Leibniz Universität Hannover
Institut für Theoretische Physik

Near-Term Quantum-Algorithmic Approaches to the Facility Location Problem

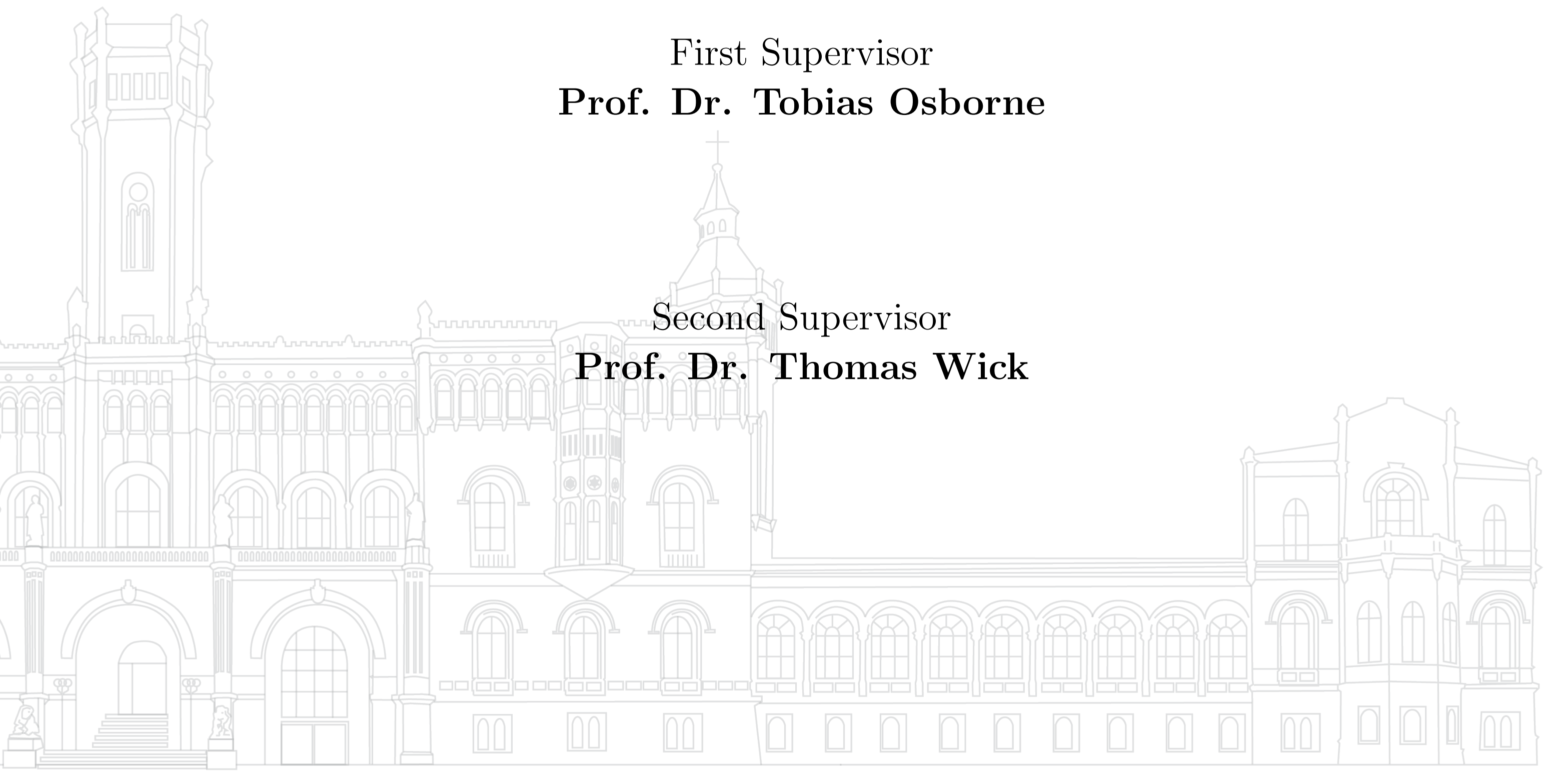
Master's Thesis

Nico Buß

10025147

First Supervisor
Prof. Dr. Tobias Osborne

Second Supervisor
Prof. Dr. Thomas Wick



Contents

1 Introduction

In this thesis, we discuss the Facility Location Problem (FLP), a combinatorial optimisation problem where a company must supply a set of customers $\mathcal{D}$. To achieve this, a predetermined set of potential facilities $\mathcal{F}$ is available. The objective is to select a subset $\mathcal{S} \subset \mathcal{F}$ of facilities to be opened. Each customer $j \in \mathcal{D}$ must be connected to exactly one open facility $i \in \mathcal{S}$, incurring a cost c_{ij}. Additionally, the opening cost for a facility i is f_i. The goal is to minimise the total costs while adhering to these constraints.
Chapter 2 explores social applications and various formulations of the FLP, including special cases like the Metric Uncapacitated FLP and generalisations such as the Capacitated FLP. For the remainder of this thesis, we will primarily focus on the Uncapacitated FLP, abbreviated as UCFLP or simply FLP. This implies that an organisation can supply any number of customers. Subsequently, other combinatorial optimisation problems are introduced, which will be referenced in later chapters.
The number of customers is usually denoted by m, and the number of facilities is denoted by n. In Chapter 3, we consider the FLP classically.
First, we define the terms solution space and optimal solution space. It is then demonstrated that the FLP is NP-hard, meaning it cannot be solved in polynomial time if P $\neq$ NP. We then introduce a greedy algorithm and a local search algorithm for solving the FLP classically. When minimising the objective function of the FLP, two main constraints must be satisfied. The assignment constraint requires that each customer is connected to exactly one facility, while the opening constraint stipulates that a customer can only be connected to an open facility. Additionally, there are equivalent constraints that do not alter the minimum of the objective function.
The main focus of this thesis is to present different quantum algorithms for the FLP. The necessary fundamentals are introduced in Chapter 4, drawing from the book "Quantum Computation and Quantum Information" [1] and the paper "Elementary gates for quantum computing" [2]. We begin with the Pauli matrices and other single-qubit operators. Next, we discuss some properties of unitary operators, especially their decomposition into rotation operators. After introducing operators for a single qubit, we will particularly consider controlled operations, which are crucial for our quantum algorithms.

In Chapter 5 on quantum approximation algorithms, we present a total of four different hybrid quantum algorithms. There are two different ways to create a hybrid quantum algorithm: softcoding and hardcoding. The main focus here is on how constraints are handled. In softcoding, constraints are replaced by a penalty, which is an additional term appended to the objective function which ensuring that states which do not fulfil the constraint receive very large function values. If both constraints are replaced by a penalty, the unconstrained FLP (UFLP) is obtained. There are no constraints for the UFLP, so the Quantum Approximate Optimisation Algorithm (QAOA) [3] can be applied to this problem. In this algorithm, a phase separator is combined with a universal QAOA mixer. Both operators depend on an angle, according to which classical optimisation is performed. When the two operators are applied consecutively p times to an initial state, the approximation improves with a larger p. This algorithm is abbreviated as UFLP from Chapter 5 onwards. QAOA is universally applicable to all unconstrained combinatorial optimisation problems.
The three other algorithms are specialised for the FLP. The first algorithm is the tailored variational algorithm presented in [4]. It is abbreviated as TV in the following. The TV algorithm is based on constructing a quantum circuit that satisfies the opening constraint. The assignment constraint is included via a penalty, resulting in the TV algorithm having both hardcoded and softcoded components. Another approach is the reverse, based on the UFLP algorithm, a new algorithm is developed which hardcodes the assignment constraint and softcodes the opening constraint. This approach is inspired by a quantum algorithm for the travelling salesperson problem presented in [5]. The idea is based on so-called SWAP gates, which exchange the values of two qubits. The adaptation is based on the fact that the SWAP gates are only used for an explicit customer j. For this customer, we swap the facility associated with them. A phase separator and a QAOA mixer are then applied to the state. As with the UFLP algorithm, these three operators can be applied to a state a total of p times, with the approximation improving with a larger p. This algorithm is referred to as SWAP due to the use of SWAP gates.
The SWAP algorithm requires a state as input that fulfils the assignment constraint. To achieve this, a quantum circuit is constructed to generate a superposition of all possible states that fulfil the assignment constraint. The opening constraint is also always fulfilled. This superposition can serve as the input for the SWAP algorithm.

This superposition can also be used to define a purely hardcoded algorithm for the FLP. For this, a decision is made for each individual customer. For each facility, an angle-dependent operator is used to determine whether they are connected or not. If they are connected, the algorithm moves on to the next customer; if not, an angle-dependent operator is applied to the connection to the next facility. This ensures that the customer is connected to the last facility at latest. Subsequently, a facility

is opened if it is connected to at least one customer. No softcoding is required for this algorithm, which is why it is referred to as constraint FLP (CFLP).
The algorithms are compared in the Chapter 6. The algorithms are first analysed with regard to the number of gates required. This is followed by an explicit example for two customers and two facilities, which is first analysed, then the algorithms are applied to it. To better compare the algorithms, we generated random instances and calculated the average approximation for different instance sizes. When comparing the algorithms, the dependency on p is shown for the SWAP and UFLP algorithms. For larger instances, the algorithms TV and CFLP are compared in particular. An alternative hardcoded algorithm is also developed for the CFLP algorithm, differing from the original CFLP algorithm for $n \geq 3$.
Finally, the distribution of the approximation is shown and the percentage of cases in which the minimum is approximated well is examined. These tests help to identify which of the algorithms performs best.

2 Facility Location Problem

When expanding, many companies face the challenge of selecting new locations. There are numerous examples of this. For instance, if a company wants to open a new facility, the ideal location would be one that has a particularly large number of potential customers nearby, allowing them to reach the shop quickly.
If we consider hospitals, for example, paramedics should reach every patient within 15 minutes, regardless of their location in Lower Saxony. Here, not only the geometric distance but also the infrastructure between hospitals and patients is crucial.
Additionally, it is important to bear in mind that hospitals have limited capacity, so there must be multiple hospitals in densely populated areas. (cf. [6])
As school attendance is compulsory in Germany, millions of children go to school every day, making the choice of school locations a complex task. Important factors include the number of pupils in the neighbourhood and the corresponding required building size. Additionally, the distribution of pupils and maximum travel time must also be taken into account.
For delivery companies, it is essential to choose ideal locations so that drivers have the shortest routes to customers. This has numerous benefits, such as reducing travel costs and journey times, which helps lower expenses and increases customer satisfaction. If only these factors are considered, it would make sense to open a large number of locations.
However, other factors must also be considered. One important factor is that each location incurs costs, such as rental or purchase expenses, which also depend on the location. For example, a facility in the city would be more expensive than one in the countryside, but there are also more potential customers in the city. If a delivery company has too many facilities, it also incurs more delivery costs for the procurement of goods.
Therefore, when choosing locations, it is always a compromise between reducing delivery costs, which include both the supplier's salary and transport costs, and the location costs. Location costs include acquisition costs for equipment as well as rental or purchase expenses.

2.1 Mathematical Modelling of Location Problems

In this chapter, various location problems are presented to give a more detailed overview of the Facility Location Problem.

Uncapacitated Facility Location Problem

First of all, we need to mathematise our problem. To do this, we first need a set of potential facilities $\mathcal{F}$ that can theoretically be opened. Additionally, there is a set of customers $\mathcal{D}$ that must be supplied. In open a factory $i \in \mathcal{F}$, we need to consider the opening costs f_i. The delivery costs from factory $i \in \mathcal{F}$ to customer $j \in \mathcal{D}$ are denoted by $c_{i,j}$. In order to minimise costs, we seek a subset of facilities $\mathcal{S} \subset \mathcal{F}$ to actually open, and an assignment σ that assigns a facility $i \in \mathcal{S}$ to each customer $j \in \mathcal{D}$.

Problem 1: Uncapacitated Facility Location Problem (UCFLP)

Given are:

- a finite set of facilities $\mathcal{F}$
- a finite set of customers $\mathcal{D}$
- opening costs $f_i \in \mathbb{R}_+$ for a facility $i \in \mathcal{F}$
- delivery costs $c_{i,j} \in \mathbb{R}_+$ for factory $i \in \mathcal{F}$ delivering customer $j \in \mathcal{D}$.

and we would like to find out:

- a subset $\mathcal{S} \subset \mathcal{F}$ of facilities to be opened
- a assignment $\sigma : \mathcal{D} \to \mathcal{S}$ which assign each customer to a open facility

such that the following sum is minimised

$$\sum_{i \in \mathcal{S}} f_i + \sum_{j \in \mathcal{D}} c_{\sigma(j),j}.$$

The notation and definitions are adapted from [7, page 13]. An instance of the UCFLP is defined by the tuple $(\mathcal{F}, \mathcal{D}, fi, ci, j)$. Many applications of UCFLP pertain to deliveries from companies to customers. Typically, a metric is used to calculate the distance, with the Euclidean metric being common for short distances. Up to this point, there are no specific restrictions on how the delivery costs are calculated or how they relate to each other.

Metric Uncapacitated Facility Location Problem

If a driver delivers from facility $i \in \mathcal{F}$ to customer $j \in \mathcal{D}$ and then drives to facility $i' \in \mathcal{F}$ to deliver to customer $j' \in \mathcal{D}$, this should be more expensive than if the driver delivers directly from facility $i \in \mathcal{F}$ to customer $j' \in \mathcal{D}$. This can be expressed as the inequality

$$c_{i,j} + c_{i',j} + c_{i',j'} \geq c_{i,j'}.$$

With this groundwork, we can define the “Metric Uncapacitated Facility Location Problem“. Here we present the definition of [7, page 15] in more detail.

Problem 2: Metric Uncapacitated Facility Location Problem (MUCFLP)

Let $(\mathcal{F}, \mathcal{D}, f_i, c_{i,j})$ be a Uncapacitated Facility Location Problem. If $c_{i,j}$ satisfied

$$c_{i,j} + c_{i',j} + c_{i',j'} \geq c_{i,j'} \tag{2.1}$$

with

- $c_{ii} = 0$ and $c_{ii'} = \min_{j \in \mathcal{D}}(c_{i,j} + c_{i',j})$ for $i, i' \in \mathcal{F}$
- $c_{j,j} = 0$ and $c_{jj'} = \min_{i \in \mathcal{F}}(c_{i,j} + c_{i,j'})$ for $j, j' \in \mathcal{D}$
- $c_{i,j} = c_{ji}$ $\forall i, j \in \mathcal{D} \cup \mathcal{F}$

we called $(\mathcal{F}, \mathcal{D}, f_i, c_{i,j})$ a Metric Uncapacitated Facility Location Problem.

The Metric Uncapacitated Facility Location Problem (MUCFLP) is named ”metric” due to the special property that $c_{i,j}$ for all $i, j \in \mathcal{D} \cup \mathcal{F}$ satisfies the characteristics of a metric. The properties of positivity and symmetry are inherently present by definition. To demonstrate the triangle inequality, consider $i, j, k \in \mathcal{D} \cup \mathcal{F}$, then we get:

$$c_{i,k} \leq c_{i,j} + c_{j,j} + c_{j,k} = c_{i,j} + c_{j,k}.$$

UCFLP as Linear Program

Now we show that the UCFLP can be formulated as an integer linear program. Formulating it as a linear program allows us to ignore the assignment function σ. Instead, we introduce the binary variables $x_{i,j}$ $\forall i \in \mathcal{F}, j \in \mathcal{D}$, which have the value 1 if the customer j is supplied by the facility i. Otherwise, the value is 0. The sum over the open facilities $\mathcal{S}$ can be replaced by the sum over all facilities. To achieve this, we introduce the binary variable y_i $\forall i \in \mathcal{F}$. This variable is only set to 1 if the facility $i \in \mathcal{F}$ is open.

Theorem 1: UCFLP as an Integer Linear Program

Let $(\mathcal{F}, \mathcal{D}, f_i, c_{i,j})$ be a Uncapacitated Facility Location Problem. Then it is equivalent to the following integer linear program:

$$\min \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{i,j} x_{i,j}$$

under the secondary conditions:

- $x_{i,j} \leq y_i \quad \forall i \in \mathcal{F}, j \in \mathcal{D}$
- $\sum\limits_{i \in \mathcal{F}} x_{i,j} = 1 \quad \forall j \in \mathcal{D}$
- $x_{i,j} \in \{0, 1\} \quad \forall i \in \mathcal{F}, j \in \mathcal{D}$
- $y_i \in \{0, 1\} \quad \forall i \in \mathcal{F}.$

Proof

Let $(\mathcal{F}, \mathcal{D}, f_i, c_{i,j})$ be a Uncapacitated Facility Location Problem. We can change the two sums in the following way:

$$\sum_{i \in \mathcal{S}} f_i = \sum_{i \in \mathcal{F}} f_i \cdot y_i, \qquad \sum_{j \in \mathcal{D}} c_{\sigma(j)j} = \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{i,j} \cdot x_{i,j}.$$

The binary variables $x_{i,j}, y_i$ are defined as follows:

$$y_i = \begin{cases} 1 \text{ if } i \in \mathcal{S} \\ 0 \text{ if } i \notin \mathcal{S} \end{cases}, \qquad x_{i,j} = \begin{cases} 1 \text{ if } \sigma(j) = i \\ 0 \text{ if } \sigma(j) \neq i \end{cases}.$$

Now we show $x_{i,j} \leq y_i$. If $x_{i,j} = 0$ then we have nothing to show. Let $x_{i,j} = 1$, then $\sigma(j) = i$ and σ maps to $\mathcal{S}$, which implies: $i \in \mathcal{S}$. It follows that $y_i = 1$.
Finally we show $\sum\limits_{i \in \mathcal{F}} x_{i,j} = 1$:

$$\sum_{i \in \mathcal{F}} x_{i,j} = \sum_{i \in \mathcal{F} \setminus \{\sigma(j)\}} x_{i,j} + x_{\sigma(j),j} = x_{\sigma(j),j} = 1.$$

$\square$

In many cases, the number of facilities and customers is given. The following problem formulation is equivalent to UCFLP. This is shown in theorem 1 considering that $|\mathcal{F}| = n, |\mathcal{D}| = m$.

Problem 3: Facility Location Problem (FLP)

Given are:

- $y_i = \begin{cases} 1 \text{ if facility } i \text{ is opened} \\ 0 \text{ otherwise} \end{cases}$ for $i \in \{1, \ldots, n\}$
- $x_{i,j} = \begin{cases} 1 \text{ if facility } i \text{ serves customer } j \\ 0 \text{ otherwise} \end{cases}$ for $i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$
- $f_i, c_{i,j} \in \mathbb{R}_+$.

The Facility Location Problem is the problem

$$\min_{x,y} \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} x_{i,j} \tag{2.2}$$

with the constraints

$$\sum_{i=1}^{n} x_{i,j} = 1, \ \forall j \in \{1, \ldots, m\} \tag{2.3}$$

$$x_{i,j} \leq y_i \ \forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}. \tag{2.4}$$

We will then denote an instance of the FLP by $(\mathcal{F}, \mathcal{D}, fi, ci, j, xi, j, yi)$. The first constraint 2.3, is referred as the assignment constraint, and the second constraint 2.4, is referred as the opening constraint.

k-Facility Location Problem

In some cases, companies cannot or choose not to open more than a certain number of facilities. This can be for various reasons, such as having too few employees or the logistics between locations becoming too complicated. To accommodate these scenarios, an additional constraint can be added to the Facility Location Problem, which is explained in the following definition.

Problem 4: k-Facility Location Problem

Let $(\mathcal{F}, \mathcal{D}, f_i, c_{i,j}, x_{i,j}, y_i)$ be a Facility Location Problem. We call it k-Facility Location Problem if y_i fulfils the condition

$$\sum_{i \in \mathcal{F}} y_i \leq k, \text{ for } k \in \mathbb{N}.$$

By definition, the k-FLP restricts the number of open facilities to a maximum of k. This definition is taken from [7, page 34]. The k-FLP is a generalisation of the FLP. If $k \geq n$ holds, then the k-FLP can also be considered as FLP.

Capacitated Facility Location Problem

To make the FLP even more realistic, several other factors must be considered. Firstly, different customers do not always require the same quantity of goods; for example, the needs of a large company can greatly differ from those of a private individual.
Additionally, an organisation cannot produce an unlimited amount of goods; there is a maximum number of goods that can be produced within a certain time frame. Therefore, customers should have the ability to order from multiple companies if, for instance, one facility is fully utilised.
Now we define the Capacitated Facility Location Problem (CFLP). We integrate the explanations from [7, pages 44/45] and [8] into a mathematical definition. For the facility $i \in \mathcal{F}$, we denote the maximum production capacity by u_i and the demand of a customer by d_j.
The first constraint ensures that the customer's demand is fulfilled. The second constraint stipulates that the facility $i \in \mathcal{F}$ must not accept more orders than its capacity. It is also specified that a customer cannot order more from a facility than they need and can only order from open facilities.
The variable $x_{i,j}$ now describes how much the customer $j \in \mathcal{D}$ orders from the facility

$i \in \mathcal{F}$. The penultimate condition prevents this value from becoming negative. As with the Facility Location Problem, y_i indicates whether the facility is open (value of 1) or not (value of 0).

Problem 5: Capacitated Facility Location Problem (CAFLP)

Given are:

- a finite set of facilities $\mathcal{F}$ and customers $\mathcal{D}$
- opening costs $f_i \in \mathbb{R}_+ \ \forall i \in \mathcal{F}$ and delivery costs $c_{i,j} \in \mathbb{R}_+ \ \forall i \in \mathcal{F}, j \in \mathcal{D}$
- a capacity $u_i \geq 0 \ \forall i \in \mathcal{F}$ and a demand $d_j \geq 0 \ \forall j \in \mathcal{D}$

and we would like to minimise

$$\sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{i,j} x_{i,j}$$

under the secoundary conditions:

- $\sum\limits_{i \in \mathcal{F}} x_{i,j} = d_j \ \forall j \in \{1, \dots, m\}$
- $\sum\limits_{j \in \mathcal{D}} x_{i,j} \leq u_i y_i \ \forall i \in \{1, \dots, n\}$
- $x_{i,j} \geq 0 \ \forall i \in \{1, \dots, n\}, j \in \{1, \dots, m\}$
- $y_i \in \{0,1\} \ \forall i \in \{1, \dots, n\}$.

Now we show that the FLP is a special case of the CFLP.

Theorem 2: FLP as a Special Case of CAFLP

Let $(\mathcal{F}, \mathcal{D}, f_i, c_{i,j}, u_i, d_j)$ be a CFLP. The CAFLP is an FLP under the following conditions.

- $d_j = 1 \ \forall j \in \mathcal{D}$
- $u_i = |\mathcal{D}| \ \forall i \in \mathcal{F}$.

Proof

We have to show that the constraints of the FLP are now fulfilled. Since $d_j = 1$, it follows from the first constraint that the assignment constraint

$$\sum_{i \in \mathcal{F}} x_{i,j} = 1$$

is fulfilled. Combined with the third constraint, it follows $x_{i,j} \in [0,1]$. $u_i = |\mathcal{D}| \; \forall i \in \mathcal{F}$ shows that each facility can deliver to all customers at the same time. Now we will show that $x_{i,j} \leq y_i$ applies. If $y_i = 1$, we do not have to show anything because $x_{i,j} \in [0,1]$ is fulfilled. If $y_i = 0$, then applies:

$$x_{i,j} \leq \sum_{j \in \mathcal{D}} x_{i,j} \leq u_i y_i = u_i \cdot 0 = 0.$$

Thus $x_{i,j} \leq y_i$.
Last we must show $x_{i,j} \in \{0,1\}$. Let $y_i = 1$. From $\sum\limits_{i \in \mathcal{F}} x_{i,j} = 1$ it follows that there is an decomposition $\sum\limits_{i \in \mathcal{F}} x_{i,j} = \sum\limits_{i=1}^{n} x_{i,j} = 1$. Let $\mathcal{F}' := \{i_1, \ldots, i_p\} \subset \mathcal{F}$ be the facilities that supply the customer j. Let k the facility with $c_{k,j} \leq c_{i,j} \forall i \in \mathcal{F}'$ then:

$$\sum_{i \in \mathcal{F}'} f_i + \sum_{i \in \mathcal{F}'} c_{i,j} x_{i,j} \geq \sum_{i \in \mathcal{F}'} f_i + c_{k,j} \sum_{i \in \mathcal{F}'} x_{i,j} = \sum_{i \in \mathcal{F}'} f_i + c_{k,j} \geq f_k + c_{k,j}.$$

It is therefore more favourable to be supplied by one facility than to be supplied by many facilities in parts. If $k = i$ it follows $x_{i,j} = 1$ otherwise $x_{i,j} = 0$. This means that a minimum of the FLP fulfils $x_{i,j} \in \{0,1\}$. $\square$

2.2 Relationship to Other Problems

Set Cover Problem

The Set Cover Problem is an NP-hard problem (more on this in the Chapter 3.3). Given a set $\mathcal{U}$ and a set of subsets from $\mathcal{U}$ which is called $\mathcal{S} = \{S_1, \dots, S_k\}$ with $S_i \subset \mathcal{U}$. We search for the smallest $\mathcal{S}' \subset \mathcal{S}$ such that $\mathcal{S}'$ covers the set $\mathcal{U}$. The problem definition is taken from [9, page 35].

Problem 6: Set Cover Problem

Given are:

- Set $\mathcal{U}$ with $|\mathcal{U}] = n$
- $\mathcal{S} = \{S_1, \dots, S_k\}, S_i \subset U$ with $i \in \{1, \dots, k\}$
- cost function $c : \mathcal{S} \to \mathbb{R}$

we want to minimise:

$$\sum_{S \in \mathcal{S}'} c(S)$$

under the secondary condition:

- $\mathcal{U} = \bigcup_{S \in \mathcal{S}'} S.$

The Set Cover Problem is used in the theorem 6 to show that the FLP is an NP-hard problem.

Traveling Salesperson Problem

Another problem that has a certain connection to the FLP is the travelling salesperson problem [10]. This problem involves determining how a salesperson can visit a certain number of cities as quickly as possible. It can be represented as a graph V, where each pair of vertices $v, w \in V$ corresponds to two cities. The cost $c_{v,w}$ represents the travel time between the cities. The objective is to visit all cities, but visit each city only once and minimise the travel time. This leads to the following problem formulation.

Problem 7: Traveling Salesperson Problem

Given are:

- set of n points V
- costs $c_{v,w} \in \mathbb{R}_+$ for traveling from city v to w
- n^2 variables $x_{v,i} = \begin{cases} 1 \text{ if the salesperson is in the city } v \text{ at time } i \\ 0 \text{ otherwise} \end{cases}$

we want to minimise

$$\sum_{i=1}^{n} \sum_{w \in V \setminus \{v\}} \sum_{v \in V} c_{v,w}(x_{v,i}x_{w,i+1} + x_{v,i+1}x_{w,i})$$

under the secondary conditions:

- $\sum_{i=1}^{n} x_{v,i} = 1 \ \forall v \in V$
- $\sum_{v \in V} x_{v,i} = 1 \ \forall i \in \{1, \ldots, n\}.$

The first constraint specifies that each city must be visited exactly once, while the second constraint describes that the salesperson can be in exactly one city at any given time i. These constraints show an interesting connection to the FLP, where each customer must be connected to exactly one facility. This connection is later used to adapt a quantum algorithm, developed for the travelling salesperson problem to the FLP. This adaptation is presented in the Chapter 5.4.

3 Classical Observations of the FLP

In this chapter, we demonstrate that there are currently no fast, exact classical algorithms for the FLP, and therefore, other approaches need to be considered.

3.1 Types of Solutions

Before we find out how well classical approximations work for the FLP, we first have to define what we want to achieve in the optimal case. To do this, we will define the concept of the combinatorial optimisation problem as well as the terms feasible and optimal solution, which are mainly taken from the work of [10]. First, we start with the definition of a combinatorial minimisation problem.

