Diese Arbeit wurde vorgelegt am Institut für Computational Geoscience and Reservoir Engineering (CGRE)

The present work was submitted to the Institute of Computational Geoscience and Reservoir Engineering (CGRE)

Titel der Arbeit
Quantum Computing for Open-Pit Optimization
Title of the thesis
Quantum Computing for Open-Pit Optimization

Bachelorarbeit
Bachelorthesis

von/presented by
Schade, Malte Leander

1. Prüfer / 1st examiner
Prof. Dr. Florian Wellmann, CGRE RWTH Aachen

2. Prüfer / 2nd examiner
Dr. Denise Degen, CGRE RWTH Aachen

Aachen, 22.03.2022

_____

Quantum Computing for Open-Pit Optimization

Malte Leander Schade

March 22, 2022

Abstract

The optimized development of open-pit mines based on underlying geologic models greatly influences their maximal runtime and profitability. Considering all influential parameters and restrictions that arise in real-world mining operations, this proves to be a mathematically complex problem to solve for classical computers. Recent advancements in the construction of quantum computers and the discovery of new quantum algorithms could lead to new optimization techniques for mine planning that reduce the computational runtime and increase the possible input model complexity.

In this work, we evaluate how basic concepts of quantum computation and their differences to standard computation can help with this classical optimization problem, considering a complex-theoretical approach. Open-pit mine development is set up as a Traveling-Salesman-Problem, a combinatorial optimization problem of finding the shortest Hamiltonian path through interconnected nodes, and a quantum phase estimation algorithm is used to solve it as a proof of concept. Furthermore, in cooperation with Rheinkalk GmbH (Lhoist Germany), the restrictions, parameters, and problems in real-world open-pit mine planning and current optimization techniques are assessed.

The experimental results show that an open-pit mining optimization problem can be solved with quantum phase estimation and is executable on real-world quantum computers today - though only at a limited scale to date, with verifiable results.

Contents

1 Introduction

The most cost-efficient development of open-pit mines is mandatory for economic viability. Increasing mining costs must be compensated through optimization to stay competitive (Caccetta and Hill, 2003), especially in countries with high labor costs and strict safety regulations. Reducing the local and global ecologic impact requires long time planning of mine development to minimize spoil tips, biotope destruction, lowering water levels, and the release of greenhouse gases. Numerous geoscientific properties of the mined area, legal approvals, and model uncertainty must be considered in developing an optimal open-pit mine, and already existing models must be refined with new insights that arise in the ongoing mining process.

Finding an optimal mine strategy, plan, and schedule based on geologic block models, factoring in real-world restrictions and requirements is a mathematically complex task that is usually impossible to solve deterministically in a reasonable amount of time with classical algorithms. Different subproblems arise in the optimization process (e.g., Cutoff Grade Problem (appx. B.2.1), Minimal Transportation Cost Problem (appx. B.2.2), Mixing Problem (appx. B.2.3) that traditionally often have been optimized separately by hand or with linear programming applications (sects. 2.4.1/2.4.2) (Barbaro and Ramani, 1986), which leads to contradictory optimization results and often sacrifices long-term mine development for suboptimal immediate economic benefit (Gershon, 1983). For a long-term economically viable mine development, it is essential to consider the restrictions that the different optimization problems impose on one another and the dynamic interdependencies between all parts of the geologic model (Eivazy and Askari-Nasab, 2012).

Metaheuristic algorithms (sect. 2.4.3) that run on classical computers have been introduced to find solutions for the combined optimization problems in the different planning phases of an open-pit mine with methods like simulated annealing (sect. 2.4.3), particle swarm optimization (appx. B.2.4), and differential evolution (appx. B.2.5) (Goodfellow and Dimitrakopoulos, 2016). These algorithms can already optimize for substantial revenue increases. However, they face runtime development and accuracy problems as the underlying models become more complex and are therefore practically limited.

Recent advancements in the scientific field of quantum algorithmic and the construction of working quantum computers with increasingly higher compu-

tational capabilities demand an interdisciplinary analysis of the possibilities of solving complex mine development optimization problems faster and with a higher degree of accuracy on quantum hardware than on classical computers (Paesani et al., 2017). Different approaches to quantum computing like quantum annealing or universal gate-based quantum computation already show advantages in specific algorithmic tasks related to real-world optimization problems (Grover, 1996).

First attempts to connect quantum computing to mining optimization have been made recently using variational quantum eigensolver algorithms and the open-source software development kit (SDK) Qiskit (Anis et al., 2021; Hindy et al., 2021).

This bachelor thesis assesses the key factors in developing open-pit mines and important parameters and restrictions in different planning phases (sect. 2.1), using real-world data and settings provided by Rheinkalk GmbH (Lhoist Germany). Current industry-standard software products related to the mine planning process and used optimization techniques are analyzed, considering uncertainty in underlying geological models (sect. 2.2). This bachelor thesis introduces basic concepts of universal gate-based quantum computation and quantum annealing (sects. 3.1-3.7,3.10), focusing on the linear algebraic description of these processes, the quantum physical background, and the mathematical complexity of quantum and classic algorithms (sect. 2.3), as they provided new ansatzes to efficiently solving mine optimization problems. A simplified quantum experiment is set up in Qiskit to solve a graph theoretical Traveling-Salesman-Problem (TSP) (sect. 5) related to mine scheduling tasks with quantum phase estimation (sect. 3.8). It is shown that a quadratic runtime speedup is archivable with gate-based quantum computation compared to deterministic optimization on classical computers (sect. 5.1). The experimental results and their implications are discussed, considering potential improvements of future quantum experiments for open-pit mine development (sect. 5.2). Finally, a conclusion is given evaluating the feasibility of quantum computational optimization for mine development tasks today and in the foreseeable future (sect. 6).

2 Open-Pit Development

2.1 Key Factors in Open-Pit Development

The development planning for open-pit mines is classically split into different phases, and different software products are used to plan and optimize those. Depending on the mined resource and the specific value chain, the importance and focus of these phases can vary. A generalized planning chain may look as follows (Figure 1):

Figure 1: General top-down planning phases of open-pit mining projects.

The different planning phases in Figure 1 are interconnected and dependent on one another; it is, therefore, essential to consider downstream processes in the uppermost planning phases to optimize the whole value chain. For global exploration companies, the planning and optimization processes in these phases (Figure 1) are usually split into divisions, often in different locations that communicate with each other giving instructions for and feedback from running factorial processes. This system, however, also holds risks for miscommunication and suboptimal decision-making because of incomplete knowledge. Ideally, all influential parameters in the value chain should be considered when optimizing day-to-day mining tasks. A global product value change can make different scheduling favorable, for example. This concept is crucial for mining companies with many adjustable parameters and impactful long- and short-term decisions. It is mainly restricted by the amount of data acquirable and the computational capabilities needed to optimize from it (Hustrulid et al., 2013).

2.2 Industry-Standard Open-Pit Planning Phases

The mining division in the administrative site Rheinkalk GmbH (Lhoist Germany) Wülfrath mainly deals with the planning phases of open-pit design, scheduling tasks, and a reduced amount of strategic exploration planning.

Strategic exploration planning focuses on the complete orebody of a mining site and the maximal volume economically extractable. A commonly used software for planning, evaluating, and optimizing this volumetric extend is GEOVIA WHITTLE™. The software uses geologic models to predict the potential net worth of an orebody and gives suggested locations for further drillings to refine predictions. On top, a financial overview about the specific location and a predicted yield is provided, which lowers the connected financial risks. GEOVIA WHITTLE™ uses strategic parameters like cut-off-grade (appx. B.2.1), mining direction, mining area, and the desired extracted volumes over a specific timeframe and can optimize pit-shells with the pseudo flow algorithm (appx. B.2.6) and linear programming (sect. 2.4.2).

The exact locations, depths, and angles of benches and ramps are defined in the phase of open-pit design. Slope stability information and the orebody extend are needed to construct viable open-pits based on the underlying strategic planning. An industry-standard software that supports the necessary workflow is GEOVIA SURPAC™. The software can use geostatistics, borehole databases, and blockmodels to construct viable open-pits following safety regulations and production demands.

The planning phase of scheduling deals with evaluating the best possible mining order to fulfill active contracts, accounting for interdependencies between blocks of the underlying geologic model, and giving day-to-day instructions for blasting and mining. The software used by Rheinkalk GmbH (Lhoist Germany) for this process is GEOVIA MINESCHED™. It supports the creation of various diagrams to predict short- and midterm costs and yields and uses graphs to connect transportation trucks, quarry regions, crushers, and landsides by transported volume.

2.3 Complexity Theory

Computational complexity theory is the scientific field that evaluates the space and time requirements needed to execute algorithms. When discussing different optimization algorithms, complexity theory proves as a helpful tool that simplifies the analysis of computational runtime and memory develop-

ment for a mathematical decision problem. Knowledge about the complexity classes a decision problem belongs to helps to determine whether algorithms can exist which can solve related real-world problems with economically feasible cost and time resources (Hidary, 2019).

A determination of the complexity class of a problem is generally based on its upper-bound worst-case run scenario for time and space noted with the big-O notation, often supported with the lower-bound worst-case run scenario noted with the big-Ω notation (Bernhardt, 2019).

2.3.1 Complexity Class P

A mathematical problem is considered economically feasible to solve when its runtime and memory demand do not increase faster than a polynomial function with the problem size. Problems that are solvable in polynomial time by deterministic Turing machines (appx. B.2.7), for example $O(n^2)$, belong to the complexity class P (Figure 2).

2.3.2 Complexity Class NP

Problems that are not solvable in polynomial time, which solution, if it is "yes", can be verified in polynomial time by a deterministic Turing machine (appx. B.2.7), belong to the complexity class NP (Figure 2). All problems of P are contained in NP. The prove if P = NP is an unsolved problem of complexity theory with relevance for various fields of mathematics, including optimization. Scheduling optimization problems, formulable as decision problems, are majorly not solvable in polynomial time and belong to NP (Lawler et al., 1993).

2.3.3 Complexity Class NP-Complete

There are problems in NP, which all other problems of NP are reducible to in polynomial time, which are called NP-Complete (Figure 2).

2.3.4 Complexity Class NP-Hard

Problems that all other NP problems are reducible to in polynomial time that do not belong to NP are called NP-Hard (Figure 2).

2.3.5 Complexity Class BPP

The complexity class BPP (Figure 2) includes all problems where a probabilistic algorithm exists, solvable in polynomial time, which solves the problem with an error rate of no more than 1/3 on a probabilistic Turing machine (appx. B.2.8). Through multiple evaluations of the same problem, it is possible to reduce the error probability of the combined result to a problem-dependent acceptable value. BPP is conjectured to equal P (Impagliazzo and Wigderson, 1997).

2.3.6 Complexity Class BQP

BPP is contained in BQP (Figure 2), the complexity class including all problems solvable on a quantum computer in polynomial time with an error rate of no more than 1/3. Under the assumption that BPP=P, all problems solvable on a deterministic Turing machine (appx. B.2.7) like a classical computer, can be solved by quantum computers. However, some problems are conjectured to be in BQP, but not in BPP, most prominently Shor's algorithm for nontrivial prime factorization (Shor, 1999). Assuming that not all problems in BQP are also in BPP, mathematical problems exist, possibly relatable to real-world problems, that are solvable on quantum computers with economical cost and time effort and unsolvable on classical computers.

2.4 Classical Optimization Techniques

Optimization of open-pit layout and scheduling tasks is often done manually by mining engineers, although modern software products support the optimization process with deterministic (sects. 2.4.1/2.4.2) or metaheuristic (sect. 2.4.3) algorithms in increasing numbers. Manual optimization is not desirable, factoring in large numbers of parameters that influence the open-pit design and scheduling, as humans can only account for five to seven variables at once in the process of decision making (Shepard, 1964), which limits the planning effectiveness and consequently increases the deviation from the cost-optimal solution. Mathematical optimization, therefore, should be preferred for complex multi-parameter mining problems.

Subsequently, three different classical optimization methods that are either already in use in industry-standard mining planning software or that have been academically used to optimize open-pit designs are explained.

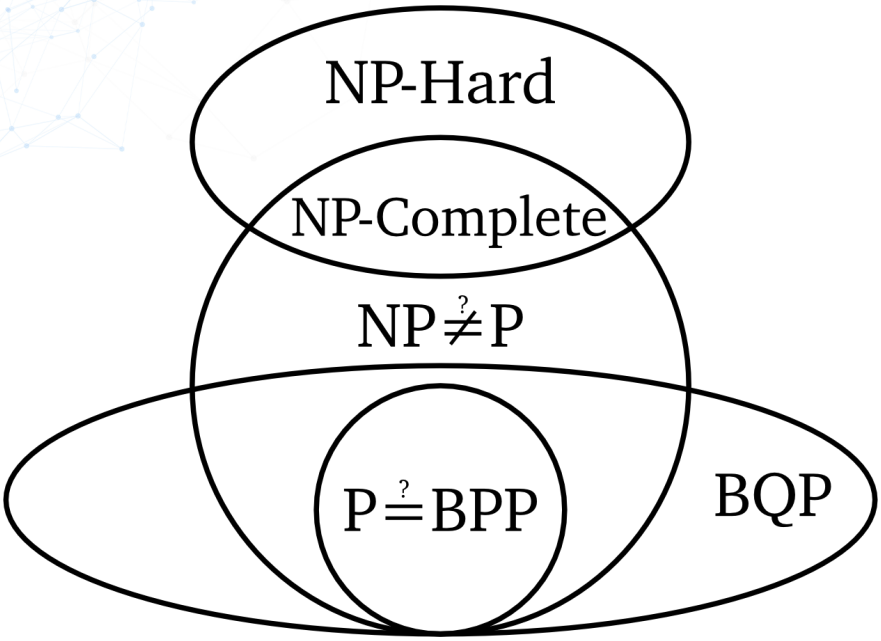

Figure 2: Different time complexity classes and their relations. Problems that are solvable by classical computers belong to the complexity class P or BPP. Problems solvable on quantum computers belong to BQP.

