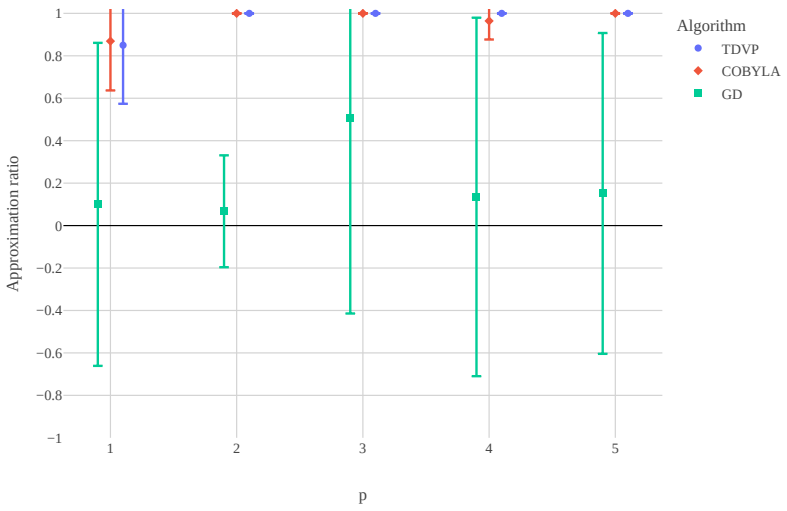
4.2.3 Analysis of efficiency (Q3)

Referring to the algorithms' efficiency (Q3), this section discusses the results of the efficiency measures M1 (Figure 4.7, p. 54) and M2 (Figure 4.8, p. 55). The expected approximation ratio per circuit (M1) shows immediately that the COBYLA (A3) is the most efficient algorithm from the practical perspective (Figure 4.7, p. 54). Here, the comparison with the TDVP shows overall large effect sizes up to 30. The results for the TDVP (A1) and GD (A2) suggest that they are both less efficient than COBYLA, even if some cases show standard deviation higher than the mean value for the GD. For $p = 1$ the TDVP shows worse values compared to the GD with a large effect size. In all other cases, both algorithms have differences with only none to small effect size indicating that they offer similar quality of the results per circuit.

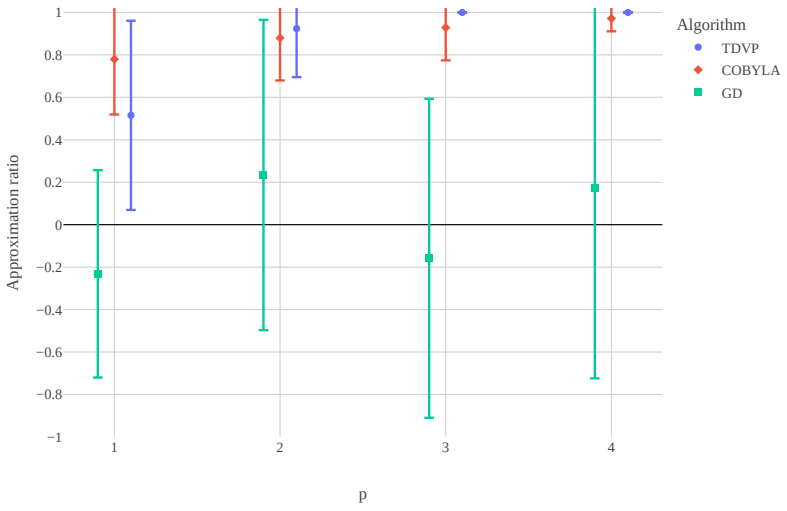
The relative path length (M2) only makes sense for the TDVP (A1) and GD (A2) and is the highest for the TDVP (A1) (Figure 4.8, p. 55). This indicates that the TDVP takes routes closer to the shortest path, compared to the GD, which takes far less efficient paths in the parameter space. The effect sizes are all large in this comparison.

The reason for the different path lengths of the TDVP and GD becomes visible when comparing the paths taken in the energy landscape (Figure 4.9, p. 56 and Figure 4.10, p. 57). The GD is more vulnerable to overshoot local minima, especially when the energy

gradient is high as in the case shown in Figure 4.9. Although the initial point is close to a local minimum, the GD jumps out of its valley, while the TDVP manages to consistently decrease towards the local minimum. Another faulty behavior of the GD can be observed at the instance discussed previously in Section 2.1.4 (Figure 2.1, p. 20). Here, the GD takes the steepest direction ending up in a shallow valley, while the TDVP starts in a different direction leading to a lower local minimum (Figure 4.10, p. 57).

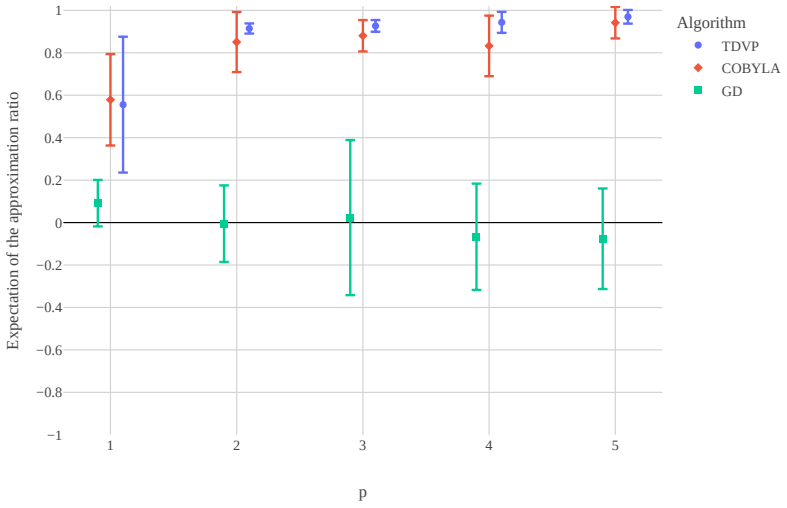


(a) Problem size $n = 4$

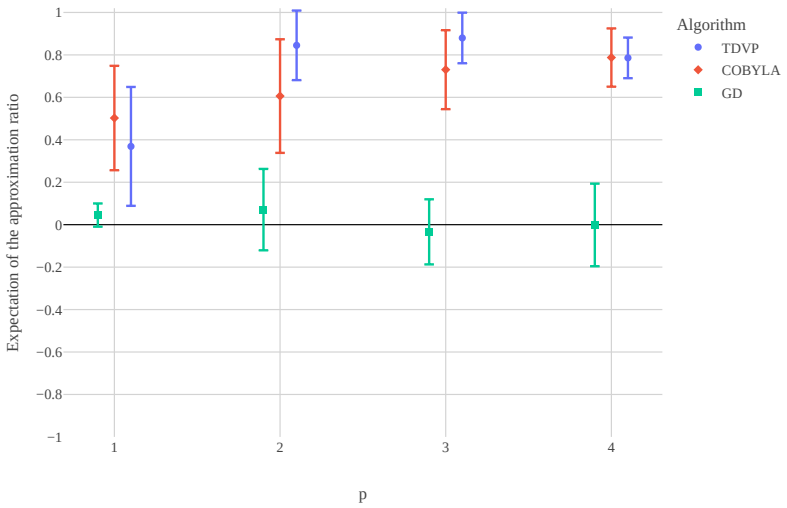


(b) problem size $n = 5$

Figure 4.1: Results of the approximation ratio M1 for different problem sizes. The different colors and symbols correspond to the three different algorithms A2 to A1. The figures show the mean values over all simulations together with their standard deviation.

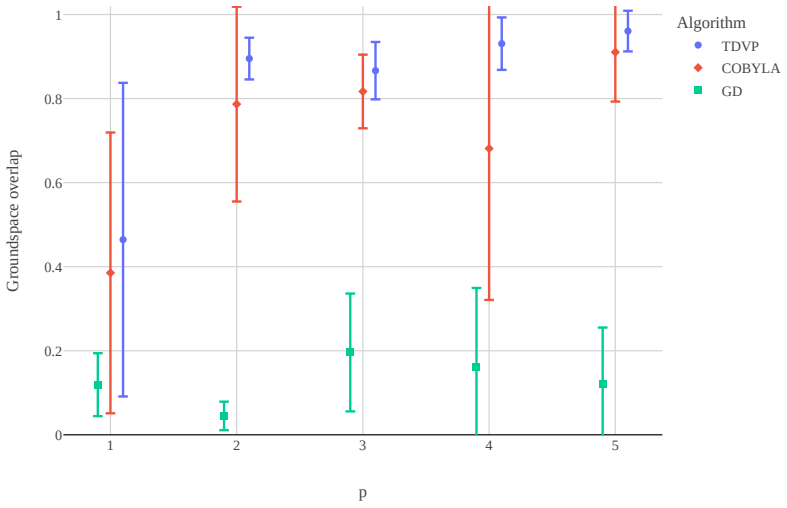


(a) Problem size $n = 4$

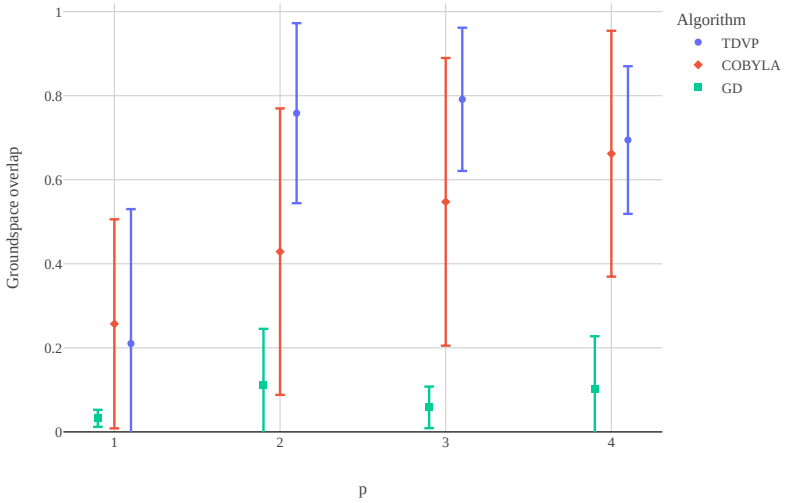


(b) Problem size $n = 5$

Figure 4.2: Results of the expected approximation ratio M2 for different problem sizes. The different colors and symbols correspond to the three different algorithms A2 to A1. The mean values over all simulations together with their standard deviation are presented.