Definition 1: Combinatorial Minimisation Problem

Given are:

- a number $N \in \mathbb{N}$ of bits
- costs $c_a \in \mathbb{R}_+, a \in \{1, \dots A\}$
- clauses $C_a : Z(N) \to \{0,1\}, a \in \{1, \dots A\}$ which are satisfied if $C_a(\mathbf{z}) = 1$
- constraints $D_b : Z(N) \to \{0,1\}, b \in \{1, \dots B\}$ with are satisfied if $D_b(\mathbf{z}) = 1$

we want to minimise the objective function:

$$C = \sum_{a=1}^{A} c_a C_a$$

under the secondary conditions:

- $D_b(\mathbf{z}) = 1 \ \forall b \in \{1, \dots B\}$.

We call the CMP := $(N, \{c_a\}_{a=1}^A, \{C_a\}_{a=1}^A, \{D_b\}_{b=1}^B)$ a combinatorial minimisation problem, for $B = 0$ we define $\{D_b\}_{b=1}^B = \emptyset$. $Z(N) = \{0{,}1\}^N$ describes the set of N-bit strings. The Facility Location Problem is a combinatorial minimisation problem, making it a good example to understand the definition better. The following function is to be minimised in the FLP:

$$\sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} x_{i,j}.$$

The objective function of the FLP can be obtained with the following assignment:

$$\begin{aligned}(i,j) &\mapsto (i-1)\cdot m + j \ \forall i \in \{1,\dots,n\}, j \in \{1,\dots,m\}\\ i &\mapsto nm + i \ \forall i \in \{1,\dots,n\}\end{aligned}$$

The costs c_a and clauses C_a can now be defined as follows:

$$\begin{aligned}c_a &= \begin{cases} c_{i,j} \text{ if } a \le nm \\ f_i \text{ if } nm < a \le nm+n \end{cases}, \ a \in \{1,\dots,nm+n\}\\ C_a &= \begin{cases} x_{i,j} \text{ if } a \le nm \\ y_i \text{ if } nm < a \le nm+n \end{cases}, \ a \in \{1,\dots,nm+n\}.\end{aligned}$$

This results in the following objective function for the FLP:

$$\sum_{a=1}^{nm+n} c_a C_a.$$

The assignment constraint can be formulated as follows:

$$\begin{aligned}D_b = &\begin{cases} 1 \text{ if } \exists!\ i \in \{1,\dots,n\} \text{ with } C_{(i-1)\cdot m + b - n\cdot m} = 1 \\ 0 \text{ otherwise} \end{cases}\\ &\text{for } b \in \{n\cdot m+1,\dots,n\cdot m+n\} \text{ with } b = nm + i.\end{aligned}$$

And the opening constraint as follows:

$$\begin{aligned}D_b = &\begin{cases} 1 \text{ if } C_{(i-1)\cdot m+j} \le C_{n\cdot m+i} \forall i \text{ for } i \in \{1,\dots,n\}, j \in \{1,\dots,m\} \\ 0 \text{ otherwise} \end{cases}\\ &\text{for } b \in \{1,\dots,n\cdot m\} \text{ with } b = (i-1)m + j.\end{aligned}$$

This shows that the FLP is a CMP with $A = nm + n$ and $B = nm + n$. There are therefore $nm + n$ binary variables and also $nm + n$ constraints. Now that the CMP has been defined, the question is what kind of solution we are particulary looking for. A feasible solution of CMP is defined as follows.

Definition 2: Feasible Solution

A bit string $\boldsymbol{z} \in Z(N)$ is a feasible solution to a CMP if the constraints are fulfils. The set of all feasible solutions is denoted as CMPsol.

If CMPsol $= \emptyset$ the CMP is infeasible. The solution we are looking for is an element from CMPsol, so that all bit stings from $Z(N) \backslash \text{CMPsol}$ do not need to be considered as optimal solutions.

Definition 3: Optimal Solution

A bit sting $\boldsymbol{z}^* \in \text{CMPsol}$ is a optimal solution, if it's fulfils:

$$\boldsymbol{z}^* = \underset{\boldsymbol{z} \in \text{CMPsol}}{\arg\min} \sum_{a=1}^{A} c_a C_a(\boldsymbol{z}).$$

The set of all optimal solutions is denoted as CMPopt.

3.2 Constraint Analysis

Alternative Constraints

The following section presents various alternative constraints for the FLP.
The assignment constraint is given by

$$\sum_{i=1}^{n} x_{i,j} = 1 \ \forall j \in \{1, \dots, m\}$$

and the opening constraint is given by:

$$x_{i,j} \leq y_i \ \forall i \in \{1, \dots, n\}, \ \forall j \in \{1, \dots, m\}.$$

We define two constraints as equivalent if exchanging these constraints does not change the minimum of the FLP.

Definition 4: Equivalent Constraints

Two constraints D and D' are called equivalent if exchanging the two constraints does not change the minimum of the objective function for every instance $\mathcal{I}$.

In the following, we always assume $f_i > 0$ and $c_{i,j} > 0$ for all customers and facilities. First consider the assignment constraint, which dictates that a customer is supplied by exactly one facility. However, another possibility would be that a customer is supplied by more than one facility.

Theorem 3: Equivalent Formulation for the Assignment Constraint

Let a instance of the FLP be given. The constraint

$$\sum_{i=1}^{n} x_{i,j} \geq 1 \ \forall j \in \{1, \dots, m\}$$

is equivalent to the assignment constraint.

Proof

Obviously, the set of feasible solutions with the assignment constraint $\mathcal{S}_{\text{sol}}$ is a subset of the set of feasible solutions of the new constraint $\mathcal{S}_{\text{new}}$. I.e. $\mathcal{S}_{\text{sol}} \subset \mathcal{S}_{\text{new}}$.
We have to show that the optimal solution of $\mathcal{S}_{\text{new}}$ is equal to the optimal solution of $\mathcal{S}_{\text{sol}}$. We proof by contradiction. Let us now assume that there is a optimal solution for which $\sum\limits_{i=1}^{n} x_{i,j} \geq 2$ holds for at least one $j \in \{1, \dots, m\}$. It must be shown that there is a smaller function value in $\mathcal{S}_{\text{new}}$ to demonstrate the assertion. Let $C_{opt}(\boldsymbol{z}') < C_{opt}(\boldsymbol{z})$ with $\boldsymbol{z}' \in \mathcal{S}_{\text{new}} \backslash \mathcal{S}_{eq}$ be given for all $\boldsymbol{z} \in \mathcal{S}_{\text{sol}}$. There are therefore $i, i' \in \{1, \dots, n\}$ with $i \neq i'$, so that $x'_{i,j} = x'_{i',j} = 1$. We now define $\boldsymbol{z}''$ as follows:

$$x''_{i,j} = \begin{cases} 0 \text{ if } \exists i_0 < i \text{ with } x'_{i_0,j} = 1 \\ x'_{i,j} \text{ otherwise} \end{cases}$$
$$y''_i = y'_i.$$

By construction, $\boldsymbol{z}'' \in \mathcal{S}_{\text{sol}}$ and $C(\boldsymbol{z}'') \leq C(\boldsymbol{z}')$ holds. This disproves the assertion.
□

The theorem shows that even if we allow a customer to be supplied by several facilities, it has no effect on the minimum. Therefore, in the best case, each customer is always supplied by exactly one facility.
The opening constraint was not changed in the previous considerations, but there are also alternatives for the opening constraint, as the following theorem shows:

Theorem 4: Equivalent Formulation for the Opening Constraint

Let a instance of the FLP be given. The constraint

$$y_i = \begin{cases} 0 \text{ if } \sum_{j=1}^{m} x_{i,j} = 0 \\ 1 \text{ otherwise} \end{cases} \quad \forall i \in \{1, \dots, n\}$$

is equivalent to the opening constraint.

Proof

First, all feasible solutions of our new constraint $\mathcal{S}_{\text{new}}$ also fulfil the opening constraint, so it holds

$$\mathcal{S}_{\text{new}} \subset \mathcal{S}_{\text{sol}}.$$

To show the assertion, we need to prove that the minimum of the objective function also satisfies the new constraint if we only require the opening constraint. We prove by contradiction. Let us now assume for a optimal solution holds $\boldsymbol{z} \in \mathcal{S}_{\text{sol}} \backslash \mathcal{S}_{\text{new}}$. Then there is an i_0 with $y_{i_0} = 1$ for which it holds that $x_{i_0,j} = 0 \ \forall j \in \{1, \dots, m\}$. We now define $\boldsymbol{z}'$ as follows:

$$y_i' = \begin{cases} 0 \text{ if } x_{i,j} = 0 \ \forall j \in \{1, \dots, m\} \\ y_i \text{ otherwise} \end{cases}$$
$$x_{i,j}' = x_{i,j}.$$

However, it is true that $C(\boldsymbol{z}') \leq C(\boldsymbol{z})$ applies and $\boldsymbol{z}' \in \boldsymbol{z} \in \mathcal{S}_{\text{sol}}$, so we have a contradiction. This shows the theorem. □

The theorem indicates that it is sufficient to check if a facility supplies any customer. If a facility does not supply any customers, it does not need to be open. However, the opening constraint does not effectively enforce this. By requiring that a facility can only be open if it supplies a customer, the number of feasible solutions is reduced, but the minimum does not change.

Number of Feasible Solutions

In total, we have n facilities and m customers in the FLP. Whether a customer j is supplied by a facility i is indicated by $x_{i,j}$. If this value is 1, a supply takes place; otherwise, it does not. This results in $n \cdot m$ parameters that can be 0 or 1. Additionally, for the opening of the facilities, $y_i = 1$ is valid only if the facility is open; otherwise, it is 0. Therefore, we have $nm + n$ parameters that can be either 0 or 1. This results in a total of 2^{nm+n} possible solutions. However, these do not necessarily fulfil one or even both constraints.
First consider the assignment constraint. There are a total of $2^n \cdot n^m$ possible states that fulfil it. The n^m holds because, for each customer j exactly one of the values x_{1j} to $x_{n,j}$ is 1 and the others are 0. There are therefore only n possible states for each customer. If we consider this for each of the m customers, there are n^m possible states. However, there is still the option of closing or opening facilities, for which there are another 2^n possibilities, so that there are ultimately $2^n \cdot n^m$ possible states that fulfil the assignment constraint. However, if the equivalent formulation of the opening constraint from Theorem 4 is added, there are exactly n^m possible states under which the minimum of the objective function is located. This is because the alternative constraint means that the value of y_i is automatically uniquely determined by the values of $x_{i,j}$.
There are a total of $(2^m + 1)^n$ states that fulfil the opening constraint. For an facility i there are two possibilities, either y_i has the value 0 or 1. If y_i has the value 1, the values of $x_{i,1}, \dots, x_{i,m}$ are irrelevant, since the opening constraint is always fulfilled. This means that there are exactly 2^m possible states for $y_i = 1$. If $y_i = 0$, there is only one possible state that fulfils the opening constraint, namely $x_{i,1} = \cdots = x_{i,m} = 0$ This results in $2^m + 1$, but since there are a total of n facilities, it follows that there are $(2^m + 1)^n$ possible states that fulfil the opening constraint.
The question now is how many possible states there are that fulfil the assignment constraint and the opening constraint. To answer this, we consider k opened facilities, there are a total of k^m possible assignments from the customers to the facilities. In addition, there are $\binom{n}{k}$ possibilities to select k facilities from n. Thus, there are $\binom{n}{k}k^m$ possible states with exactly k open facilities. In order to be able to represent all possible states, we sum over k and obtain

$$\sum_{k=1}^{n} \binom{n}{k} k^m$$

possible states that fulfil both constraints.
If we compare the number of states that fulfil the assignment and the opening constraints with those that fulfil the assignment constraint and the equivalent opening

constraint according to Theorem 4, we see that the number of states is reduced by

$$\sum_{k=1}^{n} \binom{n}{k} k^m - n^m = \sum_{k=1}^{n-1} \binom{n}{k} k^m + \binom{n}{n} n^m - n^m = \sum_{k=1}^{n-1} \binom{n}{k} k^m.$$

Thus, we have $\sum\limits_{k=1}^{n-1} \binom{n}{k} k^m$ states, which do not need to be considered if we use the equivalent opening constraint from Theorem 4 instead of the opening constraint. The following Table 3.1 can be used as an overview, which summarises the results of this section once again.

Constraints	Number of possibilities	Example for $n = 3, m = 4$
None	2^{nm+n}	32768
Only assignment	$2^n \cdot n^m$	648
Only opening	$(2^m + 1)^n$	4913
Assignment and opening	$\sum_{k=1}^{n} \binom{n}{k} k^m$	132
Assignment and Theorem 4	n^m	81

Table 3.1: Distribution of states for various constraints

The question is exists a equivalent constraints that reduce the number of possibilities even further than n^m.

Theorem 5: Minimum Number of Feasible Solutions

For the FLP there exists non constraint which is equivalent to the assignment or the opening constraint and makes the number of feasible states truly smaller than n^m.

Proof

Proof is provided by contradiction. Suppose there are constraints on the solution space with less than n^m elements. This means that there is a $\boldsymbol{z}'$ with $x_{i_1,1} = \cdots = x_{i_m,m} = y_{i_1} = \cdots = y_{i_m} = 1$, which is not in the set of feasible solutions $\boldsymbol{z}' \notin \mathcal{S}$. Now consider the following instance of the FLP $\mathcal{I}$.

$$c'_{i,j} = \begin{cases} 1 \text{ if } i = i_j \\ nm \text{ otherwise} \end{cases} \quad f'_i = \begin{cases} 1 \text{ if } i \in \{i_1, \dots i_m\} \\ n \text{ otherwise} \end{cases}.$$

By construction, $C(\boldsymbol{z}')$ is the minimum of the instance $\mathcal{I}$ if we use the assignment and opening constraint. However, $\boldsymbol{z}' \notin \mathcal{S}$. The minimum of the instance $\mathcal{I}$ is therefore not $\boldsymbol{z}'$. Let $\boldsymbol{z}$ be the minimum of the instance $\mathcal{I}$. By construction, $C(\boldsymbol{z}') < C(\boldsymbol{z})$ applies. This means that the constraints change the value of the minimum for the instance $\mathcal{I}$ and are therefore not equivalent. $\square$

The Theorem 5 shows that if we use the assignment constraint and the equivalent opening constraint from theorem 4, we have the smallest possible number of feasible solutions.
Thus we can show with simple reasoning that we have no polynomial growth. However, the question is whether there is a polynomial algorithm for the FLP at all. This will be discussed in the next section.

3.3 P and NP

The computing time required to solve a problem differs between different problems. The $\mathcal{O}$ notation is often used to represent the computing time, which is defined in [11, page 13] as follows.

Definition 5: Landau Notation

Let $g(n)$ be a function with $\lim_{n\to\infty} g(n) = \infty$. Then is $f \in \mathcal{O}(g)$ if and only if:

$$\limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty.$$

The function $g(n)$ describes the order of magnitude of computational complexity, specifically indicating how quickly the computing time increases. The primary distinction is whether the function is a polynomial or a faster-growing function. To define the complexity classes P and NP, it is essential to first understand the concepts of a formal language and a Turing machine, as these are used to describe the time required to solve a problem. A problem belongs to the complexity class P if it can be decided in polynomial time (cf. [12, page 3], [1, page 232]). More formally, a problem is decidable in polynomial time if a deterministic algorithm exists with a runtime of $\mathcal{O}(n^k)$, where $k \in \mathbb{N}$.
One of the biggest problems in computer science is whether P $=$ NP or P $\subsetneq$ NP applies. NP describes the set of problems for which a possible solution can be verified in polynomial time. An example of this is the calculation of the roots of polynomials with degree 5 or higher, for which no general formula can be specified. However, it is easy to check a possible solution by substituting the value into the polynomial. Formally, the term verifier must be introduced to define the complexity class NP (cf.

[13, page 232]). We use the following explanation: A problem is in the complexity class NP if a possible solution can be checked in polynomial time. A possible solution can be checked in polynomial time if a deterministic algorithm with a running time of $\mathcal{O}(n^k)$, where $k \in \mathbb{N}$, exists.
The complexity class P only requires the existence of an algorithm; the algorithm does not have to be known. There are many NP problems for which it is not clear if they also lie in P. Some problems are at least as complex as all others in NP, but these do not necessarily have to be in NP; these problems are called NP-hard.
Even if problems from NP-hard do not always lie in NP, there are also problems to which both holds; these problems are called NP-complete. A well-known NP-hard problem is the Set Cover Problem from Chapter 2.2. We will use this knowledge to show that the Facility Location Problem is also NP-hard. (cf. [13, page 238], [14, page 2])

Theorem 6: The FLP is NP-Hard

The Uncapacitated Facility Location Problem is NP-hard.

Proof

These proof is originates from [14, page 2].
We start with the Set Cover Problem and identify it with the FLP via: $\mathcal{U} \to \mathcal{D}$, $\mathcal{S} \to \mathcal{F}$. The costs can be defined by

$$c_{i,j} = \begin{cases} 1 \text{ if } j \in S_i \\ 3 \text{ otherwise} \end{cases} \qquad f_i = 1.$$

We take the total cost of the Set Cover Problem as

$$\sum_{S \in \mathcal{S}} c(S) x_S = C.$$

Since $f_i = 1$ the relation between the total cost of the Set Cover Problem and the total cost of the FLP is given by

$$\sum_{i \in \mathcal{S}} f_i + \sum_{j \in \mathcal{D}} c_{\sigma(j)j} = C + n.$$

Now let the costs for the Facility Location Problem be $C + n$. We must now make a case distinction.
1: Let the delivery costs be

$$\sum_{j \in \mathcal{D}} c_{\sigma(j)j} = C.$$

Then we can use

$$C = \sum_{j \in \mathcal{D}} c_{\sigma(j)j} = \sum_{S \in \mathcal{S}'} c(S).$$

Then the cost of the Set Cover Problem is given by C.
2: Let there be less than the n possible facilities open. Then the delivery costs are

$$\sum_{j \in \mathcal{D}} c_{\sigma(j)j} > C.$$

For each customer j with a delivery cost of 3, we open a facility $i \notin \mathcal{S} \subset \mathcal{F}$ with $c_{i,j} = 1$. We do this until the sum of the delivery costs is C. Then Case 1 can be applied again, and we obtain a Set Cover Problem with costs of C.
If the FLP for this particular instance were not NP-hard, the FLP could simply be solved and, with it, the Set Cover Problem. From this, it would follow that the Set Cover Problem would not be NP-hard. However, since the Set Cover Problem is NP-hard, the FLP is also NP-hard. $\square$
This theorem means that if P $\neq$ NP holds, there exists no exact polynomial algorithm for solving the Facility Location Problem. Nevertheless, there are polynomial approximation algorithms for the FLP, which are briefly described in the next section.

3.4 Approximations for the FLP

Before approximation algorithms are presented, we define what an approximation algorithm is. (cf. [15, page 1], [7, page 13])

Definition 6: k-Approximation Algorithm

Given are:

- a polynomial time algorithm $\mathcal{A}$ for an minimisation problem
- a solution $\mathcal{A}(\mathcal{I})$ for an instance $\mathcal{I}$
- the optimal solution $\mathrm{OPT}(\mathcal{I})$ of the instance $\mathcal{I}$.

Then $\mathcal{A}$ is a k-approximation algorithm if

$$\mathrm{OPT}(\mathcal{I}) \leq \mathcal{A}(\mathcal{I}) \leq k \cdot \mathrm{OPT}(\mathcal{I})$$

holds forall instances $\mathcal{I}$.

If we find a $(k = 1)$-approximation algorithm for the FLP, we automatically obtain the proof that $\mathrm{P} = \mathrm{NP}$ applies.

Greedy Algorithm

For the first approximation, we use the relationship between the Set Cover Problem and the UFLP as presented in [7, page 14]. A greedy algorithm for the Set Cover Problem works as follows: We always select the $S' \in \mathcal{S} \setminus \mathcal{S}'$ that contains the most elements from $U \setminus \bigcup\limits_{S \in \mathcal{S}'} S$ and add it to $\mathcal{S}'$. This process is repeated until $U = \bigcup\limits_{S \in \mathcal{S}'} S$ applies.

It can be represented as an algorithm as follows (cf. [16, page 2]):

Algorithm 1: Greedy Algorithm for Set Cover

```
$U = \mathcal{U}$
$S = \mathcal{S}$
$S' = \emptyset$
while($U \neq \emptyset$)do:
    select $s \in S$ such that it covers the most elements from $U$
    add $s$ to $S'$
    $U = U \setminus s$
return: $\mathcal{S}' := S'$
```

In [16] and [17, pages 233-235], it's shown that this is a $(1 + \log_2(m))$-approximation algorithm with $\mathcal{O}(mn^2)$, where $m = |\mathcal{U}|$ and $n = |\mathcal{S}|$. The greedy algorithm can be adapted to the FLP with the same approximation and run time. The greedy algorithm for the FLP is given by:

Algorithm 2: Greedy Algorithm for Facility Location

```
$D = \mathcal{D}$
$F = \mathcal{F}$
$F' = \emptyset$
while($D \neq \emptyset$) do:
    pick $(i, A)$ with minimised ratio $\frac{f_i + \sum_{j \in \mathcal{A}} c_{i,j}}{|\mathcal{A}|}$
    $F' = F' \cup i$
    $f_i = 0$
    $D = D \setminus A$
return: $\mathcal{S} := F'$
```

The Algorithm 2 works as follows. We start with all possible customers and calculate the effectiveness $\frac{f_i + \sum_{j \in \mathcal{A}} c_{i,j}}{|\mathcal{A}|}$ for a star $(i, \mathcal{A})$. In this case, a star is a pair consisting of a facility i and a subset of the as-yet unassigned customers $\mathcal{D}$. Once we have found the star with minimum effectiveness, we remove all customer $j \in \mathcal{A}$ from $\mathcal{D}$ and open the facility i. As the facility i is now open, the value for f_i is then set to 0. This process is repeated until all customers are assigned to a star.

Theorem 7: Greedy-Approximation Algorithm

Let $|\mathcal{D}| = m$.
Then the Algorithm 2 is an $(1 + \log_2(m))$-approximation algorithm.

Proof

For proof, please refer to the works [7, page 14], [18, page 1]. In particular, reference is made to the Algorithm 1 and the FLP is identified with the Set Cover Problem. □

Local Search

Another approach is the local search algorithm for the metric Facility Location Problem. Let a feasible solution be given by $\mathcal{S} \subset \mathcal{F}$ and $\sigma : \mathcal{D} \to \mathcal{S}$ be an assignment with costs of

$$\mathrm{cost}(\mathcal{S}) = \sum_{i \in \mathcal{S}} f_i + \sum_{j \in \mathcal{D}} c_{\sigma(j)j}.$$

The objective is now to reduce costs, for which a feasible solution is initially assumed. It is checked whether the costs can be reduced by adding or removing a facility. It can also be checked whether replacing the facility i with the facility i' will reduce costs. This is a local search, as only one facility can be added or removed at a time. If the algorithm can no longer be continued, there is a local minimum, which does not necessarily have to be the global minimum, i.e., the optimal solution.

Algorithm 3: Local Search for Facility Location

```
$F = \mathcal{F}$
$S \subset F$
while true do:
    If it exists $i \in F \backslash S$ s.t. $\mathrm{cost}(S \cup i) < \mathrm{cost}(S)$, then $S = S \cup i$
    If it exists $i \in S$ s.t. $\mathrm{cost}(S \backslash i) < \mathrm{cost}(S)$, then $S = S \backslash i$
    If it exists $i \in S$ and $i' \in F \backslash S$ s.t.
    $\mathrm{cost}((S \backslash i) \cup i') < \mathrm{cost}(S)$, then $S = (S \backslash i) \cup i'$
return: $\mathcal{S} := S$
```

The following theorem applies to the local search:

Theorem 8: Local Search-Approximation Algorithm

The Algorithm 3 is a 3-approximation algorithm for the MUCFLP.

Proof

The complete proof can be found in [19]. □

Currently there is the Jain-Mahdian-Saberi algorithm, a variation of the greedy algorithm, which is a 1.52-approximation algorithm for the MUCFLP, as shown by Mahdian, Ye and Zhang in 2002, see also: [7, page 28], [20, page 229-242].

4 Basics of Quantum Computing

Since there are no exact polynomial algorithms to solve the FLP, we want to use quantum algorithmic solution methods. To do this, we first need a basic knowledge of quantum computing. Unless otherwise stated, this chapter is based on [1, chapter 4].

4.1 Single Qubit Operations

Analogous to [10, page 5/6] and [21], we designate the single qubit system with $\mathbb{q}$. Obviously

$$\mathbb{q} \cong \mathbb{C}^2.$$

The tensor product of n qubits from $\mathbb{q}$ is given by

$$\bigotimes_{i=1}^{n} \mathbb{q} =: \mathbb{q}^{\otimes n}.$$

This vector space is isomorphic to $\mathbb{C}^{2^n}$. We denote the set of all n bit strings by $Z(n) := \{0,1\}^n$. With this we can define the computational basis.

Definition 7: Computational Basis

The computational basis of $\mathbb{q}^{\otimes n}$ is given by

$$\{|\boldsymbol{z}\rangle := |z_1 z_2 \dots z_n\rangle : z_i \in \{0,1\}\} = \{|\boldsymbol{z}\rangle : \boldsymbol{z} \in Z(n)\}.$$

In the next section we will look at linear operators on $\mathbb{q}^{\otimes n}$.

Unitary Operators

The norm of a linear operator T can be defined by the norm of the Hilbert space: (cf. [22, page 16/17])

$$\|T\| := \sup_{\|x\|=1} \|Tx\|.$$

In the following, the set of all linear operators on the Hilbert space $\mathcal{H}$ is denoted by $\mathcal{L}(\mathcal{H})$. The linear operators $A \in \mathcal{L}(\mathbb{C}^2)$ are described by 2×2 matrices. We want that if we apply a linear operator to a state vector, we get a state vector again. In particular, the norm of the vector must be preserved. We therefore need special types of operators, the unitary and hermitian operators. (cf. [22, page 87])

Definition 8: Types of Operators

A operator $T \in \mathcal{L}(\mathcal{H})$ is called

- unitary if T is invertible and it fulfilled $T^{-1} = T^{\dagger}$.
- hermitian if T fulfilled $T = T^{\dagger}$.

We denote the set of unitary operators on a Hilbert space $\mathcal{H}$ with $\mathcal{U}(\mathcal{H})$. If we apply a unitary operator to a state vector, we get a state vector again, as the following theorem shows.

Theorem 9: Unitary Operators are Norm Preserving

Let $\mathcal{H}$ be a Hilbert space, $|\varphi\rangle \in \mathcal{H}$ and $U \in \mathcal{U}(\mathcal{H})$ a unitary operator. Then

$$\|\varphi\| = \|U\varphi\|.$$

Proof

$$\|U\varphi\|^2 = \langle U\varphi|U\varphi\rangle = \langle\varphi|U^{\dagger}U\varphi\rangle = \langle\varphi|U^{-1}U\varphi\rangle = \langle\varphi|\varphi\rangle = \|\varphi\|^2.$$

$\square$

Unitary operators are therefore invariant with respect to the scalar product, another useful property concerns unitary operators that are applied to vectors from $\mathbb{C}^n$.

Theorem 10: Rows and Columns of a Unitary Operator Form an Orthonormal Basis

Let $U \in \mathcal{U}(\mathbb{C}^n)$ be unitary. Then the rows and columns of U form an orthonormal basis of $\mathbb{C}^n$.

Proof

From

$$\mathbb{1} = U^{-1}U = U^\dagger U$$

follows the statement. □

In [23] is an explanation for this theorem.