2.4.1 Mixed Integer Linear Programming

Mixed Integer Linear Programming (MILP) used, for example, in GEOVIA WHITTLE™ in the strategic planning phase, is a deterministic mathematical optimization technique that optimizes a defined linear objective function, which is subject to linear equality and linear inequality constraints. Various mixed integer linear programming approaches for strategic open-pit optimization and scheduling problems exist (Eivazy and Askari-Nasab, 2012), which differ in block aggregation, interdependencies, and constraint functions. The objective function is typically the maximization of the net present value (NPV), which is the sum of the differences between the net cash inflow and outflow R_t over T periods at time t, discounted back with the discount rate i to present value (Gallo, 2014):

$$NPV(i,T) = \sum_{t=0}^{T} \frac{R_t}{(1+i)^t}. \tag{1}$$

An objective function for the MILP case for a scheduling problem, not considering constraints, with $t \in \{1, ..., T\}$ as the index for scheduling periods, $n \in \{1, ..., N\}$ as the index for the blocks, d_n^t as the discounted profit generated by extracting block n in period t, and $z_n^t \in \{0, 1\}$ as a binary integer variable which equals one if block n is to be mined in period t and otherwise zero, reads as follows (Ramazan and Dimitrakopoulos, 2004):

$$max \sum_{t=1}^{T} \sum_{n=1}^{N} d_n^t \cdot z_n^t. \tag{2}$$

Several problems arise with this optimization technique, especially considering mathematical complexity (sect. 2.3). Firstly, the size of the problem connected branch-and-cut tree (appx. B.2.9) for the discrete variable case grows with $N!$ for N blocks, leading to system memory limitations for large numbers of N. Although some linear optimization techniques like the simplex algorithm (appx. B.2.10) archive polynomial complexities for the continuous decision variable case, the branch-and-cut optimization (appx. B.2.9) for binary decision variables is NP-complete (sect. 2.3.3), so it is not computationally feasible to solve on classical computers. Another problem in the given case is that a precalculated final outline of the open-pit is used in scheduling optimization to get the set of all mineable blocks N, making the optimization result suboptimal, dividing dependent optimization problems. Also, the optimization does not consider geological uncertainty that can differentiate in the geological block model and has a severe location-dependent impact on the mining process. Two possible solutions to these technique inherent problems should be considered; either finding solutions for the underlying problem heuristically while considering geological uncertainty or reformulating the objective to reduce the problem's computational complexity and space complexity. Both possibilities have been considered for classical computers, and both are also considered in quantum computational efforts (Hindy et al., 2021; Ramazan and Dimitrakopoulos, 2004).

2.4.2 Linear Programming

A simple extension of the sect. 2.4.1 case with binary decision variables, also mentioned in Eivazy and Askari-Nasab, 2012, based on Caccetta and Hill, 2003, is the introduction of continuous decision variables in the linear objective function with $u_n^t \in [0, 1]$ as a continuous variable, representing the portion of block n to be extracted and processed in period t or mined and treated as waste in period t:

$$max \sum_{t=1}^{T} \sum_{n=1}^{N} d_n^t \cdot u_n^t. \tag{3}$$

By allowing fractional extraction of blocks in scheduling, while keeping binary decision variables for block extraction ordering, it is possible to increase the optimization accuracy with more precise value choices for the continuous decision variables and, therefore, more detailed scheduling instructions. Furthermore, a substantial runtime reduction is achieved by lowering the necessary number of visited branch-and-cut nodes (appx. B.2.9) in the algorithm by optimizing branch-cut (appx. B.2.9) based on introduced continuous variables, compared to the beforementioned case. However, the algorithmic complexity of the problem is not reduced with this method, as the number of binary decision variables still scales with $N!$. Consequently, it is impossible to reduce the algorithm's computational complexity to polynomial time if binary decision variables are used, which factorially scale the branch-and-cut tree (appx. B.2.9). This conclusion also holds for merging blocks into block regions to reduce N (Goodfellow and Dimitrakopoulos, 2016).

2.4.3 Metaheuristics

A different approach to deterministic open-pit scheduling optimization is metaheuristic optimization. Metaheuristics offer the possibility to optimize scheduling with the inclusion of up and downstream processes and geologic uncertainty, as they do not require strict linearity and structure in the optimization function. There are numerous variants of metaheuristic algorithms, all having in common that they do not provide a deterministic optimal solution but decreasingly deviate ones around the cost-optimal result. Metaheuristic optimization approaches usable for open-pit scheduling optimization include simulated annealing, particle swarm optimization (appx. B.2.4), and differential evolution (appx. B.2.5)(Goodfellow and Dimitrakopoulos, 2016), of which simulated annealing is of high importance for this thesis, as it is directly comparable to quantum annealing, an optimization technique in use by numerous quantum computers.

Simulated annealing (SA) optimizes a not necessarily mathematically linear cost function through an iterative mathematical process for a discrete search space (appx. B.2.11). The cost function is modeled as a cost landscape, and probabilistic perturbations of the current estimated best state to neighboring states result in the discovery of better states in the long run. An essential difference of simulated annealing compared to deterministic algorithms is that

a transient worsening of the current state can be excepted with a certain probability, allowing a traversing through local minima of the cost function to reach better results ultimately. These probabilities change over time in the algorithmic process to support progressive refining of the optimization result. The speed of this change greatly influences the end results quality and the algorithm's computational runtime (Goodfellow and Dimitrakopoulos, 2016).

In Goodfellow and Dimitrakopoulos, 2016, the perturbations include changing block extraction periods or whether a block is extracted at all, directly influencing the predicted NPV of the mining operation. The underlying cost function additionally accounts for geologic uncertainty.

A case study of an open-pit gold mining complex presented in the paper archived an increased NPV of 6.6% compared to deterministic-equivalent techniques (sects. 2.4.1/2.4.2) and 22.6% compared to an industry-standard commercial mine planning tool.

A notable limitation for the simulated annealing approach is the tradeoff between computational runtime and the accuracy of the result, as it is directly influenced by the progressively changing speed of probabilistic acceptance for perturbations to avoid false minima in the cost function. This effect needs to be considered when optimizing the mine's planning chain in a single optimization operation and is an unavoidable restriction.

Overall, it can be seen that classical optimization techniques already archive substantial NPV increases compared to manual optimization. They are, however, limited by the inherent complexity of mine planning problems, which can be seen as NP computational problems (sect. 2.3.2) with discrete search spaces (appx. B.2.11). This complexity significantly restricts the possibility of optimization with deterministic optimization algorithms. It reduces optimization quality with metaheuristic approaches for problems with many independent decision variables, especially considering geologic uncertainty, which is inherent in all geologic models and relevant for short and long-term mine planning. Consequently, there still is a need and search for better optimization algorithms promising NPV improvements for open-pit mining operations. Quantum computation offers better ansatzes with new algorithms in solving this task (Anis et al., 2021).

3 Quantum Computation

Quantum mechanics are distinctly different in principles and laws from the macroscopic physical world (sects. 3.4/3.5/3.6/3.11). These counterintuitive principles are utilized in quantum computing and are usually hard to imagine or compare. Five quantum physical principles are the basis for the different ways of quantum computation: superposition, entanglement, reversibility, measurability, and fluctuations. They allow substantially faster computation of fitted problems on quantum computers.

3.1 Gate-Based Quantum Computing and Quantum Annealing

There are two general approaches to quantum computation demanding different quantum hardware and utilizing different quantum mechanical effects. It is essential to distinguish these, as they are both suitable for optimization problems. Gate-based quantum computing is the more general way of quantum computation. In gate-based quantum computation qubits, the basic unit of quantum computation, are manipulated by quantum gates, comparable to classical logical computer gates, achieving superposition (sect. 3.4) and entanglement (sect. 3.5) between different qubits. Through multiple gates, it is possible to manipulate the quantum information of the qubits in a way that transfers the quantum information or increases the probability of the desired outcome when evaluating the qubits' quantum state. Different companies have constructed working gate-based quantum computers, including Google, Microsoft, and IBM, with a steadily increasing number of qubits. Qiskit, an open-source Python 3 software development kit (SDK) in use in this thesis, was founded by IBM and allows the creation and execution of gate-based quantum programs on simulated or real gate-based quantum computers (Anis et al., 2021).

Quantum annealing differs distinctly from gate-based quantum computing as it is not universal and has a particular usability for discrete minimization problems. The qubits are put in a superposition state representing every possibility of the discrete problem with equal weights. Through quantum fluctuational effects, this state can evolve to the system's ground state, the state with the lowest energy, which represents the optimal solution of the optimization function (Hauke et al., 2020; Kadowaki and Nishimori, 1998). Quantum annealing is a metaheuristic and directly comparable to simulated annealing (sect. 2.4.3). Quantum computers utilizing this approach have been developed most prominently by D-Wave Systems and offer a higher count of usable qubits than gate-based quantum computers. Adiabatic quan-

tum computing is closely related to quantum annealing and utilizes the same quantum physical principles.

3.2 Formalism

A discussion of quantum computation and its principles, which is universal and unambiguous, demands a unified mathematical description and a strict formalism. A linear algebraic form of notation that was introduced by Paul Dirac for quantum computing for that purpose is the Dirac-Notation, also called Bra-Ket-Notation. It offers flexibility and ease in describing qubits and their states, as well as quantum gates. In the Dirac-Notation, a row vector $\langle a|$ is called bra and is written as follows (Hidary, 2019):

$$\langle a| = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_n \end{bmatrix}. \tag{4}$$

A ket $|b\rangle$ is a column vector and is written as follows:

$$|b\rangle = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}. \tag{5}$$

Accordingly, the inner product $\langle a|b\rangle$ of a bra and a ket of the same dimension can be written as:

$$\langle a|b\rangle = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n. \tag{6}$$

Rules for vector addition, Euclidean distance (appx. B.2.12), and scalar multiplication apply normally to bras and kets. An orthonormal base can be built with a set of unit kets, which are mutually orthogonal to one another. To assess if a basis with kets a and b of $\mathbb{R}^2$ is orthonormal three conditions must be checked; if every ket is of unitary length and if the kets are orthogonal to one another (Hidary, 2019):

$$\langle a|a\rangle = 1; \langle b|b\rangle = 1; \langle a|b\rangle = 0. \tag{7}$$

Vectors can be written as a linear combination of basis vectors of an orthonormal basis with an equal or higher dimension. In an ordered basis, the order of vectors in the set of all basis vectors is essential. An outer

product $|b\rangle\langle a|$ of two vectors of the same dimension creates a squared matrix M:

$$|b\rangle\langle a| = M. \tag{8}$$

Rules of matrix-matrix multiplication, Kronecker products (appx. B.2.13), matrix-scalar multiplication, and matrix-vector multiplication or addition apply normally. If a matrix is squared, has real entries and $M^T \cdot M = I$, with M^T the Hermitian conjugate (appx. B.2.14) of M and I the identity matrix, it is called an orthogonal matrix. A matrix that has similar properties, but complex entries is called unitary. All orthogonal matrices are unitary (Bernhardt, 2019).

In quantum computing, kets represent the quantum states of a system. Matrices represent the use of quantum gates and measurements on a quantum system. Dynamic evolutions of quantum systems, for example, due to quantum fluctuations (sect. 3.11), can be described using the Dirac-Notation.

3.3 Qubits

Unlike classical bits, which can only be in one of the two states 0 or 1, qubits can be in a linear combination of different states and accordingly in infinitely many. As this property is impossible to create with macroscopic electrical switches, quantum particles are used in quantum computers, like photons, resonating particles, trapped ions, or neutral atoms and different properties of those are utilized for the creation of logical qubits like the difference of energy levels between ground metastable states or the angular momentum of elementary particles (appx. B.2.15) called spin (appx. B.2.16) (Hidary, 2019). For easier understanding, quantum spin is considered the underlying property of qubits in this thesis.

3.4 The Principle of Superposition

A qubits state can be represented as a linear combination of two basis vectors of the complex Hilbert space $\mathbb{C}^2$ (appx. B.2.17). These basis vectors correspond, for example, to the quantum physical property of up- and down-spin of a quantum particle and can be written as follows:

$$|\uparrow\rangle = |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; |\downarrow\rangle = |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{9}$$

There are infinitely many pairs of orthonormal base vectors for $\mathbb{C}^2$, because they can be linearly transformed to one another; most noteworthy for quantum computation are apart from previously mentioned, the superpositional plus and minus states (Hidary, 2019):

$$|-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}; |+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}. \tag{10}$$

When measuring a qubit in the vertical direction, called Z-base, either $|0\rangle$ or $|1\rangle$ will be observed with specific probabilities. The state of the qubit before measurement can be written as a linear combination of both quantum states accordingly:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{11}$$

Here $|\psi\rangle$ represents the combined state of the quantum system, and α and β are the complex probability amplitudes of the specific state. By squaring the absolute probability amplitude of a state, the probability of measuring it is received. Probability amplitudes can be either positive or negative, so they can interfere in a constructive or destructive way, a principle used in some quantum algorithms (Bernhardt, 2019). Following the Born rule (appx. B.2.18), all probabilities of possible states must add up to 1. For a quantum particle with random spin, this results in the equation:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\,. \tag{12}$$

The equation shows that both measured results $|0\rangle$ and $|1\rangle$ occur with the same probability of $\frac{1}{2}$. A linear combination of different state vectors to a new state vector describing a quantum system is called superposition. An appliance of the Hadamard-Gate **H**, represented by the matrix (Bernhardt, 2019):

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{13}$$

to a particle in state $|0\rangle$, results in the superpositional state $|+\rangle$ (eq. 10). An appliance of **H** to state $|1\rangle$ results in state $|-\rangle$. This unitary operation is a generally used technique in quantum computing to archive the superposition of qubits allowing entanglement or interference.