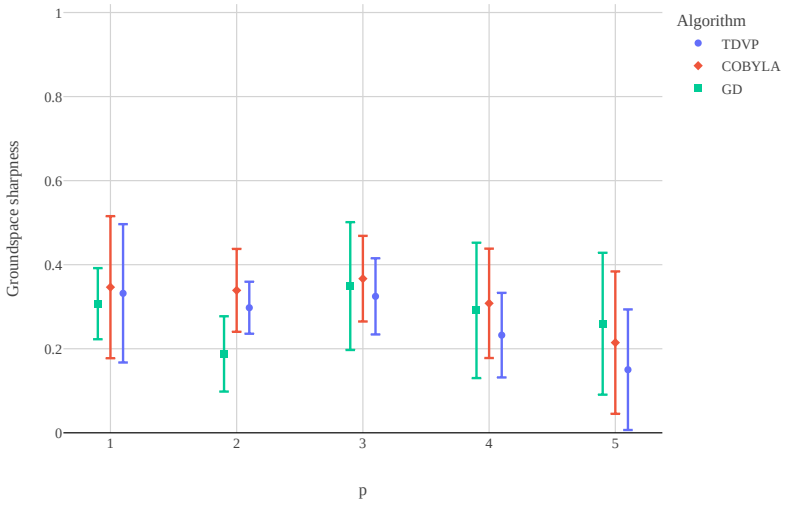


(a) Problem size $n = 4$

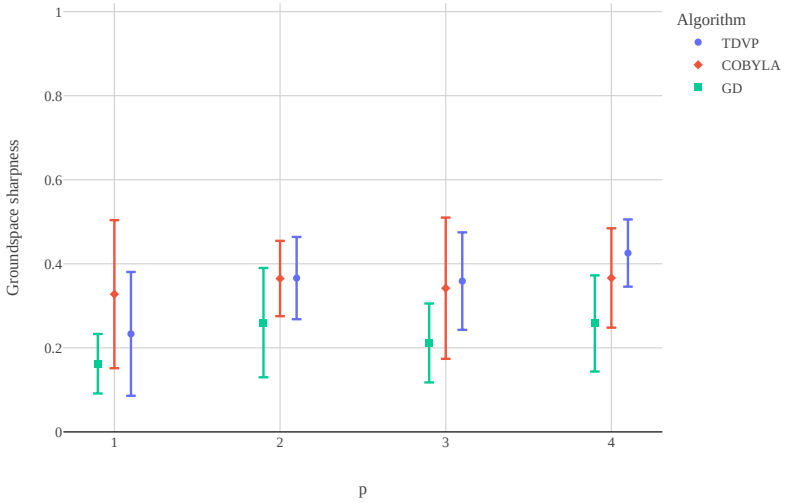


(b) Problem size $n = 5$

Figure 4.3: Results of the ground state-overlap M3 for different problem sizes. The different colors and symbols correspond to the three different algorithms A2 to A1. The figures show the mean values over all simulations together with their standard deviation.

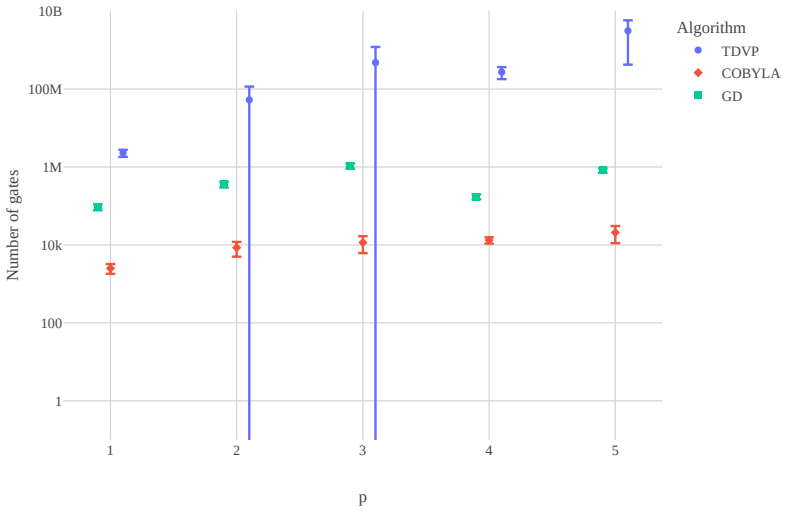


(a) Problem size $n = 4$

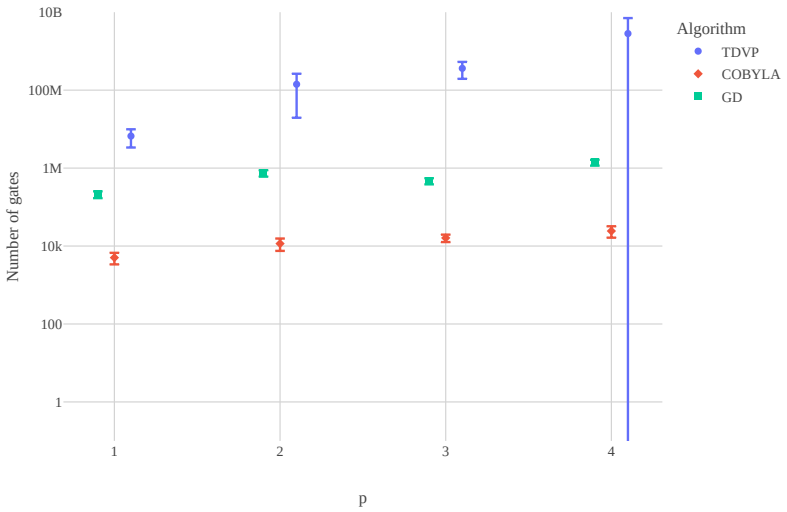


(b) Problem size $n = 5$

Figure 4.4: Results of the sharpness in the ground states M4 for different problem sizes. The different colors and symbols correspond to the three different algorithms A2 to A1. The figures show the mean values over all simulations together with their standard deviation.

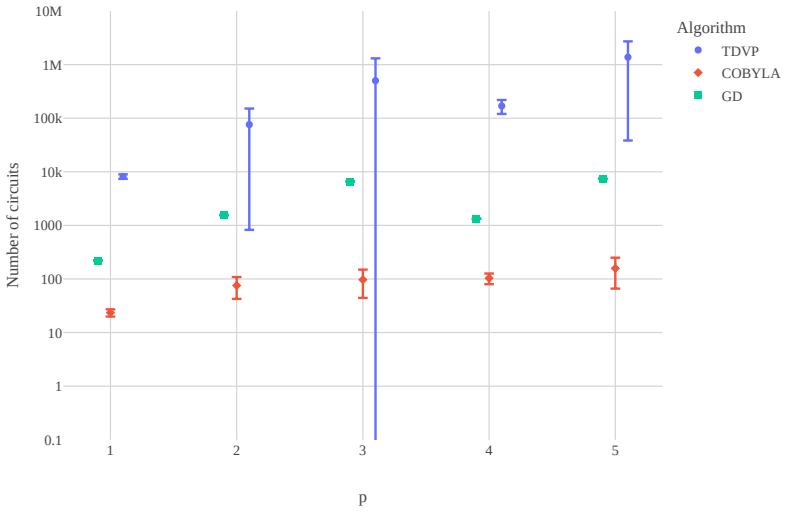


(a) Problem size $n = 4$

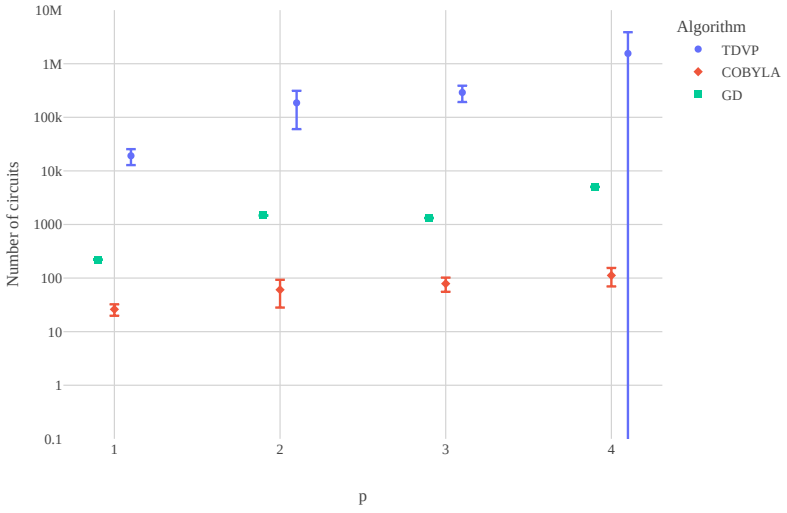


(b) Problem size $n = 5$

Figure 4.5: Results of the gate count M1 for different problem sizes. The different colors and symbols correspond to the three different algorithms A2 to A1. The figures show the mean values over all simulations together with their standard deviation in a logarithmic scale.