One of the most important properties of a linear operator are eigenvalues and eigenvectors which are defined as follows:

Definition 9: Eigenvalues and Eigenvectors

Let $T \in \mathcal{L}(\mathcal{H})$. If $|z\rangle \in \mathcal{H}$ fulfilled

$$T\,|z\rangle = t\,|z\rangle\,,\ t \in \mathbb{C}$$

then we call $|z\rangle$ eigenvector of T at eigenvalue t.

The eigenvector of an operator T is a vector $|z\rangle$ which only changes by a multiple when the operator is applied to it. The corresponding multiple t is then called an eigenvalue.

Pauli Matrices

The most important operators on $\mathbb{C}^{2\times 2}$ include the Pauli matrices, which are defined as follows: (cf. [24, page 166], [10, page 6])

Definition 10: Pauli Matrices

The Pauli matrices are define as the following matrices:

$$\sigma^0 = \mathbb{1}_{2\times 2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \sigma^1 = \sigma^x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
$$\sigma^2 = \sigma^y = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma^3 = \sigma^z = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The operators $\sigma^1, \sigma^2, \sigma^3$ are also the observables for the spin in x, y and z direction, by multiplying with $\frac{\hbar}{2}$. We can also represent the Pauli matrices by the basis vector of a qubit. The Pauli matrices have the following representation in the basis of one qubit.

$$\sigma^0 = |0\rangle\langle 0| + |1\rangle\langle 1| \qquad \sigma^1 = |1\rangle\langle 0| + |0\rangle\langle 1|$$
$$\sigma^2 = i \cdot (|1\rangle\langle 0| - |0\rangle\langle 1|) \qquad \sigma^3 = |0\rangle\langle 0| - |1\rangle\langle 1|.$$

The Pauli matrices have some useful properties which we summarise in the following theorem.

Theorem 11: Properties of the Pauli Matrices

1. The Pauli matrices are hermitian and unitary.
2. The Pauli matrices form a basis of $\mathbb{C}^{2\times 2}$.
3. The eigenvectors of σ^3 are the basis vectors of the single qubit system.

Proof

1. A calculation quickly shows that the Pauli matrices are hermitian and unitary.
2. We show that every matrix $A \in \mathbb{C}^{2\times 2}$ can be represented as a linear combination

of the Pauli matrices.

$$\begin{aligned}
A &= a\left|0\right\rangle\left\langle 0\right| + b\left|0\right\rangle\left\langle 1\right| + c\left|1\right\rangle\left\langle 0\right| + d\left|1\right\rangle\left\langle 1\right| \\
&= \frac{a+a+d-d}{2}\left|0\right\rangle\left\langle 0\right| + \frac{b+b+c-c}{2}\left|0\right\rangle\left\langle 1\right| \\
&+ \frac{b-b+c+c}{2}\left|1\right\rangle\left\langle 0\right| + \frac{a-a+d+d}{2}\left|1\right\rangle\left\langle 1\right| \\
&= \frac{a+d}{2}(\left|0\right\rangle\left\langle 0\right| + \left|1\right\rangle\left\langle 1\right|) + \frac{b+c}{2}(\left|1\right\rangle\left\langle 0\right| + \left|0\right\rangle\left\langle 1\right|) \\
&+ i^2\frac{b-c}{2}(\left|1\right\rangle\left\langle 0\right| - \left|0\right\rangle\left\langle 1\right|) + \frac{a-d}{2}(\left|0\right\rangle\left\langle 0\right| - \left|1\right\rangle\left\langle 1\right|) \\
&= \frac{a+d}{2}\sigma^0 + \frac{b+c}{2}\sigma^1 + i\frac{b-c}{2}\sigma^2 + \frac{a-d}{2}\sigma^3.
\end{aligned}$$

3. Proof is provided by recalculation:

$$\begin{aligned}
\sigma^3\left|0\right\rangle &= (\left|0\right\rangle\left\langle 0\right| - \left|1\right\rangle\left\langle 1\right|)\left|0\right\rangle = \left|0\right\rangle\left\langle 0|0\right\rangle - \left|1\right\rangle\left\langle 1|0\right\rangle = \left|0\right\rangle \\
\sigma^3\left|1\right\rangle &= (\left|0\right\rangle\left\langle 0\right| - \left|1\right\rangle\left\langle 1\right|)\left|1\right\rangle = \left|0\right\rangle\left\langle 0|1\right\rangle - \left|1\right\rangle\left\langle 1|1\right\rangle = -\left|1\right\rangle.
\end{aligned}$$

The eigenvalue for the eigenvector $\left|0\right\rangle$ is 1 and the eigenvalue for the eigenvector $\left|1\right\rangle$ is -1. □

Unitary Operators Represented by Rotation Operators

In this section, it is shown that any unitary operator can be decomposed into rotation operators. For this, the following property of the exponential function is needed: (cf. [1, page 175], [25])

Theorem 12: Euler-Formula for Unitary and Hermitian Operators

Let $A \in \mathbb{C}^{n\times n}$ be unitary and hermitian. Then the following applies:

$$\exp(iAx) = \cos(x)\mathbb{1} + i\sin(x)A.$$

Proof

For the matrix A^n we have:

$$A^n = \begin{cases} \mathbb{1} \text{ if } n = 2m, m \in \mathbb{N} \\ A \text{ if } n = 2m+1, m \in \mathbb{N} \end{cases}.$$

We now use the series representation of the matrix exponential:

$$\begin{aligned}\exp(iAx) &= \sum_{n=0}^{\infty} \frac{(iAx)^n}{n!} = \sum_{m=0}^{\infty} \frac{(iAx)^{2m}}{(2m)!} + \sum_{m=0}^{\infty} \frac{(iAx)^{2m+1}}{(2m+1)!} \\ &= \sum_{m=0}^{\infty} \frac{(-1)^m x^{2m}}{(2m)!} \mathbb{1} + i \sum_{m=0}^{\infty} \frac{(-1)^m x^{2m+1}}{(2m+1)!} A \\ &= \cos(x)\mathbb{1} + i\sin(x)A.\end{aligned}$$

□

In quantum mechanics, rotations around certain axes can be described by rotation operators.

Definition 11: Rotation Operators

These operators describe the rotation around the x, y and z axis.

$$\begin{aligned}R_x(\theta) &= \exp(-i\frac{\theta}{2}\sigma^x) \\ R_y(\theta) &= \exp(-i\frac{\theta}{2}\sigma^y) \\ R_z(\theta) &= \exp(-i\frac{\theta}{2}\sigma^z).\end{aligned}$$

We can use the Theorem 12 to reformulate the rotation operators. Thus, the rotation operator $R_j(\theta)$ can be decomposed into σ^0 and σ^j.

Theorem 13: Reformulation of the Rotation Operators

The rotation operators can rewritten in the following way:

$$\begin{aligned}R_x(\theta) &= \cos\left(\frac{\theta}{2}\right)\sigma^0 - i\sin\left(\frac{\theta}{2}\right)\sigma^x = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix} \\ R_y(\theta) &= \cos\left(\frac{\theta}{2}\right)\sigma^0 - i\sin\left(\frac{\theta}{2}\right)\sigma^y = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix} \\ R_z(\theta) &= \cos\left(\frac{\theta}{2}\right)\sigma^0 - i\sin\left(\frac{\theta}{2}\right)\sigma^z = \begin{pmatrix} \exp\left(-i\frac{\theta}{2}\right) & 0 \\ 0 & \exp\left(i\frac{\theta}{2}\right)\end{pmatrix}.\end{aligned}$$

Proof

The Pauli matrices are hermitian and unitary, so we know

$$\exp(-i\sigma^j x) = \cos(-x)\sigma^0 + i\sin(-x)\sigma^j \text{ for } j \in \{x, y, z\}.$$

Using the axis symmetry of cos and the point symmetry of sin

$$\cos(-x) = \cos(x), \quad \sin(-x) = -\sin(x),$$

we get the representations we are looking for. □

Our goal is now to establish a connection between unitary operators and rotation operators. The following theorem from [1, pages 175-176] is helpful for this.

Theorem 14: General Form of an Unitary Operator

Let $U \in \mathbb{C}^{2\times 2}$ be unitary then exists $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ such that

$$U = \begin{pmatrix} \exp(i(\alpha - \frac{\beta}{2} - \frac{\delta}{2}))\cos(\frac{\gamma}{2}) & -\exp(i(\alpha - \frac{\beta}{2} + \frac{\delta}{2}))\sin(\frac{\gamma}{2}) \\ \exp(i(\alpha + \frac{\beta}{2} - \frac{\delta}{2}))\sin(\frac{\gamma}{2}) & \exp(i(\alpha + \frac{\beta}{2} + \frac{\delta}{2}))\cos(\frac{\gamma}{2}) \end{pmatrix}.$$

Proof

This proof is also shown in [26]. Let U be unitary and represent as follows:

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Since the rows and columns form an ONB follows

$$|a|^2 + |b|^2 = 1.$$

For two complex numbers a, b that fulfil this equation, it follows that they can be represented as follows

$$a = \exp(i\alpha_{11})\cos(\theta), b = \exp(i\alpha_{12})\sin(\theta) \quad \alpha_{11}, \alpha_{12}, \theta \in \mathbb{R}.$$

Analogously, it follows from this

$$U = \begin{pmatrix} \exp(i\alpha_{11})\cos(\theta) & \exp(i\alpha_{12})\sin(\theta) \\ \exp(i\alpha_{21})\sin(\theta) & \exp(i\alpha_{22})\cos(\theta) \end{pmatrix}.$$

Now we have a representation with five parameters, from

$$0 = a\bar{b} + c\bar{d} = \exp(i\alpha_{11})\cos(\theta)\exp(-i\alpha_{12})\sin(\theta) + \exp(i\alpha_{21})\sin(\theta)\exp(-i\alpha_{22})\cos(\theta)$$

follows that

$$0 = \exp(i(\alpha_{11} - \alpha_{12})) + \exp(i(\alpha_{21} - \alpha_{22})).$$

The exponent must now fulfil the equation

$$\alpha_{11} - \alpha_{12} = \alpha_{21} - \alpha_{22} + \pi.$$

If we transform the equation to α_{11}, it follows that we can express this angle using the other angles

$$\alpha_{11} = \alpha_{12} + \alpha_{21} - \alpha_{22} + \pi.$$

Thus, we can represent a unitary operator by four operators. Now we define the angles $\alpha, \beta, \gamma, \delta$ from the theorem

$$\begin{aligned} \theta &= \frac{\gamma}{2} & \alpha_{12} &= \alpha - \frac{\beta}{2} + \frac{\delta}{2} + \pi \\ \alpha_{21} &= \alpha + \frac{\beta}{2} - \frac{\delta}{2} & \alpha_{22} &= \alpha + \frac{\beta}{2} + \frac{\delta}{2}. \end{aligned}$$

This implies for α_{11}

$$\begin{aligned} \alpha_{11} = \alpha_{12} + \alpha_{21} - \alpha_{22} + \pi &= \alpha - \frac{\beta}{2} + \frac{\delta}{2} + \pi + \alpha + \frac{\beta}{2} - \frac{\delta}{2} - \left(\alpha + \frac{\beta}{2} + \frac{\delta}{2}\right) + \pi \\ &= \alpha - \frac{\beta}{2} - \frac{\delta}{2} + 2\pi. \end{aligned}$$

If we use the following two properties of the complex exponential function

$$\exp(ix) = \exp(i(x + 2\pi)) \quad \exp(i(x + \pi)) = -\exp(ix)$$

we get the desired representation of a unitary operator. □

Now it can be shown that every unitary single qubit operator can be represented by rotation operators.

Theorem 15: $Z - Y$ Decomposition of a Unitary Operator

Let $U \in \mathbb{C}^{2\times 2}$ be unitary. Then their exists $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ such that

$$U = \exp(i\alpha) R_z(\beta) R_y(\gamma) R_z(\delta).$$

Proof

We calculate the right-hand side of the equation and show that it's equal to U:

$$\begin{aligned}
&\exp(i\alpha)R_z(\beta)R_y(\gamma)R_z(\delta)\\
&= \exp(i\alpha)\begin{pmatrix} \exp(-i\frac{\beta}{2}) & 0 \\ 0 & \exp(i\frac{\beta}{2}) \end{pmatrix}\begin{pmatrix} \cos(\frac{\gamma}{2}) & -\sin(\frac{\gamma}{2}) \\ \sin(\frac{\gamma}{2}) & \cos(\frac{\gamma}{2}) \end{pmatrix}\begin{pmatrix} \exp(-i\frac{\delta}{2}) & 0 \\ 0 & \exp(i\frac{\delta}{2}) \end{pmatrix}\\
&= \exp(i\alpha)\begin{pmatrix} \exp(-i\frac{\beta}{2})\cos(\frac{\gamma}{2}) & -\exp(-i\frac{\beta}{2})\sin(\frac{\gamma}{2}) \\ \exp(i\frac{\beta}{2})\sin(\frac{\gamma}{2}) & \exp(i\frac{\beta}{2})\cos(\frac{\gamma}{2}) \end{pmatrix}\begin{pmatrix} \exp(-i\frac{\delta}{2}) & 0 \\ 0 & \exp(i\frac{\delta}{2}) \end{pmatrix}\\
&= \begin{pmatrix} \exp(i(\alpha-\frac{\beta}{2}-\frac{\delta}{2}))\cos(\frac{\gamma}{2}) & -\exp(i(\alpha-\frac{\beta}{2}+\frac{\delta}{2}))\sin(\frac{\gamma}{2}) \\ \exp(i(\alpha+\frac{\beta}{2}-\frac{\delta}{2}))\sin(\frac{\gamma}{2}) & \exp(i(\alpha+\frac{\beta}{2}+\frac{\delta}{2}))\cos(\frac{\gamma}{2}) \end{pmatrix} = U.
\end{aligned}$$

The last equality follows from Theorem 14. □

Other Important Operators

Analogous to [1, page 174-177], we will now look at other important operators. The most important unitary operators includes the Hadamard gate H, the phase gate S and the $\pi/8$-gate T. These operators are defined by the following matrices

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad T = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\frac{\pi}{4}) \end{pmatrix}.$$

The Hadamard gate is obviously hermitian and unitary. If we apply H to the basis states of $\mathbb{q}$, we obtain a superposition of the two basis states, each with a probability of $\frac{1}{2}$.

$$H\left|0\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle + \left|1\right\rangle) \quad H\left|1\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle).$$

Since H is unitary and hermitian, it follows that $H^2 = \mathbb{1}_{2\times 2}$ and thus the basis states can also be obtained from a linear superposition by applying H again. H can be represent via the Pauli X and Z gates by:

$$H = \frac{1}{\sqrt{2}}(\sigma^1 + \sigma^3).$$

The relationship between the two operators S and T is:

$$S = T^2.$$

From Single-Qubit to Multi-Qubit Gates

Logically, classical computers require more than one bit for calculations, and we also need more than one qubit for a quantum computer. With a classical computer, however, the computing operations are generally irreversible. In a quantum computer, all operations are described as unitary operators. Since unitary operators are invertible by definition, operations on qubits are always reversible simply by applying $U^\dagger$ to the qubit. In the following we will introduce unitary operators that act on multiple qubits. For n qubits, the new quantum state can be described by the tensor product of n single qubit states

$$|\boldsymbol{z}\rangle = |z_1\rangle \otimes \cdots \otimes |z_n\rangle .$$

In the following, we abbreviate a single qubit operator A which only acts on the i-th qubit as

$$A_i = \mathbb{1} \otimes \ldots \mathbb{1} \otimes \underbrace{A}_{i-\text{th}} \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1}.$$

4.2 Controlled Operations

One of the first tasks a computer should perform is to add two numbers modulo 2. The CNOT gate is a unitary operator applied to two qubits. It maps two qubits $|z_1\rangle$ and $|z_2\rangle$ to $|z_1\rangle$ and $|z_1 \oplus z_2\rangle$, where $\oplus$ describes addition modulo 2. This means the following for the basic states:

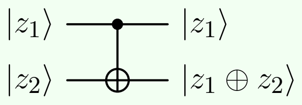
$\|z_1\rangle$	$\|z_2\rangle$	$\|z_1 \oplus z_2\rangle$
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.1: Application of the CNOT gates

The qubit $|z_1\rangle$ determines whether the other qubit is flipped or not. If $|z_1\rangle$ is 1, the value of the qubit $|z_2\rangle$ changes; if it is 0, it does not. We call $|z_1\rangle$ the control qubit and $|z_2\rangle$ the target qubit. The CNOT gate quantum circuit is defined as follows:

Definition 12: CNOT Gate

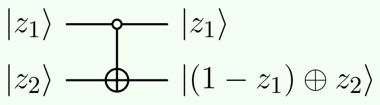
The control gate is defined as the following circuit.



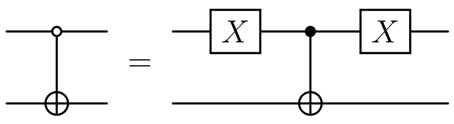
In future, the CNOT gate will be represented by CX. The CNOT gate changes the value of the target qubit if the control qubit is set to 1. However, if we want to change the target qubit when the control qubit is set to 0, we use the reverse control gate, which is defined as follows:

Definition 13: Reverse Control Gate

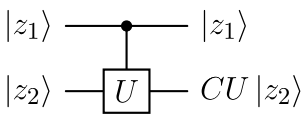
The reverse control gate is defined as the following circuit.



To indicate that we are using a reverse CNOT gate, we denoted it by $\overline{C}X$. The reverse control gate can be represented with one CNOT and two X gates. This is illustrated by the following circuit equation.



The first X gate transforms 0 into 1 and 1 into 0, after which the CNOT gate is applied. Consequently, a NOT gate is applied to the target qubit if the control qubit was initially 0. The second X gate is then applied to the control qubit to restore its initial state.
In some cases, controlled operations on a unitary operator U are also needed. This means that if the control qubit is set to 1, the operator U is applied to the target qubit. Conversely, if the control qubit is set to 0, no operator is applied to the target qubit. The quantum circuit for this operation looks like this:



Universal Quantum Gates

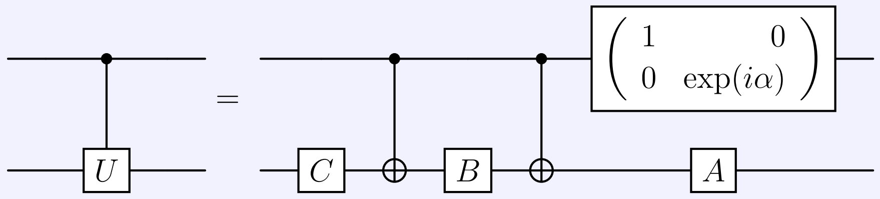
The controlled-U gate can also be described with other gates, for this we consider the equality of the following two circuits:

Theorem 16: Decomposition of the Controlled-U Gate

The controlled-U gate fulfils the identity

$$U = \exp(i\alpha) A\sigma^1 B\sigma^1 C \text{ with } ABC = \sigma^0.$$

Use this, we obtain the following circuit equivalence:



Proof

The proof is divided into two parts, first we check the equation for U and next the equality of the two circuits.
1. This part is a more detailed version of the proof of [1, page 176] and builds on the Theorem 15. We select A, B, C as

$$A = R_z(\beta)R_y\left(\frac{\gamma}{2}\right) \quad B = R_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\delta+\beta}{2}\right) \quad C = R_z\left(\frac{\delta-\beta}{2}\right).$$

First, it must be shown that $ABC = \mathbb{1}$ applies.

$$ABC = R_z(\beta)R_y\left(\frac{\gamma}{2}\right)R_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\delta+\beta}{2}\right)R_z\left(\frac{\delta-\beta}{2}\right) = \mathbb{1}.$$

The following properties of the rotation operators was used for the last equality:

$$R_i(\alpha)R_i(\alpha') = R_i(\alpha+\alpha'), \quad R_i(0) = \mathbb{1} \text{ for } i \in \{x,y,z\}.$$

We start with $\sigma^1 B\sigma^1$:

$$\begin{aligned}\sigma^1 B\sigma^1 &= \sigma^1 R_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\delta+\beta}{2}\right)\sigma^1 = \sigma^1 R_y\left(-\frac{\gamma}{2}\right)\sigma^1\sigma^1 R_z\left(-\frac{\delta+\beta}{2}\right)\sigma^1 \\ &= R_y\left(\frac{\gamma}{2}\right)R_z\left(\frac{\delta+\beta}{2}\right).\end{aligned}$$

The first equation was expanded with $(\sigma^1)^2 = \mathbb{1}$ and

$$\sigma^1 R_y(\theta)\sigma^1 = R_y(-\theta)$$

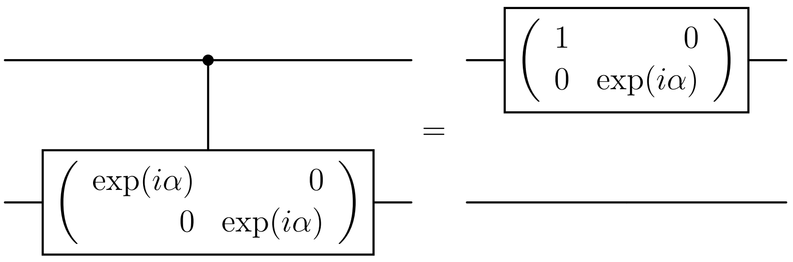
was used in the second equation. Finally:

$$\begin{aligned}\exp(i\alpha)A\sigma^1 B\sigma^1 C &= \exp(i\alpha)R_z(\beta)R_y\left(\frac{\gamma}{2}\right)R_y\left(\frac{\gamma}{2}\right)R_z\left(\frac{\delta+\beta}{2}\right)R_z\left(\frac{\delta-\beta}{2}\right)\\ &= \exp(i\alpha)R_z(\beta)R_y(\gamma)R_z(\delta) = U.\end{aligned}$$

The last equality follows from Theorem 15.
2. Now we show that the circuits match. The controlled-U operator has the following representation as a 2×2 matrix, where the elements of the matrix are again 2×2 matrices:

$$\begin{pmatrix} \mathbb{1} & 0 \\ 0 & U \end{pmatrix}.$$

We first use the following identity:


$\begin{pmatrix} \exp(i\alpha) & 0 \\ 0 & \exp(i\alpha) \end{pmatrix}$
=
$\begin{pmatrix} 1 & 0 \\ 0 & \exp(i\alpha) \end{pmatrix}$

If we use this equality to represent the right side of the circuit from the theorem by matrices, we get

$$\begin{aligned}&\begin{pmatrix} \mathbb{1} & 0 \\ 0 & \exp(i\alpha)\mathbb{1} \end{pmatrix}\begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \sigma^1 \end{pmatrix}\begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \sigma^1 \end{pmatrix}\begin{pmatrix} C & 0 \\ 0 & C \end{pmatrix}\\ &= \begin{pmatrix} \mathbb{1} & 0 \\ 0 & \exp(i\alpha)\mathbb{1} \end{pmatrix}\begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \sigma^1 \end{pmatrix}\begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix}\begin{pmatrix} C & 0 \\ 0 & \sigma^1 C \end{pmatrix}\\ &= \cdots = \begin{pmatrix} ABC & 0 \\ 0 & \exp(i\alpha)A\sigma^1 B\sigma^1 C \end{pmatrix} = \begin{pmatrix} \mathbb{1} & 0 \\ 0 & U \end{pmatrix}.\end{aligned}$$

The equality of the two circuits is shown simply by using matrix multiplication and the identity from 1. □

We see that the controlled-U gate can be decomposed into CNOT gates and single qubit operators. But what about unitary multiple qubit operators in general? Can they also be reduced to a certain set of gates? Thus, we are looking for a universal set of quantum gates that can represent all other unitary operators by combining them. First, the universal quantum gate must be defined. (cf. [27, page 2])

Definition 14: Universal Quantum Gate

$\mathcal{S} \subset \bigcup_n \mathcal{U}(\mathbb{q}^n)$ is called a universal quantum gate if for all $U \in \bigcup_n \mathcal{U}(\mathbb{q}^n)$ it holds that for $\epsilon \geq 0$ exists a subset $\{U_1, \ldots, U_m\} \subset \mathcal{S}$ such that

$$\|U - U_m \ldots U_1\| \leq \epsilon$$

is satisfied.

If $\epsilon = 0$, the universal quantum gate is called exact; otherwise, it is called approximate. There are different sets of universal quantum gates. An exact universal quantum gate is formed by all 1 and 2 qubit gates, but it still comprises an uncountable number of gates. Two possible universal quantum gates are:

- $\{\text{CNOT}, H, T\}$
- $\{\text{CNOT}, R_y(\frac{\pi}{4}), S\}$

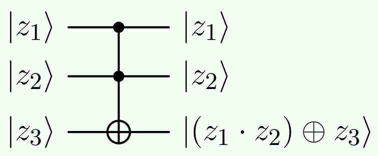
The proofs of universality are too extensive for this thesis, so please refer to [28].

Toffoli Gate

Now we use [2] to show some really important properties about multi-qubit controlled operations. We first expand the CNOT gate from one control qubit to two control qubits and obtain the Toffoli gate.

Definition 15: Toffoli Gate

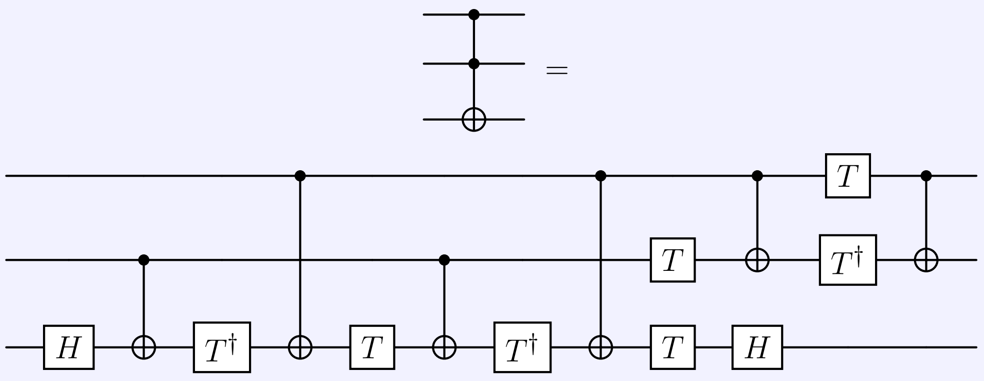
The Toffoli gate is defined as the following quantum circuit:



If both z_1 and z_2 are 1, a σ^x gate is applied to $|z_3\rangle$; otherwise, nothing happens. We now need to demonstrate the implementation of the Toffoli gate. This can be achieved using CNOT, H, and T gates, as shown by the following theorem (cf. [1, page 182]).

Theorem 17: Decomposition of the Toffoli Gate

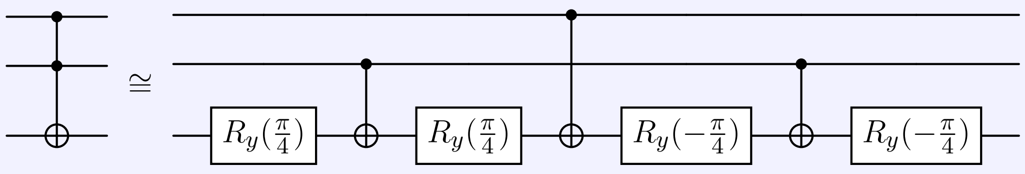
The Toffoli Gate fulfils the following circuit equality:



In the following, we use the representation from [2]. Even if the two quantum circuits only match up to a global phase, this has no subsequent consequences for the measurement of the states at the end of the quantum circuits. The probabilities match for any initial state, and we obtain the same probability for both output states. We denote this type of equality with $\cong$.