3.5 The Principle of Entanglement

A perfect correlation of the state of multiple quantum particles is known as entanglement. Entanglement can be archived through the appliance of specific quantum gates to a set of quantum states. A commonly used quantum gate for entangling two quantum states is the **CNOT**-Gate (controlled not), represented by the matrix (Hidary, 2019):

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{14}$$

This gate is applied to the combined quantum state of one qubit in superposition and one that is either $|0\rangle$ or $|1\rangle$.

To formulate a combined state of qubits, it is necessary to assess the probability of every possible combined state, which is followingly written for two qubits q_0 and q_1 in superposition with equal probabilities for $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = |q_0\rangle \otimes |q_1\rangle = \frac{1}{\sqrt{4}} |0\rangle |0\rangle + \frac{1}{\sqrt{4}} |0\rangle |1\rangle + \frac{1}{\sqrt{4}} |1\rangle |0\rangle + \frac{1}{\sqrt{4}} |1\rangle |1\rangle . \tag{15}$$

To assess if two qubits are entangled, it must be checked whether the outer and inner product of the probability amplitudes equal, meaning the probability amplitude of $|0\rangle |0\rangle$ times the probability amplitude of $|1\rangle |1\rangle$ (outer product) must not equal the probability amplitude of $|0\rangle |1\rangle$ times the probability amplitude of $|1\rangle |0\rangle$ (inner product) for the states to be entangled (Bernhardt, 2019).

As the states of two qubits can be combined to one virtual state, it is possible to describe them through a vector of higher dimension $\mathbb{C}^4$, which a quantum gate like the **CNOT**-Gate can be applied to:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}; |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}; |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}; |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{16}$$

An application of the **CNOT**-Gate with q_0 in a superposition of equal probabilities and q_1 always equal to $|0\rangle$ can be written as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}. \tag{17}$$

The probabilities of the measurement result shift from a probability amplitude of $\frac{1}{\sqrt{2}}$ for state $|00\rangle$ and state $|10\rangle$ before application of the gate to a probability amplitude of $\frac{1}{\sqrt{2}}$ for state $|00\rangle$ and state $|11\rangle$ afterward. The resulting state is perfectly correlated and therefore entangled; this two-bit state is called a Bell-State. A check can be done by comparing the inner product of the resulting state equaling 0 and the outer product equaling $\frac{1}{2}$. In the resulting quantum state, both qubits are always measured the same. If one state is measured in the vertical direction Z, physically meaning its quantum wave function collapses to one of its two possible states $|0\rangle$ or $|1\rangle$, the other qubits wavefunction collapses instantly to the same state. This property is unique to quantum particles and is a fundamental principle of gate-based quantum computing, as it can be utilized to create efficient quantum algorithms acting on multiple qubits and, therefore, their combined state vector at once (Hidary, 2019).

3.6 Quantum Information and Reversibility

A gate, which can be written as a square matrix, is reversible when its input state can be unambiguously determined from its output state. It is mandatory that all quantum gates are unitary matrices and, therefore, reversible. A unitary matrix is a squared invertible matrix, where its conjugated transpose multiplied with itself equals the identity matrix in the corresponding dimension (Abdessaied and Drechsler, 2016). Mathematically this can be written as:

$$M \cdot M^H = I. \tag{18}$$

Quantum gates need to be reversible, following information theory first postulated by Claude Shannon (Shannon, 1993) and its interaction with the no-cloning (appx. B.2.19) and no-deleting theorem (appx. B.2.20). This requirement results strictly from the permanence of quantum information, which can neither be lost nor created. Gates that are not reversible would enable copying of the state of one qubit to another or the deletion of a qubits state and would therefore permanently destroy quantum information (Bernhardt, 2019).

The impossibility of qubit-cloning is a major difference to classical computation that heavily relies on the copying and storage of bit states and shows that new computational architectures are required for quantum computation (Bernhardt, 2019).

3.7 Quantum Gates and Circuits

A quantum algorithm executable on quantum computers is constructed in the form of quantum circuits composed of quantum registers with a certain number of qubits, classical registers with classical bits for measurement, and quantum gates acting on these components. A circuit diagram for a circuit that archives entanglement between two qubits described in (eq. 17) with the following measurement qubit states can be drawn like Figure 3 using Qiskit (modified after Anis et al., 2021):

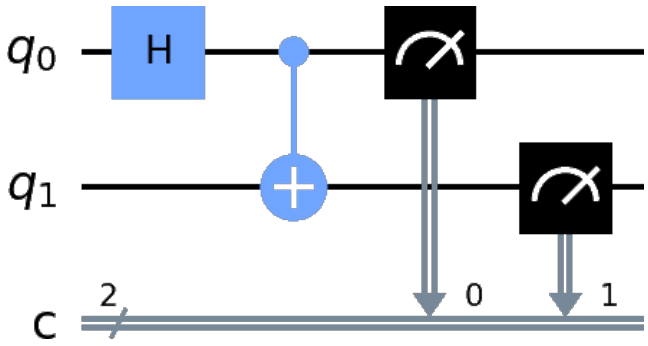

Figure 3: Qiskit circuit diagram created with the Python 3 visualization library *matplotlib* (Hunter, 2007). The quantum register holds two qubits q_0 and q_1. The classical register holds two classical bits $c_{1,2}$. The circuit is interpreted from left to right by Qiskit and describes an entanglement operation (sect. 3.5).

The two used qubits are part of a quantum register and each drawn as a horizontal line (Figure 3), where the time progression needs to be read from left to right, meaning the initialized state of the qubits, which is $|0\rangle$ by convention, is on the left and the measurement and therefore collapse of the superposition state is on the right. The two classical bits are drawn below the qubits in a classical register. Different gates and operators mentioned in earlier sections are noted down in the diagram; the Hadamard-Gate (eq. 13) acting on q_0 is denoted with the letter **H**, and the **CNOT**-Gate (eq. 14) acting on q_1 with control qubit q_0 is denoted with a directed bridge connecting

the different horizontal lines and signaling entanglement. The measurement of q_0 and q_1, denoted with black measurement symbols (Figure 3), always yields the same two quantum states of the combined system as predicted: $|00\rangle$ and $|11\rangle$.

As a part of the infinitely many possible unitary gates, there are gates having specific forms of notation and unitary matrices that follow convention. A selection, including previously mentioned, is given below, all based on Hidary, 2019:

1. **General Unitary Gate U** A general unitary gate acting on single or multiple qubits is denoted with the letter **U**. An equal number of lines is connected to the input and output of the gate (sect. 3.6).

2. **Pauli Group Gates** Sixteen unitary gates belong to the Pauli-Group (appx. B.2.21), representing either the identity operation or a reflection about an axis, all acting on a single qubit, of which four are most relevant:

   - **Identity Gate I** The identity gate is denoted with the letter **I** and represented with the matrix:

   $$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{19}$$

   - **Reflection Gate X** The reflection gate **X** indicates a flip of the probability amplitudes of $|0\rangle$ and $|1\rangle$ and is represented with the matrix:

   $$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{20}$$

   - **Reflection Gate Y** The reflection gate **Y** indicates a flip of the probability amplitudes of $|0\rangle$ and $|1\rangle$ and an additional relative phase change, meaning that the sign of the probability amplitude of $|1\rangle$ is flipped. It is represented with the matrix:

   $$Y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{21}$$

- **Reflection Gate Z** The reflection gate **Z** indicates a relative phase change of $|1\rangle$. It is represented with the matrix:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{22}$$

3. **Hadamard Gate H** The Hadamard gate is denoted with the letter **H** and indicates a defined rotation of the quantum state by 180° around a state-dependent axis. An appliance of this gate to a state not in superposition rotates it into equal superposition and therefore into the (X-Y)-plane (Figure 4): An appliance of **H** to state $|0\rangle$ results in state $|+\rangle$ (Figure 4); an appliance of **H** to state $|1\rangle$ results in state $|-\rangle$. The Hadamard gate **H** is represented with the matrix 13.

4. **CNOT Gate** The **CNOT**-gate acts on two qubits and archives entanglement between them. The first qubit is called the control qubit and changes the second one dependent on its own state vector. The second qubit is called the target qubit. The **CNOT**-gate is denoted with a directed connecting line between a plus on the line of the target qubit and a dot on the line of the control qubit. It is represented with the matrix 14.

The Bloch-Sphere is a visual representation of one qubits state vector and gates acting on it. It is a projection of the qubits state vector of dimension $\mathbb{C}^2$ into $\mathbb{R}^3$.

In the Bloch-Sphere Figure 4, the computational base state vectors in the Z-basis are denoted with $|0\rangle = |z+\rangle$ and $|1\rangle = |z-\rangle$. The superposition quantum state $|\psi\rangle$ in Figure 4 is marked with an arrow pointing in direction $|+\rangle = |x+\rangle$ opposing $|-\rangle = |x-\rangle$ in the X-basis. There are two complex phases of the quantum state: phase θ with $0 \leq \theta \leq \pi$, which is the angle between $|z+\rangle = |0\rangle$ and $|\psi\rangle$ (vertical phase), and phase ϕ with $0 \leq \phi \leq 2\pi$, which is the angle between $|x+\rangle$ and $|\psi\rangle$ (horizontal phase) (Hidary, 2019).

The following Bloch-Sphere (Figure 4) is constructed with Qiskit (Anis et al., 2021):

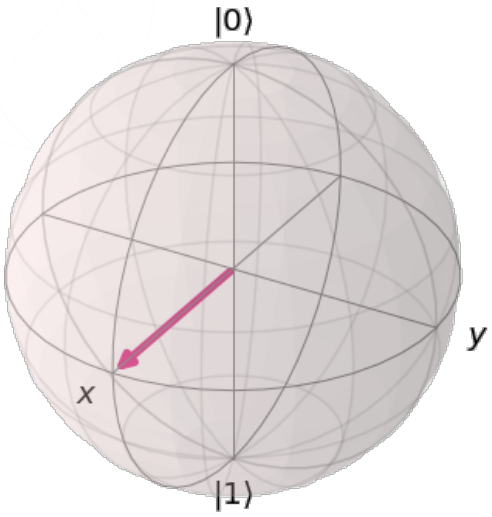

Figure 4: Bloch-Sphere representing the quantum state $|\psi\rangle = |+\rangle$ in a unitary sphere for one qubit. A combined state of multiple qubits can be visualized with multiple Bloch-Spheres. Single qubit unitary operations are describable by rotations around specific axis in the Bloch-Sphere.

3.8 Quantum Fourier Transformation

A discrete Fourier transformation (DFT) is a mathematical method of transforming an overlapped wave composed of multiple partial wavefunctions in the time domain (appx. B.2.22) to a set of frequency maxima with specific amplitudes in the frequency domain (appx. B.2.23). DFT is a reversible calculation, meaning it is possible to construct a wavefunction from a set of frequency maxima and their corresponding frequencies. In a similar sense, it is possible to transform quantum states between the computational base Z and the Fourier basis X-Y with the quantum Fourier transformation (QFT)(Anis et al., 2021):

$$QFT\,|x\rangle = |\tilde{x}\rangle\,. \tag{23}$$

To understand QFT, it is helpful to analyze how the combined state of multiple qubits in base Z is mapped to an angle around Z by the appliance of QFT, utilizing binary numeration of quantum states. All possible combined

states of n qubits in the Z-base can be enumerated with 2^n index numbers $I_{n,i}$: The state of one qubit in the Z-base, meaning it equals either $|0\rangle = I_{1,0}$ or $|1\rangle = I_{1,1}$, can be enumerated using $2^1 = 2$ index numbers. In the same way, the combined state of two qubits in the Z-base, which can be either $|00\rangle = I_{2,0}$, $|01\rangle = I_{2,1}$, $|10\rangle = I_{2,2}$, or $|11\rangle = I_{2,3}$, can be enumerated using $2^2 = 4$ index numbers. This form of enumeration is equal to a transformation of a bit number into a natural number (Anis et al., 2021).

With the appliance of QFT, the index number of every possible state in base Z for n qubits is mapped to a specific phase ϕ for every qubit. With this transformation, no information is lost as it is possible to reconstruct every qubits state in the Z-base by analyzing the phase of every qubit in the Fourier-base.

The phase ϕ of the different combined quantum states in the Fourier basis can be calculated in the following way separately for every qubit x:

$$\phi_{n,x,l} = \frac{I \cdot 2^x \cdot 2\pi}{2^n}; x \in \{0, \cdots, n-1\}; I = \textit{index number}. \tag{24}$$

Using equation 24, every qubit phase for all four index numbers in the case of two qubits can be calculated:

$$\textit{State}\ \tilde{0}: \ \phi_{2,0,0} = \frac{0 \cdot 2^0 \cdot 2\pi}{2^2} = 0\pi; \phi_{2,1,0} = \frac{0 \cdot 2^1 \cdot 2\pi}{2^2} = 0\pi, \tag{25}$$

$$\textit{State}\ \tilde{1}: \ \phi_{2,0,1} = \frac{1 \cdot 2^0 \cdot 2\pi}{2^2} = \frac{1}{2}\pi; \phi_{2,1,1} = \frac{1 \cdot 2^1 \cdot 2\pi}{2^2} = 1\pi, \tag{26}$$

$$\textit{State}\ \tilde{2}: \ \phi_{2,0,2} = \frac{2 \cdot 2^0 \cdot 2\pi}{2^2} = 1\pi; \phi_{2,1,2} = \frac{2 \cdot 2^1 \cdot 2\pi}{2^2} = 2\pi, \tag{27}$$

$$\textit{State}\ \tilde{3}: \ \phi_{2,0,3} = \frac{3 \cdot 2^0 \cdot 2\pi}{2^2} = \frac{3}{2}\pi; \phi_{2,1,3} = \frac{3 \cdot 2^1 \cdot 2\pi}{2^2} = 3\pi. \tag{28}$$

The partial term 2^x in equation 24 shows a doubling of phase ϕ for every consecutive qubit and, therefore, a doubling of frequency (modified after Anis et al., 2021).