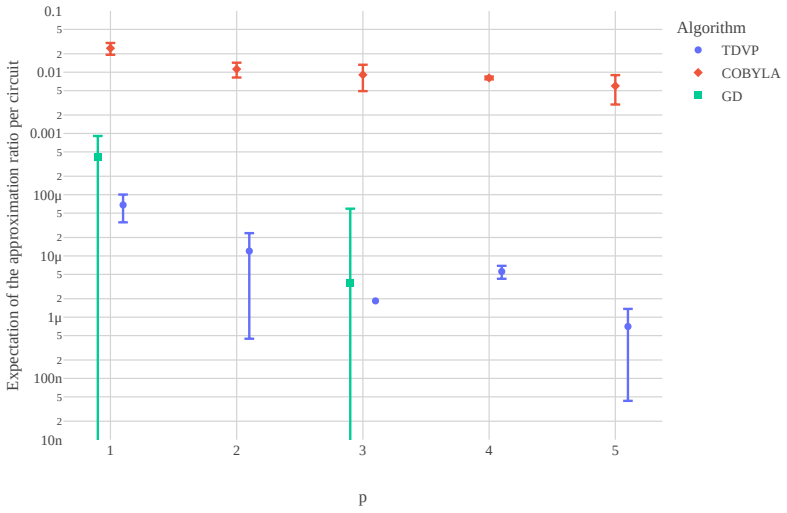


(a) Problem size $n = 4$

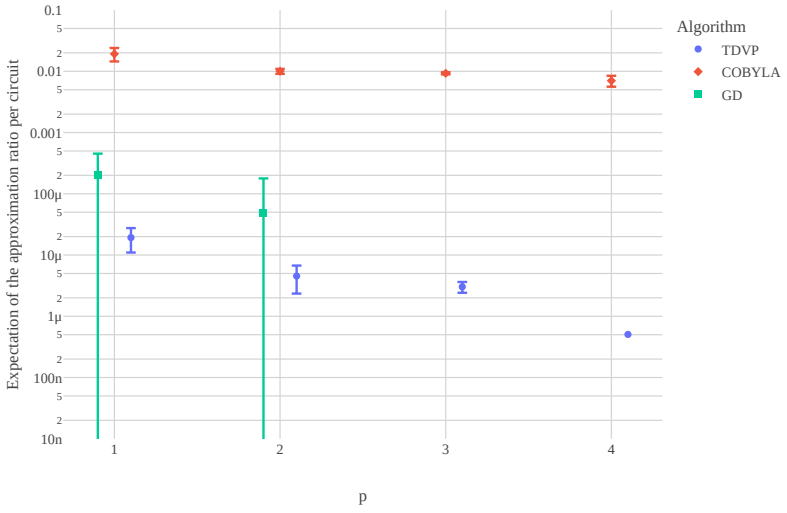


(b) Problem size $n = 5$

Figure 4.6: Results of the circuit count M2 for different problem sizes. The different colors and symbols correspond to the three different algorithms A2 to A1. The figures show the mean values over all simulations together with their standard deviation in a logarithmic scale.

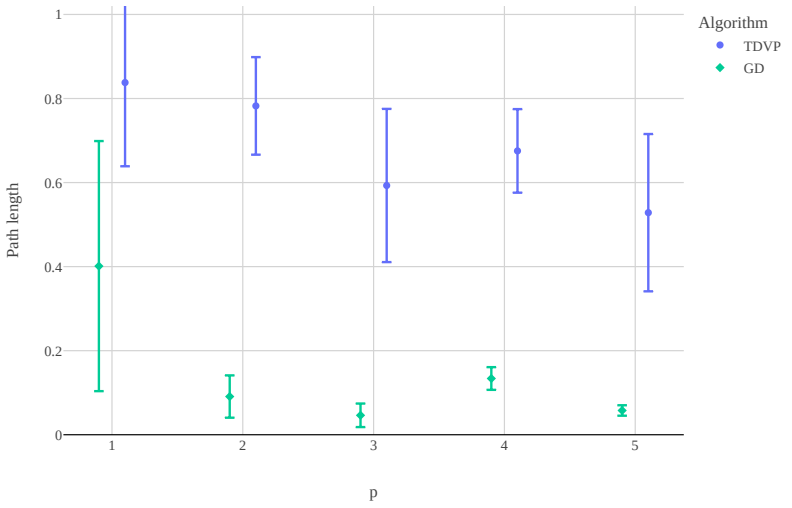


(a) Problem size $n = 4$

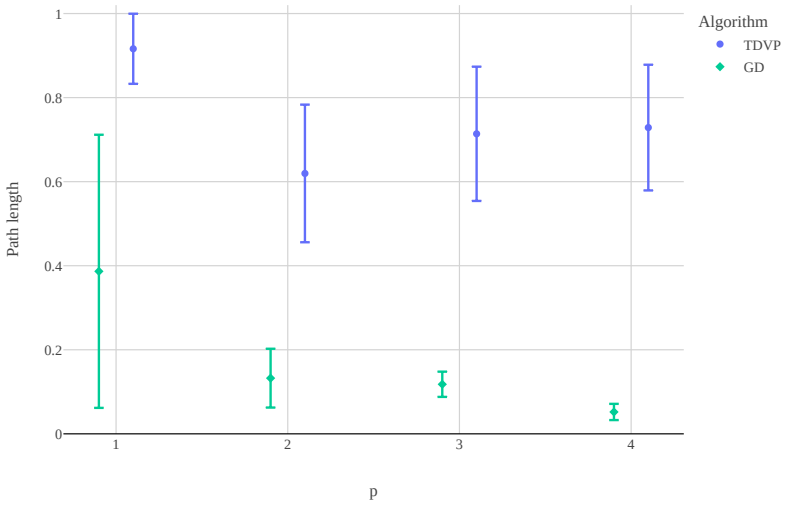


(b) Problem size $n = 5$

Figure 4.7: Results of the expected approximation ratio per circuit M1 for different problem sizes. The different colors and symbols correspond to the three different algorithms A2 to A1. The figures show the mean values over all simulations together with their standard deviation on a logarithmic scale. For GD, some values are negative and hence not visible on the logarithmic y-axis.



(a) Problem size $n = 4$



(b) Problem size $n = 5$

Figure 4.8: Results of the path length M2 for different problem sizes. The different colors and symbols correspond to the three different algorithms A2 to A1. The figures show the mean values over all simulations together with their standard deviation on a logarithmic scale.

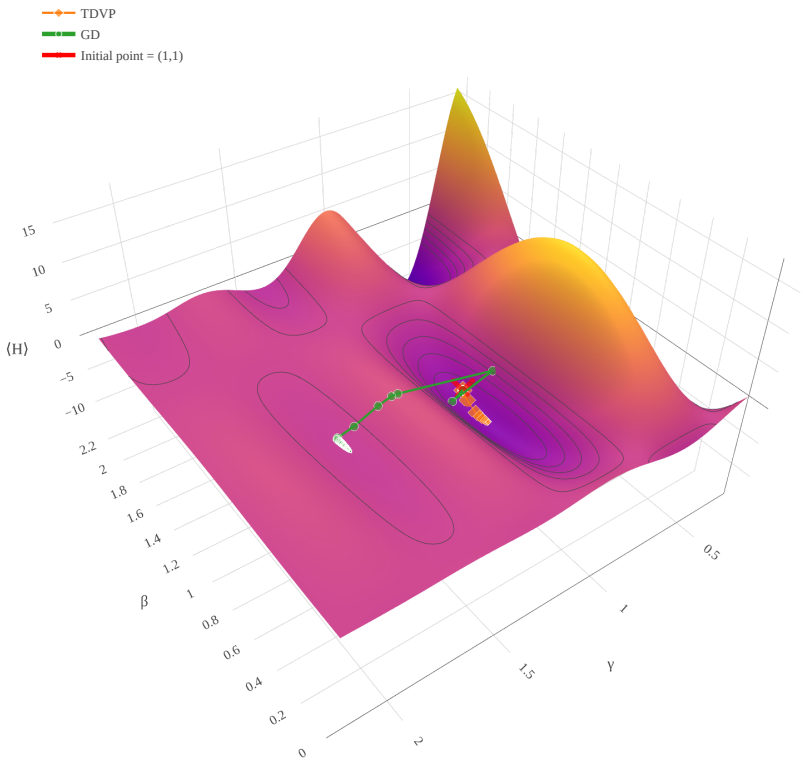


Figure 4.9: A surface plot of the energy landscape illustrating one of the benefits of the TDVP over the standard gradient descent. Shown is the energy landscape of the expectation value of the problem Hamiltonian H_P of one particular Max-Cut instance (Section 2.1.4) of size $n = 5$ with $p = 1$. The instance is chosen such that the difference in path length between the GD and TDVP out of all the considered instances is the highest. The size of the points decreases as the paths continue.

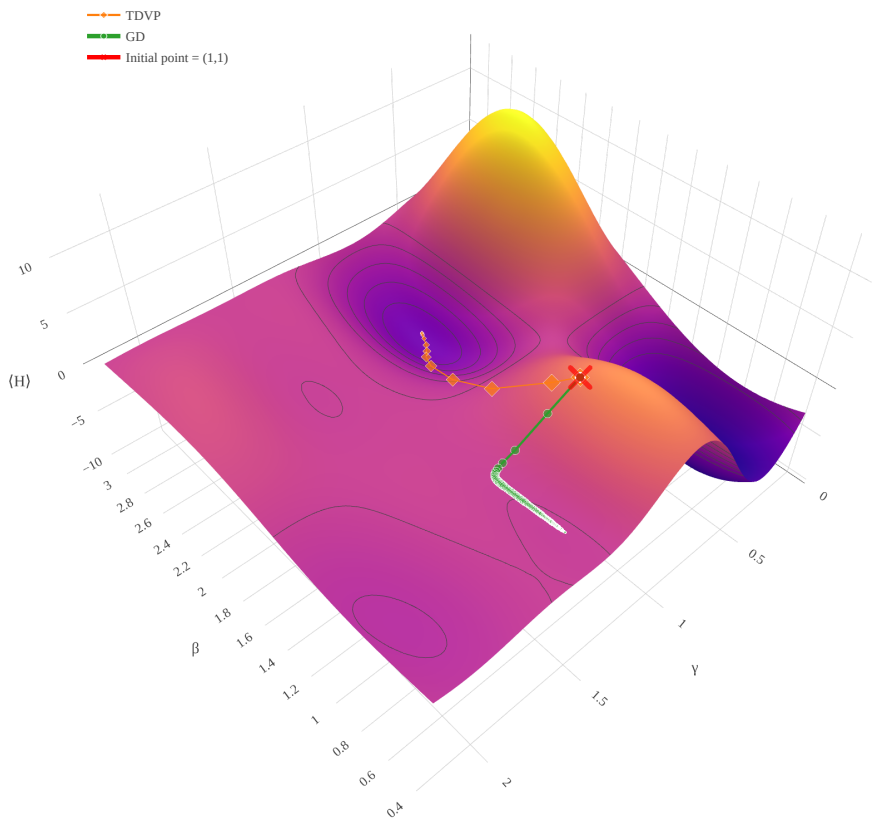


Figure 4.10: A surface plot of the energy landscape illustrating one of the benefits of the TDVP over the standard gradient descent. Shown is the energy landscape of the expectation value of the problem Hamiltonian H_P of the Max-Cut-instance corresponding to the graph in Figure 2.1 (p. 20). The size of the points decreases as the paths continue.