Theorem 18: R_y Decomposition of the Toffoli Gate

The Toffoli Gate fulfils the following circuit equality except for a global phase:



Proof

We examine four different scenarios for the four possible states that the first two qubits can have together.
1. Initial state: $|z00\rangle$:

$$R_y\left(\frac{-\pi}{4}\right) R_y\left(\frac{-\pi}{4}\right) R_y\left(\frac{\pi}{4}\right) R_y\left(\frac{\pi}{4}\right) = R_y\left(\frac{-\pi}{2}\right) R_y\left(\frac{\pi}{2}\right) = \mathbb{1}.$$

2. Initial state: $|z01\rangle$

$$R_y\left(\frac{-\pi}{4}\right) R_y\left(\frac{-\pi}{4}\right) \sigma^1 R_y\left(\frac{\pi}{4}\right) R_y\left(\frac{\pi}{4}\right) = R_y\left(\frac{-\pi}{2}\right) \sigma^1 R_y\left(\frac{\pi}{2}\right) = Z.$$

3. Initial state: $|z10\rangle$

$$R_y\left(\frac{-\pi}{4}\right) \sigma^1 R_y\left(\frac{-\pi}{4}\right) R_y\left(\frac{\pi}{4}\right) \sigma^1 R_y\left(\frac{\pi}{4}\right) = \mathbb{1}.$$

4. Initial state: $|z11\rangle$

$$R_y\left(\frac{-\pi}{4}\right) \sigma^1 R_y\left(\frac{-\pi}{4}\right) \sigma^1 R_y\left(\frac{\pi}{4}\right) \sigma^1 R_y\left(\frac{\pi}{4}\right) = X.$$

This means that the two circuits match except for one global phase. □

The question now is how to manage more than two control qubits. For this purpose, additional qubits (ancilla qubits) are required. If we have n control qubits, with $n-2$ ancilla qubits, the problem can be reduced using $2n-3$ Toffoli gates. Let's examine the following algorithm.

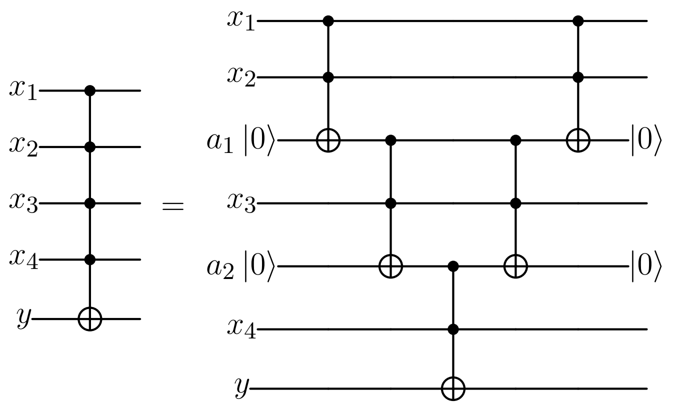
Algorithm 4: Construction of an n-Controlled NOT gate

Given are:

- label n control qubits x_i with $i \in \{1, \dots, n\}$
- target qubits y
- label $n-2$ ancilla qubits a_i with $i \in \{1, \dots, n-2\}$ set it to $|0\rangle$

```
if n=1 do
    apply CX with $x_1$ as control and $y$ as target
if n=2 do
    apply Toffoli with $x_1$ and $x_2$ as control and $y$ as target
else do
    apply Toffoli with $x_1$ and $x_2$ as control and $a_1$ as target
    for $i = 3, \dots, n-1$ do
        apply Toffoli with $a_{i-2}$ and $x_3$ as control and $a_{i-1}$ as target
    apply Toffoli with $a_{n-2}$ and $x_n$ as control and $y$ as target
    for $i = n-1, \dots, 3$ do
        apply Toffoli with $a_{i-2}$ and $x_3$ as control and $a_{i-1}$ as target
    apply Toffoli with $x_1$ and $x_2$ as control and $a_1$ as target
```

It must still be demonstrated that the Algorithm 4 indeed constructs a circuit that produces an n-controlled NOT gate. For $n = 4$, the corresponding circuit looks as follows:



We start by applying a Toffoli gate to the first two qubits x_1 and x_2. If both are set to 1, the first ancilla qubit is set to 1. This operation records whether both qubits are 1 or not. Next, another Toffoli gate is applied to the next ancilla qubit a_2, which stores whether a_1 and x_3 are both 1. Then a_2 and x_4 are checked to see if both are 1. The first ancilla qubit thus checks whether x_1 and x_2 are set to 1, and the i-th ancilla qubit always checks whether the $(i-1)$-th ancilla qubit and x_{i+1} are set to 1. In this way, an n-control NOT gate can be decomposed into $2n-3$ Toffoli gates. The last $n-2$ gates are used only to reset the ancilla qubits back to 0 if necessary.

Theorem 19: Construction of an n-Controlled NOT gate

The Algorithm 4 constructed a quantum circuit which is equal to the n-controlled NOT gate.

Proof

For $n = 1{,}2$ we showed it before. a_1 becomes 1 if and only if x_1 and x_2 are 1; otherwise, a_1 is 0. Now, for $i = 2, \ldots, n-2$, the value of a_i becomes 1 if and only if a_{i-1} and x_{i+1} are 1. Thus, to change a_i to 1, we need $x_1 = \cdots = x_{i+1} = 1$. If we consider a_{n-2}, it is 1 if and only if $x_1 = \cdots = x_{n-1} = 1$. Finally, we look at a_{n-2} and x_n. If both are 1, we apply the σ^1 gate to y. Thus, we only apply σ^1 on y if and only if $x_1 = \cdots = x_n = 1$. Except for the last Toffoli gate, we reapply the previous Toffoli gates in reverse order to reset the ancilla qubits back to 0, which has no effect on the other qubits. $\square$

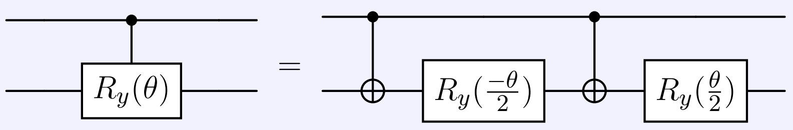
So far, we have abbreviated the CNOT gate as CX. We will now abbreviate the Toffoli gate as CCX or C^2X. The n-CNOT gate is abbreviated as C^nX. If the number of qubits is greater than $n+1$, we label each C with the corresponding controlled qubits as follows: $C_{i_1}\dots C_{i_n}X$.

Special case: "Controlled-R_y-Gate"

We now use Theorem 16 to represent the R_y gate with single qubit and CNOT gates.

Theorem 20: Controlled R_y Gate

The Controlled-R_y gate fulfil the following circuit equality:



Proof

We select α, A, B, C as follows:

$$A = R_y\left(\frac{\theta}{2}\right),\ B = R_y\left(\frac{-\theta}{2}\right),\ C = \mathbb{1},\ \alpha = 0.$$

We now check the identities from Theorem 16.

$$\begin{aligned} ABC &= R_y\left(\frac{-\theta}{2}\right) R_y\left(\frac{\theta}{2}\right) = \sigma^0 \\ \exp(i\alpha)A\sigma^1 B\sigma^1 C &= R_y\left(\frac{\theta}{2}\right)\sigma^1 R_y\left(\frac{-\theta}{2}\right)\sigma^1 = R_y(\theta). \end{aligned}$$

If we now use Theorem 16 it follows that the circuits are equal. □

Now that the n-CNOT gate has been discussed, the question is what it looks like for an n-controlled U gate. To address this, the circuit is decomposed step by step into smaller $(n-1)$-controlled V gates, as the next algorithm shows. (c.f. [2, page 21])

Algorithm 5: Circuit for an n-Controlled U Gate

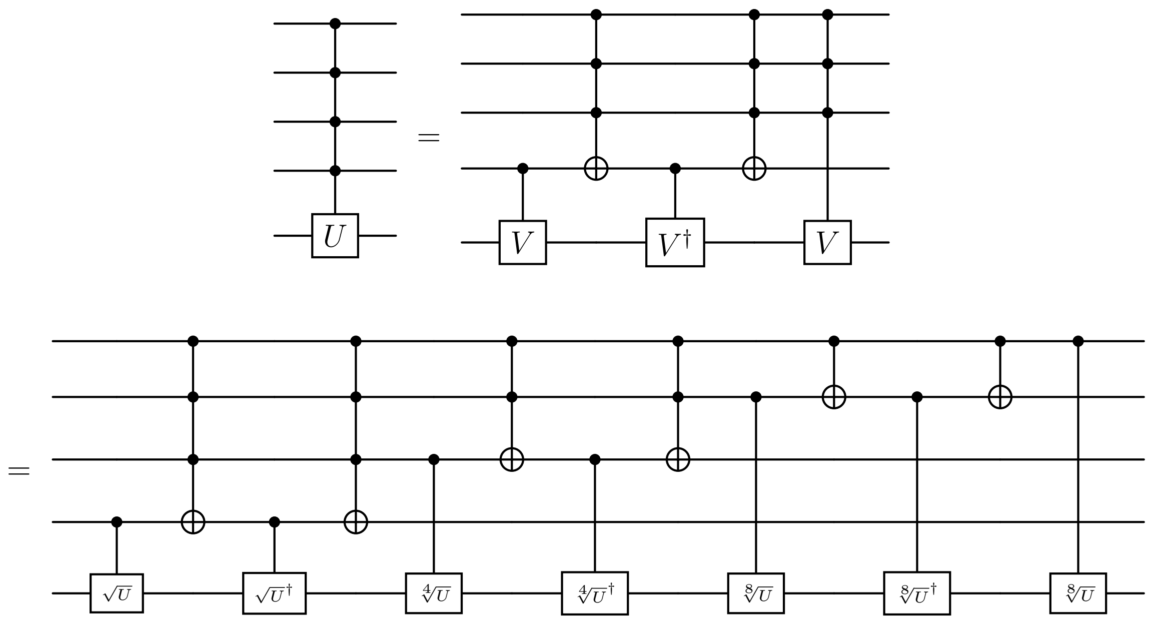
Given are:

- n controlled qubits labeled as x_i
- one target qubit y

```
apply $C_{x_n}V$ on $y$
apply $C_{x_1}\dots C_{x_{n-1}}X$ on $x_n$
apply $C_{x_n}V^\dagger$ on $y$
apply $C_{x_1}\dots C_{x_{n-1}}X$ on $x_n$
apply $C_{x_1}\dots C_{x_{n-1}}V$ on $x_n$
```

The Algorithm 5 reduces the problem of constructing an n-controlled U gate to constructing an $(n-1)$-controlled V gate, where $V^2 = U$. An n-controlled U gate can thus be reduced step by step to $C\sqrt{U}, C\sqrt[4]{U}, \dots, C\sqrt[2^{n-1}]{U}$ gates. This reduction can be formalized using Theorem 16.

For $n = 4$, the circuit looks as follows:



As can be seen after the first equal sign, we replace a C^4U gate with two C^3X gates, a CV, a $CV^\dagger$, and a C^3V gate. If the algorithm is applied to the C^3V gate in the same way, the circuit follows after the second equal sign. Here, $\sqrt[n]{U}$ is the operator V for which $V^n = U$ holds. The next step is to show that Algorithm 5 works.

Theorem 21: Construction of an n-Controlled U Gate

Let $U \in \mathcal{U}(\mathbb{q})$. The Algorithm 5 creates a n-controlled U gate, if $V^2 = U$ and $n \geq 2$.

Proof

We first consider the case where all control qubits are set to 1. It follows:

$$V \cdot V = U.$$

Now we look at the case where there is a $j \in \{1, \dots, n\}$ with $x_j = 0$. If $x_n = 0$ there are two possibilities.
1. $x_i = 1 \forall i \in \{1, \dots, n-1\}$ Then it follows:

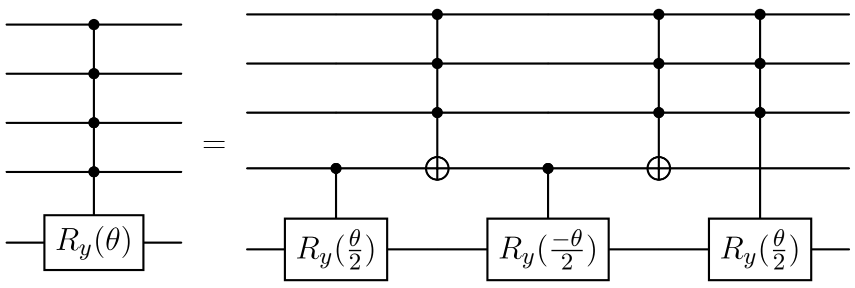
$$V \cdot V^\dagger = \mathbb{1}.$$

2. $\exists x_i$ with $x_i = 0,\ i \in \{1, \dots, n-1\}$ No operator is executed.
In the case where there is a $j \in \{1, \dots, n\}$ with $x_j = 0$, but $x_n = 1$ it follows:

$$V^\dagger V = \mathbb{1}.$$

The circuit thus constructs an n-controlled U gate. □

If we now consider the special case $U = R_y(\theta)$, then V can be determined very simply by halving the angle, because $R_y(a)R_y(b) = R_y(a+b)$. As a circuit, it looks as follows for $n = 4$:



The n-controlled R_y gates can be implemented easily since the root operation is straightforward to define. To implement it, one simply needs to halve the angle. The decomposition involves CR_y, Toffoli, and CNOT gates, which allows for a more precise estimation of the computational effort required.

Theorem 22: Decomposition of the C^nR_y Gate

The C^nR_y gate for $n \geq 2$ can be created using the Algorithm 5, whereby the following gates are used:

- $2n-1$ CR_y gates
- $2(n-2)^2$ Toffoli gates
- 2 CNOT gates

Proof

The Algorithm 5 splits the C^nR_y gate into two CR_y, two $C^{n-1}X$ and one $C^{n-1}R_y$ gate. If we do this iterativly from n to 2 , we get $2(n-2)$ CR_y gates, $2(\sum_{i=2}^{n-1}(2i-3))$ Toffoli gates and one C^2R_y gate. If we now decompose the C^2R_y gate, we get three CR_y gates and two CNOT gates. If we add up the CR_y gates, the $2n-1$ CR_y gates are shown. The number of Toffoli gates must still be shown.

$$2\left(\sum_{i=2}^{n-1}(2i-3)\right) = 2\left(\sum_{i=1}^{n-1}(2i-1)-2n+3\right)$$
$$= 2((n-1)^2-2n+3) = 2(n^2-4n+4) = 2(n-2)^2.$$

The following identity was used in the calculation $\sum_{i=1}^{n-1}(2i-1) = (n-1)^2$. □

Expected Value of an Operator

Definition 16: Expected value

Let $\mathcal{H}$ be a Hilbert space, $H \in \mathcal{L}(\mathcal{H})$ and $|\varphi\rangle \in \mathcal{H}$. The expected value from H related to $|\varphi\rangle$ is defined by

$$\langle H\rangle := \langle\varphi| H |\varphi\rangle .$$

One of the most useful properties of an operator are its eigenvalues and eigenvectors, though often only the extreme values are of interest. In our case, we seek the smallest eigenvalue E_g and the corresponding ground state $|\varphi_g\rangle$. The connection between the expected value and the ground state is that the expected value of an operator is never smaller than its smallest eigenvalue.

Theorem 23: Relationship Between Expected Value and the Smallest Eigenvalue

Let $\mathcal{H}$ be a Hilbert space, $H \in \mathcal{L}(\mathcal{H})$, $\{|\varphi_i\rangle, i \in I\}$ be the set of eigenvectors and $\{E_i, i \in I\}$ the set of eigenvalues of H. Then it holds that

$$\langle\varphi| H |\varphi\rangle \geq \langle\varphi_g| H |\varphi_g\rangle = E_g.$$

Proof

We decompose $|\varphi\rangle$ with respect to the basis of the eigenvectors of H:

$$|\varphi\rangle = \sum_{i\in I} a_i |\varphi_i\rangle .$$

Now we show the assertion by recalculating.

$$\langle\varphi| H |\varphi\rangle = \sum_{i\in I} |a_i|^2 \langle\varphi_i| H |\varphi_i\rangle = \sum_{i\in I} |a_i|^2 \underbrace{E_i}_{\geq E_g} \underbrace{\langle\varphi_i|\varphi_i\rangle}_{=1} \geq E_g \underbrace{\sum_{i\in I} |a_i|^2}_{=1} = E_g.$$

□

It is therefore obvious that the minimum expected value corresponds to the smallest eigenvalue

$$E_g = \min_{|\varphi\rangle\in\mathcal{H}} \langle\varphi| H |\varphi\rangle .$$

5 Quantum Approximation Algorithms

5.1 Quantisation of the FLP

Unconstrained FLP

The next step is to quantise the objective function of the FLP. For this we use the formulation of the FLP in Problem 3. This chapter is based on the work of [4], in particular on the explanation on page 3-8. First, we reformulate the FLP so that constraints are no longer necessary. Instead, a penalty is introduced for each constraint. If a constraint is not fulfilled, the costs are increased, for this an additional terms can be added. Now the Unconstrained Facility Location Problem can be defined. (cf. [4, page 6/7])

Problem 8: Unconstrained FLP (UFLP)

Given are:

- $y_i = \begin{cases} 1 \text{ if facility } i \text{ is opened} \\ 0 \text{ otherwise} \end{cases}$ for $i \in \{1, \dots, n\}$
- $x_{i,j} = \begin{cases} 1 \text{ if facility } i \text{ serves customer } j \\ 0 \text{ otherwise} \end{cases}$ for $i \in \{1, \dots, n\}, j \in \{1, \dots, m\}$
- $f_i, c_{i,j} \in \mathbb{R}_+$.

The Unconstrained Facility Location Problem (UFLP) is the following problem

$$\min_{x,y} \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} x_{i,j} + \lambda \sum_{j=1}^{m} \left(\left(1 - \sum_{i=1}^{n} x_{i,j} \right)^2 + \sum_{i=1}^{n} (x_{i,j} - x_{i,j} y_i) \right).$$

The following theorem demonstrates the relationship between UFLP and FLP.

Theorem 24: FLP with Penalty

Let the FLP formulated like Problem 3. Then there exists a $\lambda \in \mathbb{R}_+$ so that the optimal solution of the FLP and the UFLP (Problem 8) agree.

Proof

Let $\boldsymbol{z}$ be a optimal solution of the FLP. The question now is whether there is a state $\boldsymbol{z}'$ for the UFLP that does not fulfil a constraint of the FLP and $C(\boldsymbol{z}')$ corresponds to the minimum. For this, λ must be chosen so that there is no optimal solution $\text{CMP}_{\text{sol}} \not\ni (x', y') \in \{0,1\}^{nm+n}$ of the UFLP with

$$\begin{aligned}\sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}x_{i,j} > \sum_{i=1}^{n} f_i y_i' + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}x_{i,j}' \\ + \lambda \sum_{j=1}^{m}\left(\left(1-\sum_{i=1}^{n} x_{i,j}'\right)^2 + \sum_{i=1}^{n}(x_{i,j}' - x_{i,j}'y_i')\right).\end{aligned}$$

whereby $(x, y) \in \text{CMP}_{\text{sol}}$. We now select λ as follows:

$$\lambda = \sum_{i=1}^{n} f_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}.$$

If $\sum\limits_{j=1}^{m}\left(\left(1-\sum\limits_{i=1}^{n} x_{i,j}'\right)^2 + \sum\limits_{i=1}^{n}(x_{i,j}' - x_{i,j}'y_i')\right) = 0$, the optimal solution of the FLP with penalty is equal to that of the FLP.

Now we assume $\sum\limits_{j=1}^{m}\left(\left(1-\sum\limits_{i=1}^{n} x_{i,j}'\right)^2 + \sum\limits_{i=1}^{n}(x_{i,j}' - x_{i,j}'y_i')\right) > 0$,

since $x_{i,j}', y_i' \in \{0,1\}$ it follows $\sum\limits_{j=1}^{m}\left(\left(1-\sum\limits_{i=1}^{n} x_{i,j}'\right)^2 + \sum\limits_{i=1}^{n}(x_{i,j}' - x_{i,j}'y_i')\right) \geq 1$.

The statement can be demonstrated with the following calculation.

$$\begin{aligned}
&\sum_{i=1}^{n} f_i y_i' + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} x_{i,j}' + \lambda \underbrace{\sum_{j=1}^{m}\left(\left(1-\sum_{i=1}^{n} x_{i,j}'\right)^2 + \sum_{i=1}^{n}(x_{i,j}' - x_{i,j}' y_i')\right)}_{\geq 1} \\
&\geq \sum_{i=1}^{n} f_i y_i' + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} x_{i,j}' + \lambda \geq \lambda = \sum_{i=1}^{n} f_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} \\
&\geq \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} x_{i,j}.
\end{aligned}$$

The optimal solutions of the FLP therefore corresponds to the UFLP. $\square$

If we attach a penalty only for one of the two constraints, we get the following objective functions:

$$\begin{aligned}
C_{\text{TV}} &= \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} x_{i,j} + \lambda \sum_{j=1}^{m}\left(1-\sum_{i=1}^{n} x_{i,j}\right)^2 \\
C_{\text{SWAP}} &= \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} x_{i,j} + \lambda \sum_{j=1}^{m}\left(\sum_{i=1}^{n}(x_{i,j} - x_{i,j} y_i)\right).
\end{aligned}$$

C_{TV} only has a penalty for the assignment constraint and C_{SWAP} has a penalty for the opening constraint. The naming will become clear in the course of this chapter. From now on we always choose $\lambda = \sum_{i=1}^{n} f_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}$.

Quantisation of the Objective Function

In combinatorial optimisation problems, the goal is to minimise the objective function, which must first be quantised. For this, the objective function C is considered as a Hamiltonian in the computational basis $\mathrm{q}^{\otimes n}$. We define the effect of C on $|\boldsymbol{z}\rangle$ as follows: (cf. [10, page 10])

Definition 17: Objective Hamiltonian

The objective Hamiltonian is defined by the objective function as

$$C\,|\boldsymbol{z}\rangle := C(\boldsymbol{z})\,|\boldsymbol{z}\rangle\,,\ \boldsymbol{z} \in Z(N).$$

By definition, C is an orthogonal operator in the computational basis. Simply calculating the ground state of the objective Hamiltonian is not sufficient to solve the CMP. It must also be ensured that the solution is located within $\mathrm{CMP}_{\mathrm{sol}}$. Therefore, we define the solution space S. (cf. [10, page 10])

Definition 18: Solution Space

The solution space S is defined as

$$S := \mathrm{span}\{|\boldsymbol{z}\rangle : \ \boldsymbol{z} \in \mathrm{CMP}_{\mathrm{sol}}\} \subset q^{\otimes n}.$$

We define analogously: $S_{\min} := \mathrm{span}\{|\boldsymbol{z}\rangle : \boldsymbol{z} \in \mathrm{CMP}_{\mathrm{opt}}\} \subset S$. The task is to find the ground state of $C|_S$. The vector $|\boldsymbol{z}\rangle$ is the ground state of $C|_S$ if and only if $\boldsymbol{z}$ is an optimal solution.
Assuming $|\boldsymbol{z}\rangle$ is the ground state, then $C(\boldsymbol{z})$ is the smallest eigenvalue of $C|_S$, which by definition corresponds to the smallest function value of the objective function $C(\boldsymbol{z})|_{\boldsymbol{z}\in\mathrm{CMP}_{\mathrm{sol}}}$. This represents an optimal solution. Conversely, if $\boldsymbol{z}$ is an optimal solution, then $|\boldsymbol{z}\rangle$ is the eigenvector of the ground state, since $C(\boldsymbol{z})$ is the smallest possible function value.

Hamiltonian for the FLP

The next step is to quantise the two binary variables $x_{i,j}, y_i$, for this we are looking for a unitary operator with the eigenvalue 0 to the eigenvector $|0\rangle$ and the eigenvalue 1 to the eigenvector $|1\rangle$, for this we consider (cf.[29, page 4])

$$\hat{x} := \frac{\sigma^0 - \sigma^3}{2} = |1\rangle\langle 1| .$$

Theorem 25: Eigenvalues and Eigenvectors of $\hat{x}$

The operator $\hat{x}$ has the eigenvector $|0\rangle$ with eigenvalue 0 and the eigenvectors $|1\rangle$ with eigenvalue 1.

Proof

$$\hat{x}|0\rangle = |1\rangle\langle 1|0\rangle = 0\,|1\rangle = 0\,|0\rangle \qquad \hat{x}|1\rangle = |1\rangle\langle 1|1\rangle = |1\rangle = 1\,|1\rangle .$$

$\square$

Let $a \in \{1, \ldots, nm+n\}$, the operator $\hat{x}$ can be generalised for the qubit a

$$\hat{x}_a := \frac{\sigma_a^0 - \sigma_a^3}{2}.$$

Now $x_{i,j}, y_i$ can be quantised in the following way:

$$x_{i,j} \mapsto \hat{x}_{(i-1)\cdot m+j} \qquad y_i \mapsto \hat{x}_{nm+i}.$$

The indices are oriented to the Chapter 3.1 and can thus use the quantised version of the FLP. The objective Hamiltonian is

$$H_{\text{CFLP}} = \sum_{i=1}^{n} f_i \hat{x}_{nm+i} + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} \hat{x}_{(i-1)\cdot m+j}.$$

This Hamiltonian can also be specified for the unconstrained FLP as

$$H_{\text{UFLP}} = H_{\text{CFLP}} + \lambda \sum_{j=1}^{m} \left(\left(\mathbb{1} - \sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j} \right)^2 + \sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j} (\mathbb{1} - \hat{x}_{nm+i}) \right).$$

H_{CFLP} describes the Facility Location Problem with constraints, the C in CFLP emphasises this. The costs for the open facilities are added together with the delivery costs. For H_{UFLP} the constraints are then added as penalties.
The term $\left(\mathbb{1} - \sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j} \right)^2$ is different to the zero matrix if not exactly one facility supplies the customer. The term $\sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j}(\mathbb{1} - \hat{x}_{nm+i})$ penalises if customer j is supplied by a not open facility i. These two terms are added together for each individual customer. The pre-factor λ ensures that the ground states of H_{CFLP} and H_{UFLP} match.

5.2 Quantum Approximate Optimisation Algorithm

There are generally two methods to deal with constraints. First, consider the objective Hamiltonian H_{CFLP} for the FLP. It is crucial not only to find the minimum itself but also to ensure it is within the solution space. To account for the constraints, specific mixers are required for each problem.
Hardcoding is a method where a special quantum circuit ensures that only states fulfilling the constraints are generated. More on this can be found in Chapter 5.3. Alternatively, there is softcoding, where constraints are incorporated into the Hamiltonian as additional terms in the form of penalties.
In Chapter 5.1, an unconstrained FLP was introduced with H_{UFLP}. This chapter presents the Quantum Approximate Optimisation Algorithm (QAOA), which can be used to approximate the minimum of the FLP using softcoding. This section is guided by the following papers: [10, chapter 3] and [3].

QAOA is a variational quantum algorithm, which combines quantum computing with classical optimisation. It involves a natural number $p \geq 1$, which determines the number of angles to optimise. In our case, $2p$ angles are optimised to approximate the minimum of the FLP. In QAOA we first apply a phase separator $U_P(H, \gamma)$ to the initial state, followed by a mixer $U_M(B, \beta)$. The initial state of QAOA is the plus state

$$|+\rangle := \bigotimes_{i=1}^{N} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2^N}} \sum_{\boldsymbol{z} \in Z(N)} |\boldsymbol{z}\rangle$$

which is the linear superposition of all states. First of all, we need to define what a phase separator is. The following definition originates from [5].

Definition 19: Phase Separator

Let H be a Hamiltonian. We call H a phase separator Hamiltonian if H fulfils:

- H is diagonal in the computational basis.
- The eigenspace of the smallest eigenvalue from $H|_S$ is $S_{\min}$.

Then

$$U_P(H, \gamma) := \exp(-i\gamma H)$$

is the corresponding phase separator.