Utilizing Euler's formula (appx. B.2.24), which links trigonometric and complex functions, it is possible to generalize equation 24 by introducing the complex number i:

$$QFT_N \left|x\right\rangle = \frac{1}{\sqrt{N}} \sum_{y=o}^{N-1} e^{\frac{2\pi i x y}{2^n}} \left|\tilde{x}\right\rangle . \tag{29}$$

By mapping every binary qubit state in the Z-base of $|x\rangle = |x_0 \cdots x_{-1}\rangle$ from which there are $N = 2^n$ many for n qubits to states in the Fourier-base of $|\tilde{x}\rangle = |\tilde{x}_0 \cdots \tilde{x}_{-1}\rangle$ with equation 29, it is possible to calculate the probability amplitudes and the corresponding state vectors in the new basis. This operation can be implemented using a unitary operator U_{QFT} acting on the state vector $|\psi\rangle$. In the simplest nontrivial case with $n = 1$ and $|\psi\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle$ this results in (modified after Anis et al., 2021):

$$\left|\tilde{x}\right\rangle = \frac{1}{\sqrt{2^1}}(\alpha e^{\frac{2\pi i 0 \cdot 0}{2^1}} + \beta e^{\frac{2\pi i 1 \cdot 0}{2^1}})\left|0\right\rangle + \frac{1}{\sqrt{2^1}}(\alpha e^{\frac{2\pi i 0 \cdot 1}{2^1}} + \beta e^{\frac{2\pi i 1 \cdot 1}{2^1}})\left|1\right\rangle , \tag{30}$$

which can be simplified to:

$$\left|\tilde{x}\right\rangle = U_{QFT}\left|\psi\right\rangle = \frac{1}{\sqrt{2}}(\alpha + \beta)\left|0\right\rangle + \frac{1}{\sqrt{2}}(\alpha - \beta)\left|1\right\rangle . \tag{31}$$

In this case, the unitary operator U_{QFT} acting on the qubit state vector can be expressed by:

$$U_{QFT} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H. \tag{32}$$

With equation 32, it is shown that the unitary U_{QFT} for the QFT of one qubit exactly equals the Hadamard gate **H** explained in section 3.4 (eq. 13).

When constructing a quantum circuit for a QFT that acts on multiple qubits, it is necessary to implement a unitary for rotation of the phase of one qubit in relation to others. The gate that implements this is the **CROT**-gate, where the control qubit is qubit x (Anis et al., 2021):

$$CROT_x = \begin{bmatrix} I & 0 \\ 0 & UROT_x \end{bmatrix}, with\, UROT_x = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^x}} \end{bmatrix} . \tag{33}$$

Additionally, **SWAP**-gates are needed, which archive a swap of the quantum states of two qubits (Hidary, 2019):

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{34}$$

With a combination of **H**-gates (eq. 13), **CROT**-gates (eq. 33), and **SWAP**-gates (eq. 34), it is possible to archive a quantum Fourier transformation for n qubits. As an example, the circuit for $QFT\,|1101\rangle = |\tilde{13}\rangle$ can be drawn in the following way with Qiskit (modified after Anis et al., 2021):

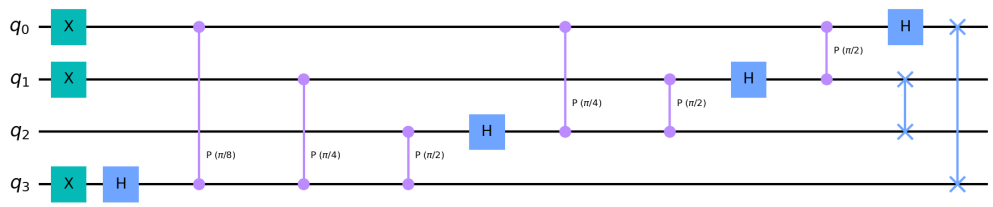

Figure 5: Circuit for the QFT of 4 qubits transforming the state $|1101\rangle$ in the Z-base to the state $|\tilde{13}\rangle$ in the Fourier-base. The line connections labeled with a P and a relative phase denote PHASE-gates, which are generalized versions of CROT-gates. The lines connected with crosses denote SWAP-gates. PHASE-gates and SWAP-Gates are undirected as no control qubit, and target qubit can be distinguished.

An inverse quantum Fourier transformation can be implemented similarly to reverse the QFT and transform the combined state vector back to the Z-base. The quantum Fourier transformation and inverse quantum Fourier transformation are commonly used tools in quantum algorithmic and are also needed for the following quantum phase estimation algorithm.

3.9 Quantum Phase Kickback

Quantum phase kickback is a commonly used effect induced in controlled gates changing the relative phase of a qubit, e.g., the **P**-gate (Figure 5) or generalized **CROT**-gates (eq. 33). A phase change is kicked back onto the

control qubit, while the target qubit is left unchanged (Anis et al., 2021). A combination of **H**-gates and **CNOT**-gates provides a simple example. When applying a **CNOT**-gate (eq. 17) to two qubits in state $|+\rangle$ (sect. 3.7), their states stay unchanged:

$$|{+}{+}\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle), \tag{35}$$

$$CNOT\,|{+}{+}\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = |{+}{+}\rangle\,. \tag{36}$$

The resulting quantum state after the **CNOT** operation is different when the control qubit is in state $|+\rangle$ and the target qubit in state $|-\rangle$ beforehand:

$$|{-}{+}\rangle = \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle), \tag{37}$$

$$CNOT\,|{-}{+}\rangle = \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle) = |{-}{-}\rangle\,. \tag{38}$$

The phase of the control qubit in equation 38 is changed from $|+\rangle$ to $|-\rangle$, while the target qubit of the operation is still $|-\rangle$. This effect is utilized in phase rotation gates, which apply a relative phase change to the control qubit of the operation.

3.10 Quantum Phase Estimation

Quantum phase estimation (QPE) is an approximation algorithm used as a subroutine in various complex quantum algorithms. With QPE, the relative phase ϕ of an eigenvector $|\psi\rangle$ of a unitary U can be estimated to a finite degree of accuracy depending on the number of used qubits. QPE utilizes that a phase ϕ can be formulated as an eigenvalue λ of U with eigenvector $|\psi\rangle$ (Hidary, 2019):

$$U \cdot |\psi\rangle = \lambda \cdot |\psi\rangle \rightarrow U \cdot |\psi\rangle = e^{2\pi i\phi} \cdot |\psi\rangle\,. \tag{39}$$

In the form of controlled unitary phase rotations, quantum phase kickback transfers the quantum phase information of U, dependent on $|\psi\rangle$, to a quantum register in the Fourier-base. Afterward, an inverse Fourier transformation (IQFT) is applied to return the state back to the Z-base, where it can be read out as a bitstring (Anis et al., 2021). The number of qubits n in

the quantum register determines the QPE accuracy as only $2^n = N$ different quantum states can be distinguished. Phases of U that transform $|\psi\rangle$ to state vectors not in N result in the state vector collapsing to a state vector in N with a probability depending on its proximity to near state vectors in N when measuring.

A general QPE algorithm can be implemented with two quantum registers and one classical register in six distinct consecutive steps (Anis et al., 2021):

1. **Initialization** Two quantum registers and one classical register are initialized. The first register holds a varying number of qubits n, depending on the required accuracy of the resulting QPE. The second quantum register holds a varying number of qubits, which can encode a specific eigenstate of the unitary phase rotation matrix U. The classical register can hold the resulting measurement of the qubits of quantum register one; it, therefore, has the same number of bits n. All qubits are initialized in state $|0\rangle$.

2. **Eigenvector Encoding** A specific eigenvector $|\psi\rangle$ of U, also named eigenstate in quantum computation, is encoded bitwise to the second qubit register. The encoding happens through the appliance of **X**-gates (eq. 20) to states that require the state $|1\rangle$ and no gates (or **I**-gates (eq. 19)) to states that require state $|0\rangle$. The combined state $|\psi\rangle$ of the second qubit register in the Z-base corresponds to the bitwise notation of the specific eigenstate of U.

3. **Superposition** All qubits in qubit register one are set to state $|+\rangle$ with the appliance of **H**-gates. This process can be denoted for multiple qubits n with:

$$H^{\otimes n}. \tag{40}$$

4. **Phase Kickback Gates** There is a controlled unitary gate applied for every qubit in register one. Every qubit in register two acts as a control qubit, defining the phase ϕ kicked back to the qubit in register one in the Fourier-base (sect. 3.9).

   This operation leaves the combined state vector of qubit register two $|\psi\rangle$ unchanged, so the unitary can be applied for every consecutive qubit of register one, archiving an information transfer of the phase ϕ of $|\psi\rangle$ of the overall unitary U with high precision.

5. **Inverse quantum Fourier transformation** An IQFT (sect. 3.8) is applied, which transforms the state vectors of the qubits in quantum register one from the Fourier-base into the Z-base where they can be measured.

6. **Measurement** The qubits in quantum register one are measured in the Z-base, collapsing the qubits' corresponding wave functions into a bitstring. This bitstring is stored in the classical register and can be read.

3.11 Quantum Fluctuations

Quantum particles have probabilistic properties like location and momentum that are not measurable with absolute precision. This quantum-inherent effect is described by the relativistic Schrödinger equation, stating that quantum properties behave wavelike (Kadowaki and Nishimori, 1998). Quantum particles have higher probabilities of observation where they are exhibiting the strongest interactive effect; they can, however, also be observed distant from that place, even with barriers that show very low observation probability in between. This effect is called quantum fluctuation or, in the case of location measurement, quantum tunneling (Hauke et al., 2020).

Quantum tunneling effects are utilized in quantum annealing computers. A cost landscape is defined with an objective function for a discrete optimization problem. At the beginning of the experiment, all possible states are put in equal superposition (sect. 3.4). The ground state (appx. B.2.25) of the quantum system with the lowest energy represents the cost-optimal solution and the desired measurement result. Through progressively slowing evolution of the quantum system and its probability wavefunctions under the right conditions, the combined quantum state of the system can change to lower energy and, therefore, more favorable quantum states. Quantum tunneling effects allow traversing through local minima of the cost landscape. The system reaches its ground state with an objective-dependent probability when altered slowly and kept entangled.

This method of optimization is directly comparable to simulated annealing (sect. 2.4.3). Experimental data suggest that quantum annealing generally produces better results with a higher probability for the same optimization problems than simulated annealing (Denchev et al., 2016; Hauke et al., 2020).

4 TSP Experiment

As quantum computation and quantum programming are widely unknown concepts, at first, the general details of the experimental setup are discussed in sections 4.1 - 4.7, followed by the elaboration of the experimental results in section 4.8.

4.1 Qiskit

Qiskit (Anis et al., 2021) is an open-source quantum programming SDK developed by IBM Research. The main version of Qiskit is written in the Python 3 programming language. Qiskit provides development libraries for quantum programs, allowing circuit construction, algorithm utilization, and program execution on simulated or real, cloud-accessed quantum computers. Qiskit follows the standard circuit model of universal gate-based quantum computation explained in section 3 and compiles its models to the low-level intermediate representation (appx. B.2.26) for quantum instructions OpenQASM (appx. B.2.27). The SDK allows direct and free access to a set of real quantum computers with a small number of qubits for code execution via an application programming interface (API) and paid access to newly constructed ones with a higher number of qubits (Anis et al., 2021).

Qiskit is divided into different Python 3 modules and submodules offering numerous tools for quantum development, documented in the corresponding API Reference webpages and guides. The practical part of this thesis is based on an official Qiskit guide to solving a TSP, a discrete mathematical optimization problem, with phase estimation on a quantum computer (Anis et al., 2021), which is in turn based on the paper by Srinivasan et al., 2018.

The provided programming code is strongly modified to correct errors and archive verifiable generalized results. The views expressed are those of the author and do not reflect the official policy or position of IBM or the IBM Quantum team.

4.2 Experimental Setup

In the subsequent experiment, a general open-pit mine optimization problem is formulated as a simplified cost minimization problem. A defined number of nodes representing geologic blocks or block units is set. Every node has a connection to every other node apart from itself. Each connection is linked with a cost value, representing the total cost of the specific mine ordering of

two nodes, considering the distance of the nodes, legal- and safety obligations, geologic factors, and all other cost-influential parameters. These costs are directed, meaning the cost from node A to node B can vary from node B to node A. This direction dependence strongly represents real-world situations where opposing mining directions offer varying practicability. The provided quantum algorithm calculates the total distance of all distinct paths through all nodes. Note that the calculation of the exact cost value of every node connection based on real-world parameters is not part of this thesis and is therefore not specified in more detail.

Problems that are formulatable as a defined number of nodes N, interconnected to all other nodes with certain costs, that need to be traversed in the most cost-efficient way, reaching all nodes exactly once, are known as Traveling-Salesman-Problems (TSP). A path that visits every node of a TSP exactly once is called a Hamiltonian path. TSPs are seen in different economic branches and scientific fields and are computationally difficult to solve as they belong to the NP-Complete complexity class (sect. 2.3.3) (Anis et al., 2021).

4.3 Problem Generation

The costs that connect every node of a TSP can be written as an $N \cdot N$ cost matrix. This cost matrix has diagonal entries of zero, signaling that there are no self-referential node paths. The cost matrix is generated randomly with a seeded integer random number generator for the experiment. It is necessary to provide the node count N, the cost range that all randomized integer values belong to, and a definition if the returned array should be symmetric. While symmetric cost matrices do not represent a real-world cost scenario as stated in section 4.2, they nevertheless are important in this thesis, enabling verification of the correct code execution, as every forward Hamiltonian path length needs to exactly equal the corresponding backward Hamiltonian path length.