Chapter 5

Conclusion and outlook

In this thesis, the *Time-Dependent Variational Principle* is considered for optimizing the parameters of the *Quantum Approximate Optimization Algorihtm*. The metric is calculated by quantum circuits very similar to the idea of QAOA and independently of McArdle et al. [67]. The performance of the QAOA with the parameters optimized by the TDVP is compared to the performance with the established COBYLA optimizer and the standard gradient descent GD.

As discussed in the preceding sections, there are differences in the quality and efficiency of all three algorithms. As may be expected, the QAOA with the well-established COBYLA algorithm (A3) is by far the most efficient one. This might be due to a more sophisticated and polished implementation in the `scipy` library. But more importantly, the main drawback of the TDVP is the need for computing the metric (3.7) in every single step. The size of the metric grows of order $\mathcal{O}\left(p^2\right)$ and thus quickly leads to a number of quantum circuit evaluations per step that is relatively large compared to the single quantum circuit evaluation per step in the COBYLA. On the other hand, the quality of the results of the TDVP reflects the more specialized nature of the algorithm. While COBYLA only takes the parameter space into account and handles the QAOA only as a function, the TDVP also recognizes the geometric structure of the QAOA-state space. This may be suspected to be the reason for the slightly better quality of the results.

Moreover, the convergence behavior of the TDVP is significantly better than the convergence of the GD. This convergence behavior shows the benefits of recognizing the geometry of the state space and taking higher orders during the solving of the ODEs into account. However, while the quality of the results is way better for the TDVP, the resource consumption is also significantly higher. This results in better gate efficiency of the GD compared to the TDVP. Recall that in the simulation of the GD the number of steps is limited to the average number of steps it took for the TDVP to terminate, so both algorithms were allowed to take the same number of steps on average. In this sense, the better quality of the results of the TDVP compared to the results of the GD indicates that the TDVP converges faster.

As discussed in Section 2.2, the main differences between the algorithms are the appearance of the metric and the higher order algorithm for solving the differential equation in the

TDVP. These differences are suspected to result in the shorter path length of the TDVP confirming the optimality of the curve taken. This becomes visible when we compare the paths taken in the energy landscape (Figure 4.9, p. 56 and Figure 4.10, p. 57). In addition, the observation that the TDVP is less vulnerable to increasing parameter space dimension compared to the GD is justified by this interpretation. The information about the global geometry of the state space could help to prevent walking in the wrong direction. The GD is exclusively local and becomes increasingly blind to its surroundings when increasing the dimension. Future simulations of larger problem instances could show that the TDVP converges faster than the GD.

As a side effect, another conclusion is that the expectation of the approximation ratio M2 seems to be a better performance indicator compared to the approximation ratio. By not only considering the answer to the problem but taking the entire final state of the algorithm into account, more differences between different parameter optimizations could be observed.

In summary, the TDVP proved to be beneficial over the GD only in some perspectives. Compared to the well-established COBYLA the quality of the results is surprisingly better, but with less practical relevance of the results indicated by the effect size. However, this slight benefit in quality comes with the cost of a huge increase in the computational resources needed. In the end, the decision of which algorithm is suited better may depend on the situation, but the slight increase in quality does not justify the additional resource consumption in most situations. Further research considering the performance of the algorithm on real quantum hardware might yield additional insights into this trade-off.

An aspect to highlight in the present work is that the simulation results are interpreted not only by mean values but also by the effect size from statistics. Proper statistical analysis of simulations may give more reliable results, also in future work. However, this statistical analysis can be improved by adapting the common practices from behavioral and medical sciences to the needs and circumstances of quantum algorithms. For example, the categories of the effect size for rating the relevance of results introduced by Cohen [69] might be chosen as suboptimal in this context. As highlighted by Cohen [69], the interpretation of effect sizes depends on the circumstances and assumptions of each particular statistical experiment. Future research could establish common interpretation guidelines for the statistical analysis of quantum algorithms in this context.

A downside is the implementation of the TDVP-optimization algorithm used for simulations, which might be far from optimized. For example, the implementation checks whether to continue by computing the gradient separately from the main algorithm during run time, which needs computation of the metric. This leads to an unnecessary increase in the circuit and gate count. More optimized implementations may avoid this problem by checking the norm of the gradient in each step at the same time it is computed for the optimization step. The failed simulations of the TDVP (Table 4.2) may be caused by another bug or by some intrinsic problem of the algorithm with these particular instances.

Using the pseudoinverse of the metric rather than the inverse may solve this problem. A detailed analysis of affected instances and also of the generally hard instances of the Max-Cut problem might give insights into the convergence behavior of the TDVP.

A deep differential geometric analysis of the QAOA state space following Haegeman, Mariën, Osborne and Verstraete [59] might give interesting explanations for the behavior of the QAOA. Additionally, being able to compute the metric not only enables algorithms like the TDVP to be applied but may also give access to the numerical analysis of many geometric properties of the QAOA-state space. This might prove as an interesting research direction in the future.

Most importantly, only exact simulations of the quantum circuits are considered and only pure quantum states are considered in the derivation of the TDVP. However, on real quantum devices noise resulting in mixed states is unavoidable in the current state of technology. In future research, the behavior of the TDVP under noise and on mixed quantum states needs to be analyzed by simulations as well as by rigorous mathematical analysis. For this purpose, the mathematical formulation should cover mixed quantum states, rather than restricting it to pure states. Does the TDVP still converge to a state in a local minimum of the energy landscape when starting in a mixed state or when the metric entries are given by probability distributions? Which conditions may lead to convergence under noise?

Appendix A

Proofs

This appendix presents proofs that are skipped in the main text.

Proof of Lemma 2.4 (p. 21). We give an explicit construction of a QUBO matrix. Let an arbitray instance of Max-Cut be given by a graph $G = (V, E)$ and $\{w_{j,k} \mid j, k \in V,\, w_{j,k} = 0, \text{for } jk \notin E\}$. Define the function $C\colon\ \mathcal{P}(V) \to \mathbb{R};\ C(U) = \sum_{j,k \in V,\, j \in U,\, k \notin U} w_{j,k}$. Encrypt which vertices lie in a given subset U by a binary string $x(U) \in \{0,1\}^{|V|}$, s.t.

$$x(U) \coloneqq x \coloneqq x_1 \dots x_n, \quad x_j = \begin{cases} 1 & (j \in U) \\ 0 & (j \notin U) \end{cases}.$$

$x_j \wedge (\neg x_k) = 1$ being equivalent to $x_j \neq x_k$, allows us to rewrite C as a function on binary strings.

$$C(U) = \sum_{\substack{j,k \in V \\ x_j \neq x_k}} w_{j,k} = \sum_{j,k \in V} w_{j,k}\big((x_j \wedge (\neg x_k)) \vee ((\neg x_j) \wedge x_k)\big) =: C(x(U))$$

$$\Rightarrow \arg\max_{U \in \mathcal{P}(V)} C(U) = \arg\max_{x \in \{0,1\}^{|V|}} C(x)$$

This already shows that Max-Cut can be formulated as a binary combinatorial optimization problem.

It remains to be shown that $C(x)$ is quadratic. Note that $\neg x = 1 - x$, $x \wedge y = xy$, $x \vee y = x + y - xy$ and either $x = 0$ or $1 - x = 0$, for $x, y \in \{0,1\}$. With this we have

$$\begin{aligned} C(x) &= \sum_{j,k \in V} w_{j,k}\big((x_j \wedge (\neg x_k)) \vee ((\neg x_j) \wedge x_k)\big) \\ &= \sum_{j,k \in V} w_{j,k}\big(x_j(1 - x_k) + (1 - x_j)x_k - \underbrace{x_j x_k (1 - x_k)(1 - x_j)}_{=0}\big) \\ &= -2 \sum_{j,k \in V} w_{j,k} x_j x_k + 2 \sum_{j,k \in V} w_{j,k} x_j \\ &= x^T Q x, \end{aligned}$$

where

$$Q_{j,k} = \begin{cases} w_{j,k} & (j \neq k) \\ \sum_{l=1}^{|V|} w_{j,l} & (j = k) \end{cases}. \tag{A.1}$$