The UFLP is an unconstrained minimisation problem. For a sufficiently large λ, it holds that $H|_S = H$. The eigenspace of the smallest eigenvalue corresponds to $S_{\min}$. After defining the objective Hamiltonian, we also have a diagonal operator in the computational basis. Thus, H_{UFLP} is a phase separator Hamiltonian.
We define the operator B as

$$B = \sum_{i=1}^{N} \sigma_i^1.$$

The corresponding mixer is given by

$$U_M(B, \beta) = \prod_{i=1}^{N} \exp(-i\beta\sigma_i^1).$$

The mixer can be used to change the variables from zero to one or vice versa, depending on the angle β. It should mix the states but only produce feasible solutions. The following explanations of the QAOA are based on [10, page 20]. The initial state is always the superposition of all possible states. First, a phase separator is applied, followed by a mixer. These two unitary operators are applied p times. The associated angles are given by $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p), \boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$. The output state $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ is defined as follows:

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle := U_M(B, \beta_p)U_P(H, \gamma_p) \ldots U_M(B, \beta_1)U_P(H, \gamma_1) |+\rangle .$$

We then calculate the expected value of the objective function with respect to the output and obtain

$$F(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \langle \boldsymbol{\beta}, \boldsymbol{\gamma}| C |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle .$$

In general, the aim is now to calculate an extreme value of $F(\boldsymbol{\beta}, \boldsymbol{\gamma})$. In the context of the FLP, we are interested in the minimum of $F(\boldsymbol{\beta}, \boldsymbol{\gamma})$,

$$M = \min_{\boldsymbol{\beta}, \boldsymbol{\gamma}} F(\boldsymbol{\beta}, \boldsymbol{\gamma}).$$

The QAOA works as follows:

```
Algorithm 6: QAOA
Given are:
  • initial angles $(\boldsymbol{\beta}, \boldsymbol{\gamma})$
while until the termination condition are satified:
    1. Prepare $|+\rangle$ on a quantum computer
    2. Calculate $|\boldsymbol{\beta_i}, \boldsymbol{\gamma_i}\rangle$
    3. Calculate $F(\boldsymbol{\beta}, \boldsymbol{\gamma})$
    4. Update the angle $\boldsymbol{\beta_i}, \boldsymbol{\gamma_i} \to \boldsymbol{\beta_{i+1}}, \boldsymbol{\gamma_{i+1}}$ with a classical optimisation
Repeatedly measure the final output $|\boldsymbol{\beta}_{\boldsymbol{i}_{\text{end}}}, \boldsymbol{\gamma}_{\boldsymbol{i}_{\text{end}}}\rangle$ in the CB
```

Convergence towards the ground state follows for $p \to \infty$.

$$\lim_{p\to\infty} M = \min_{\boldsymbol{z}\in Z(N)} C(\boldsymbol{z}).$$

The proof is shown in [10, appendix B].
If QAOA is to be used for the FLP, there are a few things to consider. First, as already mentioned, the algorithm can only handle unconstrained problems. Therefore, we need to use the softcoded version of the FLP, referred to as UFLP. The mixer can connect or disconnect arbitrary customers with facilities by changing the value of $x_{i,j}$. Similarly, any facility can be opened or closed by changing the value of y_i. Now let's look at the unconstrained Hamiltonian. First, the quadratic term is rewritten:

$$\begin{aligned}\left(\mathbb{1} - \sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j}\right)^2 &= \mathbb{1} - 2\sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j} + \sum_{i=1}^{n}\sum_{i'=1}^{n} \hat{x}_{(i-1)\cdot m+j}\hat{x}_{(i'-1)\cdot m+j} \\ &= \mathbb{1} - 2\sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j} + 2\sum_{i'=1}^{n}\sum_{i=1}^{i'-1} \hat{x}_{(i-1)\cdot m+j}\hat{x}_{(i'-1)\cdot m+j} + \sum_{i=1}^{n} \hat{x}^2_{(i-1)\cdot m+j} \\ &= \mathbb{1} - \sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j} + 2\sum_{i'=1}^{n}\sum_{i=1}^{i'-1} \hat{x}_{(i-1)\cdot m+j}\hat{x}_{(i'-1)\cdot m+j}.\end{aligned}$$

Now it can be decomposed as

$$\begin{aligned}H_{\text{UFLP}} &= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}\hat{x}_{(i-1)\cdot m+j} + \sum_{i=1}^{n} f_i \hat{x}_{nm+i} \\ &\quad + 2\lambda \sum_{j=1}^{m}\sum_{i'=1}^{n}\sum_{i=1}^{i'-1} \hat{x}_{(i-1)\cdot m+j}\hat{x}_{(i'-1)\cdot m+j} + m\lambda\mathbb{1} - \lambda\sum_{i=1}^{n}\sum_{j=1}^{m} \hat{x}_{(i-1)\cdot m+j}\hat{x}_{nm+i}.\end{aligned}$$

The system does not verify whether the two constraints are fulfilled. However, the penalty causes violations of the constraints to yield a larger value of $C(\boldsymbol{z})$. This can result in very high function values, complicating the search for the global minimum due to large local minima. Although it is true that as $p \to \infty$, the minimum is found, there is no information about the convergence rate.

5.3 Mixer for the Opening Constraint

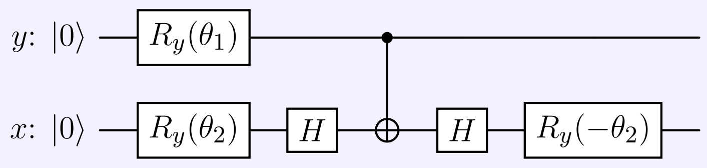
QAOA is a very general algorithm that works exclusively for softcoded problems. The FLP has an assignment constraint 2.3 and an opening constraint 2.4. In the following sections, different possibilities are explored to find better mixers specifically focused on the FLP. For the opening constraint, a quantum circuit is presented in [4]. This circuit is detailed and proven in the following section.

Qubit Inequalities

If $x_{i,j}$ and y_i are two qubits then the inequality $x_{i,j} \leq y_i$ can also be represented as a circuit, as the following theorem shows: (cf. [4, page 7])

Theorem 26: Quantum Circuit for $x \leq y$

The output of the following quantum circuit consisting of a control qubit y and target qubit x satisfies $x \leq y$.



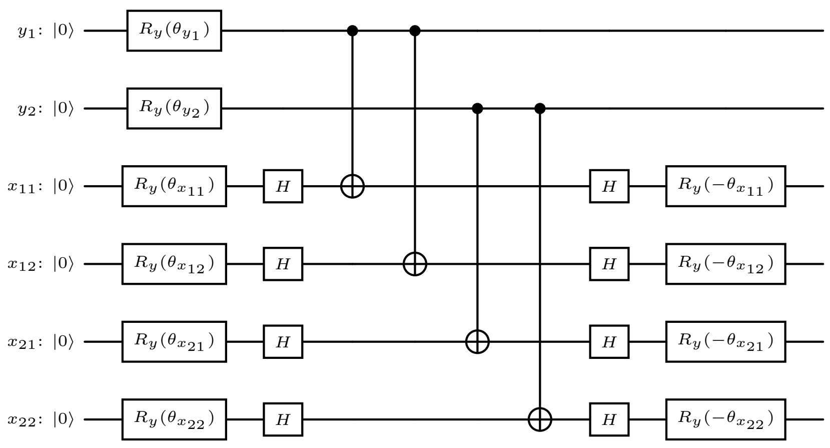
Proof

Two cases are considered. The first case is trivial, if the control qubit y is in the state $|1\rangle$ before the control gate, the target qubit x can either be in the state $|0\rangle$ or $|1\rangle$ at the end, either way the inequality is fulfilled.
The second case describes the case in which the control qubit y is in the state $|0\rangle$. The following operators are then applied to the qubit x.

$$R_y(-\theta_2) \cdot \underbrace{H \cdot H}_{=\mathbb{1}} \cdot R_y(\theta_2) |0\rangle = \underbrace{R_y(-\theta_2) \cdot R_y(\theta_2)}_{=\mathbb{1}} |0\rangle = |0\rangle \,.$$

This means that if the qubit y has the value 0, the qubit x also has the value 0. □

Theorem 26 shows that there is a circuit that guarantees for two qubits y_i and $x_{i,j}$ that $x_{i,j} \leq y_i$ holds. In relation to the FLP, this circuit ensures that if a facility y_i is closed, the customer $x_{i,j}$ is not supplied, thus fulfilling the opening constraint. This circuit can be extended for n facilities and m customers. As a small example, we consider the following case: $(n = 2, m = 2)$.



It can be seen that for each connection between a customer j and a facility i, the value of $x_{i,j}$ depends only on the qubit y_i representing facility i. The following algorithm describes how the circuit can be created. (cf. [4, page 7])

Algorithm 7: Quantum Circuit for $x_{i,j} \leq y_i$

Given are:

- n facilities and m customers
- The first $n \cdot m$ qubits labeled as $x_{i,j}$ $\forall i \in \{1, \dots, n\},\ j \in \{1, \dots, m\}$
- The last n qubits labeled as y_i $\forall i \in \{1, \dots, n\}$
- Angles $(\theta_{x_{11}}, \dots, \theta_{x_{nm}}, \theta_{y_1}, \dots, \theta_{y_n}) \in \mathbb{R}^{nm+n}$

```
Initialise: $\underbrace{|0\dots0\rangle}_{nm+n \text{ times}}$
for $i = 1, \dots, n$ do
    apply $R_y(\theta_{y_i})$ on $y_i$
    for $j = 1, \dots, m$ do
        apply $R_y(\theta_{x_{i,j}})$ on $x_{i,j}$
        apply $H$ on $x_{i,j}$
        apply $C_{y_i}X$ with $x_{i,j}$ as target
        apply $H$ on $x_{i,j}$
        apply $R_y(-\theta_{x_{i,j}})$ on $x_{i,j}$
```

Algorithm 7 iterates over each facility i and applies an $R_y(\theta_{y_i})$ gate to the qubit y_i, where θ_{y_i} is specific to the facility i. Each customer j is considered through the qubit $x_{i,j}$. It must be proven that Algorithm 7 always satisfies the opening constraint.

Theorem 27: Quantum Circuit for $x_{i,j} \leq y_i$

The Algorithm 7 creates a quantum circuit which satisfies for any $(\theta_{x_{11}}, \dots, \theta_{x_{nm}}, \theta_{y_1}, \dots, \theta_{y_n}) \in \mathbb{R}^{nm+n}$ the following inequalities:

$$x_{i,j} \leq y_i \;\forall i \in \{1, \dots, n\},\; j \in \{1, \dots, m\}$$

Proof

Let an arbitrary qubit $x_{i,j}$ be given. If the qubit y_i is in the state $|1\rangle$, there is nothing to show. Let y_i therefore be in the state $|0\rangle$. Then the following circuit is applied to the qubit $x_{i,j}$.

$$R_y(-\theta_{x_{i,j}}) \cdot \underbrace{H \cdot H}_{=\mathbb{1}} \cdot R_y(\theta_{x_{i,j}}) |0\rangle = \underbrace{R_y(-\theta_{x_{i,j}}) \cdot R_y(\theta_{x_{i,j}})}_{=\mathbb{1}} |0\rangle = |0\rangle \, .$$

The output of the qubit $x_{i,j}$ is therefore the state $|0\rangle$ if the qubit y_i is in the state $|0\rangle$. This is valid for all customers j and facilities i. □

Sum Representation of the Hamiltonian

As only the opening constraint is always fulfilled, the assignment constraint must still be softcoded. Classically, this corresponds to the objective function

$$C_{\text{TV}} = \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} x_{i,j} + \lambda \sum_{j=1}^{m} \left(1 - \sum_{i=1}^{n} x_{i,j}\right)^2 .$$

The unconstrained Hamiltonian can be modified to obtain the Taylorised Variational Hamiltonian H_{TV}. The Hamiltonian is named after the article in which the corresponding algorithm was presented. H_{TV} is defined as

$$H_{\text{TV}} = \sum_{i=1}^{n} f_i \hat{x}_{nm+i} + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} \hat{x}_{(i-1)\cdot m+j} + \lambda \sum_{j=1}^{m} \left(\mathbb{1} - \sum_{i=1}^{n} \hat{x}_{(i-1)\cdot m+j}\right)^2 .$$

The Hamiltonian can be rewritten as

$$H_{\text{TV}} = \sum_{i=1}^{n}\sum_{j=1}^{m}(c_{i,j} - \lambda)\hat{x}_{(i-1)\cdot m+j} + \sum_{i=1}^{n} f_i \hat{x}_{nm+i}$$
$$+ 2\lambda \sum_{j=1}^{m}\sum_{i'=1}^{n}\sum_{i=1}^{i'-1} \hat{x}_{(i-1)\cdot m+j}\hat{x}_{(i'-1)\cdot m+j} + m\lambda\mathbb{1}.$$

Let $|\varphi(\boldsymbol{\theta})\rangle := |\varphi(\theta_{x_{11}}, \dots, \theta_{x_{nm}}, \theta_{y_1}, \dots, \theta_{y_n})\rangle$ be the output of Algorithm 7. Obviously, the output of the quantum circuit depends on the angles
$\boldsymbol{\theta} := (\theta_{x_{11}}, \dots, \theta_{x_{nm}}, \theta_{y_1}, \dots, \theta_{y_n}) \in \mathbb{R}^{nm+n}$. The objective is to minimise the expected value of C_{TV} in the states $|\varphi(\boldsymbol{\theta})\rangle$. This results in a minimisation problem in the $nm+n$ angles $\boldsymbol{\theta}$.

Theorem 28: Minimum from C_{TV}

Given is the solution space $S \subset q^{\otimes nm+n}$ of the FLP. Let E_{opt} a optimal solution of the FLP, defined by:

$$E_{\text{opt}} := \min_{|\varphi\rangle \in S} \langle\varphi| C_{\text{FLP}} |\varphi\rangle$$

The following equation holds:

$$E_{\text{opt}} = \min_{\boldsymbol{\theta}\in\mathbb{R}^{nm+n}} \langle\varphi(\boldsymbol{\theta})| C_{\text{TV}} |\varphi(\boldsymbol{\theta})\rangle$$

Proof

Let the ground state of the FLP be given by $|\boldsymbol{z}\rangle$. First, it must be shown that there are angles $\boldsymbol{\theta}$ so that $|\varphi(\boldsymbol{\theta})\rangle = |\boldsymbol{z}\rangle$. Since the optimal solution $\boldsymbol{z}$ fulfils both constraints, there is $i_1, \dots, i_m \in \{1, \dots, n\}$ with $x_{i_1,1} = \dots = x_{i_m,m} = y_{i_1} = \dots = y_{i_m} = 1$. All other $x_{i,j}, y_i$ are set to 0.
Since $R_y(0) = \mathbb{1}$, we select the angle $\theta_{y_i} = 0$ for each facility i with $y_i = 0$. Then $x_{i,j} = 0\ \forall i \neq i_j$ holds.
Let us consider an arbitrary open facility i. Since $R_y(\pi) \cong X$ is valid, we choose $\theta_{y_i} = \pi$, so the qubit y_i is changed from $|0\rangle$ to $|1\rangle$. For all customers j that are supplied by the facility i, we select $\theta_{x_{i,j}} = \frac{3\pi}{2}$. This follows from the equality

$$R_y\left(-\frac{3\pi}{2}\right) \cdot \underbrace{H \cdot X \cdot H}_{=Z} \cdot R_y\left(\frac{3\pi}{2}\right)|0\rangle = \underbrace{R_y\left(-\frac{3\pi}{2}\right) \cdot Z \cdot R_y\left(\frac{3\pi}{2}\right)}_{=X}|0\rangle = |1\rangle\,.$$

For all other customers k we choose $\theta_{x_{ik}} = 0$, then the equality

$$R_y(0) \cdot \underbrace{H \cdot X \cdot H}_{=Z} \cdot R_y(0) |0\rangle = Z |0\rangle = |0\rangle$$

holds. If we select the angles as described, the statement follows. The penalty term prevents states that do not fulfil the assignment constraint from becoming the minimum. This means that $E_{\text{opt}} = \min_{\boldsymbol{\theta} \in \mathbb{R}^{nm+n}} \langle \varphi(\boldsymbol{\theta}) | C_{\text{TV}} | \varphi(\boldsymbol{\theta}) \rangle$ holds. □

The objective is now to determine $\boldsymbol{\theta}$ in such a way that the expected value of C_{TV} is minimised. To achieve this, Algorithm 7 is used iteratively: first, the expected value is calculated, and then the angles are optimised classically.

Algorithm 8: Taylored Variational Algorithm

Given are:

- n facilities and m customers
- The first $n \cdot m$ qubits labels as $x_{i,j}$ $\forall i \in \{1, \dots, n\}, \ j \in \{1, \dots, m\}$
- The last n qubits labels as y_i $\forall i \in \{1, \dots, n\}$
- initial angles $\boldsymbol{\theta}_0 \in \mathbb{R}^{nm+n}$

Carry out the following steps, starting with $i = 0$:

1. Perform the Algorithm 7 with the angles $\boldsymbol{\theta}_i$ and call the output $|\varphi(\boldsymbol{\theta}_i)\rangle$
2. Calculate using a measurement: $\langle \varphi(\boldsymbol{\theta}_i) | C_{\text{TV}} | \varphi(\boldsymbol{\theta}_i) \rangle$.
3. Fit the angles to $\boldsymbol{\theta}_i \mapsto \boldsymbol{\theta}_{i+1}$ with a classic optimisation algorithm.
4. Return to step 1. until the maximum iterations are reached. Name the final vector $\boldsymbol{\theta}_{i_{end}}$.

Algorithm 8 combines Algorithm 7 with Theorem 28. First, Algorithm 7 is applied to ensure that the opening constraint is fulfilled in the output state $|\varphi(\boldsymbol{\theta}_i)\rangle$. Next, the expected value of C_{TV} is calculated in the output state. The angles are then classically optimised to minimise the next expected value $\langle \varphi(\boldsymbol{\theta}_{i+1}) | C_{\text{TV}} | \varphi(\boldsymbol{\theta}_{i+1}) \rangle$. This process is repeated several times. Various criteria can be used to determine when to stop, such as a specified number of iterations or the absence of significant changes.

5.4 Mixer for the Assignment Constraint

Now, we are shifting our perspective. Previously, we used either the standard QAOA mixer with an unconstrained FLP or a mixer specifically tailored to the opening constraint 2.4. Now, a mixer for the assignment constraint 2.3 will be introduced. This approach is based on the Traveling Salesperson Problem (TSP), which was discussed in Chapter 2.2. As demonstrated in [5], the optimal solution for the TSP can be found by swapping qubits. For the FLP, we first included the opening constraint 2.4 in the objective function and Hamiltonian.

$$H_{\text{SWAP}} = \sum_{i=1}^{n}\sum_{j=1}^{m}(c_{i,j}+\lambda)\hat{x}_{(i-1)\cdot m+j} + \sum_{i=1}^{n} f_i \hat{x}_{nm+i} - \lambda \sum_{i=1}^{n}\sum_{j=1}^{m}\hat{x}_{(i-1)\cdot m+j}\hat{x}_{nm+i}$$

$$C_{\text{SWAP}} = \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}x_{i,j} + \lambda \sum_{j=1}^{m}\left(\sum_{i=1}^{n}(x_{i,j} - x_{i,j}y_i)\right).$$

In the constraint 2.3, the main point is that exactly one facility i supplies each customer j. The idea is that we first assign a facility i for each customer j, classically setting $x_{i,j} = 1$. Now we remain with the same customer j and change the facility to i'. Therefore, the following change is made classically:

$$(x_{i,j} = 1, x_{i'j} = 0) \Rightarrow (x_{i,j} = 0, x_{i'j} = 1).$$

For a quantum computer, we therefore have to swap the associated qubits so that we can define the SWAP gate, inspired by [30]

$$\begin{aligned}\text{SWAP}_{ii'j} = \frac{1}{2}(\mathbb{1} &+ \sigma^1_{(i-1)\cdot m+j}\sigma^1_{(i'-1)\cdot m+j} \\ &+ \sigma^2_{(i-1)\cdot m+j}\sigma^2_{(i'-1)\cdot m+j} + \sigma^3_{(i-1)\cdot m+j}\sigma^3_{(i'-1)\cdot m+j}).\end{aligned}$$

With this gate, the qubits $(i-1)\cdot m + j$ and $(i'-1)\cdot m + j$ are swapped. The qubit $(i-1)\cdot m + j$ corresponds to $x_{i,j}$ and the qubit $(i'-1)\cdot m + j$ corresponds to $x_{i',j}$. The following mechanism can be used to decide, based on an angle β, whether two qubits should be swapped, depending on which configuration yields a smaller expected value.

$$U_{ii'j}(\beta_{ii'j}) = \exp(-i\beta_{ii'j} \cdot \text{SWAP}_{ii'j}).$$

Now the product is formed using all possible swaps of the facilities

$$U_{\text{SWAP}}(\boldsymbol{\beta}) = \prod_{j=1}^{m}\prod_{i=1}^{n}\prod_{i'=i+1}^{n} U_{ii'j}(\beta_{ii'j}) \text{ with } \boldsymbol{\beta} \in \mathbb{R}^{m\cdot\frac{n^2-n}{2}}.$$

$U_{\text{SWAP}}(\boldsymbol{\beta})$ can swap the values of $x_{i,j}$ while complying with the assignment constraint. However, it must also be possible to open or close facilities. For this, we use the QAOA mixer, but we only apply it to the y_i qubits, influencing whether the facilities are opened or closed.

$$U_{\text{QAOA}}(\alpha) = \prod_{i=nm+1}^{nm+n} \exp(-i\alpha\sigma_i^1).$$

If we apply the two mixers and the phase separator to a state $|\varphi\rangle$ that fulfils the assignment constraint, we get the output

$$|\alpha, \boldsymbol{\beta}, \gamma\rangle = U_{\text{QAOA}}(\alpha) U_P(H_{\text{SWAP}}, \gamma) U_{\text{SWAP}}(\boldsymbol{\beta}) |\varphi\rangle .$$

Since both the input and the output fulfil the assignment constraint, it can be applied multiple times, resulting in a modified version of QAOA. Thus, we can choose a value $p \geq 1$ and apply the mixer p times to our initial state. Then we get:

$$|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = \prod_{i=1}^{p} U_{\text{QAOA}}(\alpha_i) U_P(H_{\text{SWAP}}, \gamma_i) U_{\text{SWAP}}(\boldsymbol{\beta}_i) |\varphi\rangle$$

The algorithm works as follows:

Algorithm 9: SWAP Mixer

Given are:

- $p \geq 1$
- initial state $|\varphi\rangle$ which fulfilled the assignment constraint
- initial angles $\boldsymbol{\alpha}_0 = (\alpha_{1_0}, \dots, \alpha_{p_0}), \boldsymbol{\beta}_0 = (\boldsymbol{\beta}_{1_0}, \dots, \boldsymbol{\beta}_{p_0}), \boldsymbol{\gamma}_0 = (\gamma_{1_0}, \dots, \gamma_{p_0})$

```
while until the termination conditions are satisfied:
    1. Calculate $|\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\gamma}_i\rangle$
    2. Calculate $F(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\gamma}_i) = \langle\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\gamma}_i| C_{\text{SWAP}} |\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\gamma}_i\rangle$
    3. Update the angle $(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, , \boldsymbol{\gamma}_i) \to (\boldsymbol{\alpha}_{i+1}, \boldsymbol{\beta}_{i+1}, \boldsymbol{\gamma}_{i+1})$ with a classical
       optimisation
Repeatedly measure the final output $|\boldsymbol{\alpha}_{\boldsymbol{i}_{\text{end}}}, \boldsymbol{\beta}_{\boldsymbol{i}_{\text{end}}}, \boldsymbol{\gamma}_{\boldsymbol{i}_{\text{end}}}\rangle$ in the CB
```

We take a few steps back. Let's look at the assignments of facilities and customers. These are represented by the $x_{i,j}$. What output states can be achieved with $U_{\text{SWAP}}(\boldsymbol{\beta})$? The question answers the next theorem:

Theorem 29: Range from $U_{\mathbf{SWAP}}(\boldsymbol{\beta})$

Let $|E_g\rangle = |x_{11} = a_{11}, \dots, x_{nm} = a_{nm}, y_1 = b_1, \dots, y_n = b_n\rangle$ with $a_{11}, \dots, a_{nm}, b_1, \dots, b_n \in \{0{,}1\}$ be the ground state of a FLP and let $|\varphi\rangle$ be an initial state which fulfils the assignment constraint. Then there are angles $\boldsymbol{\beta} \in \mathbb{R}^{pm\frac{n^2-n}{2}}$ such that $|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ matches with $|E_g\rangle$ for $x_{11}, \dots, x_{nm}$, i.e. $|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = |x_{11} = a_{11}, \dots, x_{nm} = a_{nm}, y_1 = c_1, \dots, y_n = c_n\rangle$ with $c_1, \dots, c_n \in \{0{,}1\}$.