A cost matrix A for a random non-trivial case for three nodes, generated with the appended code, is provided:

$$A = \begin{bmatrix} 0 & 75 & 15 \\ 91 & 0 & 17 \\ 79 & 67 & 0 \end{bmatrix}. \tag{41}$$

The asymmetric cost matrix (eq. 41) was generated with the seed 54123

in the closed interval $I = [0, 100]$ for $N = 3$. The row indices determine the starting node of a specific path in the cost matrix, and the column indices the end node. The integer indices count up from zero, following general computer scientific convention:

$$A = \begin{bmatrix} \Delta_{0\to0} & \Delta_{0\to1} & \Delta_{0\to2} \\ \Delta_{1\to0} & \Delta_{1\to1} & \Delta_{1\to2} \\ \Delta_{2\to0} & \Delta_{2\to1} & \Delta_{2\to2} \end{bmatrix}. \tag{42}$$

To utilize this cost matrix in a quantum phase estimation algorithm, it is necessary to make it unitary (sect. 3.6), which requires all matrix entries to be formulated as phases. In the first step, the matrix entries are min-max normalized to the interval $[0, 1]$, using the generation interval I:

$$\tilde{\Delta}_{n\to m} = \frac{\Delta_{n\to m} - I_{min}}{I_{max} - I_{min}}. \tag{43}$$

It is essential to account for the limited image of phases in quantum phase estimation, as they can only take values in the half-open interval $(0, 2\pi]$. In the second step, it is, therefore, the objective to ensure that the length of a Hamiltonian path cannot exceed 2π by dividing every cost matrix entry by N:

$$\phi_{n\to m} = \frac{\tilde{\Delta}_{n\to m}}{N}. \tag{44}$$

This calculation gives us a preperated cost matrix with costs as phases ϕ that can be easily formulated as a unitary matrix U by introducing Euler's number e and the complex number i:

$$U(\phi_{n\to m}) = e^{2\cdot\pi\cdot i\cdot\phi_{n\to m}}. \tag{45}$$

Every cost value ϕ is now encoded as a rotation around the unit circle, meaning that for every value of ϕ, it is true that:

$$\sqrt{Re(z)^2 + Im(z)^2} = 1. \tag{46}$$

The cost matrix (eq. 41), with the appliance of the beforementioned steps, transforms to:

$$U(A) = \begin{bmatrix} e^{2\pi i \cdot 0.000} & e^{2\pi i \cdot 0.250} & e^{2\pi i \cdot 0.050} \\ e^{2\pi i \cdot 0.30\overline{3}} & e^{2\pi i \cdot 0.000} & e^{2\pi i \cdot 0.05\overline{6}} \\ e^{2\pi i \cdot 0.26\overline{3}} & e^{2\pi i \cdot 0.22\overline{3}} & e^{2\pi i \cdot 0.000} \end{bmatrix}. \tag{47}$$

Certain specific values for $e^{2\pi i \cdot \phi_{n \to m}}$ are easily calculatable:

$$e^{2\pi i \cdot 0} = 1; e^{2\pi i \cdot 0.25} = i; e^{2\pi i \cdot 0.5} = -1; e^{2\pi i \cdot 0.75} = -i; e^{2\pi i \cdot 1} = 1. \tag{48}$$

These values of ϕ reveal that a higher ϕ in $I = [0, 1]$ corresponds to a counterclockwise rotation of the unit vector in $\mathbb{C}^2$ and that $\phi = 1$ describes one full rotation around the unit circle, proving that the unit vector for $\phi = 0$ and $\phi = 1$ is equivalent.

4.4 Graph Transformation

Following Graph theory, the mathematical study of relations between pairs of objects, a TSP can be described with a directed or undirected graph (Anis et al., 2021). An undirected graph is a graph where all object relations are equal in both directions. This equality is no requirement for undirected graphs.

The graph in Figure 6 visually shows the three-node directed TSP calculated in equation 47, the simplest nontrivial case of the problem. The arrows and labels show the connected costs as phases ϕ for every edge, representing the real-world cost for a mining operation in this specific order. The graph is scalable with a higher node count N. In the three-node problem, there are two distinct routes to compare: the clockwise and counterclockwise Hamiltonian path with differing lengths. In this thesis quantum phase estimation is used to estimate the path length of both possible Hamiltonian paths.

The following directed graph (Figure 6) was constructed with the Python 3 module *networkx* (Hagberg et al., 2008), and the corresponding code is provided in the appendix:

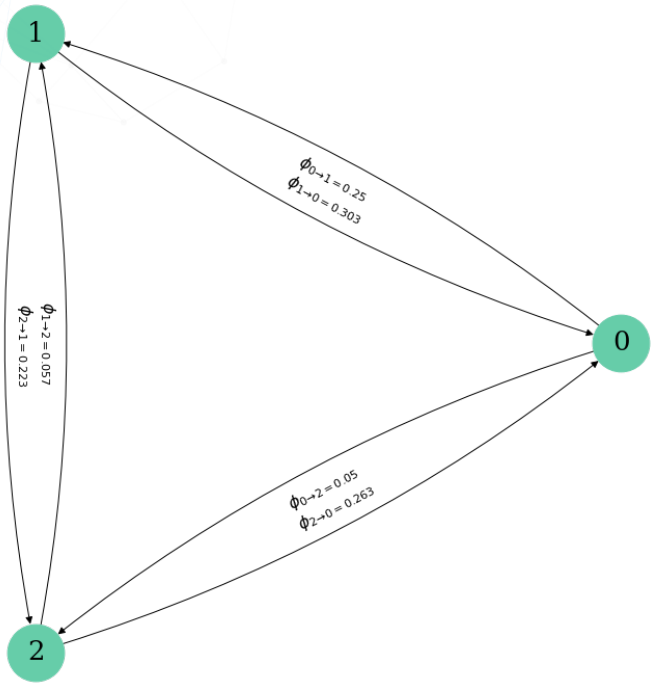

Figure 6: The directed Traveling-Salesman-Problem (eq. 41) with three nodes displayed as a graph. Every edge is annotated with its corresponding phase value ϕ, calculated in equation 44.

4.5 Kronecker Product Unitary

According to equation 39, quantum phase estimation requires the creation of a unitary gate $\mathbf{U}$ in which the cost information of all possible paths is encoded. As there can only be one phase ϕ transferred with phase kickback at a time, this requires the combination of all the costs as phases of one path to a single value for all possible Hamiltonian paths, achievable with the Kronecker product. In the first step, the transformed cost matrix $U(A)$ is split into N diagonal matrices, as for every node n in N, there is precisely one step in the Hamiltonian path. It is, however, required to scale these matrices to $a \cdot a$ with:

$$a = 2^b; b \in \mathbb{N}_0, \tag{49}$$

where b is the number of binary digits needed to represent every two-node path in a single step, with N possibilities. This means for our given problem with $N = 3$:

$$N = 3_{10} = 11_2 \rightarrow a = 2^2 = 4, \; with \; b = 2. \tag{50}$$

This scaling is necessary, as the corresponding eigenstates $|\psi\rangle$ to the unitary U need to be defined in a binary way, described in section 4.8 in more detail. The created rows and columns are filled with ones on the diagonal and zeros on every other position. When splitting $U(A)$(eq. 47) into its three diagonal column matrices U_n, following the rules stated above, these unitary matrices result:

$$U_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{2\pi i \cdot 0.30\overline{3}} & 0 & 0 \\ 0 & 0 & e^{2\pi i \cdot 0.26\overline{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{51}$$

$$U_2 = \begin{bmatrix} e^{2\pi i \cdot 0.250} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{2\pi i \cdot 0.22\overline{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{52}$$

$$U_3 = \begin{bmatrix} e^{2\pi i \cdot 0.050} & 0 & 0 & 0 \\ 0 & e^{2\pi i \cdot 0.05\overline{6}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{53}$$

To combine the matrices into a unitary matrix containing the combined path lengths of every possible three-step path, including self-referential steps, it is necessary to calculate their Kronecker product. As the Kronecker product is not commutative, meaning that in general:

$$A \otimes B \neq B \otimes A, \tag{54}$$

it is necessary to keep their correct order in the calculation, in this case, from left columns to right columns:

$$U = U_1 \otimes U_2 \otimes U_3. \tag{55}$$

The correct ordering is critical to ensure that the eigenstates constructed in section 4.6 kick back the correct phase of the unitary. The resulting diagonal matrix U for $N = 3$ has the size $u \cdot u$ with:

$$u = 4^3 = 64. \tag{56}$$

The calculation of the Kronecker product in the provided code is done using *NumPy* (Harris et al., 2020); it is, however, also directly implementable in the quantum circuit in an iterative process, which drastically increases its efficiency.

The resulting unitary U has an important property utilized in the quantum phase estimation. All its diagonal entries, representing each a unique path through the transformed cost matrix, are indexable with a binary number with $N \cdot b = 6$ digits, exactly corresponding to the binary digit count of the input eigenvectors. Every path is a multiplication of its sub-paths, which equals an addition of their phases. To allow the desired phase kickback onto a control qubit described in section 3.9, it is required to transform the unitary U into a controlled unitary CU, which is done by enlarging the matrix to double diagonal size and filling every second diagonal entry with one, starting from index $[0, 0]$. The resulting controlled unitary matrix CU for the case $n = 3$ has the size $128 \cdot 128$ and acts on $N \cdot b + 1 = log_2(128) = 7$ qubits. A possible code, implementing this controlled unitary into a quantum circuit in Qiskit is provided below:

Listing 1: Controlled Unitary Circuit, Qiskit Implementation

```
1 def unitary_as_gate(cost_matrix, reg_size):
2   q_reg = QuantumRegister(reg_size)
3   qc = QuantumCircuit(q_reg)
4   qc.diagonal(cost_matrix, q_reg)
5   qc = qc.reverse_bits()
6   return qc.to_gate(label='CU')
```

As Qiskit allows the usage of quantum circuits as gates, it is often the best subroutine to construct certain desired unitary operations as gates in functions and return them to the primary circuit. In line 2, a quantum register with the desired register size, equaling in the beforementioned case seven, is created. This register is used to construct an empty quantum circuit object in line 3. It is possible to add new gates to the quantum circuit object by

calling one of its gate functions and providing the necessary parameters, seen for example in line 4, where a general diagonal unitary gate is implemented based on the input array CU, which addresses all qubits of the quantum register when provided with the $128 \cdot 128$ array.

Line 5 can be required, dependent on the chosen programming approach, as Qiskit interprets qubits with reversed index order. This interpretation order makes strict control of the input states, resulting states and unitaries necessary, as a wrong ordering can quickly occur when implementing not commutative operations (eq. 54). In line 6, the circuit is returned as a general controlled unitary gate **CU**, appendable in the primary quantum circuit.

4.6 Eigenstates of the Unitary

As seen in equation 39, it is necessary to provide specific eigenstates of the cost unitary in the quantum phase estimation circuit to calculate their corresponding eigenvalues. As only the distinct Hamiltonian paths are of interest, as a part of all possible paths, this reduces the number of eigenstates that need to be evaluated for a directed TSP to (Anis et al., 2021):

$$E = (N-1)!, \tag{57}$$

and for an undirected TSP to:

$$E = \frac{(N-1)!}{2}, \tag{58}$$

as all forward and backward paths in the undirected case are equivalent. As all rotational variants of a Hamiltonian path result in the same eigenstate, meaning that for example $-0-1-2-$ is equivalent to $-1-2-0-$, the Hamiltonian paths of the directed TSP can be calculated in an algorithmically simple way. All permutations of the set N, without 0, with $n-1$ elements, are calculated and appended to the element 0, which in the three-node directed TSP results in:

$$E_1 = [0] + [1,2] = [0,1,2], \tag{59}$$

$$E_2 = [0] + [2,1] = [0,2,1]. \tag{60}$$

The number of two calculated distinct Hamiltonian paths as lists exactly equals the predicted number in equation 57. To convert these Hamiltonian paths into binary eigenvectors of U, the corresponding non-binary integer representation of the eigenstate must be calculated. In a first step, all two-node paths of a specific Hamiltonian path are determined:

$$E_1 = (0,1) + (1,2) + (2,0). \tag{61}$$

The steps are sorted in reverse order by their first element:

$$E_1 = (2,0) + (1,2) + (0,1). \tag{62}$$

It is necessary to calculate a sum of every step's second element multiplied by four to the power of its index position, to calculate the eigenvector integer representation:

$$E_{1,10} = 0 \cdot 4^0 + 2 \cdot 4^1 + 1 \cdot 4^2 = 0 + 8 + 16 = 24. \tag{63}$$

The calculated $E_{1,10} = 24$ represents the index $I_{E,E}$ in the corresponding diagonal unitary matrix U (sect. 4.5). The targeted value equals the combined length of the corresponding Hamiltonian path E_1 and is kicked back to the control qubit of **CU** in the quantum phase estimation algorithm.

As the eigenvector needs to be introduced into the quantum circuit with qubits, it must be formulated as a state vector, which can be done by converting $E_{1,10}$ to $E_{1,2}$ with $N \cdot b = 6$ digits:

$$E_{1,2} = 011000 = |\psi_1\rangle = |000110\rangle\,. \tag{64}$$

This quantum state can be easily represented with qubits in a quantum register in Qiskit. The state is in reversed order (sect. 4.5).

4.7 Circuit for Quantum Phase Estimation

The general circuit for quantum phase estimation can be divided into six consecutive steps described in section 3.10 that need to be implemented with Qiskit. A generalized circuit drawing for quantum phase estimation (appx. B.3.2) and a simplified one (appx. B.3.1) corresponding to the programmed implementation are appended. All steps are divided by barriers in the generalized circuit diagram (appx. B.3.2), which will be underlying

for further explanation, as the simplified Qiskit implementation utilizes the same concepts and steps. Both circuit diagrams display the same three-node Traveling-Salesman-Problem (eq. 41) with an equaling Hamiltonian path $|\psi_1\rangle$ with six qubits used for measurement.