□

Proof of Lemma 2.5 (p. 23). Assume $\psi(t)$ to be a curve for which $S\left[\bar{\psi}(t), \psi(t), t\right]$ is being extremalized. Consider $\bar{\psi}$ and ψ as independent variables and let $\bar{\psi}_\epsilon$ be an arbitrary variation of $\bar{\psi}$ s.t. $\frac{\mathrm{d}}{\mathrm{d}\epsilon}\psi_\epsilon\big|_{\epsilon=0}$ and $\psi_0 = \psi$. We omit all t dependencies for the sake of brevity. Now, as S is stationary for $\bar{\psi}$ we have:

$$\begin{aligned} 0 = \frac{\mathrm{d}}{\mathrm{d}\epsilon} S[\bar{\psi}_\epsilon, \psi]\Big|_{\epsilon=0} &= \int_{t_1}^{t_2} \frac{i}{2}\frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\dot{\psi}_\epsilon|\psi\rangle - \frac{i}{2}\frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\psi_\epsilon|\dot{\psi}\rangle - \frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\bar{\psi}|\, H\,|\psi\rangle\Big|_{\epsilon=0} \\ &= \left[\frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\psi_\epsilon|\psi\rangle\Big|_{\epsilon=0}\right]_{t_1}^{t_2} - \int_{t_1}^{t_2} \frac{i}{2}\frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\psi_\epsilon|\dot{\psi}\rangle \\ &\quad + \frac{i}{2}\frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\psi_\epsilon|\dot{\psi}\rangle + \frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\psi|\, H\,|\psi\rangle \\ &= -\int_{t_1}^{t_2} \frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\psi_\epsilon|\Big|_{\epsilon=0}\left(i\frac{\mathrm{d}}{\mathrm{d}t}|\psi\rangle - H\,|\psi\rangle\right) \end{aligned}$$

$\frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\psi_\psi|\big|_{\epsilon=0}$ is arbitrary as so is the variation $\bar{\psi}_\epsilon$. Therefore, ψ must satisfy the TDSE. □

Proof of Lemma 2.6 (p. 24). Assume $\psi\left(x(t)\right)$ extremalizes the action functional (2.35). Consider a variation $x(t) \mapsto x_\epsilon(t) = x(t) + \epsilon x'(t)$, with x' arbitrary except $x'(t_{1,2}) = 0$. We omit all time dependencies. Then, together with the chain rule, this gives $\frac{\mathrm{d}}{\mathrm{d}\epsilon}\langle\psi(x+\epsilon x')|\big|_{\epsilon=0} = x'^j\left\langle\partial_j\psi(x)\right|$ and $\dot{\psi}(x) = \dot{x}^k\partial_k\psi(x)$, so we can compute the variation to

$$\begin{aligned} 0 &= \frac{\mathrm{d}}{\mathrm{d}\epsilon} S\left(x_\epsilon\right)\Big|_{\epsilon=0} \\ &= \int_{t_1}^{t_2} \frac{i}{2}\left(\left\langle x'^j\partial_j\psi\middle|\dot{\psi}\right\rangle + \left\langle\psi\middle|\frac{\mathrm{d}}{\mathrm{d}t}x'^j\partial_j\psi\right\rangle\right) \\ &\quad - \frac{i}{2}\left(\left\langle\frac{\mathrm{d}}{\mathrm{d}t}x'^j\partial_j\psi\middle|\psi\right\rangle + \left\langle\dot{\psi}\middle|x'^j\partial_j\psi\right\rangle\right) \\ &\quad - \frac{i}{2}\left(\left\langle x'^j\partial_j\psi\right|H\,|\psi\rangle + \langle\psi|\,H\left|x'^j\partial_j\psi\right\rangle\right)\mathrm{d}t \\ &= \int_{t_1}^{t_2} \frac{i}{2}\left(x'^j\left\langle\partial_j\psi\middle|\dot{\psi}\right\rangle - x'^j\left\langle\dot{\psi}\middle|\partial_j\psi\right\rangle + x'^j\left\langle\partial_j\psi\middle|\dot{\psi}\right\rangle - x'^j\left\langle\dot{\psi}\middle|\partial_j\psi\right\rangle\right) \\ &\quad - x'^j\left(\langle\partial_j\psi|\,H\,|\psi\rangle + \langle\psi|\,H\,|\partial_j\psi\rangle\right)\mathrm{d}t \\ &= \int_{t_1}^{t_2} x'^j\left[i\left(\left\langle\partial_j\psi\middle|\dot{\psi}\right\rangle - \left\langle\dot{\psi}\middle|\partial_j\psi\right\rangle\right)\right. \\ &\quad \left. - \left(\langle\partial_j\psi|\,H\,|\psi\rangle + \langle\psi|\,H\,|\partial_j\psi\rangle\right)\right]\mathrm{d}t \end{aligned}$$

$$= -2 \int_{t_1}^{t_2} x'^j \left(\operatorname{Im} \left(\langle \partial_j \psi | \partial_k \psi \rangle \, \dot{x}^k \right) + \operatorname{Re} \left(\langle \partial_j \psi | \, H \, | \psi \rangle \right) \right) \mathrm{d}t \, .$$

Since x' is arbitrary, this implies Equation (2.37).

□

Proof of Theorem 2.3 (p. 27). We follow the proof given by Hackl et al. [58]. As described in Section 2.2.2, the proximum of $(E - H) |\psi\rangle$ to the tangent plane $T_\psi \mathcal{M}$ is given by the orthogonal projection

$$\begin{aligned} P_\psi((E - H) |\psi\rangle) &= 2 \, |\partial_j \psi\rangle \, g^{j,k} \operatorname{Re} \langle \partial_k \psi | \, (E - H) \, |\psi\rangle \\ &= -2 \, |\partial_j \psi\rangle \, g^{j,k} \operatorname{Re} \langle \partial_k \psi | \, H \, |\psi\rangle \, , \end{aligned}$$

where we used that $\partial_k \psi \in T_\psi \mathcal{M} \simeq \mathcal{H}_\psi^\perp$ is in the orthogonal complement

$$\mathcal{H}_\psi^\perp \coloneqq \{ \phi \in \mathcal{H} \mid \langle \phi | \psi_p(\delta) \rangle = 0 \}$$

of ψ. As the unique proximum in $T_\psi \mathcal{M}$ to $(E - H) |\psi\rangle$ is given by $P_\psi \left((E - H) |\psi\rangle \right)$ and as we have

$$\frac{\mathrm{d}}{\mathrm{d}\tau} |\psi(x(\tau))\rangle = \sum_j \dot{x}^j \, |\partial_j \psi(x(\tau))\rangle \, , \tag{A.2}$$

a comparison of coordinates in the frame $\{|\partial_j \psi\rangle\}$ implies Equation (2.49).

To see that the energy monotonically decreases notice that the metric $g_{j,k}$ is positive semidefinite as so is the inner product. This means $g_{j,k}$ has only positive eigenvalues and so has $g^{j,k}$. Therefore, the inverse $g^{j,k}$ is positive semidefinite, too. With this, we can estimate the differential of the energy by:

$$\begin{aligned} \frac{\mathrm{d}E}{\mathrm{d}\tau} = (\partial_j E) \dot{x}^k &= -(\partial_j E) g^{j,k} (2 \operatorname{Re} \langle \partial_k \psi | \, H \, |\psi\rangle) \\ &= -(\partial_j E) g^{j,k} \partial_j (\langle \psi | \, H \, |\psi\rangle) \\ &= -(\partial_j E) g^{j,k} (\partial_k E) \leq 0 \end{aligned}$$

□

Proof of Theorem 3.1 (p. 30) and 3.2 (p. 32). Only the proofs of Equations (3.10) and (3.17) are left open in the main text. Denote the number of single qubit R_X and R_Z rotations and CNOT-gates that a unitary circuit U consists of by $\#(U)$. Extend this notation for expectation values $\langle U \rangle$, by setting $\#(\langle U \rangle) \coloneqq \#(U)$. Then $\#(UU') = \#(U) + \#(U')$ and $\#(\langle UU' \rangle) = \#(\langle U \rangle) + \#(\langle U' \rangle)$. We use that the number of the first n integers is

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \tag{A.3}$$

and the sum of the first n squares is

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(n+2)}{6} = \frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n. \tag{A.4}$$

The number of gates in the QAOA-gates U_B and U_P as in (2.20) and (2.17) are $\#(U_B) = n$ and $\#(U_P) = n + 3\#(Q)$, where $\#(Q) \leq n^2$ is the number of non-zero elements in the qubo-matrix Q. Therefore, each QAOA-block consists of $N \coloneqq \#\left(U^{(p)}(\delta)\right) = p(2n + 3\#(Q))$ gates. Due to $U_B(-\delta_j)U_B(\delta_j) = \mathbb{I} = U_P(-\delta_j)U_P(\delta_j)$ for all j, a total of $p - k < p$ QAOA-blocks annihilate themselve in the middle of $U_j^{(p)}U_k^{(p)}$, $\tilde{U}_j^{(p)}U_k^{(p)}$, $U_j^{(p)}\tilde{U}_k^{(p)}$ and $\tilde{U}_j^{(p)}\tilde{U}_k^{(p)}$ (3.8) under the assumption that $j < k$.

Consider the expression for the QAOA-metric in (3.7). The annihilation of gates as described above means that in the expectations the initial number of gates for the adjusted QAOA-blocks $U_j^{(p)}$ and $\tilde{U}_j^{(p)}$ is reduced from N to

$$N(k) \coloneqq \begin{cases} (p - (p-k))(2n + 3\#(Q)) = \frac{k}{p}N & (k \leq p) \\ (p - (p - (k-p)))(2n + 3\#(Q)) = \frac{k-p}{p}N & (k > p), \end{cases}$$

when neglecting the inserted gates X_l, Z_l and Z_lZ_m. Taking the inserted gates into account and counting the gates in $\mathfrak{g}_{j,k}^{(p)}$ (3.3) using (A.3) yields a total of

$$\begin{aligned} \#\left(\mathfrak{g}_{j,k}^{(p)}\right) &= 3n^2\left(N(k) + 1 + N(k) + 1\right) + 3n\frac{n(n-1)}{2}\left(N(k) + 1 + N(k) + 2\right) \\ &\quad + \left(\frac{n(n-1)}{2}\right)^2\left(N(k) + 2 + N(k) + 2\right) \\ &= N(k)C_1(n) + C_2(n), \end{aligned} \tag{A.5}$$

where $C_1(n) = \frac{1}{2}n^4 + 2n^3 + \frac{7}{2}n^2$ and $C_2(n) = n^4 + \frac{5}{2}n^3 + \frac{5}{2}n^2$, and we used that

$$\sum_{l \leq m} 1 = \sum_{m=1}^{n}\sum_{l=1}^{m} 1 = \sum_{m=1}^{n} m \overset{(A.3)}{=} \frac{n(n+1)}{2}. \tag{A.6}$$