Proof

Let the initial state be given as

$$|\varphi\rangle = |x_{11} = d_{11}, \dots, x_{nm} = d_{nm}, y_1 = e_1, \dots, y_n = e_n\rangle$$

with $d_{11}, \dots, d_{nm}, e_1, \dots, e_n \in \{0{,}1\}$. Let $d_{k_1,1}, \dots, d_{k_m,m} = 1$ be given. Since $|\varphi\rangle$ satisfies the assignment constraint, this applies to all other $d_{ij} = 0$. For the ground state $|E_g\rangle$ applies $a_{i_1,1}, \dots, a_{i_m,m} = 1$ and all others $a_{ij} = 0$. With this knowledge, the angles can now be selected as

$$\beta_{ii'j} = \begin{cases} \pi \text{ if } d_{ij} = a_{i'j} = 1 \\ 0 \text{ otherwise} \end{cases}.$$

The rule is that only one facility i supplies the customer j. Thus, from $x_{i,j}$ to $x_{n,j}$ there is only one $x_{i,j} = 1$, the others are 0. We swap this value of 1 from the position of our initial state to the position at which it is located in the ground state and obtain

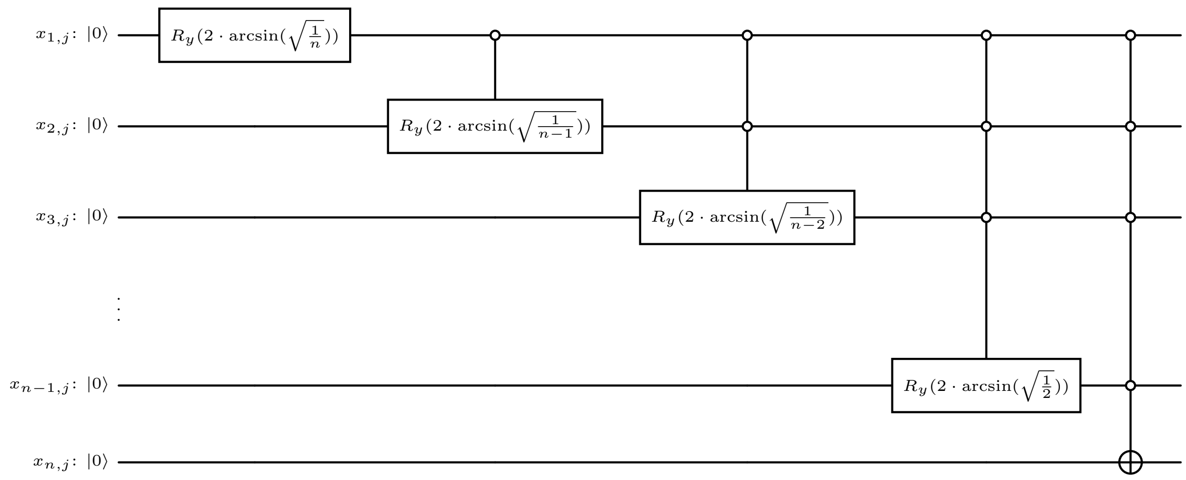
$$U_{\text{SWAP}}(\boldsymbol{\beta})\,|\varphi\rangle = |x_{11} = a_{11}, \dots, x_{nm} = a_{nm}, y_1 = e_1, \dots, y_n = e_n\rangle\,.$$

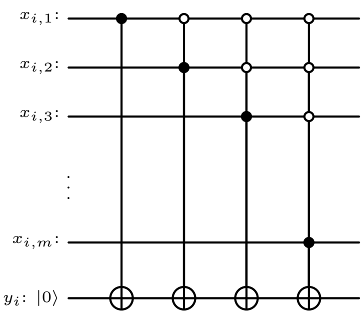
The Operator $U_P(H_{\text{SWAP}})$ does not open or close any facilities or disconnect any connections. The operator $U_{\text{QAOA}}(\alpha)$ only changes the values of $y_1, \dots, y_n$ and thus the assertion is shown. □

The question is what constitutes a good initial state. If the opening cost were zero, Theorem 29 shows that we can start with any state as long as the assignment constraint holds. Since there are opening costs and associated qubits y_i, the QAOA mixer is applied to these qubits. Therefore, starting with a superposition state makes the most sense. To achieve this superposition, a quantum circuit is required, which is presented in the next section

Superposition of All Feasible States

First, it is necessary to define what is meant by all feasible states. Guided by Chapter 3.2, all feasible states are those that fulfill both the assignment constraint and the equivalent opening constraint from Theorem 4. To achieve a superposition of all feasible states, we consider each customer step by step. For a given customer, we assign a probability of $1/n$ for each possible connected facility. As a quantum circuit, this looks as follows:


$x_{1,j}$: $|0\rangle$
$x_{2,j}$: $|0\rangle$
$x_{3,j}$: $|0\rangle$
$x_{n-1,j}$: $|0\rangle$
$x_{n,j}$: $|0\rangle$
$R_y(2 \cdot \arcsin(\sqrt{\frac{1}{n}}))$
$R_y(2 \cdot \arcsin(\sqrt{\frac{1}{n-1}}))$
$R_y(2 \cdot \arcsin(\sqrt{\frac{1}{n-2}}))$
$R_y(2 \cdot \arcsin(\sqrt{\frac{1}{2}}))$



A customer is associated with exactly one facility. For example, for customer j, the operator $R_y\left(2\arcsin\left(\sqrt{\frac{1}{n}}\right)\right)$ is applied to $x_{1,j}$, which generates the state $|1\rangle$ with a probability of $\frac{1}{n}$ and remains in the state $|0\rangle$ with a probability of $\frac{n-1}{n}$. Iteratively, reverse-controlled $R_y\left(2\arcsin\left(\sqrt{\frac{1}{n-i}}\right)\right)$ are applied to $x_{i+1,j}$. The operator is only applied if $x_{1,j} = \cdots = x_{i,j} = 0$. If we apply this quantum circuit to all customers, we get a superposition of all feasible $x_{1,1}, \ldots, x_{n,m}$. However, the opening constraint must still be considered. This means that if one of the $x_{i,1}, \ldots, x_{i,m}$ has the value 1, then facility i must also be opened. Therefore, we go through the $x_{i,j}$ one after the other, and if the value of y_i is still 0, a CNOT gate is applied with $x_{i,j}$ as the control qubit and y_i as the target qubit. The algorithm that generates the superposition of all possible feasible solutions is as follows:

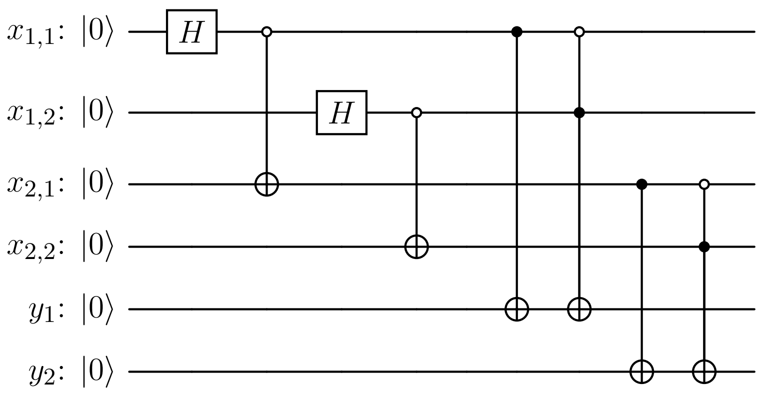
Algorithm 10: Quantum Circuit for FLP Superposition

Given are:

- n facilities and m customers
- The first $n \cdot m$ qubits labels as $x_{i,j}\ \forall i \in \{1,\dots,n\},\ j \in \{1,\dots,m\}$
- The last n qubits labels as $y_i\ \forall i \in \{1,\dots,n\}$

```
Initialise: $\underbrace{|0\dots0\rangle}_{nm+n \text{ times}}$
for $j = 1,\dots,m$ do
    apply $R_y\left(2\arcsin\left(\sqrt{\frac{1}{n}}\right)\right)$ on $x_{1,j}$
    for $i = 1,\dots,n-1$ do
        apply $\overline{C}_{x_{1,j}}\dots\overline{C}_{x_{i,j}}R_y\left(2\arcsin\left(\sqrt{\frac{1}{n-i}}\right)\right)$ to the target $x_{i+1,j}$
for $i = 1,\dots,n$ do
    apply $C_{x_{i,1}}X$ to the target $y_i$
    for $j = 2,\dots,m$ do
        apply $\overline{C}_{x_{i,1}}...\overline{C}_{x_{i,j-1}}C_{x_{i,j}}X$ to the target $y_i$
```

Now we look at the case $n = m = 2$:



In the case where $n = m = 2$, we can use $R_y(2\cdot\arcsin(\sqrt{\frac{1}{2}})) = R_y\left(\frac{\pi}{2}\right) \cong H$. Initially, there is a 50% probability that the first customer is assigned to the first facility; if not, they are assigned to the second facility. This process is repeated for the second customer. Afterward, a superposition of all possible assignments of customers to facilities is generated. It is then checked whether facility 1 supplies any customers, and if so, it is opened. The same process is repeated for the second facility. Next, it is demonstrated that the algorithm indeed generates a superposition of all feasible states.

Theorem 30: Circuit for Superposition of All Feasible States

Let an arbitrary instance of an FLP be given. Then Algorithm 10 creates a superposition of all states which fulfils the assignment constraint and the constraint from Theorem 4.

Proof

We split the proof into two parts. The first part deals with the assignments of customers to the facilities and the second part deals with the opening of the facilities.
1. We look at the first for loop over j. In this loop, the individual customers j are independent of each other. This means that it is sufficient to look at one customer j. The following gate is applied to the qubit $x_{1,j}$:

$$R_y\left(2\arcsin\left(\sqrt{\frac{1}{n}}\right)\right)|0\rangle_{x_{1,j}} = \sqrt{\frac{n-1}{n}}\,|0\rangle_{x_{1,j}} + \sqrt{\frac{1}{n}}\,|1\rangle_{x_{1,j}}\,.$$

This results in $P(x_{1,j}=0)=\frac{n-1}{n}, P(x_{1,j}=1)=\frac{1}{n}$. Next we apply a for loop over each facility i. Here, if the customer j is not yet connected to the facilities $1,\dots,n$, the following operator is applied to the qubit $x_{i+1,j}$:

$$R_y\left(2\arcsin\left(\sqrt{\frac{1}{n-i}}\right)\right)|0\rangle_{x_{i+1,j}} = \sqrt{\frac{n-i-1}{n-i}}\,|0\rangle_{x_{i+1,j}} + \sqrt{\frac{1}{n-i}}\,|1\rangle_{x_{i+1,j}}\,.$$

We now calculate the probability that $x_{i+1,j}$ switches into the state 1:

$$P(x_{i+1,j}=1)=\prod_{k=0}^{i-1}\frac{n-k-1}{n-k}\cdot\frac{1}{n-i}=\frac{n-1}{n}\cdot\frac{n-2}{n-1}\cdot\dots\cdot\frac{n-i}{n-i+1}\cdot\frac{1}{n-i}=\frac{1}{n}.$$

The product at the beginning stands for the probability that all qubits before it remain at 0. It is then multiplied with the probability that the state $x_{i+1,j}$ switches to 1. After the second equals sign, we can see that all numbers except 1 and n are cancel each other out. Logically it follows

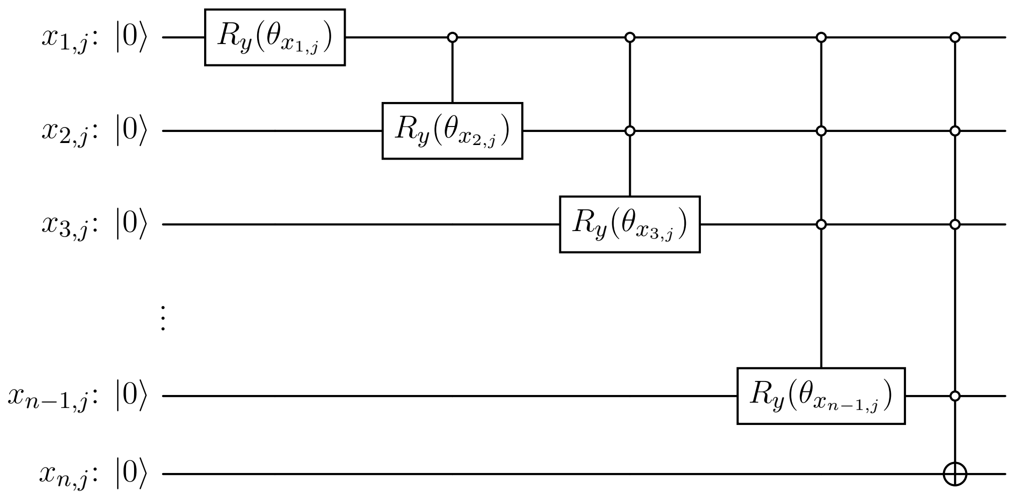
$$P(x_{i+1,j} = 0) = 1 - P(x_{i+1,j} = 1) = 1 - \frac{1}{n} = \frac{n-1}{n}.$$

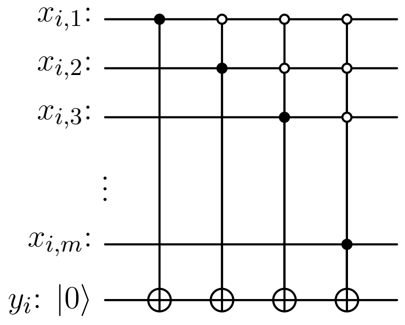
2. The second step is to show how the facilities are opened. To do this, we examine the for loop over i. The individual facilities are independent of each other, so we can simply look at a single facility i. First, we check whether the first customer is connected to facility i (i.e., if $x_{1,j} = 1$). If so, the facility is set to 1. For customer j, the algorithm first checks whether customers $1, \ldots, j-1$ are connected to the facility. If none of these customers are connected, the algorithm checks whether customer j is connected to facility i, and if so, facility i is opened. This prevents an open facility from being closed again.
Thus, in part 1, we have shown how all possible combinations of facilities and customers are generated as a superposition, and in part 2, we have shown how the connected facilities are opened. $\square$

The greatest challenges in creating the circuit involve constructing the n-controlled R_y and C^nX gates. For the n-controlled R_y gate, Algorithm 5 can be used, where the special case of the controlled R_y gate is explained. For the C^nX gates, their implementation is detailed in Algorithm 4. The $\overline{C}X$ gates can simply be changed to CX gates with two X gates.

5.5 Mixer for the Facility Location Problem

So far, we have presented mixers that require softcoding for one or both constraints. Next, we will introduce a mixer that operates without softcoding. The inspiration for this approach comes from Algorithm 10, where a superposition of all feasible states was created. In Algorithm 10, all R_y gates have specified angles that ensure each state has the same probability. However, by keeping the angles as variables and optimising them, we can achieve an algorithm where any specific feasible state can be obtained for particular angles, as will be demonstrated later. The quantum circuit is designed as follows:





The upper circuit is executed for each customer in turn, followed by the lower circuit for each facility. In total, we need $(n-1)\cdot m$ angles for optimisation. The idea is that if we look at customer j and connect it to the i-th facility, we set $\theta_{x_{ij}} = \pi$ and leave all other angles $\theta_{x_{kj}}$ $\forall k \in \{1,\dots,m\}\backslash\{i\}$ to 0. The corresponding algorithm is as follows:

Algorithm 11: Quantum Circuit for FLP

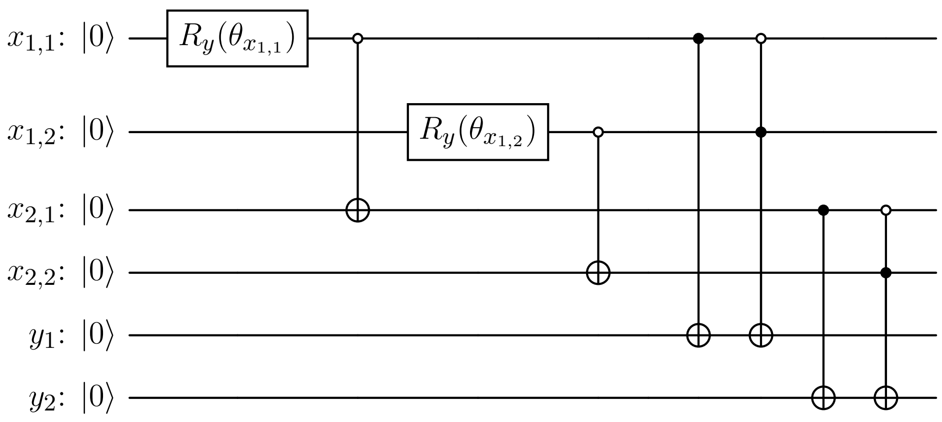
Given are:

- n facilities and m customers
- The first $n \cdot m$ qubits labels as $x_{i,j}$ $\forall i \in \{1, \dots, n\},\ j \in \{1, \dots, m\}$
- The last n qubits labels as y_i $\forall i \in \{1, \dots, n\}$
- Angles $(\theta_{x_{1,1}}, \dots, \theta_{x_{n-1,m}}) \in \mathbb{R}^{(n-1)m}$

```
Initialise: $\underbrace{|0\dots0\rangle}_{nm+n \text{ times}}$
for $j = 1, \dots, m$ do
    apply $R_y(\theta_{x_{1,j}})$ on $x_{1,j}$
    for $i = 1, \dots, n-2$ do
        apply $\overline{C}_{x_{1,j}} \dots \overline{C}_{x_{i,j}} R_y(\theta_{x_{i+1,j}})$ to the target $x_{i+1,j}$
    apply $\overline{C}_{x_{1,j}} \dots \overline{C}_{x_{n-1,j}} X$ to the target $x_{n,j}$
for $i = 1, \dots, n$ do
    apply $C_{x_{i,1}} X$ to the target $y_i$
    for $j = 2, \dots, m$ do
        apply $\overline{C}_{x_{i,1}} \dots \overline{C}_{x_{i,j-1}} C_{x_{i,j}} X$ to the target $y_i$
```

It is important that for each customer j, if they are not connected to facilities $1, \dots, n-1$, they are connected to facility n in order to fulfil the assignment constraint. Incidentally, this is also the case in Algorithm 10 if we take into account the fact that $R_y(2\arcsin(1)) = R_y(\pi) \cong X$ applies, although this is not directly obvious from the algorithm. The circuit generated by Algorithm 10 is therefore a special case where the angles are specified.

Let's take a look at the case $n = m = 2$. In this circuit, an R_y gate is first applied to $x_{1,1}$, which corresponds to connecting customer 1 with facility 1. If we leave the qubit $x_{1,1}$ at 0, the qubit $x_{2,1}$ is changed to 1. The same then happens with the second customer. This guarantees the assignment constraint. The first facility is then opened (y_1 changed to 1) if at least one of the customers is supplied by the facility. This process is repeated for the second facility. In this way, the opening constraint is also always fulfilled.



The question now is whether every possible ground state of an FLP can be achieved with Algorithm 11 by selecting the appropriate angles. The following theorem addresses this question:

Theorem 31: Quantum Circuit for FLP

Let an arbitrary instance of an FLP be given and let $\mathcal{S}$ be the set of all states which fulfils the assignment constraint and the equivalent opening constraint from Theorem 4. Then the following two points holds:

1. For all states $\boldsymbol{z} \in \mathcal{S}$ there are angles $\boldsymbol{\theta} \in \mathbb{R}^{(n-1)m}$, such that the Algorithm 11 generates the state $\boldsymbol{z}$.

2. The Algorithm 11 cannot generate a non-feasible state $\boldsymbol{z}' \notin \mathcal{S}$.

Proof

1. We consider an arbitrary state which, fulfils the assignment constraint and the equivalent opening constraint. Then there are $i_1, \ldots, i_m \in \{1, \ldots, n\}$ with $x_{i_1,1} = \cdots = x_{i_m,m} = y_{i_1} = y_{i_m} = 1$. We now define the angles $\theta_{x_{i,j}}$ as

$$\theta_{x_{i,j}} = \begin{cases} \pi \text{ if } i = i_j \\ 0 \text{ otherwise} \end{cases} .$$

These angles are chosen because the output of an R_y gate is unique in the following way

$$R_y(\pi) \cong X \quad R_y(0) = \mathbb{1}.$$

Since a customer j is only connected to one facility i_j, a non-trivial operator is only applied to the qubit $x_{i_j,j}$. After initialisation, $x_{1,j} = \cdots = x_{i_j-1,j} = 0$ applies. The operator X is then applied to the qubit $x_{i_j,j}$, which changes the qubit to 1. This means that we connect customer j to facility i_j for all customers. With the second large for loop, the associated facility is also automatically opened ($y_{i_j} = 1$).
2. Now, let's consider two cases. The first case is generating a state where the assignment constraint is not fulfilled. The second case is generating a state where the opening constraint is not fulfilled.
i. If all $x_{1,j}, \dots, x_{n-1,j}$ are set to 0, $x_{n,j}$ is always set to 1. Therefore, it is not possible to use the algorithm to create a state that does not fulfill the assignment constraint.
ii. States that fulfill the assignment constraint also automatically fulfill the opening constraint. This is ensured by the second large for loop. The y_i are always set to 1 if at least one of the $x_{i,1}, \dots, x_{i,m}$ is set to 1. $\square$

The goal is to use Algorithm 11 to approximate the minimum of the FLP. To achieve this, a classic optimisation algorithm is used after each step to optimise the angles.

Algorithm 12: FLP Approximation Algorithm

Given are:

- n facilities and m customers
- The first $n \cdot m$ qubits labels as $x_{i,j}$ $\forall i \in \{1, \dots, n\},\ j \in \{1, \dots, m\}$
- The last n qubits labels as y_i $\forall i \in \{1, \dots, n\}$
- initial angles $\boldsymbol{\theta}_0 \in \mathbb{R}^{(n-1)m}$

Carry out the following steps, starting with $i = 0$:

1. Perform the Algorithm 11 with the angles $\boldsymbol{\theta}_i$ and call the output $|\varphi(\boldsymbol{\theta}_i)\rangle$
2. Calculate using a measurement: $\langle\varphi(\boldsymbol{\theta}_i)| C_{\text{CFLP}} |\varphi(\boldsymbol{\theta}_i)\rangle$.
3. Fit the angles to $\boldsymbol{\theta}_i \mapsto \boldsymbol{\theta}_{i+1}$ with a classic optimisation algorithm.
4. Return to step 1. until the maximum iterations are reached. Name the final vector $\boldsymbol{\theta}_{i_{end}}$.

Algorithm 12 can be used to approximate the minimum of the FLP. To understand this, let's examine the second part of Theorem 31. No infeasible states can be created with Algorithm 12. This means that there can be no states with smaller eigenvalues

than the ground state.
The next question is whether there are angles that allow us to reach the ground state. For this, we consider the first part of Theorem 31, which explains that every feasible state can be generated, including the ground state. Since there is no smaller eigenvalue, the minimum can be approximated by optimising the angles.

Now that several quantum algorithms have been introduced to approximate the solution of the FLP, the question arises as to which algorithm performs best. These algorithms will be tested and compared in the next chapter.

6 Testing of the FLP Quantum Algorithms

This chapter compares the algorithms presented in Chapter 5. The table below lists the algorithms along with their associated objective functions and Hamiltonians.

Name of the Algorithm	Unconstraint Algorithm	TV Algorithm	SWAP Algorithm	Constraint Algorithm
Objektiv Function	C_{UFLP}	C_{TV}	C_{SWAP}	C_{CFLP}
Objectiv Hamiltonian	H_{UFLP}	H_{TV}	H_{SWAP}	H_{CFLP}
Softcoded Constraints	both	assignment	opening	none
Algorithms	6	8	9	12

Table 6.1: Overview of some approximation algorithms for the FLP which was presented in Chapter 5.

6.1 Theoretical Analysis of the Algorithms

Representation of Hamiltonian's in the Pauli Basis

For the implementation of phase separators it is useful to decompose the Hamiltonian into Pauli matrices. It's holds

$$\hat{x}_a = \frac{1}{2}\mathbb{1} - \frac{1}{2}\sigma^3_a,\ a \in \{1, \dots, nm+n\}.$$

If we apply the two equations and summarise them, we get

$$H_{\text{CFLP}} = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{-c_{i,j}}{2}\sigma^3_{(i-1)\cdot m+j} + \sum_{i=1}^{n} \frac{-f_i}{2}\sigma^3_{nm+i} + \frac{\lambda}{2}\mathbb{1}.$$

To rewrite the penalty for the assignment constraint, the following conversion is required.

$$\sum_{i=1}^{n}\sum_{i'=1}^{i-1}\hat{x}_{(i-1)\cdot m+j}\hat{x}_{(i'-1)\cdot m+j}=\sum_{i=1}^{n}\sum_{i'=1}^{i-1}\frac{1}{4}(\mathbb{1}-\hat{x}_{(i-1)\cdot m+j})(\mathbb{1}-\hat{x}_{(i'-1)\cdot m+j})$$
$$=\frac{1}{4}\sum_{i=1}^{n}\sum_{i'=1}^{i-1}\left(\mathbb{1}-\sigma^3_{(i-1)\cdot m+j}-\sigma^3_{(i'-1)\cdot m+j}+\sigma^3_{(i-1)\cdot m+j}\sigma^3_{(i'-1)\cdot m+j}\right)$$
$$=\frac{1}{4}\left(\sum_{i=1}^{n}\sum_{i'=1}^{i-1}\sigma^3_{(i-1)\cdot m+j}\sigma^3_{(i'-1)\cdot m+j}-\sum_{i=1}^{n}(n-1)\sigma^3_{(i-1)\cdot m+j}+\frac{n^2-n}{2}\mathbb{1}\right).$$

The decomposed into Pauli matrices, of the Hamiltonians look as follows:

$$H_{\text{TV}}=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{-c_{i,j}+\lambda(2-n)}{2}\sigma^3_{(i-1)\cdot m+j}+\sum_{i=1}^{n}\frac{-f_i}{2}\sigma^3_{nm+i}$$
$$+\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{i'=1}^{i-1}\frac{\lambda}{2}\sigma^3_{(i-1)\cdot m+j}\sigma^3_{(i'-1)\cdot m+j}+\frac{\lambda}{2}\left(m\frac{n^2-3n+4}{2}+1\right)\mathbb{1}.$$
$$H_{\text{SWAP}}=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{-2c_{i,j}-\lambda}{4}\sigma^3_{(i-1)\cdot m+j}+\sum_{i=1}^{n}\frac{-2f_i+m\lambda}{4}\sigma^3_{nm+i}$$
$$-\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\lambda}{4}\sigma^3_{(i-1)\cdot m+j}\sigma^3_{nm+i}+\frac{(2+n\cdot m)\lambda}{4}\mathbb{1}.$$
$$H_{\text{UFLP}}=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{-2c_{i,j}+\lambda(3-2n)}{4}\sigma^3_{(i-1)\cdot m+j}+\sum_{i=1}^{n}\frac{-2f_i+m\lambda}{4}\sigma^3_{nm+i}$$
$$+\sum_{j=1}^{m}\sum_{i'=1}^{n}\sum_{i=1}^{i'-1}\frac{\lambda}{2}\sigma^3_{(i-1)\cdot m+j}\sigma^3_{(i'-1)\cdot m+j}-\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\lambda}{4}\sigma^3_{(i-1)\cdot m+j}\sigma^3_{nm+i}$$
$$+\frac{\lambda}{2}\left(m\frac{n^2-2n+4}{2}+1\right)\mathbb{1}.$$

The following table shows the number of summands for each Hamiltonian.

In Table 6.2, it is shown that the Hamiltonian without softcoding naturally requires the fewest terms. However, softcoding the opening constraint does not change the order of magnitude, which remains $\mathcal{O}(nm)$. In contrast, softcoding the assignment constraint increases the order of magnitude to $\mathcal{O}(n^2m)$. The computational effort increases with the number of terms, so H_{SWAP} and H_{TV} perform worse in this regard.

Hamiltonian	H_{CFLP}	H_{TV}	H_{SWAP}	H_{SWAP}
Terms	$nm+n+1$	$n(\frac{m}{2}$ $+\frac{nm}{2}+1)+1$	$2nm+n+1$	$n(\frac{3m}{2}$ $+\frac{nm}{2}+1)+1$

Table 6.2: Comparison of the number of terms of the different Hamiltonian.

Number of Unitaries in Mixers

In Table 6.3 the number of qubits, angles and gates become compared.

Problem	UFLP	TV	SWAP	CFLP
Number of Qubits	$nm+n$	$nm+n$	$nm+n$	$nm+n$ $+\max(n-1,m)-2$
Angles	$2p$	$nm+n$	$p(m\frac{n^2-n}{2}+2)$	$(n-1)m$
Phase separator	p		p	
Single qubit operator	$(nm+n)p$	$5nm+n$	np	$2nm(n$ $+m-2)+m$
CNOT		nm		$4m(n-1)$
CR_y				$m(n-1)^2$
Toffoli				$m\frac{(n-3)(n-2)(2n-5)}{3}$ $+nm^2+n^2-5m$
$e^{i\beta\text{SWAP}}$			$pm\frac{n^2-n}{2}$	

Table 6.3: Comparison of the different algorithms in terms of number of qubits, required angles and unitary operators used, for $\max(n-1,m)\geq 3$ in the case of CFLP.

Exactly $nm+n$ qubits are required for each algorithm. Due to the multi-controlled gates, in the case that $\max(n-1,m)\geq 3$, we need an additional $\max(n-1,m)-2$ ancilla qubits. The number of angles required depends heavily on the algorithm. For the UFLP and SWAP algorithms, the number of angles is also strongly dependent on the chosen value of p. Only UFLP and SWAP require a phase separator, but we will analyse later how the addition of a phase separator affects the TV and CFLP algorithms.
The composition of the circuits is relatively easy to read from the algorithms for UFLP, TV, and SWAP. However, the construction of the CFLP circuit consists of many i-controlled NOT gates with $i\in\{1,\ldots,\max(n-1,m)\}$. The $\overline{C}^iX$ gates must first be converted into C^iX gates, for which $2nm(n+m-2)\,\sigma^1$ gates are required. In addition, an R_y gate is added for each $x_{1,j}$, so that $2nm(n+m-2)+m$ single-qubit gates are required.