4.7.1 Circuit Initialization

It is necessary to define all classical and quantum registers used in the primary circuit before the Qiskit circuit is initialized. Two quantum registers and one classical register are required for the quantum phase estimation algorithm. Following equation 64, the quantum register introducing the eigenstate into the phase estimation consists of six qubits for the general three-node case. This register is named *eigen* in the following. The second quantum register, followingly named *unit*, has a varying size, dependent on the precision of the result required. Generally, with q qubits in the *unit* register 2^q different result states can be distinguished, as only binary values can be measured. In the described example, six *unit*-qubits are used with $2^6 = 64$ distinguishable states in the range:

$$[0, (1 - \frac{1}{2^q})] = [0, (1 - \frac{1}{2^6})] = [0, 0.984375]. \tag{65}$$

The range does not equal $[0, 1]$, as the result ϕ_E is calculated based on the measured state vector $\psi_{E,m}$ with:

$$\phi_E = \frac{(\psi_{E,m})_{10}}{2^q}. \tag{66}$$

The classical register, named *unit_classical*, is constructed with equal size to the *unit* register and holds six classical bits.

The Qiskit primary quantum circuit called *qc* is initialized with the classical and both quantum registers. All twelve qubits are initialized in the state $|0\rangle$ and all six classical bits in the state 0.

4.7.2 Circuit Eigenvector Encoding

The quantum circuit must be constructed and executed once for every eigenstate in the conventional quantum phase estimation approach. Extended phase estimation algorithms like Iterative Quantum Phase Estimation (appx. B.2.28) solve the eigenstate encoding differently but are not discussed further in this thesis.

For every iteration, an eigenvalue $E_{n,2}$, calculated in section 4.6, is encoded into the qubit register *eigen* with reversed order to create the eigenstate $|\psi_n\rangle$. In the case of $|\psi_1\rangle = |000110\rangle$ (eq. 64), displayed in the circuit diagrams, this means that qubits 0,1,2, and 5 in register *eigen* are left unchanged in state $|0\rangle$, and qubits 3 and 4 are put in state $|1\rangle$ by the appliance of **X**-gates to their qubit states.

4.7.3 Circuit Superposition

To apply the quantum phase estimation, all qubits in the *unit* register must be in state $|+\rangle$. This state is required, as for any phase kickback to be possible to the control qubit, which in this case is every qubit in the quantum register *unit*, it needs to be in the Fourier basis (sects. 3.7, 3.9). Accordingly, a Hadamard-gate **H** is applied to every *unit* qubit to change its initialization state $|0\rangle$ to the state $|+\rangle$ (sect. 3.4).

4.7.4 Circuit Phase Kickback Gates

The quantum phase kickback is induced iteratively with q unitary gates, where q equals the number of qubits in the *unit* register. In every iteration $i \in \{1, 2, \cdots, q\}$, the controlled unitary gate **CU** constructed in section 4.5 is applied 2^{i-1} consecutive times, where the qubit acting as the control qubit is the qubit with index $q - i$ of the quantum register *unit*. The target qubits are all qubits in the quantum register *eigen*.

This appliance kicks back the desired phase corresponding to the input eigenstate to the control qubit in the *unit* register 2^{i-1} times in the Fourier basis. The consecutive appliance adds the kicked-back phases together and is required in the IQFT with more than one qubit. For the provided example E_1 in the three-node case with a predicted total:

$$\phi_E = 0.250 + 0.05\overline{6} + 0.26\overline{3} = 0.57, \tag{67}$$

which is precisely the phase ϕ in the unitary U, having its position addressed by the eigenstate E_1, following phases are kicked back to each of the six qubits in the *unit* register:

$$unit_5 = e^{2\pi i \cdot 0.57 \cdot 1}; unit_4 = e^{2\pi i \cdot 0.57 \cdot 2}; unit_3 = e^{2\pi i \cdot 0.57 \cdot 4}; \tag{68}$$

$$unit_2 = e^{2\pi i \cdot 0.57 \cdot 8}; unit_1 = e^{2\pi i \cdot 0.57 \cdot 16}; unit_0 = e^{2\pi i \cdot 0.57 \cdot 32}. \tag{69}$$

Note that the phases in the quantum register *unit* are indexed in reversed order. The repeated appliance of the **CU**-gate is possible as it does not change the eigenstate $|\psi_1\rangle$ (sect. 3.9). The phases in *unit* are followingly transformed to the Z-base with an inverse quantum Fourier transformation, as they cannot be read out in the Fourier base. A measurement in the Fourier base would give $|0\rangle$ and $|1\rangle$ with the same probability of $\frac{1}{2}$ for every qubit (sect. 3.4).

4.7.5 Circuit Inverse quantum Fourier transformation

The IQFT archives a transformation of a quantum state vector $|\psi_n\rangle$, with consecutively doubling phase in every qubit, from the Fourier base, equal to the (X, Y)-plane, to the Z-base, where it can be measured (sect. 3.8). If the transformed $|\psi_n\rangle$ does not equal one of the distinguishable states (sect. 4.7.1), the IQFT returns a state vector that is not aligned with the Z-axis for some qubits and is therefore still in a superpositional state, resulting in a probabilistic measurement result corresponding to a bell curve. The qubits of the lower index and, therefore, lower significance have a higher variance than the more significant higher indexed ones.

4.7.6 Circuit Measurement and Result Calculation

To complete the quantum phase estimation, all qubits in the *unit* register are measured, which results in a collapse of their state vectors corresponding quantum wave function to a binary value. This value is stored in the classical register and saved to a list for every repetition of the experiment. Every eigenstate is calculated numerous times to archive its probabilistic bell curve with high accuracy. In the beforementioned example, the number of repetitions for each eigenstate is 8192. The experiment is repeated for every eigenstate while only changing the qubits in the *eigen* quantum register to the corresponding eigenvalue (sect. 4.7.2). Each eigenvectors most common binary measurement result is converted to the corresponding total angle of the Hamiltonian path ϕ_E and saved in a list.

4.8 Experimental Results

4.8.1 Simulated Quantum Computation

The described three-node TSP (eq. 41) was calculated with the simulated Qiskit backend *aer_simulator*. The appended histogram (appx. B.3.3) for E_1 was drawn in Qiskit utilizing the Python 3 library *matplotlib* (Hunter, 2007) for every binary state from 000000 to 111111. The histogram describes

a bell curve that does not have a predominant peak for one measurement state. However, this variance is precisely as expected in section 4.7.5, and a reasonable estimation of ϕ_E can be calculated based on the histogram when evaluating the most and second most measured quantum state. Following equation 66, the exact angle ϕ_n both states represent can be received:

$$\phi_1 = \frac{(100100)_{10}}{2^6} = \frac{36}{64} = 0.5625, \tag{70}$$

$$\phi_2 = \frac{(100101)_{10}}{2^6} = \frac{37}{64} = 0.578125. \tag{71}$$

When accounting for both measurement probabilities $p_1 = 0.438$ and $p_2 = 0.377$, it is possible to estimate ϕ_E with high accuracy:

$$\phi_E \approx \frac{p_1 \cdot \phi_1 + p_2 \cdot \phi_2}{p_1 + p_2} = \frac{0.438 \cdot 0.5625 + 0.377 \cdot 0.578125}{0.438 + 0.377} = 0.5697. \tag{72}$$

The accuracy of the estimation can be increased by either increasing the number of qubits in the measurement qubit register *unit* or by factoring in more measured states with their corresponding probability. With equation 71, the measured angle deviates $\sim 0.05\%$ from the predicted value of 0.57 (eq. 67). This accuracy is sufficient to distinguish it from the other eigenstate E_2 having a predicted ϕ_E of $0.57\overline{6}$ and a measured one of 0.578125. However, it should be noted that in a case with similar Hamiltonian pathlengths, a higher qubit count in the *unit* register should be preferred to counteract probabilistic errors and increase result resolution.

The minimum of all ϕ_E is searched with a classical algorithm in the provided code. This Hamilton path has the smallest cost sum and is preferable as a schedule for open-pit development in the specific cost case.

4.8.2 Real Quantum Computation

As Qiskit allows the free utilization of real quantum computers in the IBMQ-cloud, accessible over an API, the provided quantum program has been executed on the IBM quantum computer *ibmq_belem*. As this quantum computer only provides five qubits, the TSP was modified to the trivial case of two nodes (appx. B.3.4). In this problems quantum circuit, the *eigen* quantum register holds two qubits necessary for the eigenvector encodement, and

the *unit* quantum register holds three qubits for phase estimation and measurement. The costs of both node connections are set to $\phi_{1,2} = \frac{1}{4}$; therefore, the expected resulting Hamiltonian path length is $\phi_E = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$. The following results (Figure 7) were measured with 8192 repetitions:

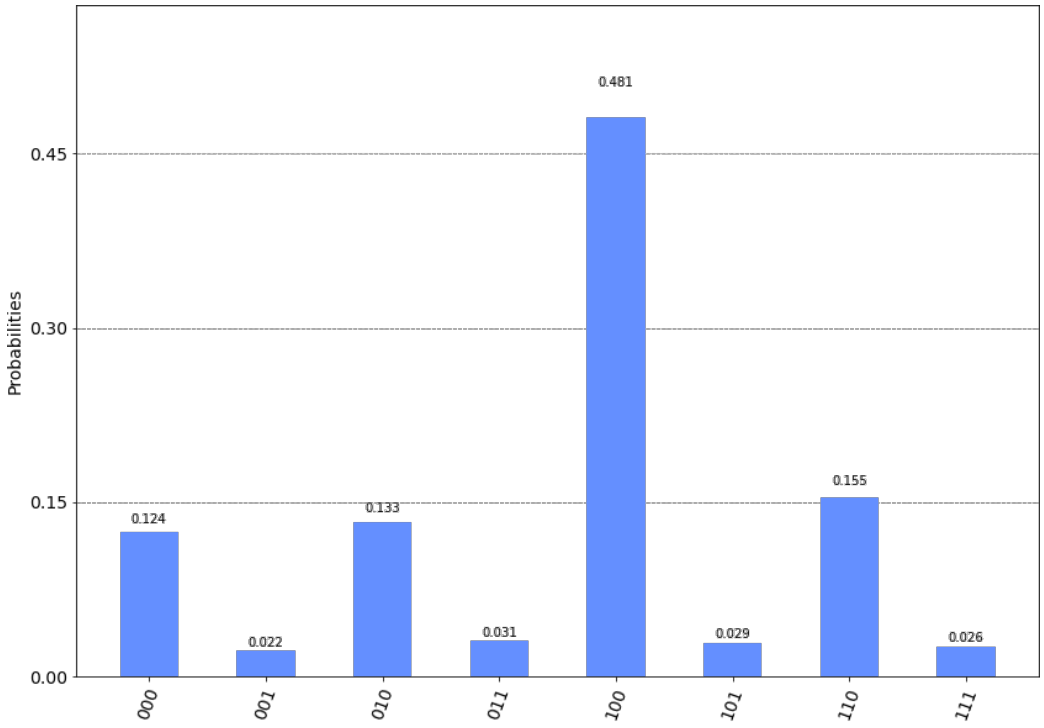

Figure 7: Measurement results of a two-node TSP with the provided code on the IBMQ quantum computer ibmq_belem. The results show a predominantly measured state $|100\rangle$.

The histogram Figure 7 shows a more significant variance compared to the results of the simulated quantum computer backend (appx. B.3.3) but a prominence of state $|100\rangle$. The calculation of ϕ_E is accordingly (eq. 66):

$$\phi_E = \frac{(100)_{10}}{2^3} = \frac{4}{8} = \frac{1}{2}. \tag{73}$$

The expected ϕ_E and measured ϕ_E for the Hamiltonian path length are equal.

5 Discussion and Outlook

5.1 Discussion of Experimental Results

The implemented code for quantum phase estimation is universal and gives reproducible and verifiable results.

Quantum phase estimation provides a quadratic speedup over classical algorithmic methods when solving a Traveling-Salesman-Problem with a large number of nodes N if the algorithm is implemented efficiently and together with a quantum search algorithm to find the shortest Hamiltonian path length (Anis et al., 2021; Srinivasan et al., 2018). This speedup is substantial.

However, numerous factors and considerations limit the real-world usability of the provided implementation and the algorithmic approach in general when considering runtime and space complexity. The three most limiting factors can be described.

5.1.1 Code Implementation

The provided code is implemented in a less than optimal way. By calculating the unitary matrix U with classical algorithmic methods rather than building it directly and iteratively in the quantum circuit, the runtime is increased dramatically. Therefore, the code in its stated form does not provide a quadratic speedup, especially for large N. Additionally, using a simulated quantum computer for the calculation reduces the algorithmic execution efficiency dramatically, as the quantum inherent computing properties cannot be utilized. Nevertheless, the provided experiment proves the general applicability of quantum computation for open-pit optimization problems with today's methods and underlines the potential of quantum algorithmics for computational mining-related problems in general.

5.1.2 Complexity

Solving TSPs with quantum phase estimation reduces the computational runtime substantially compared to classical deterministic methods. However, the Traveling- Salesman-Problem is still not computationally feasible to solve. As the node count increases, the number of Hamiltonian paths increases factorially. Even finding all Hamiltonian paths and their corresponding eigenvectors is an NP-Complete subproblem (Anis et al., 2021).