As g is self-adjoint, only the upper-triangular half of g need to be computed directly. For this we have

$$\begin{aligned} \sum_{j \leq k} \#\left(\mathfrak{g}_{j,k}^{(p)}\right) &\overset{(A.5),(A.6)}{=} \sum_{k=1}^{p}\sum_{j=1}^{k}\left(\frac{k}{p}NC_1(n) + C_2(n)\right) \\ &\quad + \sum_{k=p+1}^{2p}\sum_{j=1}^{k}\left(\frac{k-p}{p}NC_1(n) + C_2(n)\right) \\ &= \frac{N}{p}C_1(n)\sum_{k=1}^{p}\sum_{j=1}^{k} k + C_2(n)\sum_{k=1}^{p}\sum_{j=1}^{k} 1 \end{aligned}$$

$$+\frac{N}{p}C_1(n)\sum_{k=p+1}^{2p}\sum_{j=1}^{k}k+(C_2(n)-NC_1(n))\sum_{k=p+1}^{2p}\sum_{j=1}^{k}1$$

$$\overset{\text{(A.3),(A.4)}}{=}\frac{N}{p}C_1(n)\left(\frac{7}{3}p^3+2p^2+\frac{1}{3}p\right)+(2C_2(n)-NC_1(n))\left(\frac{1}{2}p^2+\frac{3}{2}p\right)$$

which yields (3.10).
Inserting N, $C_1(n)$, $C_2(n)$ and $\#(Q)\leq n^2$ directly implies the asymptotic behavior. Counting the gates for computing the gradient by (3.15) is analagous to the above by first computing $\#(\partial_j f_p)$ and then summing $\#(\nabla f_p)=\sum_j^p \#(\partial_j f_p)$. □

Appendix B

Differential geometry

The discussion of the Time-Dependent Variational Principle in Section 2.2 covers some notions of differential geometry and functional analysis. In this appendix definitions and explanations for some of those notions are collected. These explanations are simplified and do not claim completeness. For example, we neglect the complex structure or projective character of quantum states and focus on the intuition behind the discussed notions. One of the main points is to justify the orthogonal projection from a Hilbert space onto the tangent space of some submanifold used in Section 2.2.2. For more details, refer to the literature [59, 78–81].

The most important definition in differential geometry is that of a smooth manifold. Manifolds are sets that look locally like the euclidean space $\mathbb{R}^n$. This structure allows differentiating and has countless implications. In this thesis, we only consider manifolds that are subsets of finite-dimensional Banach spaces. Here, completeness ensures things behave nicely. In particular, we can differentiate on such vector spaces.

Definition B.1 (Completeness)**.** *Let $(V, \|\cdot\|)$ be a normed vectorspace over $\mathbb{C}$.*

(a) A sequence $(x_n)_{n\in\mathbb{N}} \subset V$ is called a Cauchy sequence*, if for all $\epsilon > 0$ there is $N \in \mathbb{N}$, s.t.*

$$\|x_n - x_m\| < \epsilon, \quad \forall n, m \geq N. \tag{B.1}$$

(b) We call V complete *or a* Banach space*, if every Cauchy sequence converges in V.*

(c) We call a subset $U \in V$ open*, if for every $x \in U$, there is $\epsilon > 0$ s.t. $B_\epsilon(x) \coloneqq \{y \in V \mid \|x - y\| < 1\} \subseteq U$.*

For example, every Hilbert space is a Banach space by definition.

Let $(V, \|\cdot\|)$ be an n-dimensional Banach space over $\mathbb{C}$. Given V, we may define a special kind of manifold, namely the notion of submanifolds embedded into V.

Definition B.2 (Manifolds)**.** *Let $\Omega \subseteq \mathbb{R}^m$ with $m \leq n$, be some open domain.*

(a) An immersion *from $\Omega \subseteq \mathbb{R}^m$ to V is a smooth map $\psi\colon \Omega \to V$ with* $\operatorname{rank}\psi = m$*,*

i.e. with an injective differential.

(b) An embedding *from $\Omega \subseteq \mathbb{R}^m$ into V, is an immersion $\psi\colon\ \Omega \to V$, for which $\psi\colon\ \Omega \to \psi(\Omega)$ is a homeomorphism, i.e. it is a continuous bijective map with continuous inverse.*

*(c) Let $\psi\colon\ \Omega \to V$ be an embedding for open $\Omega \subseteq \mathbb{R}^m$. Then the image $\mathcal{M} = f(\Omega)$ is called an m-*dimensional embedded submanifold *of V.*

An example of an embedded submanifold is the unit circle (see Figure B.1) $\mathbb{S}^1$ in $\mathbb{C} \cong \mathbb{R}^2$, parametrized by

$$\psi\colon\ \Omega = (0, 2\pi) \to \mathbb{C}, \quad \psi(\theta) = e^{i\theta}. \tag{B.2}$$

In the following, let $\mathcal{M}$ be an m-dimensional embedded submanifold of V, embedded by $\psi\colon\ \Omega \to V$ for open $\Omega \subseteq \mathbb{R}^m$. As $\mathcal{M}$ is embedded into a Banach space V, the notion for functions on $\mathcal{M}$ to be smooth can be inherited from the corresponding notion on V. We call a function $f\colon\ \mathcal{M} \to \mathbb{R}$ smooth, if $f \circ \psi\colon\ \mathbb{R}^m \supseteq \Omega \to \mathbb{R}$ is smooth. The partial derivatives are defined in the same way by setting

$$\partial_j f(p) \coloneqq \frac{\partial}{\partial x_j} f \circ \psi \Big|_{\psi^{-1}(p)}, \quad (j \leq m) \tag{B.3}$$

as for the usual partial derivatives on $\mathbb{R}^m$.

The notion of derivatives, i.e. "directional changes" in $\mathcal{M}$, leads to the notion of the *tangent space* of $\mathcal{M}$.

Definition B.3 (Tangentspace and frames)**.** *Let $\mathcal{M}$ be an m-dimensional submanifold embedded by $\psi\colon\ \mathbb{R}^m \supseteq \Omega \to V$ and $p \in \mathcal{M}$ be a point in the manifold.*

(a) The tangentspace *$T_p\mathcal{M}$ is the m-dimensional vectorspace of all "directions", i.e.*

$$T_p\mathcal{M} = \operatorname{span}_{\mathbb{R}}\{[f\colon\ \mathcal{M} \to \mathbb{R}] \mapsto \partial_j f(p) \mid j \leq m\}, \tag{B.4}$$

where f are smooth functions on $\mathcal{M}$.

(b) A frame of $\mathcal{M}$ is a set of smooth maps $E_j\colon\ \mathcal{M} \ni p \mapsto T_p\mathcal{M}$, $j \leq m$, such that $\{E_j(p) \mid j \leq m\}$ is a basis of $T_p\mathcal{M}$ for each $p \in \mathcal{M}$.

For example, the partial derivatives ∂_j form a frame, when regarded as maps $\partial_j(p) = \partial_j(\cdot)\big|_p$. We can identify the tangent space with $\mathbb{R}^m$ as linear spaces. Even further we can embedd it into V by considering it as the image $T_p\mathcal{M} = \mathrm{d}\psi_{\psi^{-1}(p)}(R^m)$ of the jacobian $\mathrm{d}\psi_{\psi^{-1}(p)}$ of the embedding ψ. Then $p + T_p\mathcal{M}$ is the best linear approximation to $\mathcal{M}$ in V at the point p. This justifies the construction of the TDVP for the imaginary time evolution in Section 2.2.3 by projecting from the Hilbert space onto the tangent space orthogonally.

The embedding of the tangent space can be visualized on the unit sphere. The tangent space of the unit circle at $p = e^{i\theta_0} \in \mathbb{S}^1$ is given by $T_p\mathbb{S}^1 = \left\{\lambda \frac{\partial}{\partial\theta} \;\middle|\; \lambda \in \mathbb{R}\right\}$. Then, the jacobian is a 1×1 matrix given by

$$\mathrm{d}\psi_{\theta_0} = ie^{i\theta_0} = e^{i(\theta_0 + \frac{\pi}{2})}. \tag{B.5}$$

As suggested by the visual intuition the image $\mathrm{d}\psi_{\theta_0}(\mathbb{R}) = \left\{\lambda e^{i(\theta_0+\frac{\pi}{2})} \;\middle|\; \lambda \in \mathbb{R}\right\}$ is perpendicular to p in $\mathbb{C}$. The space $p + T_p\mathcal{M}$ is the tangent line of $\mathbb{S}^1$ at p.

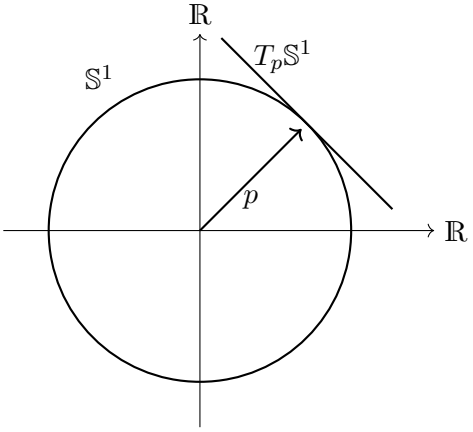


Figure B.1: The unit circle in $\mathbb{S}^1 \subset \mathbb{C} \cong \mathbb{R}^2$ with its tangent space $T_p\mathcal{M}$ to $p = e^{i\pi/4}$.

Tangent spaces at different points of a manifold are a priori completely unrelated vector spaces. In order to find a natural connection between them, additional structure is necessary. This structure is given by a *Riemannian metric*, which characterizes the Riemannian geometry of $\mathcal{M}$.