For the C^iR_y gates for $i \in \{1, \dots, n-1\}$, we can use Theorem 22 to calculate the number of necessary gates. First, we calculate the number of CR_y gates:

$$1 + \sum_{i=2}^{n-1}(2i-1) = \sum_{i=1}^{n-1}(2i-1) = (n-1)^2.$$

Since we use this algorithm for every j, we get $m(n-1)^2$ CR_y gates. The number of CNOT gates required for a C^iR_y gate is 2 for $i \geq 2$, so a total of $2m(n-2)$ CNOT gates are required. For the Toffoli gates, let's first look at the first part of the algorithm. The number of Toffoli gates is

$$2\sum_{i=2}^{n-1}(i-2)^2 = 2\sum_{i=1}^{n-3} i^2 = \frac{(n-3)(n-2)(2n-5)}{3}.$$

This value must still be multiplied by the number of customers m. The following identity was used in the calculation: $\sum\limits_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ (cf. [31, page 73]).
The last $C^{n-1}X$ gate for $x_{n,j}$ can be represented by $2(n-1)-3$ Toffoli gates. Since we need this many Toffoli gates for each customer, we have $m(2n-5)$ more Toffoli gates. However, further Toffoli gates are required, specifically in the second large for loop, where the C^iX gates must be decomposed into Toffoli gates. Then we get an additional

$$\begin{aligned}\sum_{i=2}^{m}(2i-3) = \sum_{i=2}^{m}(2i-1) - 2m + 2 &= \sum_{i=1}^{m}(2i-1) - 2m + 1\\ = m^2 - 2m + 1 &= (m-1)^2\end{aligned}$$

Toffoli gates. Since we have a total of n facilities, we have $n(m-1)^2$ more Toffoli gates. In total, $m\frac{(n-3)(n-2)(2n-5)}{3} + nm^2 + n^2 - 5m$ Toffoli gates are required. Additionally, there are n more CNOT gates.
If we compare the four circuits, we observe that the CFLP circuit has the largest order of magnitude in terms of the number of gates. If we compare the number of single-qubit gates alone, we have an order of magnitude of $\mathcal{O}(n^2m + nm^2)$. For $p \approx n \geq m$, this is comparable to the order of magnitude of single-qubit gates in the UFLP circuit. However, the number of Toffoli gates is particularly large. Here, the order of magnitude for CFLP is $\mathcal{O}(n^3m + nm^2)$, which is larger than the order of magnitude of all other circuits for $n \approx m$. If we set $p \approx n$, the order of magnitude of unitary operators in the SWAP algorithm can be comparable to that of the CFLP circuit.
After each iteration step, a classical optimisation of the angles takes place, which

depends significantly on the number of angles. These are the largest with SWAP, where the order of magnitude is $\mathcal{O}(pmn^2)$. Depending on the choice of p, CFLP or UFLP has the least number of angles. The smaller the number of angles, the faster the classical optimisation works.
For exact problems, the TV and CFLP algorithms can always be applied but cannot be improved further because they do not depend on a parameter p. Therefore, their quality remains fixed. Nevertheless, the comparison is worthwhile. With TV, we have an order of magnitude of $\mathcal{O}(nm)$, compared to $\mathcal{O}(n^3m + nm^2)$ with CFLP, resulting in considerably more gates with CFLP than with the TV algorithm. On the one hand the construction of the quantum circuit for TV need less gates than that of CFLP. On the other hand, CFLP has slightly fewer angles that need to be classically optimised, even if the order of magnitude remains the same. When considering the SWAP or UFLP algorithms, the approximation quality can be adjusted. If the obtained approximation is unsatisfactory, the parameter p can be increased to achieve a better approximation, although it is unclear how high p needs to be set.

6.2 Implementation of the FLP for $n = m = 2$

This chapter addresses the practical implementation of the previously presented algorithms for approximating the FLP. For this purpose, a class called "QuantumApproximateOptimisation" in Python3 is used, which was provided by ITP Hannover. One of the inputs is a problem class, which in our case is the FLP. The various objective functions, Hamiltonians, and mixers are implemented in the "FLP" problem class. This simulation handles qubits as vectors. Instead of using $nm+n$ qubits, we work with a vector of size 2^{nm+n} to simulate a quantum computer.

Additionally, two other useful classes were provided by ITP Hannover: "PureState" and "PauliString", which shorten the computing times. "PureState" runs most matrix-vector multiplications on C++, and "PauliString" represents tensor product Pauli matrices as strings. Because the interactions between Pauli matrices are known, calculations between different Pauli matrices can be performed quickly. This section analyses how well the different algorithms perform.

Direct Comparison of the Algorithms

We begin by comparing the algorithms with a simple example. This instance consists of 2 facilities and 2 customers. Let the following instance of an FLP be given:

$$\begin{aligned}
\min_{x,y} 1x_{1,1} + 4x_{1,2} + 2x_{2,1} + 10x_{2,2} + 3y_1 + 7y_2 \\
\text{s.t.} \qquad\qquad x_{1,1} + x_{2,1} = 1 \\
x_{1,2} + x_{2,2} = 1 \\
x_{1,1} \leq y_1 \\
x_{1,2} \leq y_1 \\
x_{2,1} \leq y_2 \\
x_{2,2} \leq y_2
\end{aligned}$$

There are four possible configurations in which the customers can be assigned to the facilities, as illustrated in Table 6.4.

The optimal solution has a minimum value of 8. Figure 6.1 shows the approximation of this minimum using various algorithms. It is evident that both the TV and CFLP algorithms quickly reach the minimum. In contrast, for SWAP and UFLP, the approximation of the minimum improves with larger values of p. For UFLP, the rate of improvement is much slower compared to SWAP, whereas for $p = 4$, the

Scenario	$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	y_1	y_2	Total cost
Customer 1 and 2 are both supplied by facility 1	1	1	0	0	1	0	8
Customer 1 and 2 are both supplied by facility 2	0	0	1	1	0	1	19
Customer 1 is supplied by facility 1 and customer 2 is supplied by facility 2	1	0	0	1	1	1	21
Customer 1 is supplied by facility 2 and customer 2 is supplied by facility 1	0	1	1	0	1	1	16

Table 6.4: All possible combinations for the assignment of two customers to two facilities, which do not leave any facility open unnecessarily, as well as the associated costs.

approximation achieved with SWAP matches that of CFLP and TV. This example highlights that pure softcoding performs worse than at least partially hardcoded algorithms. This will be confirmed in the following sections. It is important to note that only the integer values count in the figures, and not those in between, but these points have been connected for a better overview.

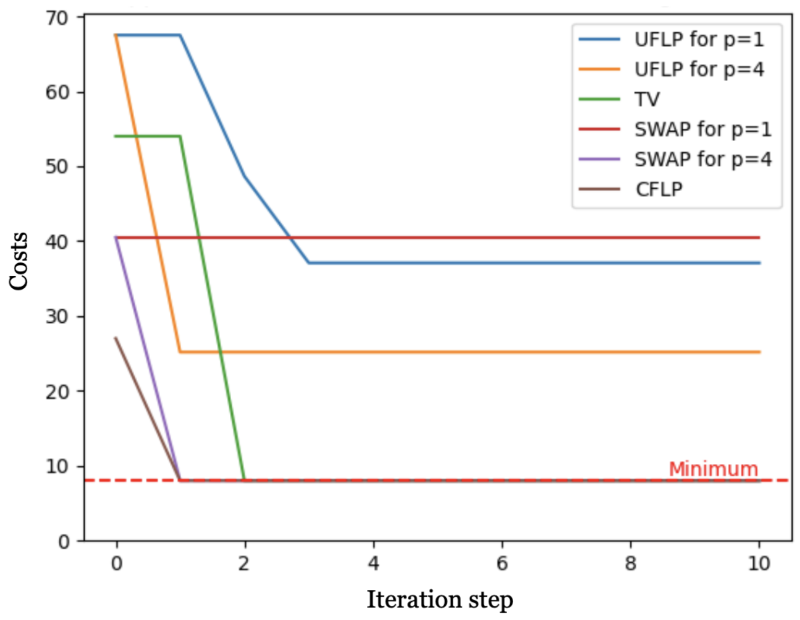


Figure 6.1: Approximation of the minimum of an explicit example for $n = m = 2$ with different FLP algorithms

So far, only a single example has been considered, but the algorithms are designed to work across a variety of instances. To test their general applicability, 1000 FLP instances were generated pseudo-randomly using the Python package "random". These 1000 instances were then averaged for each algorithm to assess their performance. To ensure a fair comparison, all instances were normalised as follows:

$$\text{Normalised cost} = \frac{\text{Minimal cost}}{\text{Approximated cost}}$$

Because the approximated costs are always greater or equal as the minimum, the algorithms approximate better if the normalised costs are closer to 1.

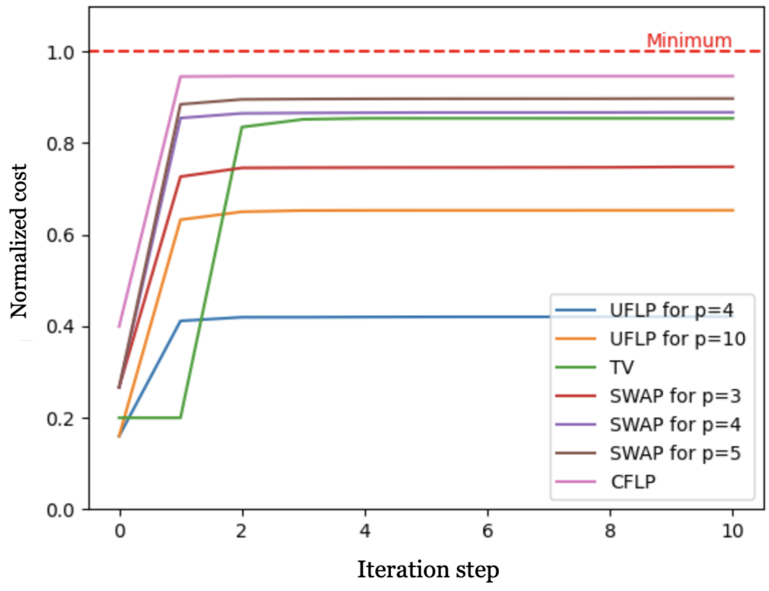


Figure 6.2: Average approximate of 1000 random instances for $n = m = 2$ for different FLP algorithms.

The algorithms can be effectively compared using Figure 6.2. In this figure, the average approximation of CFLP performs the best. Based on the results from Figure 6.1, the values of p for SWAP and UFLP were deliberately chosen to be larger. It can be seen that SWAP for $p = 4$ and TV have very similar approximations. With a smaller p, SWAP performs worse, while increasing p improves the approximation, making it better than TV. As previously assumed, UFLP also approximates much worse for large p, with an average normalised cost of 0.6, compared to the other algorithms, which are at a normalised cost of 0.85. CFLP even achieves a normalised cost of 0.95.
To analyse the quality of the algorithms even more precisely, it is also necessary to compare how many gates and angles are required by the different algorithms.

The Table 6.5 illustrates the gates they are used for the various algorithms.

Algorithm	Angles	Single qubit gates	CNOT	Two qubit gates	Toffoli
UFLP $p = 4$	8	76			
UFLP $p = 10$	20	190			
TV	6	22	4		
SWAP $p = 3$	9	39		24	
SWAP $p = 4$	12	52		32	
SWAP $p = 5$	15	65		40	
CFLP	2	10	4		2

Table 6.5: Decomposition of the mixers of different algorithms into gates.

For SWAP, the $\exp(i\beta\text{SWAP})$ gates in SWAP were broken down into four two-qubit Pauli matrices. The reverse NOT gates are decomposed into CNOT and two X gates. UFLP and SWAP require a substantial number of angles and gates to match the approximation performance of the TV and CFLP algorithms. Specifically, UFLP exhibits the worst approximation and the highest gate count for $p = 10$ compared to the algorithms TV and SWAP for $p \in \{3,4,5\}$ and CFLP. When comparing SWAP with TV, both offer similar approximation quality for $p = 4$, but TV needs fewer angles and gates. Therefore, SWAP should be used for $p > 4$, as it provides a slightly better approximation than TV but involves more angles and gates. To minimise the number of angles, TV and CFLP are preferable. CFLP requires fewer angles than TV for each choice of n and m, but the order of magnitude of the gates in CFLP grows much faster than that of TV, as already shown in Table 6.3.
In the case of $n = m = 2$, we use Theorem 18 to decompose the two Toffoli gates. One Toffoli gate corresponds to 3 CNOT and 4 single-qubit gates, so we have 18 single-qubit gates and 10 CNOT gates in CFLP as opposed to 22 single-qubit gates and 4 CNOT gates in TV. Thus, the two algorithms differ mainly in the number of CNOT gates, which are lower in the TV algorithm. This means that CFLP requires slightly more gates, but on average, it also provides a better approximation.

Dependence of the SWAP and UFLP Algorithms on p

For larger values of p, it is worth considering whether the SWAP algorithm could potentially outperform the CFLP algorithm or whether the UFLP approximation for very large p could be superior to that of TV. To investigate this, we will calculate the normalised minimal costs for SWAP and UFLP for different values of p and present them as a function of p.

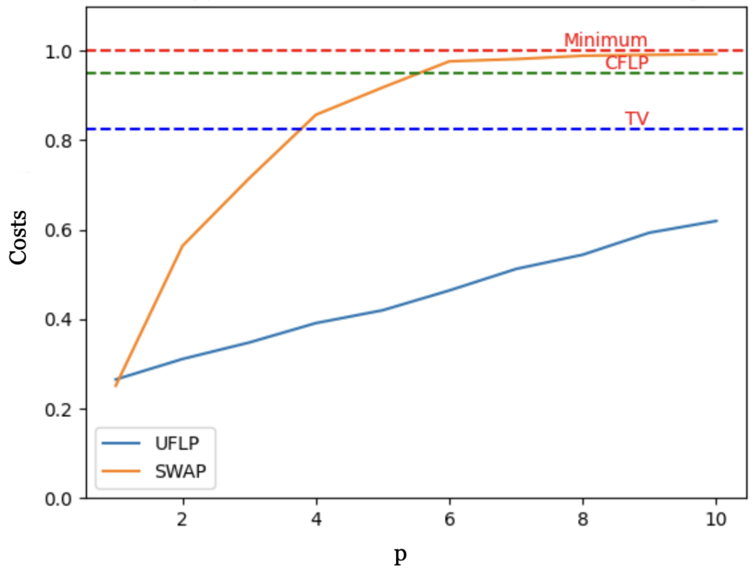


Figure 6.3: p-dependence of the average approximation for the SWAP and UFLP algorithm.

Figure 6.3 shows that the average approximation improves consistently, at least up to $p = 10$. For $p \geq 6$, the SWAP algorithm, on average, surpasses the pure hardcoded CFLP algorithm. However, it must also be noted that p was chosen to be larger than n or m, resulting in a higher number of gates required to implement the algorithm compared to the TV and CFLP algorithms. The UFLP algorithm also shows continuous improvement for larger p values, yet remains the last in terms of average approximation up to $p = 10$.

6.3 Approximation Quality as a Dependency of the Size of the FLP

Having previously only considered the case $n = m = 2$, we will now examine scenarios with more facilities and customers. From previous considerations, it became evident that the UFLP algorithm exhibits poor approximation values even with two customers and two facilities. Therefore, we will exclude it from further analysis. We have also observed that the SWAP algorithm requires a significant number of gates for $n = m = 2$. To compare larger instances, we will focus on the CFLP and TV algorithms.

Dependence on the Number of Customers

Firstly, the dependency on the number of customers is examined by varying the number of customers while keeping the number of facilities constant. The number of facilities is set to the minimum value of 2. If there were only one facility, the assignment would be trivial since all customers would be assigned to that single facility.

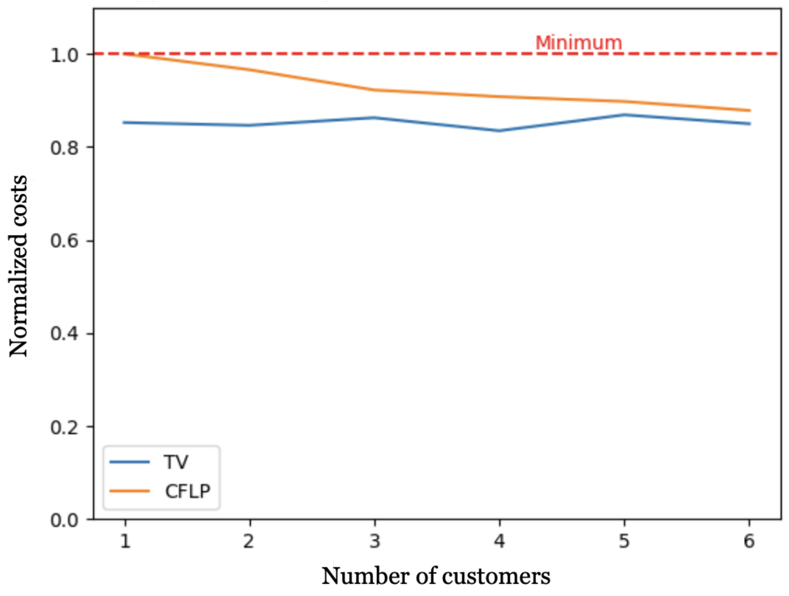


Figure 6.4: Approximation of 100 random instances for $n = 2, m \in \{1, \ldots, 6\}$.

Figure 6.4 illustrates that as the number of customers increases, the approximation accuracy for the CFLP algorithm slightly declines, from 0.95 for $n = 2$ to approximately 0.85 for $n = 6$ on average. The TV algorithm's performance remains relatively stable, fluctuating between 0.8 and 0.85. Up to $m = 6$, the CFLP algorithm generally outperforms the TV algorithm on average. Despite the decrease in performance for the CFLP algorithm, both algorithms maintain stability even with an increased number of customers.

In future studies, it may be possible to analyse the FLP with both algorithms on a quantum computer across a significantly larger number of customers and over more than 100 different instances. An interesting question to explore would be whether there exists a value of m for which the CFLP and TV algorithms achieve the same approximation accuracy, or if, for larger m, the TV algorithm might outperform the CFLP algorithm.

Dependence on the Number of Facilities

Having considered the dependency of the approximation on the number of customers, we now examine the dependency on the number of facilities. We consider both the cases $m = 1$ and $m = 2$. It is important to note that the number of qubits is given by $nm + n$. If the number of customers is increased by one, the number of qubits increases by n. Conversely, if the number of facilities is increased by one, the number of qubits increases by $m + 1$. This implies that the number of qubits grows faster with increasing n than with increasing m, when the other value (either the number of customers or the number of facilities) is held constant. For example, comparing $n = 2, m = 5$ with $n = 5, m = 2$ results in 12 and 15 qubits, respectively. We first consider the minimum number of customers to be 1 and then increase the number of facilities.

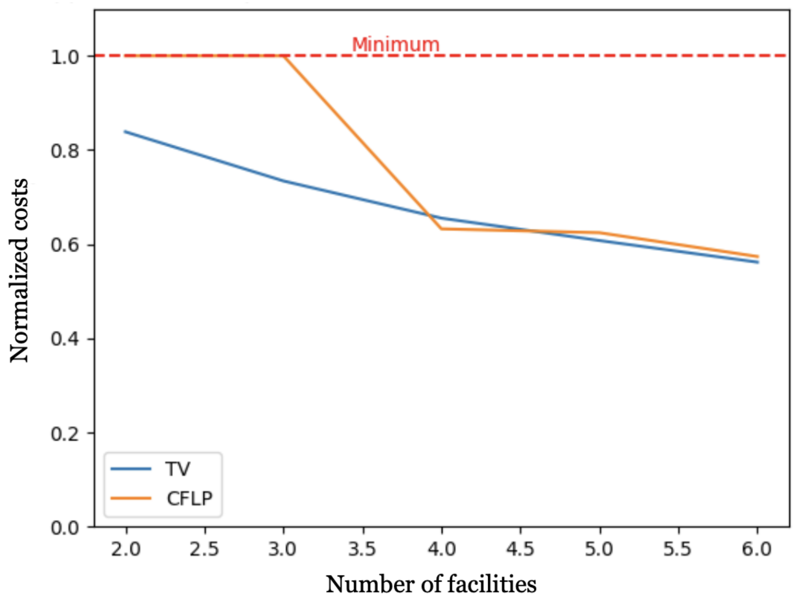


Figure 6.5: Approximation of 400 random instances for $m = 1, n \in \{2, \ldots, 6\}$.

Figure 6.5 demonstrates that for $n \leq 3$, the CFLP algorithm provides a better average approximation than the TV algorithm. From $n = 4$ onwards, the difference between the two algorithms becomes negligible. It is evident that for $m = 1$ and $n \in \{2, 3\}$, the CFLP algorithm converges for all 400 random instances. This raises the question of whether the approximation also deteriorates for larger values of m when $n \geq 4$.

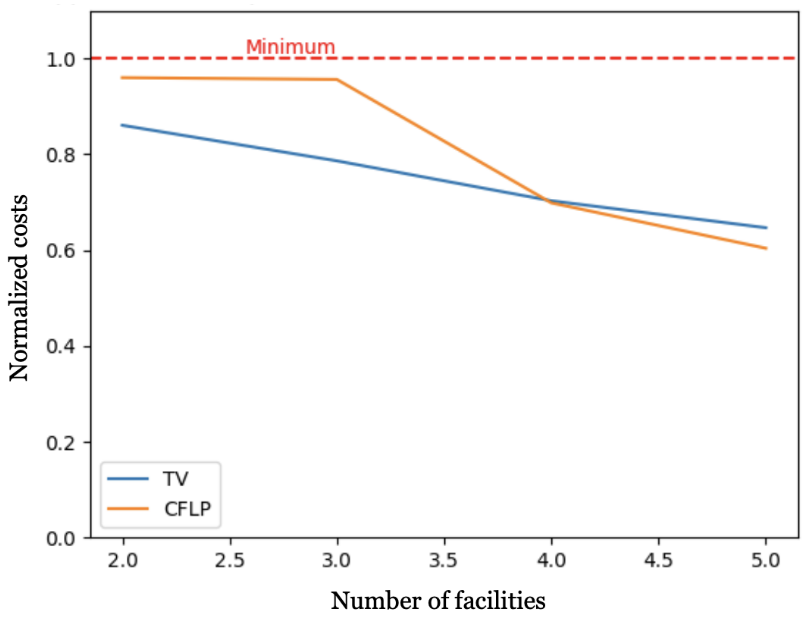


Figure 6.6: Approximation of 100 random instances for $m = 2, n \in \{2, \ldots, 5\}$.

Figure 6.6 indicates that the approximation accuracy decreases when there are two customers and four or more facilities. In the cases considered, the approximation decreases more significantly when the number of facilities is increased compared to when the number of customers is increased.

This observation raises the question of whether this trend holds for larger values of n or m. Specifically, if $\max(n, m) \geq 4$, do the TV and CFLP algorithms provide the same approximation or decreases more.

Higher Order Problems

After previously setting one of the parameters n or m to a maximum of 2, the TV and CFLP algorithms are now applied to problems with $\min(n, m) = 3$.
Figure 6.7 shows the average approximation for three different instances. It is evident that, especially for $m \geq n$, the approximation of CFLP is better than that of the TV algorithm. In the case of $m < n$, there is no significant difference between the two algorithms, consistent with the previously considered cases.

Thus, in the scenarios examined, the CFLP algorithm performs better, particularly when there are more customers than potential facilities. In many everyday applications, companies often have a limited number of potential locations for their facilities but significantly more customers. A special case occurs when a facility is to be built to supply one or only a few other companies.

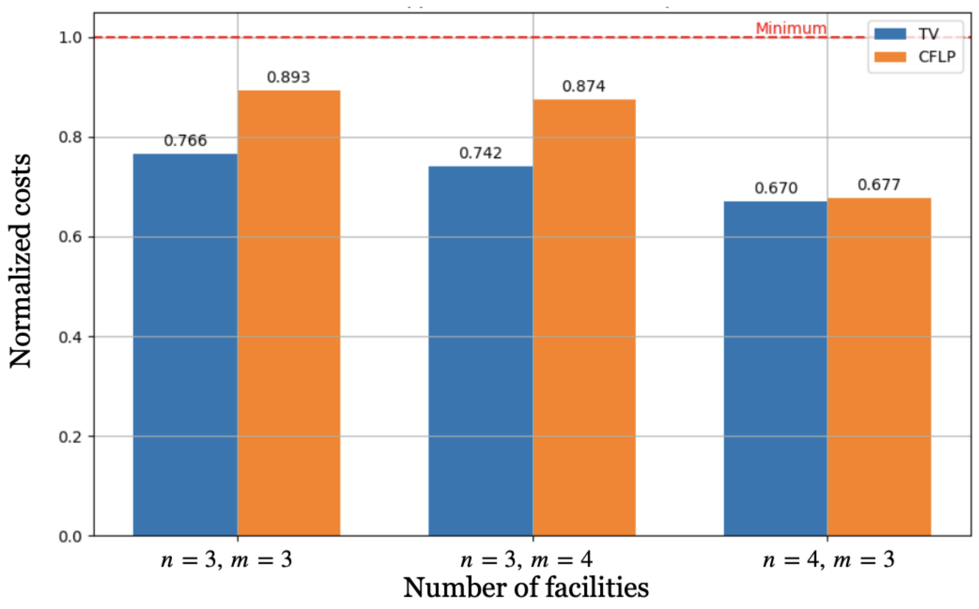


Figure 6.7: Approximation of 100 random instances for $\min(n, m) = 3$ from the CFLP and TV algorithm

6.4 Customisation of the Algorithms

Dependence on the Number of Angles

Thus far, we have only examined the overall performance of the algorithms. To analyse the CFLP algorithm in more detail, we will vary the number of angles used for optimisation in the specific case where $n = m = 3$. As we know, the CFLP algorithm requires a total of $(n-1)m = 2 \cdot 3 = 6$ angles, while the TV algorithm necessitates $n \cdot m + n = 12$ angles. We will simulate how the CFLP algorithm performs when fewer angles are optimised, and the remaining angles are selected as in Algorithm 10.

Figure 6.8 demonstrates that only every second additional angle causes a change in the approximation. In each case, two angles are responsible for connecting one of

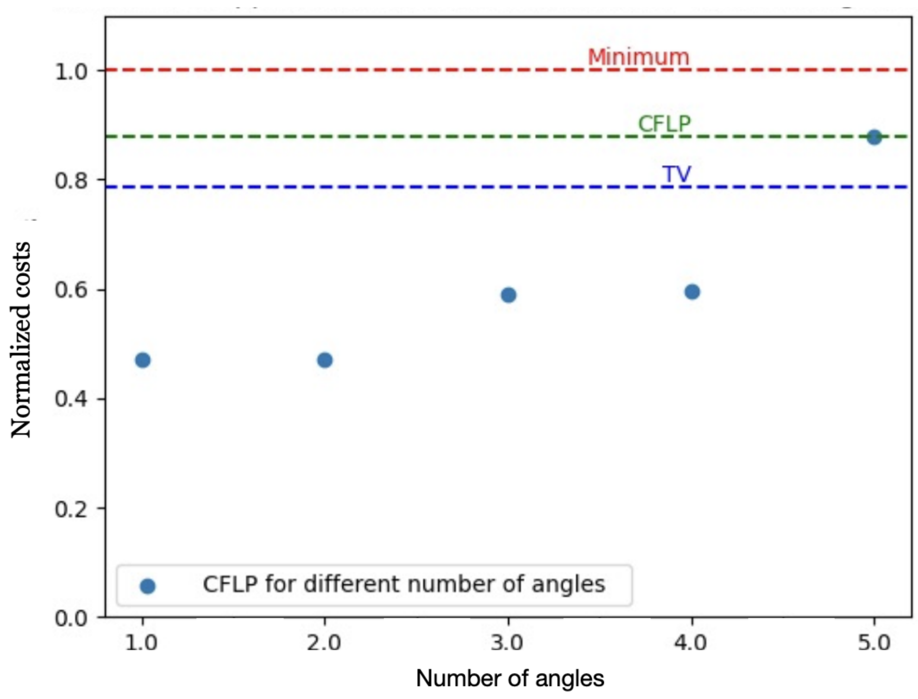


Figure 6.8: Normalised approximation for a different number of optimised angles for the CFLP algorithm for $m = n = 3$.

the three customers to the possible facilities. For $n = m = 3$, adding an angle to be optimised results in a better approximation if it affects a new customer. This is evidenced by the staircase structure of the figure.
It is now interesting to determine if this phenomenon occurs for each instance when the number of facilities is increased by one.