The memory space needed to store all eigenvectors and results also scales with $N!$. There is not a subproblem that archives polynomial runtime with the stated approach. As an example, the number of Hamiltonian paths for $N = 1000$, a value for N orders of magnitude less than typical block counts in geologic block models, equals:

$$(1000 - 1)! \approx 4 \cdot 10^{2564}. \tag{74}$$

Comparingly, the number of atoms in the observable universe is estimated to be around $5.3 \cdot 10^{79}$ (Ade et al., 2016). It is evident that solving a TSP utilizing the proposed form of quantum phase estimation is infeasible, even for small to medium numbers of N.

5.1.3 Executability

While the simulated backend, used for calculation of the quantum algorithm, simulates an ideal quantum computer without errors, this does not represent the real-world conditions. Real-world quantum computers utilize complex error correction to counteract unentanglement and environment interaction effects of real-world qubits, requiring substantially more total qubits for the complete calculation (Reed et al., 2012). The not error corrected experiment (sect. 4.8.2), executed on a real quantum computer, resulted in wrong results with a probability of over 50%, even with a trivial experimental setup.

5.2 Improvements of Future Experiments

Numerous improvements are implementable in future research connecting quantum computation with mining. First, the provided algorithm needs to be refined and optimized to scalably archive the described runtime speedup. While the provided quantum algorithm does not archive exponential runtime reduction for TSPs, it nevertheless may be usable with economic benefits for open-pit optimization problems formulated differently or for other optimization problems arising in the mining industry.

A quantum error correction must be implemented in the quantum circuit for industrial usability of the described quantum algorithm on real-world quantum computers. It is also required for the code to give predictions about the estimated error probability and result accuracy. The runtime and quantum hardware requirements are optimizable by tweaking algorithmic parameters to real-world requirements.

Other quantum algorithms must be evaluated to assess their usability for open-pit mine optimization. This evaluation includes algorithms executable on universal gate-based quantum computers like the quantum approximate optimization algorithm (QAOA)(appx. B.2.29) or the variational quantum eigensolver (VQE)(appx. B.2.30)(Hindy et al., 2021), but especially optimizations on quantum annealing quantum computers.

To improve the provided implementation and different quantum implementations in general, it is essential to access modern real-world quantum computers and utilize them for experiments, providing firsthand feedback and easing refinement of computational methods, as real-world quantum computers often behave differently than their simulated counterparts. Additionally, the simulation of larger quantum computers on classical ones is computationally impossible.

It is of great importance to focus the research on new quantum algorithms. As the research field is comparatively young, it must be assumed that new quantum algorithms will be found, tackling general computational problems, which can be reformed to represent mining-related problems or geoscientific problems in general.

5.3 Consequences for Open-Pit Mining

In recent years, substantial advancements in the construction and accessibility of gate-based quantum computers and quantum annealing quantum computers have been made. Many leading global technology companies like Google, IBM, Amazon, Intel, and Microsoft have invested substantially in constructing new quantum computers and inventing new quantum algorithms with the foundation of new sub-companies and research groups. Numerous worldwide known corporations and institutions already utilize quantum cloud services to research the applicability of quantum computation for their specific use case in the future. For example, the list of IBMQ partners includes financial companies like JPMorgan Chase and Goldman Sachs, technology companies like Daimler, Samsung, and Boeing, and also resource companies like ExxonMobil, showing the predicted relevance of this computational technology for multiple economic branches, including mining (Anis et al., 2021).

Quantum computers already demonstrate substantially beneficial runtimes compared to classical computation for selected problems. Runtime reductions in problem-solving are archived with numerous algorithms utilizing quantum

computation. For specific problems (sect. 2.3), this reduction is conjectured to be exponential, rendering computational problems that were previously estimated to be infeasible on classical hardware feasible with quantum computation.

Quantum computation will be a relevant technology for mine planning and mine optimization with a high probability in the future. However, it is required to distinguish between gate-based quantum computers and quantum annealing quantum computers. Quantum annealing quantum computers already show substantial advantages in solving optimization problems modeled as cost landscapes, heavily outperforming methods used in modern open-pit mining optimization software products. A study conducted by Google in 2015 shows that quantum annealing outperforms simulated annealing (sect. 2.4.3), an algorithmic method already outperforming industry-standard optimization software, by a factor of $\sim 10^8$ considering runtime with a 99% success probability on D-Wave quantum annealing computers for specific problems (Denchev et al., 2016).

The NPV increase through open-pit optimization that is possible utilizing quantum computation is hard to estimate. However, even simulated annealing, a technology far inferior to quantum annealing as stated above, archives NPV increases more than 20% compared to industry-standard optimization software applications (sect. 2.4). The potential revenue increase through quantum computer supported mining optimization can therefore be expected to be substantial.

It needs to be noted that quantum computational hardware is not yet advanced enough to outperform conventional supercomputers for complex optimization problems. This is especially true for gate-based quantum computers, which are more challenging to construct and need to be fitted with quantum error corrective algorithms. The most potent gate-based quantum computer at work today is the 127 Qubit quantum computer eagle constructed and provided by IBMQ. This quantum computer can, for example, only encode the provided quantum phase estimation code described in section 5 with a maximal amount of around 40 nodes, not considering runtime development, which is far from industrial usability. Quantum annealing quantum computers, by contrast, already offer substantially more usable qubits. The D-Wave computer Advantage, developed in 2020 and in one version constructed in January 2022 in Jülich, Germany as the first European-based quantum computer, provides 5640 usable qubits.

6 Conclusion

In conclusion, quantum computers have distinct runtime and accuracy advantages in calculating optimization problems hard or impossible to compute on classical computers, including many problems arising in mining optimization. The research field linking quantum computation to mining is new, and there are nearly no publications yet. However, general quantum computational research underlines the proposed advantages. It is required to distinguish gate-based quantum computation and quantum annealing. Gate-based quantum computation is more general but still requires generations of hardware improvements and implementation of quantum error correction. Quantum annealing is especially suited to solving optimization problems and will archive industrial usability in the next decade with high probability.

Implementation of quantum computation for mining optimization will result in relevant revenue increases with high probability and should be investigated in the future to stay competitive and cost-efficient.

Malte Leander Schade, Hattingen, March 22, 2022

A References

Abdessaied, N., & Drechsler, R. (2016). Reversible and quantum circuits [Publisher: Springer].

Ade, P. A., Aghanim, N., Arnaud, M., Ashdown, M., Aumont, J., Baccigalupi, C., Banday, A., Barreiro, R., Bartlett, J., Bartolo, N., et al. (2016). Planck 2015 results - XIII. cosmological parameters [Publisher: EDP Sciences]. *Astronomy & Astrophysics*, *594*, A13.

Anis, M. S., Abraham, H., AduOffei, Agarwal, R., Agliardi, G., Aharoni, M., Akhalwaya, I. Y., Aleksandrowicz, G., Alexander, T., Amy, M., Anagolum, S., Arbel, E., Asfaw, A., Athalye, A., Avkhadiev, A., Azaustre, C., Bhole, P., Banerjee, A., Banerjee, S., et al. (2021). Qiskit: An open-source framework for quantum computing. https://doi.org/10.5281/zenodo.2573505

Barbaro, R., & Ramani, R. (1986). Generalized multiperiod MIP model for production scheduling and processing facilities selection and location [Publisher: SME]. *Mining Engineering*, *38*(2), 107–114.

Bernhardt, C. (2019). *Quantum computing for everyone* [Publisher: MIT Press].

Caccetta, L., & Hill, S. P. (2003). An application of branch and cut to open pit mine scheduling [Publisher: Springer]. *Journal of global optimization*, *27*(2), 349–365.

Denchev, V. S., Boixo, S., Isakov, S. V., Ding, N., Babbush, R., Smelyanskiy, V., Martinis, J., & Neven, H. (2016). What is the computational value of finite-range tunneling? [Publisher: APS]. *Physical Review X*, *6*(3). https://doi.org/10.1103/physrevx.6.031015

Eivazy, H., & Askari-Nasab, H. (2012). A mixed integer linear programming model for short-term open pit mine production scheduling [Publisher: Taylor & Francis]. *Mining Technology*, *121*(2), 97–108.

Gallo, A. (2014). A refresher on net present value [Publisher: Harvard Business Publishing]. *Harvard Business Review*, *19*.

Gershon, M. E. (1983). Optimal mine production scheduling: Evaluation of large scale mathematical programming approaches [Publisher: Springer]. *International journal of mining engineering*, *1*(4), 315–329.

Goodfellow, R. C., & Dimitrakopoulos, R. (2016). Global optimization of open pit mining complexes with uncertainty [Publisher: Elsevier]. *Applied Soft Computing*, *40*, 292–304.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search [Publisher: ACM]. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 212–219.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th python in science conference* (pp. 11–15).

Harris, C. R., Millman, K. J., Walt, S. J. v. d., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. v., Brett, M., Haldane, A., Río, J. F. d., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy [Publisher: Springer]. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hauke, P., Katzgraber, H. G., Lechner, W., Nishimori, H., & Oliver, W. D. (2020). Perspectives of quantum annealing: Methods and implementations [Publisher: IOP Publishing]. *Reports on Progress in Physics*, *83*(5), 054401.

Hidary, J. D. (2019). *Quantum computing: An applied approach* [Publisher: Springer].

Hindy, Y., Pointing, J., Tolunay, M., Venkatarao, S., Motta, M., & Latone, J. A. (2021). A quantum computational approach to the open-pit mining problem. *arXiv preprint arXiv:2107.11345*.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment [Publisher: IEEE Computer Society]. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hustrulid, W. A., Kuchta, M., & Martin, R. K. (2013). *Open pit mine planning and design* [Publisher: CRC Press].

Impagliazzo, R., & Wigderson, A. (1997). P = BPP if E requires exponential circuits: Derandomizing the XOR lemma [Publisher: ACM]. *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 220–229.

Kadowaki, T., & Nishimori, H. (1998). Quantum annealing in the transverse ising model [Publisher: APS]. *Physical Review E*, *58*(5), 5355.

Lawler, E. L., Lenstra, J. K., Kan, A. H. R., & Shmoys, D. B. (1993). Sequencing and scheduling: Algorithms and complexity [Publisher: Elsevier]. *Handbooks in operations research and management science*, *4*, 445–522.

Paesani, S., Gentile, A. A., Santagati, R., Wang, J., Wiebe, N., Tew, D. P., O'Brien, J. L., & Thompson, M. G. (2017). Experimental bayesian quantum phase estimation on a silicon photonic chip [Publisher: APS]. *Physical review letters*, *118*(10), 100503.

Ramazan, S., & Dimitrakopoulos, R. (2004). Traditional and new MIP models for production scheduling with in-situ grade variability [Publisher:

Taylor & Francis]. *International Journal of Surface Mining*, *18*(2), 85–98.

Reed, M. D., DiCarlo, L., Nigg, S. E., Sun, L., Frunzio, L., Girvin, S. M., & Schoelkopf, R. J. (2012). Realization of three-qubit quantum error correction with superconducting circuits [Publisher: Nature Publishing Group]. *Nature*, *482*(7385), 382–385.

Shannon, C. E. (1993). *Claude elwood shannon: Collected papers* [Publisher: IEEE press].

Shepard, R. N. (1964). On subjectively optimum selection among multiattribute alternatives [Publisher: APA]. *Human judgments and optimality*, 257–281.

Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer [Publisher: SIAM]. *SIAM review*, *41*(2), 303–332.

Srinivasan, K., Satyajit, S., Behera, B. K., & Panigrahi, P. K. (2018). Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience. *arXiv preprint arXiv:1805.10928*.

B Appendices

B.1 Abbreviations

(Sorted in alphabetical order.)

API	=	Application Programming Interface
CNOT	=	Controlled Not Gate
$CROT_x$	=	Controlled Rotation Gate (Around X-Axis)
CU	=	Controlled Unitary Gate
DFT	=	Discrete Fourier Transformation
H	=	Hadamard Gate
IQFT	=	Inverse Quantum Fourier Transformation
LP	=	Linear Programming
MILP	=	Mixed Integer Linear Programming
NPV	=	Net Present Value
P	=	Phase Rotation Gate
QAOA	=	Quantum Approximate Optimization Algorithm
QFT	=	Quantum Fourier Transformation
QPE	=	Quantum Phase Estimation
SA	=	Simulated Annealing
SDK	=	Software Development Kit
TSP	=	Traveling Salesman Problem
U	=	General Unitary Gate
$UROT_x$	=	Rotation Gate (Around X-Axis)
VQE	=	Variational Quantum Eigensolver

B.2 Nomenclature

B.2.1 Cutoff Grade Problem

Optimization sub-problem arising in mining, where the optimization function is to maximize all mining of ore above a specific ore purity/cutoff ore grade (Barbaro and Ramani, 1986).

B.2.2 Minimal Transport Cost Problem

Optimization sub-problem arising in mining, where the optimization function is the minimization of ore and waste transportation length, count, and height difference (Barbaro and Ramani, 1986).

B.2.3 Mixing Problem

Optimization sub-problem arising in mining, where the optimization function is keeping a mixed ore grade in the rock crusher over a certain threshold (Barbaro and Ramani, 1986).

B.2.4 Particle Swarm Optimization

Metaheuristic optimization algorithm that is inspired by swarm intelligence of animals (Goodfellow and Dimitrakopoulos, 2016).

B.2.5 Differential Evolution

Multidimensional metaheuristic optimization algorithm that combines existing solutions to new optimization candidates (Goodfellow and Dimitrakopoulos, 2016).

B.2.6 Pseudo Flow Algorithm

Metaheuristic optimization algorithm, which is setting up the optimization problem as a combination of sources and sinks and tries to find points with maximum flow (Goodfellow and Dimitrakopoulos, 2016).

B.2.7 Deterministic Turing Machines

Abstract mathematical concept of a machine that acts on a string of numbers or letters in a deterministic and predictable way. Classical computers are deterministic Turing machines (Hidary, 2019).

B.2.8 Probabilistic Turing Machines

Turing machines that do not act deterministically but with connected probabilities, usable for the description of randomized algorithms (Hidary, 2019).