Definition B.4 (Riemannian metric)**.** *Let $\mathcal{M}$ be a manifold (for example an embedded submanifold of V) and $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ a field. A* Riemannian metric *g, is a family of maps $g_p \colon T_p\mathcal{M} \otimes T_p\mathcal{M} \to \mathbb{K}$ for every $p \in \mathcal{M}$, where each g_p is an inner product on $T_p\mathcal{M}$ that depends smoothly on p.*

A Riemmanian metric g_p on an m-dimensional manifold $\mathcal{M}$ can be represented by an $m \times m$ matrix g with $g_{j,k} = g_p(E_j(p), E_k(p))$, where $\{E_j \mid j \leq m\}$ is some frame of $\mathcal{M}$. This representation is used in the main text. With that matrix representation, we have

$$g_p(X, Y) = \sum_{j,k=1}^{m} X^j g_{j,k} Y^k, \tag{B.6}$$

for $X, Y \in T_p\mathcal{M}$ with $X = \sum_{j=1}^m X^j E_j$ and $Y = \sum_{k=1}^m X^k E_k$.

A natural way to transport tangent vectors from one tangent space to another is given by a Riemannian metric called the *Levi-Civita connection*. Also, a notion of distance between points in $\mathcal{M}$ is defined by the structure of a Riemannian metric. The locally shortest paths connecting two points are given by so-called geodesics. Geodesics are special curves in $\mathcal{M}$ that are "as straight as possible", where "straight" is again defined by the Riemannian metric. Without going into detail about how these geometric properties arise from g, this outlook should give an idea of the importance of the Riemannian metric and its appearance in the TDVP (Section 2.2). It should also hint, that by evaluating the Riemannian metric of the QAOA-manifold in Section 3.1 not only the TDVP becomes accessible, but also many other geometric properties of the QAOA may be derived from the Riemannian metric.

References

[1] R. P. Feynman, 'Simulating physics with computers', International Journal of Theoretical Physics **21**, 467–488 (1982) 10.1007/BF02650179.

[2] P. Benioff, 'The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines', Journal of Statistical Physics **22**, 563–591 (1980) 10.1007/BF01011339.

[3] P. Benioff, 'Quantum Mechanical Models of Turing Machines That Dissipate No Energy', Physical Review Letters **48**, 1581–1585 (1982) 10.1103/PhysRevLett.48.1581.

[4] M. Brooks, 'Beyond quantum supremacy: the hunt for useful quantum computers', Nature **574**, 19–21 (2019) 10.1038/d41586-019-02936-3.

[5] S. Ebadi et al., 'Quantum optimization of maximum independent set using Rydberg atom arrays', Science **376**, 1209–1215 (2022) 10.1126/science.abo6587.

[6] C. Grange, M. Poss and E. Bourreau, *An introduction to variational quantum algorithms on gate-based quantum computing for combinatorial optimization problems*, (22 December 2022) arXiv:2212.11734 [quant-ph], http://arxiv.org/abs/2212.11734 (visited on 03/01/2023), pre-published.

[7] D. Amaro, M. Rosenkranz, N. Fitzpatrick, K. Hirano and M. Fiorentini, 'A case study of variational quantum algorithms for a job shop scheduling problem', EPJ Quantum Technology **9**, 5 (2022) 10.1140/epjqt/s40507-022-00123-4.

[8] E. Farhi, J. Goldstone and S. Gutmann, 'A Quantum Approximate Optimization Algorithm', 14 November 2014, arXiv:1411.4028 [quant-ph].

[9] R. Shaydulin, S. Hadfield, T. Hogg and I. Safro, 'Classical symmetries and the Quantum Approximate Optimization Algorithm', Quantum Information Processing **20**, 359 (2021) 10.1007/s11128-021-03298-4, arXiv:2012.04713 [quant-ph].

[10] F. G. S. L. Brandao, M. Broughton, E. Farhi, S. Gutmann and H. Neven, *For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances*, (10 December 2018) arXiv:1812.04170 [quant-ph], http://arxiv.org/abs/1812.04170 (visited on 05/10/2022), pre-published.

[11] G. E. Crooks, *Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem*, (20 November 2018) arXiv:1811.08419 [quant-ph], http://arxiv.org/abs/1811.08419 (visited on 05/10/2022), pre-published.

[12] S. Hadfield, *Quantum Algorithms for Scientific Computing and Approximate Optimization*, (8 May 2018) arXiv:1805.03265 [quant-ph], http://arxiv.org/abs/1805.03265 (visited on 15/07/2022), pre-published.

[13] R. Herrman et al., 'Impact of graph structures for QAOA on MaxCut', Quantum Information Processing **20**, 289 (2021) 10.1007/s11128-021-03232-8.

[14] R. Herrman, P. C. Lotshaw, J. Ostrowski, T. S. Humble and G. Siopsis, *Multi-angle Quantum Approximate Optimization Algorithm*, (23 September 2021) arXiv:2109.11455 [quant-ph], http://arxiv.org/abs/2109.11455 (visited on 05/10/2022), pre-published.

[15] M. Medvidovi and G. Carleo, 'Classical variational simulation of the Quantum Approximate Optimization Algorithm', npj Quantum Information **7**, 1–7 (2021) 10.1038/s41534-021-00440-z.

[16] J. Ostrowski, R. Herrman, T. S. Humble and G. Siopsis, *Lower Bounds on Circuit Depth of the Quantum Approximate Optimization Algorithm*, (12 August 2020) arXiv:2008.01820 [quant-ph], http://arxiv.org/abs/2008.01820 (visited on 05/10/2022), pre-published.

[17] G. Pagano et al., 'Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator', Proceedings of the National Academy of Sciences **117**, 25396–25401 (2020) 10.1073/pnas.2006373117.

[18] *Phys. Rev. Lett. 124, 090504 (2020) - Reachability Deficits in Quantum Approximate Optimization*, https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.124.090504 (visited on 05/10/2022).

[19] G. G. Guerreschi and A. Y. Matsuura, 'QAOA for Max-Cut requires hundreds of qubits for quantum speed-up', Scientific Reports **9**, 6903 (2019) 10.1038/s41598-019-43176-9.

[20] R. Shaydulin and Y. Alexeev, 'Evaluating Quantum Approximate Optimization Algorithm: A Case Study', in 2019 Tenth International Green and Sustainable Computing Conference (IGSC) (October 2019), pp. 1–6, 10.1109/IGSC48788.2019.8957201, arXiv:1910.04881 [quant-ph].

[21] M. Szegedy, *What do QAOA energies reveal about graphs?*, (31 December 2019) arXiv:1912.12277 [quant-ph], http://arxiv.org/abs/1912.12277 (visited on 05/10/2022), pre-published.

[22] R. Tate, M. Farhadi, C. Herold, G. Mohler and S. Gupta, *Bridging Classical and Quantum with SDP initialized warm-starts for QAOA*, (6 June 2022) arXiv:2010.14021 [quant-ph], http://arxiv.org/abs/2010.14021 (visited on 05/10/2022), pre-published.

[23] Z. Wang, S. Hadfield, Z. Jiang and E. G. Rieffel, 'Quantum approximate optimization algorithm for MaxCut: A fermionic view', Physical Review A **97**, 022304 (2018) 10.1103/PhysRevA.97.022304.

[24] Z. Wang, N. C. Rubin, J. M. Dominy and E. G. Rieffel, 'XY mixers: Analytical and numerical results for the quantum alternating operator ansatz', Physical Review A **101**, 012320 (2020) 10.1103/PhysRevA.101.012320.

[25] J. Wurtz and P. J. Love, *MAXCUT QAOA performance guarantees for $p > 1$*, (2 February 2021) 10.1103/PhysRevA.103.042612, arXiv:2010.11209 [quant-ph], http://arxiv.org/abs/2010.11209 (visited on 05/10/2022), pre-published.

[26] L. Zhou, S.-T. Wang, S. Choi, H. Pichler and M. D. Lukin, 'Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices', Physical Review X **10**, 021067 (2020) 10.1103/PhysRevX.10.021067.

[27] L. Zhu et al., *An adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer*, (7 July 2022) arXiv:2005.10258 [quant-ph], http://arxiv.org/abs/2005.10258 (visited on 05/10/2022), pre-published.

[28] J. Haegeman et al., 'Time-dependent variational principle for quantum lattices', Physical Review Letters **107**, 070601 (2011) 10.1103/PhysRevLett.107.070601, arXiv:1103.0936.

[29] M. J. D. Powell, 'Direct search algorithms for optimization calculations', Acta Numerica **7**, 287–336 (1998) 10.1017/S0962492900002841.

[30] M. Powell, 'A View of Algorithms for Optimization Without Derivatives', Mathematics TODAY **43** (2007).

[31] D. J. Moore and N. Ikromov, 'A Real Options Approach to Distressed Property Borrower-Lender Reconciliation', Journal of Mathematical Finance **05**, 73–81 (2015) 10.4236/jmf.2015.51007.

[32] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information* (Cambridge University Press, Cambridge, 2010).

[33] R. de Wolf, 'Quantum Computing: Lecture Notes', 2 August 2022, arXiv:1907.09415 [quant-ph].

[34] P. A. M. Dirac, 'A new notation for quantum mechanics', Mathematical Proceedings of the Cambridge Philosophical Society **35**, 416–418 (1939) 10.1017/S0305004100021162.

[35] J. W. Z. Lau, K. H. Lim, H. Shrotriya and L. C. Kwek, 'NISQ computing: where are we and where do we go?', AAPPS Bulletin **32**, 27 (2022) 10.1007/s43673-022-00058-z.

[36] T. Toffoli, 'Reversible computing', in *Automata, Languages and Programming*, Vol. 85, edited by J. Bakker and J. Leeuwen, red. by G. Goos et al. (Springer Berlin Heidelberg, Berlin, Heidelberg, 1980), pp. 632–644, 10.1007/3-540-10003-2_104.

[37] D. P. Divincenzo, 'Two-Bit Gates are Universal for Quantum Computation', Physical Review A **51**, 1015–1022 (1995) 10.1103/PhysRevA.51.1015, arXiv:cond-mat/9407022.