Influence of a Phase Separator to CFLP and TV

The phase separator is indispensable for QAOA, which is why it is included in the UFLP algorithm. It is also fully implemented for the SWAP algorithm, and the QAOA mixer is at least partially implemented. Without a phase separator, the approximations using the SWAP algorithm are significantly worse. Even for $n = m = 2$, without a phase separator, the value of the normalised costs of the superposition of all feasible states is often in the range of 0.2-0.3. This raises the question of whether the use of a phase separator can improve the approximations following the respective algorithms of CFLP and TV. The algorithms are compared both with and without a phase separator.

An additional angle is required for the phase separator, and the number of additional

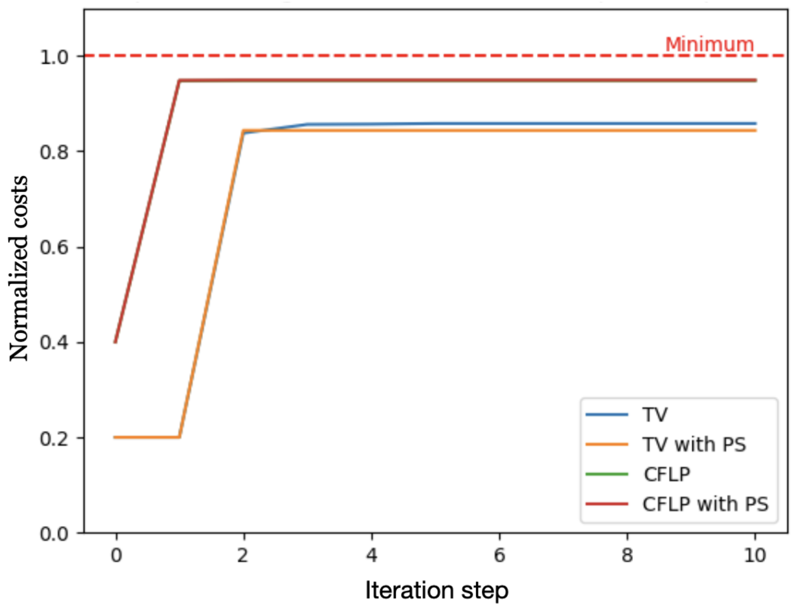


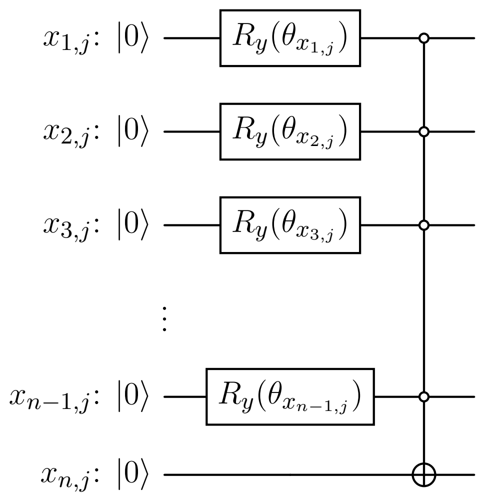
Figure 6.9: Comparison of the CFLP and the TV algorithm with and without phase separator for $m = n = 2$.

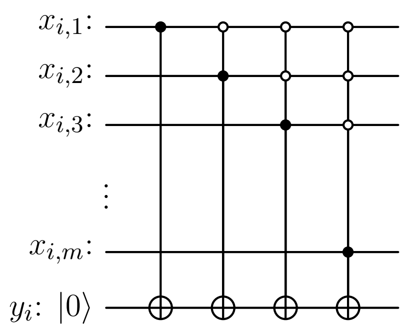
single-qubit gates needed can be found in Table 6.2. The phase separator was employed in each iteration step following the respective algorithms and was then optimised classically according to the angles.
When simulating various random instances, it is observed that the phase separator has minimal influence on the CFLP algorithm. The average approximation of the CFLP without a phase separator is 0.951, while with a phase separator, it is 0.953. This difference is not easily discernible in Figure 6.9. In the figure, the TV algorithm without a phase separator performs slightly better, but the difference is very minimal. Given that the phase separator does not significantly affect the approximation, the versions of the TV and CFLP algorithms without a phase separator were used in all other analyses. This approach also helps in reducing the computation time.

Simplification of the CFLP Algorithm

Algorithm 12 describes the CFLP algorithm, which consists of two parts: the first part connects customers with facilities, and the second part opens the facilities. It is evident that many controlled operations occur in both parts. These operations ensure that only states in which a customer is connected to exactly one facility are considered, and that a facility is opened only if it supplies at least one customer, as demonstrated by Theorem 31.
Consider Theorem 3, which states that it does not matter if a customer is connected

to exactly one facility or at least one facility. The minimum of the FLP remains unchanged. The following circuit illustrates that the number of gates is reduced, which results in an increased number of feasible states.





For each individual customer, connections to other customers $1, \ldots, n-1$ are managed only by applying an R_y gate and without additional checks to determine if the customer is already connected to another facility. To ensure that the customer is connected to at least one facility, a $\overline{C}^{n-1}X$ gate is applied to the last connection. This means that the customer is connected to facility n at the latest. The opening of the facilities remains the same like the CFLP algorithm. All states which can be created with the circuit fulfils the equivalent assignment constraints from theorem 3 and the equivalent opening constraint from theorem 4.

Algorithm 13: Simplified Quantum Circuit for FLP

Given are:

- n facilities and m customers
- The first $n \cdot m$ qubits labels as $x_{i,j}$ $\forall i \in \{1, \ldots, n\},\ j \in \{1, \ldots, m\}$
- The last n qubits labels as y_i $\forall i \in \{1, \ldots, n\}$
- Angles $(\theta_{x_{1,1}}, \ldots, \theta_{x_{n-1,m}}) \in \mathbb{R}^{(n-1)m}$

```
Initialise: $\underbrace{|0 \ldots 0\rangle}_{nm+n \text{ times}}$
for $j = 1, \ldots, m$ do
    for $i = 1, \ldots, n-1$ do
        apply $R_y(\theta_{x_{i,j}})$ on $x_{i,j}$
    apply $\overline{C}_{x_{1,j}} \ldots \overline{C}_{x_{n-1,j}} X$ to the target $x_{n,j}$
for $i = 1, \ldots, n$ do
    apply $C_{x_{i,1}} X$ to the target $y_i$
    for $j = 2, \ldots, m$ do
        apply $\overline{C}_{x_{i,1}} \ldots \overline{C}_{x_{i,j-1}} C_{x_{i,j}} X$ to the target $y_i$
```

The question now is how the Simplified CFLP (SCFLP) Algorithm 13 performs against the CFLP Algorithm 11.

Gates	Single qubit	CNOT	Toffoli
Number of gates	$4mn^2 - nm - 3m$	n	$nm^2 - 5m + 1$

Table 6.6: Number of gates of the SCFLP algorithm for $\min(n-1, m) \geq 3$.

To compare the gates of the SCFLP and CFLP algorithms, we present the number of gates for SCFLP in Table 6.6 alongside those for CFLP (see Table 6.3). It is evident that the order of magnitude of Toffoli gates for SCFLP is $\mathcal{O}(mn^2)$, compared to $\mathcal{O}(mn^3)$ for the CFLP algorithm. This significantly reduces the number of gates, especially when there are many facilities.

Since the SCFLP algorithm can result in more feasible states, it is necessary to evaluate how the two algorithms perform against each other when implemented. The CFLP and SCFLP algorithms differ only for $n \geq 3$, so we compare the two algorithms for instances with $n \geq 3$.

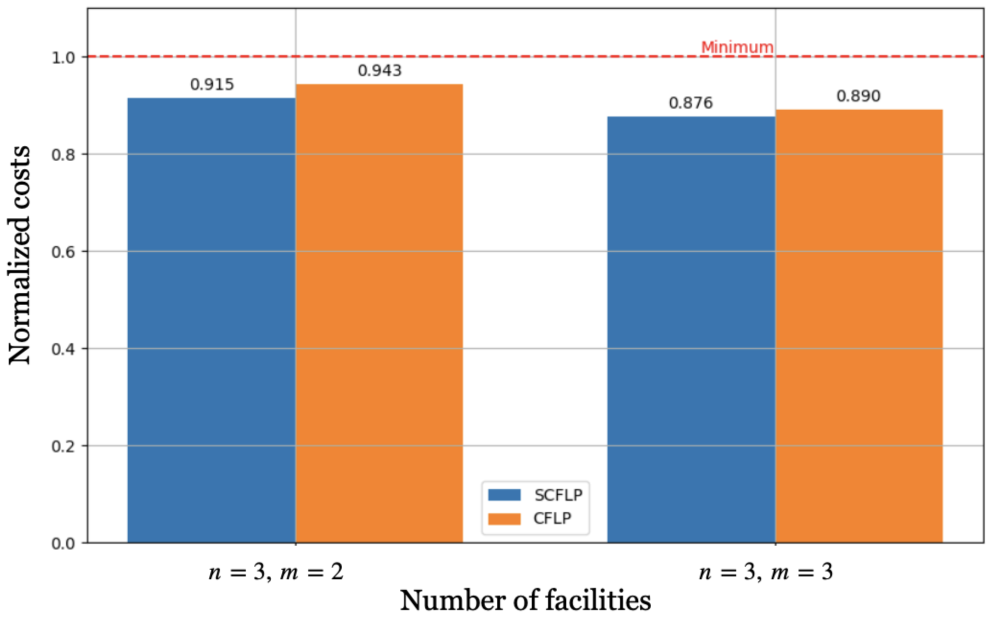


Figure 6.10: Distribution of the CFLP and SCFLP algorithm for 1000 random instances for $n = 3$

When comparing the two algorithms in Figure 6.10, it is evident that the SCFLP algorithm performs slightly worse on average than the CFLP algorithm. Nevertheless, both algorithms achieve a better average approximation than the TV algorithm in the considered cases.

6.5 Distribution of the Approximations

Thus far, we have compared the different algorithms based on their average approximation. However, we have not determined the frequency with which each algorithm finds the exact minimum. Since the distribution of the approximations is not known, we cannot make direct statements about standard deviation or variance. Therefore, various previously shown instances are compared again with regard to their distribution.

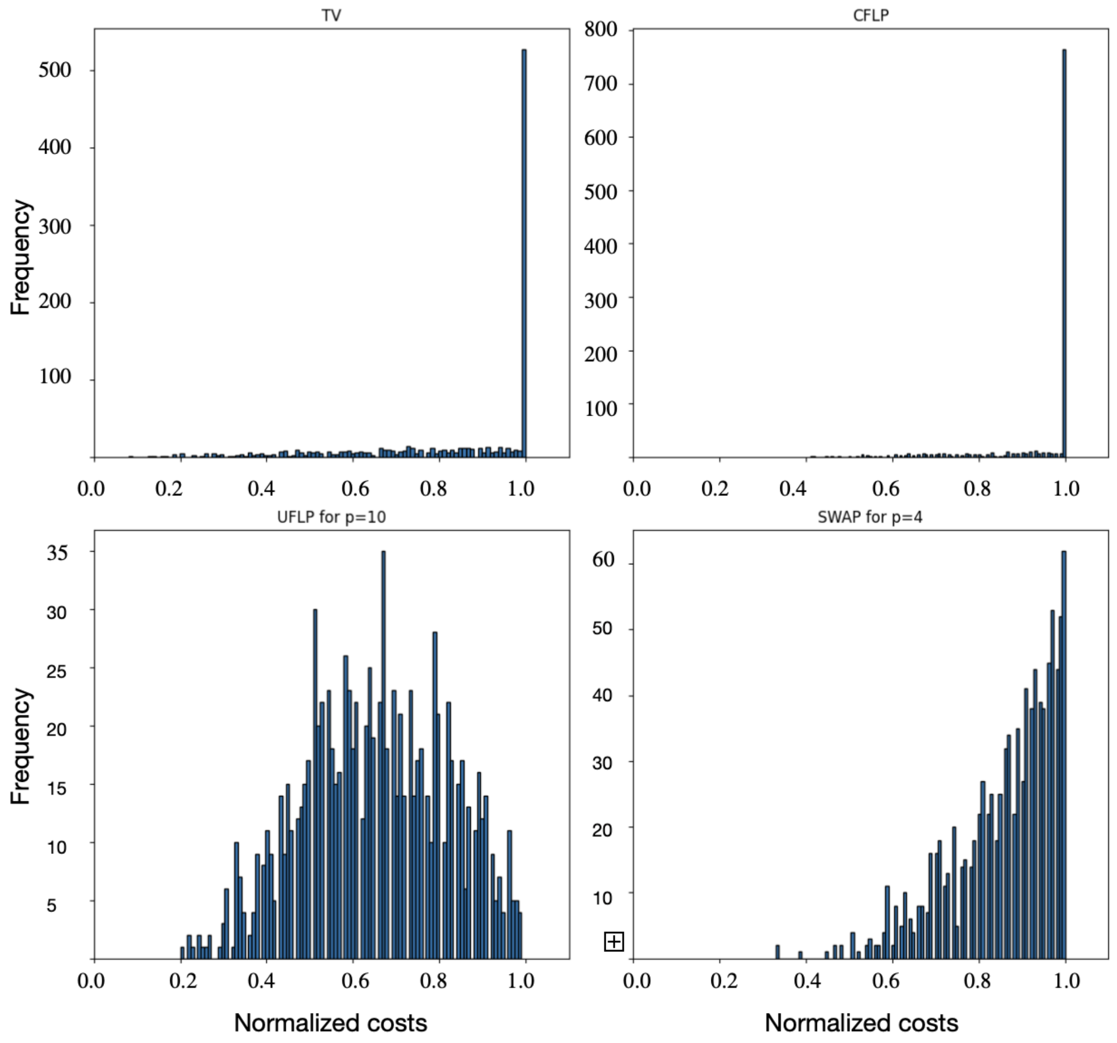


Figure 6.11: Distribution of the CFLP, TV, SWAP and UFLP algorithm for 1000 random instances with $n = m = 2$.

The distributions illustrated in Figure 6.11 reveal how frequently each particular approximation was achieved. For clarity, the normalised approximations were rounded to two decimal places. The CFLP algorithm identified the minimum for $n = m = 2$ in over 75% of the cases considered. In contrast, the TV algorithm found the minimum in around 50% of cases. The UFLP algorithm never found the minimum for $p = 10$, with approximations clustering around 0.6. The SWAP algorithm often failed to find the exact minimum, but usually achieved approximations between 0.8 and 1.0. The remaining 50% of the states in the TV algorithm exhibit an approximately uniform distribution of approximations between 0.2 and 1.0. While the SWAP and TV algorithms yield a similar mean value, their distributions differ. It is evident that the CFLP algorithm performs best in this scenario.
The distributions for larger orders of magnitude are also of great interest, for which we compare the CFLP, SCFLP, and TV algorithms in two cases.

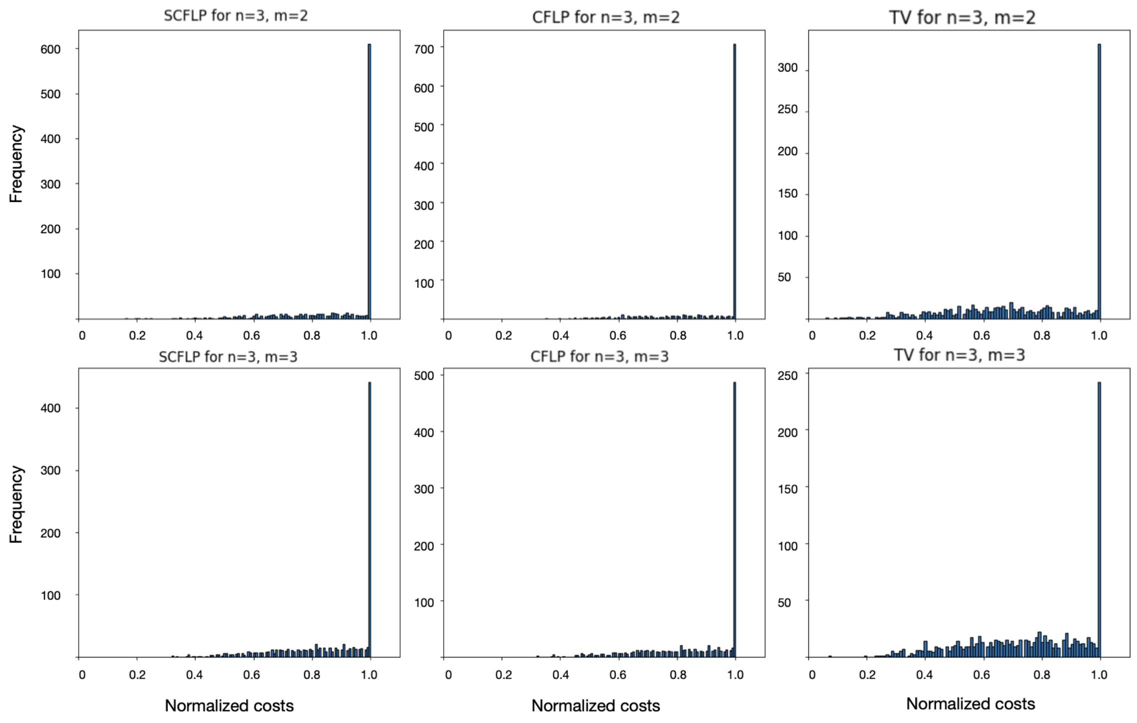


Figure 6.12: Distribution of the CFLP, TV and SCFLP algorithm for 1000 random instances for $n = 3$.

When comparing the three algorithms, it is evident that the CFLP algorithm achieves the exact minimum most frequently, followed by the SCFLP algorithm, which approximates the minimum in about 10% fewer cases. The TV algorithm approximates the minimum about half as often as the CFLP algorithm.

7 Conclusion and Outlook

In this work, we explored various quantum algorithms to approximate the Facility Location Problem (FLP). Given that the FLP is an NP-hard problem and assuming P $\neq$ NP, there is no polynomial-time algorithm for solving it. Quantum algorithms present a fascinating approach to approximating the FLP.
We investigated four different algorithms, aiming to determine whether soft-coded algorithms (UFLP) or hard-coded quantum algorithms (CFLP) perform better.
Additionally, we examined algorithms that combine both soft-coded and hard-coded components (TV, SWAP).
Our analysis revealed that the TV algorithm is the most efficient in terms of gate count. The UFLP and SWAP algorithms are effective only for large values of p, requiring substantial gate counts, thus performing worst in this regard. Among the algorithms, CFLP and TV cannot be further improved for a given specific instance. Conversely, the SWAP and UFLP algorithms, based on QAOA, depend on p.
Increasing the value of p enhances the approximation, but even for high values of p, the UFLP algorithm performs worst and requires the most gates. This indicates that purely soft-coded algorithms provide the least effective approximation, as QAOA was developed for arbitrary unconstrained combinatorial optimisation problems. All other algorithms are specially adapted to the FLP, yielding better results.
For instances where the number of customers exceeds the number of facilities, the CFLP algorithm performs better on average than the TV algorithm. This also applies to the simplified CFLP version, SCFLP, suggesting that purely hard-coded algorithms can be more effective than partially soft-coded ones in such scenarios. However, when comparing the CFLP algorithm with the SWAP algorithm, SWAP achieves a better approximation for $n = m = 2$ and $p \geq 6$ than CFLP, though at the cost of a significantly higher gate count. For larger problem instances, the value of p must be increased substantially for SWAP to perform comparably to CFLP. If there are significantly more potential facilities than customers, the difference between the TV and CFLP algorithms diminishes. The advantage lies in everyday problems where the number of customers typically exceeds the number of facilities, making CFLP approximations generally superior to TV. If the number of angles is reduced according to the CFLP algorithm, the approximation accuracy decreases. However,

for $m = n = 3$, the number of angles can be reduced by one without significant loss in accuracy. Although a phase separator is used in the SWAP and UFLP algorithms, applying it to the TV or CFLP algorithm does not result in a significant difference in approximations with or without the phase separator.
Examining the distribution of the different approximations further highlights the distinction between the TV and CFLP algorithms and the SWAP and UFLP algorithms. The first two algorithms often achieve the exact minimum, while the latter two exhibit a greater spread in approximations. Hence, for achieving the exact minimum, the TV and CFLP algorithms are the most suitable. The SWAP algorithm also performs well if a good approximation suffices.
In summary, the hard-coded CFLP algorithm generally performs best. However, the TV algorithm, as presented in [4], has clear advantages, particularly in its low gate count requirements. Due to computational constraints, only small instances were simulated. This is because the TV algorithm necessitates vectors of size 2^{nm+n} to simulate $nm+n$ qubits. In our simulation of the CFLP, an additional $\max(n-1, m)-2$ qubits were also required. As future quantum computers evolve, larger instances could be executed, enabling a more detailed investigation of the differences between the TV and CFLP algorithms. For some instances, the algorithms do not approximate as well as for others, so it would be interesting to determine whether there is a way to characterise which instances can be approximated well and which cannot. Further research in this area could focus on whether and how the hard-coded algorithm for the FLP can be improved. This work has already presented an algorithm, SCFLP, which reduces the number of gates required. The approximation is only slightly reduced compared to the CFLP algorithm. Both the CFLP and SCFLP algorithms consist of two parts. The first part assigns customers to facilities. Here, the SCFLP algorithm has succeeded in greatly reducing the number of gates. One possible approach would be to determine how the number of gates in the second part of the CFLP algorithm, which involves opening the facilities that supply a customer, can be reduced.
Some further formulations for the FLP were presented in Chapter 2. The MUCFLP is a special case of the FLP where the cost c is a metric. All the algorithms considered can also be applied to this problem. However, it is important to find out how the algorithms approximate instances of the MUCFLP. Some classical algorithms only work for FLP instances that fulfil the triangle inequality. It would be interesting to find out how the quantum algorithms could be improved, especially for MUCFLP instances. The k-FLP is a possible generalisation of the FLP where a maximum of k facilities are opened. Adapting the algorithms to this scenario would be worthwhile. In reality, a single facility cannot supply an unlimited number of customers. Another possible generalisation of the algorithms for the Capacitated FLP would be a fascinating research topic as well.

Bibliography

[1] Isaac L. Chuang Michael A. Nielsen. *Quantum Computation and Quantum Information.* 2010.

[2] Adriano Barenco **andothers**. *Elementary gates for quantum computation.* **november** 1995. DOI: 10.1103/physreva.52.3457. URL: http://dx.doi.org/10.1103/PhysRevA.52.3457.

[3] Edward Farhi, Jeffrey Goldstone **and** Sam Gutmann. *A Quantum Approximate Optimization Algorithm.* 2014. arXiv: 1411.4028 [quant-ph].

[4] Miguel Paredes Quinones **and** Catarina Junqueira. *Modeling Linear Inequality Constraints in Quadratic Binary Optimization for Variational Quantum Eigensolver.* 2020. arXiv: 2007.13245 [quant-ph].

[5] Gereon Koßmann **andothers**. *Open-Shop Scheduling With Hard Constraints.* 2023. arXiv: 2211.05822 [quant-ph].

[6] Dennis L. *So lange braucht ein Rettungswagen bis zum Eintreffen im Notfall.* 2021. URL: https://www.forschung-und-wissen.de/nachrichten/oekonomie/so-lange-braucht-ein-rettungswagen-bis-zum-eintreffen-im-notfall-13375259#. (Letzter Zugriff: 01.02.24).

[7] Jens Vygen. *Approximation Algorithms for Facility Location Problems. Lecture Notes.* Research Institute for Discrete Mathematics,, University of Bonn. 2005.

[8] João Pedro Pedroso **andothers**. *Facility location problems.* 2012. URL: https://scipbook.readthedocs.io/en/latest/flp.html. (Letzter Zugriff: 14.02.24).

[9] lak. *Set Cover.* 2012. URL: https://ac.informatik.uni-freiburg.de/lak_teaching/ws11_12/combopt/notes/set_cover.pdf. (Letzter Zugriff: 17.02.24).

[10] Lennart Binkowski. *Constraint Graph Model Analysis of the Quantum Alternating Operator Ansatz.* 2022.

[11] Thomas Wick Thomas Richter. *Einführung in die Numerische Mathematik.* 2017.

[12] Kim Kern. *Komplexitätsklasse P*. 2012. URL: `http://www.inf.fu-berlin.de/lehre/WS11/ProSem-ThInf/Komplexittsklasse_P.pdf`.

[13] Juraj Hromkovic. *Theoretische Informatik*. 2011.

[14] Bishal Thapa San Tan. *Algorithmic Power Tools*. 2009. URL: `https://www.khoury.northeastern.edu/home/rraj/Courses/7880/F09/Lectures/FacLocFiltering.pdf`.

[15] Arindam Khan. *Approximation Algorithm for Set Cover*. 2014. URL: `https://www14.in.tum.de/personen/khan/Arindam%20Khan_files/1.%20Set%20Cover.pdf`.

[16] Dave Mount. *Greedy Approximation: Set Cover*. 2017. URL: `https://www.cs.umd.edu/class/fall2017/cmsc451-0101/Lects/lect09-set-cover.pdf`.

[17] Chvatal. *Mathematics of Operations Research 4*. 1979.

[18] Anupam Gupta. *Lecture 6: Facility location: greedy and local search algorithms*. 2008. URL: `https://www.cs.cmu.edu/~anupamg/adv-approx/lecture6.pdf`.

[19] Deeparnab Chakrabarty. *Local Search Algorithms for Facility Location*. 2022. URL: `https://www.cs.dartmouth.edu/~deepc/LecNotes/Appx/2b.%20Local%20Search%20Algorithms%20for%20Facility%20Location.pdf`.

[20] Vijay Vazirani Klaus Jansen Stefano Leonardi. *Approximation Algorithms for Combinatorial Optimization*. springer, 2002. ISBN: 3-540-44186-7.

[21] Vazirani. *Lecture 2: Quantum Algorithms*. 2009. URL: `https://people.eecs.berkeley.edu/~vazirani/s09quantum/notes/lecture2.pdf`.

[22] Wolfram Bauer. *Funktionalanalysis*. 2023.

[23] *Unitäre und orthogonale Abbildungen*. 2021. URL: `https://www2.physik.uni-muenchen.de/lehre/vorlesungen/wise_21_22/r_rechenmethoden_21_22/skript/auth/16-L8_1-2-vor-Matrizen-V-UnitaerOrthogonalHermiteschSymmetrisch.pdf`.

[24] Prof. Dr. Wolfgang von der Linden. *Quantenmechanik*. 2000. URL: `https://itp.tugraz.at/LV/arrigoni/QM/a_qm_all.pdf`.

[25] Stefan Boresch. *Taylorreihen*. 2011. URL: `https://www.mdy.univie.ac.at/lehre/mathe/molbio/folien/taylor.pdf`.

[26] *General parametrisation of an arbitrary 2×2 unitary matrix*. 2023. URL: `https://quantumcomputing.stackexchange.com/questions/5199/general-parametrisation-of-an-arbitrary-2-times-2-unitary-matrix`.

[27] Andrew M. Childs. *Lecture Notes on Quantum Algorithms*. 2022. URL: `https://www.cs.umd.edu/~amchilds/qa/qa.pdf`.

[28] Adam Sawicki1 **and** Katarzyna Karnas. *Criteria for universality of quantum gates*. 2017. URL: `https://arxiv.org/pdf/1610.00547.pdf`.

[29] Sha-Sha Wang **andothers**. *Variational quantum algorithm-preserving feasible space for solving the uncapacitated facility location problem*. 2024. arXiv: `2312.06922 [quant-ph]`.

[30] Maximilian Balthasar Mansky **andothers**. *Decomposition Algorithm of an Arbitrary Pauli Exponential through a Quantum Circuit*. 2023. arXiv: `2305.04807 [quant-ph]`. URL: `https://arxiv.org/abs/2305.04807`.

[31] Merziger. *Formeln + Hilfen Höhere Mathematik*. 2014.