B.2.9 Branch-and-Cut algorithm

Optimization algorithm for mixed integer linear programming applications. The target is to minimize a function over a cost polygon (Ramazan and Dimitrakopoulos, 2004).

B.2.10 Simplex algorithm

Linear programming optimization algorithm that tries to find the cost optimal solution which is encoded in the vertices of a polygon (Ramazan and Dimitrakopoulos, 2004).

B.2.11 Discrete Search space

A finite amount of possible solutions to an optimization problem (Bernhardt, 2019).

B.2.12 Euclidean Distance

The distance of two points in Euclidean geometry. For $\mathbb{R}^2$ it is the Pythagorean theorem.

B.2.13 Kronecker Product

The matrix-matrix product, that results in a larger matrix having a field for every multiplied field combination exactly once.

B.2.14 Hermitian Conjugate

Analogous mathematic operator for matrix transposing that is generalized for Hilbert spaces (Hidary, 2019).

B.2.15 Elementary Particles

Fundamental building parts of the universe for example photons, electrons, gluons, and leptons.

B.2.16 Spin

Angular momentum of elementary particles that can be utilized in quantum computation. It is not a mechanical rotation (Hidary, 2019).

B.2.17 Hilbert Space

Generalized vector space with real or complex numbers in multiple dimensions. It offers a scalar product (Abdessaied and Drechsler, 2016).

B.2.18 Born Rule

Quantum mechanical law linking the probability amplitude of quantum systems with measurement probabilities when observing it (Hidary, 2019).

B.2.19 No Cloning Theorem

Quantum mechanical law that states that it is impossible to perfectly copy the quantum information of one quantum particle to another without altering the first ones quantum state (Hidary, 2019).

B.2.20 No Deleting Theorem

Quantum mechanical law that states that it is impossible to delete the information of one quantum state which is a copy of another one, without altering the second. It is a time reversed version of the No-Cloning-Theorem (Hidary, 2019).

B.2.21 Pauli Group

Matrix group of 16 matrices which act on the state of a single qubit and alter it (Bernhardt, 2019).

B.2.22 Time Domain

Mathematical function space with evolution time of the system on one of its axis (Anis et al., 2021).

B.2.23 Frequency Domain

Mathematical function space with measurement frequency on one of its axis opposed to time (Anis et al., 2021).

B.2.24 Euler's Formula

Formula linking trigonometric and complex functions (Anis et al., 2021).

$$e^{ix} = cos(x) + i \cdot sin(x); x \in \mathbb{R}. \tag{75}$$

B.2.25 Ground state

State of a quantum system that has its lowest intrinsic energy, which is also called the vacuum state (Hidary, 2019).

B.2.26 Intermediate Representation

Programming code that is interpreted from the high-level programming environment syntax to code that is nearer to the machine execution level (Anis et al., 2021).

B.2.27 OpenQASM

Intermediate representation for quantum circuits. All Qiskit programs are compiled down to OpenQASM instructions (Anis et al., 2021).

B.2.28 Iterative Quantum Phase Estimation

Quantum phase estimation algorithm that reduces the amount of required qubits and circuit depth by iteratively reading out phases with only one auxiliary qubit (Anis et al., 2021).

B.2.29 Quantum Approximate Optimization Algorithm

Metaheuristic quantum algorithm which reversely estimates a solution by introducing changed setup parameters (Anis et al., 2021).

B.2.30 Quantum Variational Eigensolver

Metaheuristic quantum algorithm that estimates the eigenvalue of a matrix (Anis et al., 2021).

B.3 Graphs and Figures

B.3.1 QPE Simple Circuit

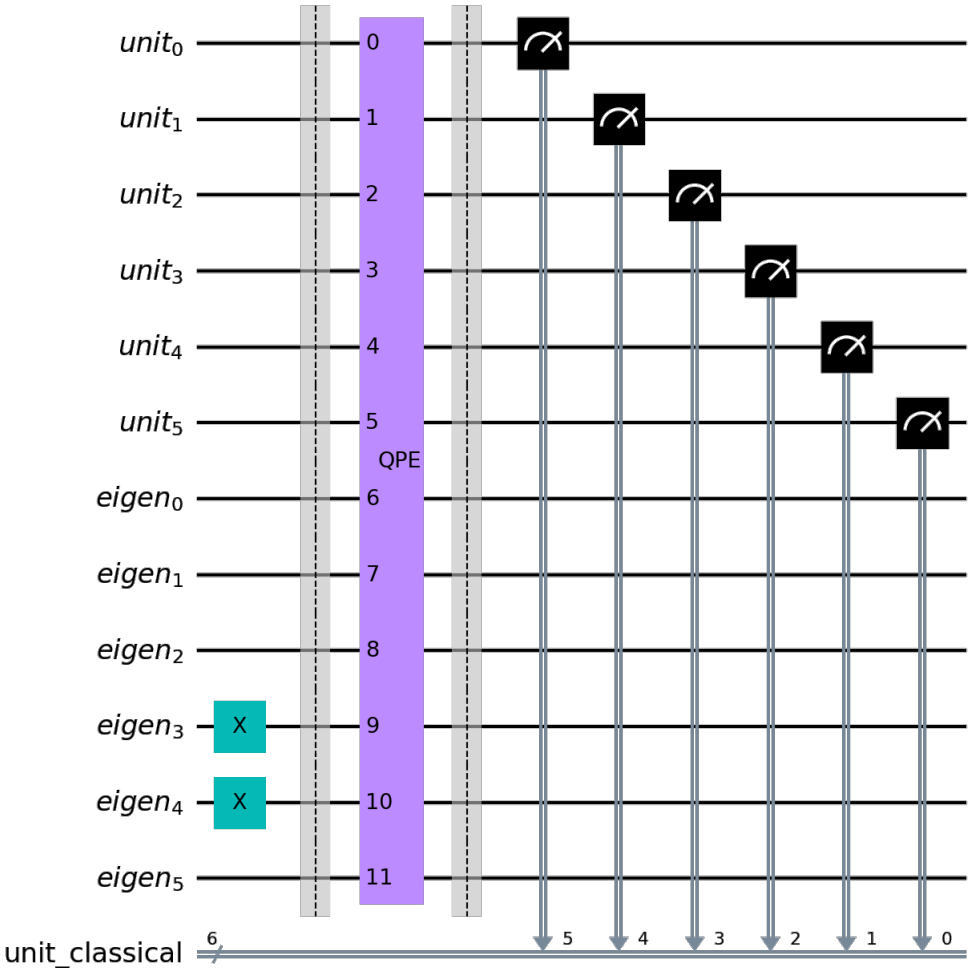

Figure 8: Simplified quantum circuit for quantum phase estimation. Both quantum registers hold 6 qubits. In the eigen register the quantum state $|000110\rangle$ is encoded through the appliance of **X**-gates. The 6 qubits of register unit are measured to 6 classical bits. The vertical barriers ensure that every circuit step is executed separately.

B.3.2 QPE Extended Circuit

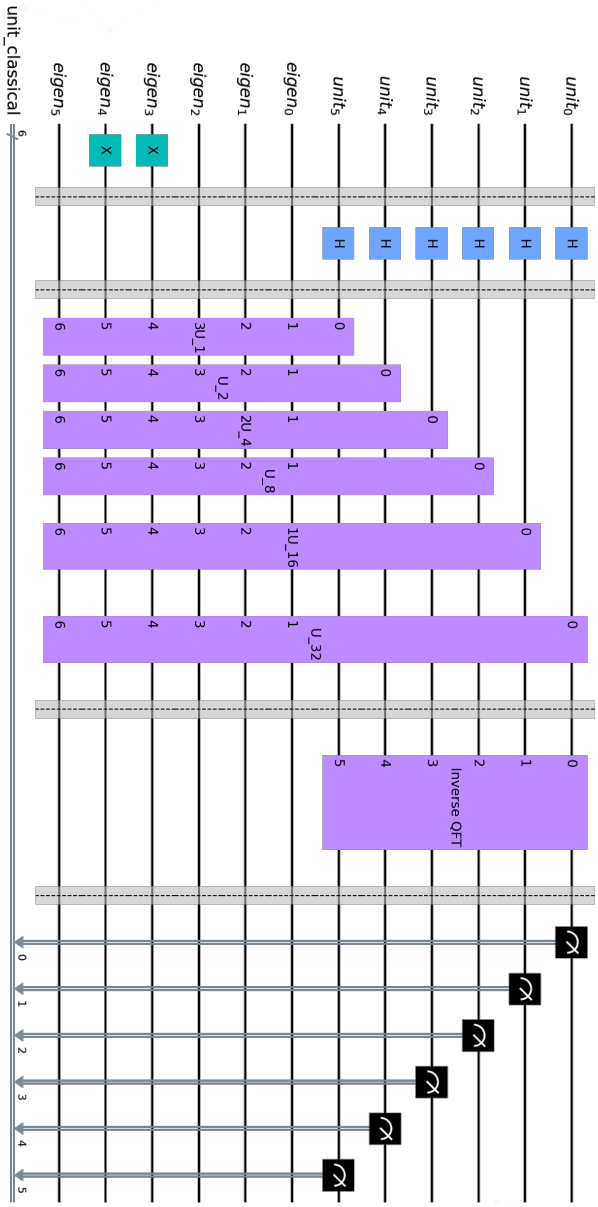

Figure 9: Analog quantum circuit to appendix B.3.1 with a differentiated QPE unitary gate. For every consecutive unit qubit, the calculated unitary U is applied twice the previous number of times. .

B.3.3 Simulator Measurements

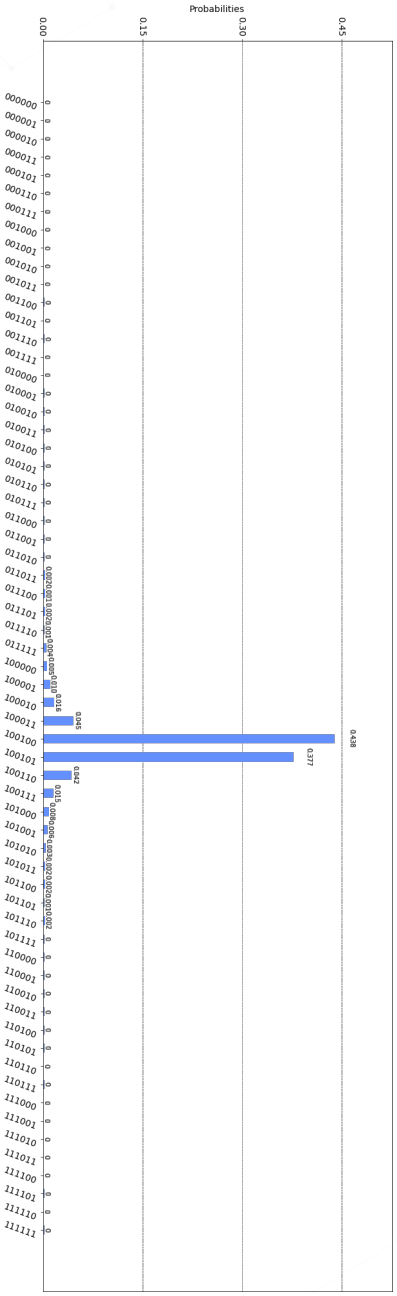

Figure 10: Measurement results for the TSP experiment with the cost matrix eq. 41, executed on the simulated Qiskit quantum computer backend aer_simulator.

B.3.4 IBM Quantum Computer Circuit

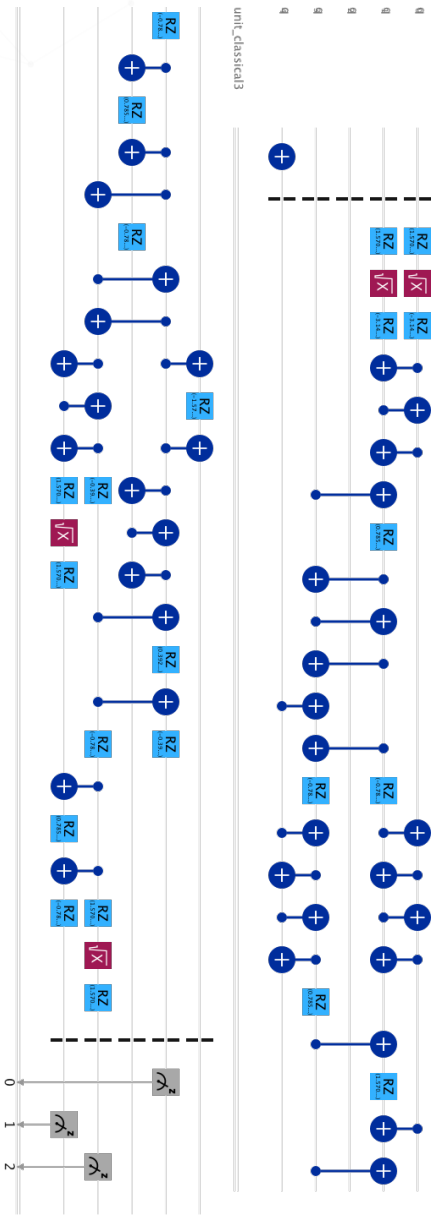

Figure 11: Hardware implementation of the provided code (appx. B.4) in the execution on *ibmq_belem*. The code is formulated as an arrangement of basic unitary gates and describable with QASM. Only three gates are realized on hardware level: **H**-like-gates, **RZ**-gates, and **CNOT**-gates.

B.4 Python 3 Code

The Python 3 code implementation of the described TSP in section 4 is provided in GitLab under the link (*private_git_hub*). The code offers a detailed documentation and is free to use and modify. It is licensed under the MIT License. No warranty of any kind is given. Questions considering the implementation of the quantum phase estimation algorithm and thesis in general can be send to: *contact@malteschade.com*.