[38] S. Jansen, M.-B. Ruskai and R. Seiler, 'Bounds for the adiabatic approximation with applications to quantum computation', version 3, Journal of Mathematical Physics **48**, 102111 (2007) 10.1063/1.2798382, arXiv:quant-ph/0603175.

[39] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser, *Quantum Computation by Adiabatic Evolution*, version 1, (28 January 2000) 10.48550/arXiv.quant-ph/0001106, arXiv:quant-ph/0001106, http://arxiv.org/abs/quant-ph/0001106 (visited on 30/09/2022), pre-published.

[40] T. Kadowaki and H. Nishimori, 'Quantum Annealing in the Transverse Ising Model', Physical Review E **58**, 5355–5363 (1998) 10.1103/PhysRevE.58.5355, arXiv:cond-mat/9804280.

[41] T. Albash and D. A. Lidar, 'Adiabatic Quantum Computing', Reviews of Modern Physics **90**, 015002 (2018) 10.1103/RevModPhys.90.015002, arXiv:1611.04471 [quant-ph].

[42] A. Botea, A. Kishimoto and R. Marinescu, 'On the complexity of quantum circuit compilation', in SOCS (2018).

[43] H. F. Trotter, 'On the product of semi-groups of operators', Proceedings of the American Mathematical Society **10**, 545–551 (1959) 10.1090/S0002-9939-1959-0108732-6.

[44] S. Yarkoni, E. Raponi, T. Bäck and S. Schmitt, 'Quantum Annealing for Industry Applications: Introduction and Review', Reports on Progress in Physics **85**, 104001 (2022) 10.1088/1361-6633/ac8c54, arXiv:2112.07491 [quant-ph].

[45] M. W. Johnson et al., 'A scalable control system for a superconducting adiabatic quantum optimization processor', Superconductor Science and Technology **23**, 065004 (2010) 10.1088/0953-2048/23/6/065004, arXiv:0907.3757 [cond-mat, physics:physics, physics:quant-ph].

[46] *Solving combinatorial optimization problems using QAOA*, https://community.qiskit.org/textbook/ch-applications/qaoa.html (visited on 07/01/2023).

[47] S. Hadfield et al., 'From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz', Algorithms **12**, 34 (2019) 10.3390/a12020034, arXiv:1709.03489 [quant-ph].

[48] S. Hadfield, 'On the representation of Boolean and real functions as Hamiltonians for quantum computing', ACM Transactions on Quantum Computing **2**, 1–21 (2021) 10.1145/3478519, arXiv:1804.09130 [quant-ph].

[49] R. M. Karp, 'Reducibility among Combinatorial Problems', in *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 2022, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, edited by R. E. Miller, J. W. Thatcher and J. D. Bohlinger, The IBM Research Symposia Series (Springer US, Boston, MA, 1972), pp. 85–103, 10.1007/978-1-4684-2001-2_9.

[50] M. J. D. Powell, 'A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation', in *Advances in Optimization and Numerical Analysis*, edited by S. Gomez and J.-P. Hennart, Mathematics and Its Applications (Springer Netherlands, Dordrecht, 1994), pp. 51–67, 10.1007/978-94-015-8330-5_4.

[51] D. Kraft, *A software package for sequential quadratic programming, ein softwarepaket zur sequentiellen quadratischen optimierung, forschungsbericht. Deutsche forschungs- und versuchsanstalt für luft- und raumfahrt, DFVLR* (Institut für Dynamik der Flugsysteme, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (DFVLR), Oberpfaffenhofen, Köln, 1988).

[52] 'ANALYSE MATHÉMATIQUE. Méthodc générale pour la résolution des systèmes d'équations simultanées', in *Oeuvres complètes: Series 1*, Vol. 10, edited by A.-L. Cauchy, Cambridge Library Collection - Mathematics (Cambridge University Press, Cambridge, 2009), pp. 399–402, 10.1017/CBO9780511702396.063.

[53] J. Hadamard, *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*, Mémoires Présentés Par Divers Savants à l'Académie Des Sciences de l'Institut de France, Éxtrait Du Tome XXXIII (Imprimerie nationale, Paris, 1908), 128 pp.

[54] C. Lemarechal, 'Cauchy and the Gradient Method', Documenta Mathematica (2012).

[55] H. B. Curry, 'The method of steepest descent for non-linear minimization problems', Quarterly of Applied Mathematics **2**, 258–261 (1944) 10.1090/qam/10667.

[56] P. a. M. Dirac, 'Note on Exchange Phenomena in the Thomas Atom', Mathematical Proceedings of the Cambridge Philosophical Society **26**, 376–385 (1930) 10.1017/S0305004100016108.

[57] P. W. LANGHOFF, S. T. EPSTEIN and M. KARPLUS, 'Aspects of Time-Dependent Perturbation Theory', Reviews of Modern Physics **44**, 602–644 (1972) 10.1103/RevModPhys.44.602.

[58] L. Hackl et al., 'Geometry of variational methods: dynamics of closed quantum systems', SciPost Physics **9**, 048 (2020) 10.21468/SciPostPhys.9.4.048, arXiv:2004.01015 [cond-mat, physics:quant-ph].

[59] J. Haegeman, M. Mariën, T. J. Osborne and F. Verstraete, 'Geometry of Matrix Product States: metric, parallel transport and curvature', Journal of Mathematical Physics **55**, 021902 (2014) 10.1063/1.4862851, arXiv:1210.7710.

[60] P. Kramer and M. Saraceno, eds., *Geometry of the Time-Dependent Variational Principle in Quantum Mechanics*, Vol. 140, Lecture Notes in Physics (Springer Berlin Heidelberg, Berlin, Heidelberg, 1981), 10.1007/3-540-10579-4.

[61] D. H. Kobe, 'Lagrangian Densities and Principle of Least Action in Nonrelativistic Quantum Mechanics', 10 December 2007, arXiv:0712.1608 [quant-ph].

[62] V. I. Arnold, *Mathematical Methods of Classical Mechanics*, red. by C. C. Moore, Vol. 60, Graduate Texts in Mathematics (Springer New York, New York, NY, 1978), 10.1007/978-1-4757-1693-1.

[63] A. McLachlan, 'A variational solution of the time-dependent Schrodinger equation', Molecular Physics **8**, 39–44 (1964) 10.1080/00268976400100041.

[64] D. Werner, *Funktionalanalysis*, Springer-Lehrbuch (Springer, Berlin, Heidelberg, 2018), 10.1007/978-3-662-55407-4.

[65] S.-i. Amari, 'Natural Gradient Works Efficiently in Learning', Neural Computation **10**, 251–276 (1998) 10.1162/089976698300017746.

[66] J. Stokes, J. Izaac, N. Killoran and G. Carleo, 'Quantum Natural Gradient', Quantum **4**, 269 (2020) 10.22331/q-2020-05-25-269, arXiv:1909.02108 [quant-ph, stat].

[67] S. McArdle et al., 'Variational ansatz-based quantum simulation of imaginary time evolution', npj Quantum Information **5**, 75 (2019) 10.1038/s41534-019-0187-2, arXiv:1804.03023 [quant-ph].

[68] J. Larkin, M. Jonsson, D. Justice and G. G. Guerreschi, 'Evaluation of QAOA based on the approximation ratio of individual samples', Quantum Science and Technology **7**, 045014 (2022) 10.1088/2058-9565/ac6973.

[69] J. Cohen, *Statistical power analysis for the behavioral sciences*, 2nd ed (L. Erlbaum Associates, Hillsdale, N.J, 1988), 567 pp.

[70] L. V. Hedges, 'Distribution Theory for Glass's Estimator of Effect size and Related Estimators', Journal of Educational Statistics **6**, 107–128 (1981) 10.3102/10769986006002107.

[71] 'Effect Size and Precision', in *Introduction to Meta-Analysis* (John Wiley & Sons, Ltd, 2009), pp. 21–32, 10.1002/9780470743386.

[72] 'Effect Sizes Based on Means', in *Introduction to Meta-Analysis* (John Wiley & Sons, Ltd, 2009), pp. 21–32, 10.1002/9780470743386.ch4.

[73] J. Johansson, P. Nation and F. Nori, 'QuTiP: An open-source Python framework for the dynamics of open quantum systems', Computer Physics Communications **183**, 1760–1772 (2012) 10.1016/j.cpc.2012.02.021.

[74] J. Johansson, P. Nation and F. Nori, 'QuTiP 2: A Python framework for the dynamics of open quantum systems', Computer Physics Communications **184**, 1234–1240 (2013) 10.1016/j.cpc.2012.11.019.

[75] P. Virtanen et al., 'SciPy 1.0: fundamental algorithms for scientific computing in Python', Nature Methods **17**, 261–272 (2020) 10.1038/s41592-019-0686-2.

[76] L. F. Richter, *Implemenation of the Quantum Approximate Optimization Algorithm and the Time-Dependent Variational Principle*, (January 2023) https://github.com/LeonhardRichter/TDVP_and_QAOA/releases/tag/v0.1.0-alpha (visited on 25/01/2023).

[77] J. Dormand and P. Prince, 'A family of embedded Runge-Kutta formulae', Journal of Computational and Applied Mathematics **6**, 19–26 (1980) 10.1016/0771-050X(80)90013-3.

[78] J. M. Lee, *Introduction to Smooth Manifolds*, Vol. 218, Graduate Texts in Mathematics (Springer New York, New York, NY, 2012), 10.1007/978-1-4419-9982-5.

[79] J. M. Lee, *Introduction to Riemannian Manifolds*, Vol. 176, Graduate Texts in Mathematics (Springer International Publishing, Cham, 2018), 10.1007/978-3-319-91755-9.

[80] K. Fritzsche and H. Grauert, *From Holomorphic Functions to Complex Manifolds*, Vol. 213, Graduate Texts in Mathematics (Springer New York, New York, NY, 2002), 10.1007/978-1-4684-9273-6.

[81] A. Moroianu, *Lectures on Kähler Geometry*, 1st ed. (Cambridge University Press, 29 March 2007), 10.1017/CBO9780511618666.